

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**HIGH-LEVEL POWER ESTIMATION OF DSP
CIRCUITS IMPLEMENTED IN FPGAS**

TESIS DOCTORAL

Ružica Jevtić
Ingeniera en Electrónica

2009

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE MADRID



PH.D. THESIS

HIGH-LEVEL POWER ESTIMATION OF DSP CIRCUITS
IMPLEMENTED IN FPGAs

Author:

Ružica Jevtić
Electrical Engineer

Advisor:

Carlos Carreras Vaquer
Telecommunication Engineer, Ph.D.
Universidad Politécnica de Madrid

2009

Ph.D. THESIS: High-level Power Estimation of DSP Circuits Implemented in FPGAs

AUTHOR: Ružica Jevtić

ADVISOR: Carlos Carreras Vaquer

El tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día de de 200...., para juzgar la Tesis arriba indicada, compuesto por los siguientes doctores:

PRESIDENTE: Dr.

VOCALES: Dr.

Dr.

Dr.

SECRETARIO: Dr.

Realizado el acto de lectura y defensa de la Tesis el día de de 200.... en la E.T.S. de Ingenieros de Telecomunicación, acuerda otorgarle la calificación de:

.....

EL PRESIDENTE :

EL SECRETARIO :

LOS VOCALES :

To my mother

Contents

Contents	i
Abstract	v
Resumen	vii
Acknowledgments	ix
List of Figures	xi
List of Tables	xv
List of Acronyms	xvii
List of Notations	xix
1 Introduction	1
1.1 Motivation	1
1.1.1 FPGA structure	2
1.1.2 Power consumption	4
1.1.3 Power optimization	8
1.1.4 Power estimation	10
1.2 Objectives	11
1.3 Book organization	12
1.4 Publications	14
2 Model characterization and verification	15
2.1 FPGA Power estimation based on on-board measurements: background	16
2.2 Measurement setup	18
2.2.1 Wire capacitance extraction	21
2.2.2 Logic and Input buffer power	23
2.2.3 Effective wire capacitances	24
2.3 XPower	25

3	Power estimation models for interconnections	27
3.1	Interconnect power estimation: background	27
3.1.1	Post-placement techniques	28
3.1.2	Pre-placement techniques	30
3.1.3	Rent's rule	33
3.1.4	Summary of the previous work on interconnect estimation	34
3.2	Interconnection Power Analysis	36
3.3	High-level Point-to-point Interconnect Power Model	39
3.4	High-level Multi-point Interconnect Power Model	42
3.5	Experimental results	44
3.5.1	Point-to-point interconnect model evaluation	45
3.5.2	Multi-point interconnect power model evaluation	50
3.5.3	Model evaluation for DSP test designs	51
3.6	Conclusions	54
3.6.1	Future work	55
4	Power estimation models for logic	57
4.1	Signal model	59
4.1.1	Zero-mean signals	59
4.1.2	Non-zero mean signals	65
4.2	Switching activity computation	67
4.2.1	Switching activity: background	67
4.2.2	Methodology for switching activity computation	70
4.2.3	Structural model	73
4.3	Glitching model	80
4.3.1	Glitching: background	81
4.3.2	Glitching methodology	83
4.4	Logic power estimation	85
4.4.1	High-level logic power estimation: background	85
4.4.2	Logic power model	93
4.5	Experimental results	97
4.5.1	Signal model accuracy	98
4.5.2	Logic power model for LUT-based components : XPower	101
4.5.3	Logic power model for LUT-based components: on-board measurements	114
4.5.4	Logic power model for embedded blocks: on-board measurements	116
4.5.5	Correlation factor for logic power estimates	118
4.6	Conclusions	120
4.6.1	Future work	122
5	Complete power estimation flow	123
5.1	High-level FPGA estimation flow: background	124
5.1.1	Summary of the previous work on power estimation flow	125
5.2	Power estimation flow	125
5.3	Experimental results	129
5.3.1	Small test DSP circuits	129
5.3.2	Word-level statistics: Large DSP designs	134
5.4	Bit-level statistics: Large DSP designs	138

5.5	Conclusions	142
5.5.1	Future work	143
6	Conclusions	145
6.1	Measurement system: conclusions	146
6.2	Interconnect power estimation: conclusions	147
6.3	Logic power estimation: conclusions	148
6.4	Complete estimation flow: conclusions	150
6.5	Future work	150
	Appendices	155
	Appendix A	155
	Bibliography	163

Abstract

Power consumption in microelectronic devices and circuits has become a critical design concern in recent years due to the rapid growth of personal wireless communications, battery-powered devices and portable digital applications. Reliability concerns and packaging costs have made power optimization relevant not only for battery-powered applications.

The largest impact on power reduction can be achieved at the system level where architecture and algorithms are to be defined. Selecting the most power efficient algorithm out of many available requires a fast and accurate way to estimate the power consumption of any implementation, so as to avoid time-consuming low-level implementations of each possible design architecture.

This work is oriented towards the high-level dynamic power estimation of DSP-oriented designs implemented in a chosen target hardware architecture. According to the different power features of logic and communication design segments, the presented power estimation methodology includes two different models. One is used for power estimation of the global routing employed for interconnections between the components. The other is used for both, local interconnect and logic, power estimation of the components. The complete methodology in this work has been applied to DSP circuits implemented in modern Field Programmable Gate Array devices (FPGAs).

High-level interconnection power estimation is a difficult task due to the extremely scarce information on global routes available at these levels of abstraction. This work proposes a power model that depends on the mutual distance of the components and their shape. There are only two input parameters to the model: the relative position of the components and the ordering of the pins on the components' boundaries. Although simple, the model takes advantage of the router properties and is capable of giving fast and highly accurate estimates for DSP-oriented designs, with the mean relative error of the interconnect power model lying within 10% of the physical measurements.

High-level logic power estimation is a more studied topic in the literature. However, it often includes exhaustive low-level simulations needed for the model characterization. This work presents the methodology for high-level logic power estimation which is based on the component's structure and the analytical computation of the switching activity produced inside the component in the presence of correlated inputs. Glitching generated inside the component is also included in the model by using a novel approach for its estimation. Compared to other proposed power estimation methods, the number of circuit simulations needed for characteriz-

ing the power model of the component is highly reduced. Another important contribution is the application of the presented methodology to heterogeneous FPGA resources, so it can be used for both configurable logic-based blocks and specialized embedded blocks. Again, the mean relative error of the component power models lies within 10% of the physical measurements.

The complete model that includes both interconnect and logic power models, has been characterized and verified by on-board power measurements, instead of using low-level estimation tools which often lack the required accuracy. The measurement system in this work is designed in order to facilitate the separation of the static power, the clock power, the power of the global interconnects and the power consumed in the logic. Consequently, it has been used for the verification of both models separately and also when they were used together in order to estimate the total dynamic power consumption of the DSP circuits.

Finally, the complete model performance has been explored over a wide range of input parameters, signal components and design positions on a chip. The accuracy of the model has also been verified for some DSP benchmarks. The results suggest that the proposed model is suitable for integration with high-level power optimization techniques, where accurate estimates are needed in the shortest possible time.

Resumen

El consumo de potencia se ha convertido en un aspecto muy relevante del diseño microelectrónica debido en gran medida al gran éxito de los sistemas portátiles, como la telefonía móvil, las redes inalámbricas y los sistemas multimedia, que han inundado los mercados de todo el mundo. En las complejas aplicaciones actuales los costes de encapsulado así como de los accesorios de refrigeración han aumentado, mientras que la duración de las baterías ha disminuido y la fiabilidad de los circuitos ha sido seriamente amenazada. Por consiguiente, el consumo se ha convertido en un aspecto más crítico incluso que el área o las prestaciones para mayoría de las aplicaciones.

Las herramientas de diseño asistido por ordenador (CAD) son necesarias para abordar el problema de la reducción del consumo debido a la alta complejidad de las aplicaciones modernas. Los cambios en la arquitectura son más fáciles en los niveles de abstracción más altos. El diseñador puede investigar un número elevado de arquitecturas diferentes en estos niveles, por lo que aumentan las posibilidades de encontrar alternativas de bajo consumo. Para evitar los tiempos de diseño excesivos que son necesarios para la implementación de cada posible arquitectura, es imprescindible desarrollar modelos de alto nivel, rápidos y precisos, para la estimación del consumo.

El objetivo de esta tesis es la construcción de una tecnología eficiente para la estimación del consumo dinámico a alto nivel y que se pueda integrar fácilmente con los algoritmos de alto nivel orientados a la optimización del consumo.

Cabe distinguir dos modelos de estimación diferentes, debido a que la capacidad de carga tiene diferentes propiedades en la lógica y las interconexiones. Así pues, un modelo se usa para la estimación del consumo de las conexiones entre los bloques lógicos y depende de la distancia entre los bloques y de su forma. El otro estima consumo en la lógica junto con las conexiones locales. Ambas metodologías se han adaptado a la arquitectura habitual de las FPGAs más modernas.

La estimación a alto nivel del consumo de las interconexiones es una tarea extremadamente complicada, debido a la escasa información disponible sobre las conexiones antes de realizar el posicionamiento de los bloques y de que el circuito se haya rutado. En este trabajo se propone una nueva metodología que tiene sólo dos parámetros de entrada: la posición relativa entre los componentes y el orden de los terminales en los bordes de componente. Aunque simple, el modelo aproxima bien el comportamiento del rutador en alto nivel y es rápido y preciso con sólo un 10% de error medio.

Las técnicas de estimación del consumo en lógica existentes para FPGAs a menudo necesitan simulaciones extensivas del circuito a nivel de transistores o de puertas, como paso previo a la síntesis de alto nivel. Este trabajo presenta una nueva metodología que se basa en la estructura del componente y en el cálculo de la actividad de las transiciones dentro del componente cuando las señales de entrada están correladas. La cantidad de glitching también se incluye en el modelo mediante una técnica novedosa. El número de simulaciones del circuito a nivel de puertas necesarias para caracterizar el modelo es muy reducido comparado con otros métodos. Además, cabe destacar una contribución importante que se basa en la adaptación del modelo a la estructura de los bloques empotrados. Por lo tanto, los modelos desarrollados para la estimación del consumo en lógica se pueden usar para los recursos heterogéneos de las FPGAs actuales. El error medio es un 10% comparado con las medidas de consumo reales. El modelo completo que se obtiene al integrar los dos modelos anteriores se ha evaluado con medidas reales, en vez de usar herramientas de bajo nivel que a menudo no tienen la precisión necesaria para la validación de modelos de consumo y requieren gran cantidad de memoria y largos tiempos de ejecución.

El sistema de medidas desarrollado en este trabajo es capaz de separar los valores de consumo estático, consumo de reloj, consumo en las conexiones y consumo en lógica. Este sistema se ha usado para la validación de los dos modelos por separado y también del modelo conjunto cuando éstos se han aplicado a la estimación del consumo de circuitos de procesado digital de señal (DSP). Finalmente, el modelo completo se ha evaluado con un número elevado de circuitos de prueba, cambiando las posiciones de los bloques y las estadísticas de las señales de entrada. Los resultados que proporciona el modelo son muy favorables y permiten su integración con técnicas de optimización del consumo a alto nivel que requieren la obtención rápida y precisa de estimaciones de consumo.

Acknowledgments

I would like to thank to Carlos Carreras, for his invaluable advices that have enriched this thesis, and for his constant encouragement throughout the research work. I doubt this thesis would have been so successful without the benefit of his knowledge and experience. I consider myself very fortunate to have had him as advisor.

I would also like to thank to all the people from the LSI group, in Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid for their help and moral support. It has been a great pleasure to share meals in cafetería with Javier, Zorana, Pablo, Pedro, Roberto, Gabriel, Marisa, Angel and Juan Antonio. Many thanks to Vule for his enormous help on hardware part of this thesis, and to Pedro and Alvaro for their generous help in the lab. Special mention deserve Enrique Calleja from the ISOM group and Octavio Nieto-Taladriz from the LSI group, Departamento de Ingeniería Electrónica.

Jelena Popović from the Electronics Department, Faculty of Electrical Engineering, University of Belgrade, had made possible my first stay in Spain at the Universidad Politécnica de Madrid, which later led to the realization of this thesis. I owe her a lot, for she had helped me find a place where I could enjoy in my research.

I thank Saša, Maja and Milena, the best friends one can have, for being there for me in difficult times and also for patiently hearing out the ideas about my investigation throughout the whole education, even though they weren't precisely the most interesting conversation topics shared over a cup of tea.

I thank my family for all the support throughout my education and my life. My mother deserves a special mention since she never stopped believing in me. Without her support, everything would have been much more difficult. This book is for her.

And I thank Jon, for knowing exactly what I am going through and not letting me doubt in myself and my work for one second. For being there for me always in all aspects of my life. I am most fortunate to be with him.

List of Figures

1.1	Virtex II Pro architecture	3
1.2	Routing resources in Xilinx Virtex devices [PHB06]	4
1.3	Leakage current components [RMMM03]	5
1.4	Charging and discharging of a load capacitance	6
1.5	Power savings and iteration times at different levels of a design flow [RJD98]	9
1.6	Thesis organization	13
2.1	Methodology used in [SKB02] and [DT05]	16
2.2	Cycle-by-cycle energy measurement	17
2.3	Measurement setup	18
2.4	Block scheme of the measurement setup	19
2.5	Methodology for effective capacitance extraction	20
2.6	Two different module positions in FPGA	21
2.7	XPower design flow	26
3.1	fGREP: net demands for 2-terminal net	28
3.2	Communication link between modules and their different types of connections	30
3.3	Slice floorplan: binary tree and rectangular representation	32
3.4	Recursive partition scheme of a design and the physical architecture	33
3.5	Power per interconnect between the modules A and B	37
3.6	Simulation setup when the interconnects between two modules are considered	38
3.7	Routing zones	39
3.8	Simulation setup when interconnects between a module and I/O pins are considered	40
3.9	Power per interconnect between the module A and the I/O pins	41
3.10	Rectilinear Steiner trees	42
3.11	Second routing zone divided equally between the modules	43
3.12	Maximum power difference for different placements	45
3.13	HLPM performance when applied to the connections between modules A and B	46
3.14	HLPM average relative error when applied to the connections between modules A and B	47

3.15	HLPM performance when unique set of coefficients is applied to the connections between the module and I/O pins	48
3.16	Errors for the HLPM when applied to the connections between the modules A and B by varying their distance in only one direction	49
3.17	Errors for the HLPM when applied to the connections between the module A and I/O pins by varying their distance in only one direction	50
3.18	Errors for the HLMP when applied to a) 3-module design, and b) 5-module design	51
3.19	Design positions for DSP_1	52
4.1	Chapter 4 organization	58
4.2	Bit transition activity vs. bit position in a word for zero-mean signals	61
4.3	Bit transition activity vs. bit position in a word for non-zero mean signals	66
4.4	Statistical and probabilistic switching activity computation	68
4.5	Full-adder cell	71
4.6	Probability methodology	71
4.7	Ripple-carry subtracter with misaligned breakpoints	74
4.8	a) Regionally decomposed array multiplier, b) Full-adder cell	75
4.9	Row-adder tree multiplier	76
4.10	Row-adder tree multiplier implemented in FPGAs	77
4.11	Modified Booth Algorithm	78
4.12	Ripple-carry adder	80
4.13	Module decomposition in function of MSB-LSB breakpoint of the longer operand	80
4.14	Glitch generation in a logic circuit	81
4.15	Repetitive delays at the entrance of the basic cells	82
4.16	The distribution of glitching generated inside the array multiplier	84
4.17	Power macromodelling	86
4.18	Errors for two signal models: ARMA and Dual-bit type, when considering zero-mean gaussian signals and various sizes of the MSB activity region. The arrow indicates how errors behave when the number of sign bits increases	98
4.19	Errors for three signal models for gaussian signals with mean equal to 10	99
4.20	Errors for three signal models for gaussian signals with mean equal to 125	100
4.21	Error performance of multipliers for various autocorrelation coefficients	103
4.22	Error performance of adders for various autocorrelation coefficients	104
4.23	Error performance for multipliers with operands of different sizes	105
4.24	Error performance for adders with operands of different sizes	106
4.25	Error performance for multipliers and adders with operands of the same size	107
4.26	Error performance for multipliers - a), b), c) and adders - d), e) with different size operands for various signal statistics	108
4.27	Error performance for multipliers and adders for various signal statistics considering two approaches: the model presented here and the model presented in [JCC07b]	109
4.28	Errors with respect to measurements of the logic model for a) multipliers, b) adders	114
4.29	Errors for a) multiplier, and b) adder logic power given by XPower	115
4.30	Measured and estimated power for various component sizes	120
5.1	Power estimation flow	127

5.2	Estimated vs. measured power values for DSP circuits and individual components	131
5.3	Power distribution according to the measurements and XPower for DSP_1 design in a) position 1, and b) position 2	132
5.4	Error distribution for HLM considering total dynamic power, and its components: logic and interconnect power	133
5.5	Error distribution for XPower considering total dynamic power, and its components: logic and interconnect power	134
5.6	SYSTEM block schematic	135
5.7	Transition activity vs. bit position in the signal word at the output of the multiplier	138
5.8	Estimated and measured power values for <i>SYSTEM</i> designs	141
1	Measurement system	156
2	a) Measurement setup b) Buck-switching PWM regulator	157
3	XDL file syntax: part I	158
4	XDL file syntax: part II	158
5	MARWEL structure	160

List of Tables

2.1	Effective capacitances for different wire types.	24
2.2	Error for the interconnect power computed with the effective capacitance values.	25
3.1	Summary of features of the interconnection models.	35
3.2	Coefficient values for connections between modules, k^{mod}	46
3.3	Coefficient values for connections between modules and I/O pins, k^{io}	48
3.4	Relative errors for the proposed model (HLIM) and XPower (XP), for different autocorrelation coefficients.	53
4.1	Probabilistic method for AND-gate	69
4.2	Capacitance coefficients for a ripple-carry subtracter [LR95]	74
4.3	Summary of features of the logic power models.	92
4.4	Summary of features of the embedded power models.	93
4.5	Comparison of two models for multipliers.	111
4.6	Comparison of two models for adders	111
4.7	Comparison of the two models when component size differs from the input signal bit-width	111
4.8	Computation times for the two models applied to 16×16 multiplier	112
4.9	Summary of features for both models when applied to real-world applications.	113
4.10	Computational times for logic power models.	115
4.11	Autocorrelation coefficients for real-world data.	116
4.12	Power values and relative errors for the proposed high-level model (HLLM) and the low-level tool XPower (XPwr).	117
4.13	Computational times in seconds for embedded power models.	118
5.1	Summary of features of the high-level power estimation.	126
5.2	Relative errors for the proposed model (HLM) and XPower (XP), for different autocorrelation coefficients.	130
5.3	The number and size of arithmetic operators in DSP_2 and DSP_3 designs	135
5.4	The number of different arithmetic module types in the three configurations of <i>SYSTEM</i>	136

- 5.5 Relative errors for the proposed model (HLM) and XPower (XP), for different autocorrelation coefficients. 136
- 5.6 Relative HLM errors for the internal modules of *SYSTEM_{MIX}* 137
- 5.7 Computational times in seconds for bit- and word-level power models. 140
- 5.8 HLM errors for the three *SYSTEM* configurations when bit-level statistics are used 141
- 5.9 Relative HLM errors when applied to each component module in *SYSTEM_{MIX}* separately for both bit-level and word-level statistics 142
- 5.10 HLM errors for the first *DSP₂* design when using bit-level instead of word-level statistics 142

List of Acronyms

ARMA	Autoregressive Moving Average.
ASIC	Application Specific Integrated Circuit.
CAD	Computer-Aided Design.
CLB	Configurable Logic Block.
CMOS	Complementary Metal Oxide Semiconductor.
DBT	Dual-bit Type.
DFG	Data Flow Graph.
DSP	Digital Signal Processing.
FDCT	Fast Discrete Cosine Transform.
FFT	Fast Fourier Transform.
FPGA	Field Programmable Gate Array.
GTL	Graph Template Library.
HLIM	High-Level Interconnect Model.
HLLM	High-Level Logic Model.
HLM	High-Level Model.
HLMM	High-Level Multi-point Model.
HLPM	High-Level Point-to-point Model.
HLS	High-Level Synthesis.
HW	Hardware.
IIR	Infinite Impulse Response.
IO	Input/Output.
IP	Intellectual Property.
LMS	Least Mean Squares.
LSB	Least Significant Bit.
LUT	Look-Up Table.
MARWEL	Measurement of ARchitectural WirE Length.
MSB	Most Significant Bit.
MOSFET	Metal Oxide Semiconductor Field Effect Transistor.
MUX	Multiplexer.
NP	Non-Polynomial.
NMOS	Negative-channel Metal Oxide Semiconductor.
PDF	Probability Density Function.

PMOS	Positive-channel Metal Oxide Semiconductor.
PWM	Pulse-Width Modulation.
RAM	Random Access Memory.
RISC	Reduced Instruction Set Computer.
RTL	Register Transfer Logic.
RST	Rectilinear Steiner Tree.
SRAM	Static Random Access Memory.
SA	Simulated Annealing.
SW	Software.
VLSI	Very Large Scale Integration.
VPR	Versatile Place and Route.
WDM	Wire Distribution Map.

List of Notations

C_l	Load capacitance.
f	Frequency.
V_{dd}	Power supply.
α, sw	Switching activity.
P	Power.
G	Glitching.
$ x $	Absolute value of x .
$det(M)$	Determinant of matrix M .
σ	Variance.
ρ	Autocorrelation coefficient.
μ	Mean.
$E\{\}$	Mathematical expectation.
$[x]$	Round to the integer closest to x .
$A \oplus B$	Logic XOR of A and B .
$\lceil x \rceil$	Round to the closest integer greater than x .
H_d	Hamming distance.
S_d	Signal distance.
V_E	Set of input vectors for embedded block.

CHAPTER 1

Introduction

The current microelectronics design trend focuses on decreasing transistor size in order to increase circuit speed and chip capacity. However, higher circuit speed produces faster charging and discharging of design load capacitors, resulting in increased design power. As the chip density grows larger, it also allows for the implementation of more complex and thus, more power hungry electronics applications. As a consequence, the packaging and cooling costs have increased, the battery life has decreased and the circuit reliability has been seriously endangered. Power consumption has become a more critical design concern than area and performance for many applications. Furthermore, in order to achieve ultra high performance, as for example in massive computing systems, power reduction is obligatory. Otherwise, cooling requires more space and such increased distance limits performance.

In this chapter, basic definitions of power consumption, optimization and estimation are presented in section 1.1, together with the motivations of this thesis. Next, the objectives of the research are enumerated in section 1.2. Section 1.3 presents the organization of the thesis. Finally, section 1.4 lists the publications in international journals and conferences that resulted from this thesis.

1.1. Motivation

In the last decade, communications systems have experienced rapid development, which goes beyond the expectations made in the 80's and 90's. Apart from the appearance of Internet, many other applications like mobile telephones, wireless and multimedia systems have become the main stream of the world market. This work focuses on Digital Signal Processing (DSP) circuits as they are the essential components in these communication systems.

The high processing rates of DSP circuits have become possible by the use of application specific hardware (ASICs) instead of microprocessor-based implementations. Besides, programmable devices like Complex Programmable Logic Devices (CPLDs) and Field-Programmable

Gate Arrays (FPGAs) have emerged as a new type of semiconductor devices that can be classified to be in between these two categories. FPGAs allow the implementation of dedicated architectures that can be reconfigured. Parallelism can be fully exploited to allow multiple implementations of the same design, thus increasing the throughput significantly. Also, they can be configured at the transistor level for special-purpose designs. As a result, their computing performance, area and power are more optimized than in microprocessors. However, as the FPGA architecture is not specialized for only one single function, ASICs have a clear advantage regarding these constraints [KR06].

Still, FPGAs have become an attractive solution for various embedded designs and designs susceptible to changes due to their ability for reconfiguration and significantly lower cost compared to ASICs. They have gained special relevance in past few years, not only as prototyping platforms, but also as final product implementations. For these reasons, they have been chosen as a target technology for this study.

1.1.1. FPGA structure

An FPGA is a two dimensional array of logic blocks and flip-flops with electrically programmable interconnections between logic blocks. Unlike ASICs that can perform only one single function during their lifetime, FPGAs can be reprogrammed for a different functionality in a matter of milliseconds [WBG⁺06]. However, this flexibility in using FPGAs comes at the expense of the increased complexity in their structure. In recent years, new types of resources have been added to FPGAs making them capable of implementing many high-performance computing applications.

There are two basic types of FPGAs depending on the technology process used for their production. The first type is the so-called SRAM-based FPGA where the programmability is achieved through specialized volatile SRAM cells that activate or deactivate the programmable logic. The major SRAM-based FPGA vendors are Xilinx [Xil], Altera [Alt], Atmel [Atm] and Lattice Semiconductor Corporation [Cor]. The second type of FPGAs are antifuse and flash-based FPGAs, where the FPGAs are programmed by creating permanent conductive electrical paths as in a case of anti-fuse, or where the programming is achieved through non-volatile flash cells that activate or deactivate the programmable logic. The vendors of these chips are Actel [Act] and QuickLogic corporation [Qui].

However, the leading global producers of programmable logic are Xilinx and Altera, who hold more than 80 % of the programmable market. Consequently, we chose Xilinx devices as a target platform.

The first important implementations of double-precision floating point applications arrived in 2002 with Xilinx Virtex II Pro devices which contain up to one hundred thousand logic cells, embedded 18-bit by 18-bit multipliers and embedded PowerPC RISC processor blocks. This

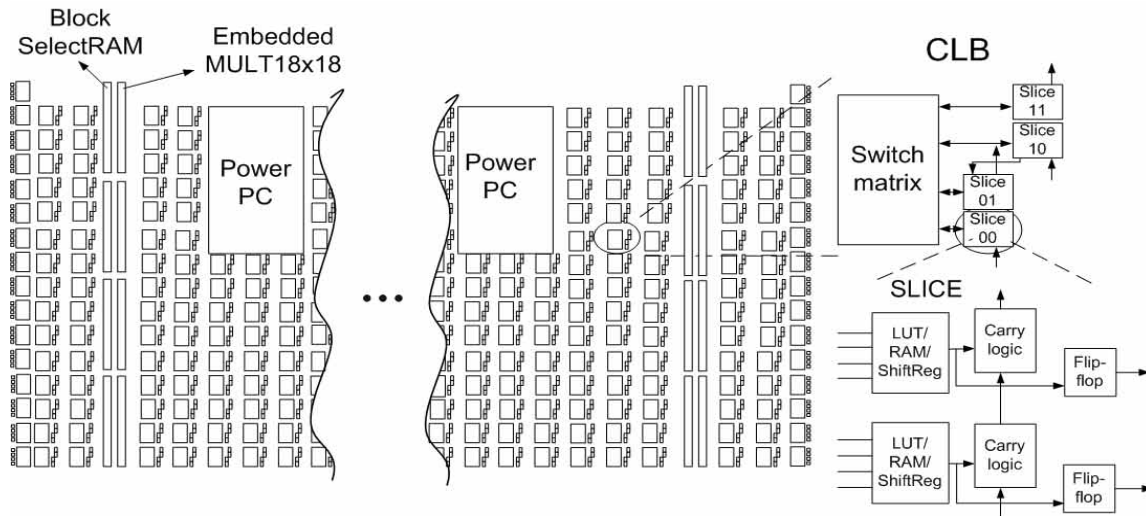


Figure 1.1: Virtex II Pro architecture

is the reason that these particular devices have been chosen as a target platform in this work. Virtex II architecture is identical to Virtex II Pro architecture (except that they do not contain the PowerPC processor blocks) and therefore, the methodology presented in this work can be also applied to these devices. The Virtex 4, released in 2004, was built considerably on Virtex II Pro architecture with up to two hundred thousand logic cells, and embedded DSP blocks capable of implementing multiply-and-accumulate (MAC) operations. Unfortunately, the multiplier size (18×18) is not well suited for the requirements of floating point IP cores. Consequently, Virtex 5 devices were released in 2008 and they contain a larger number of 25-bit by 18-bit embedded multipliers.

Although the architecture considered in this work is the Virtex II Pro architecture, the methodology presented here can be easily extended in order to consider the most recently released high-speed FPGA devices, as their structure is built upon the Virtex II Pro device architecture.

Virtex II Pro architecture

Virtex II Pro architecture is depicted in Fig. 1.1. The programmable fabric of an FPGA consists of an array of configurable logic blocks (CLBs). Each CLB contains four slices and is tied to a switch matrix in order to access global routing resources. The slices are split in two columns of two slices each with two independent carry logic chains and one common shift chain. Each slice has two look-up tables (LUTs). A LUT serves for generating any logic function that can have up to four input bits. They can be also configured as shift registers or distributed SelectRAM memory. Beside LUTs, the slices contain flip-flops, logic gates and multiplexers.

The Block SelectRAM resources are 18Kb of Dual-port RAM, programmable from $16K \times$

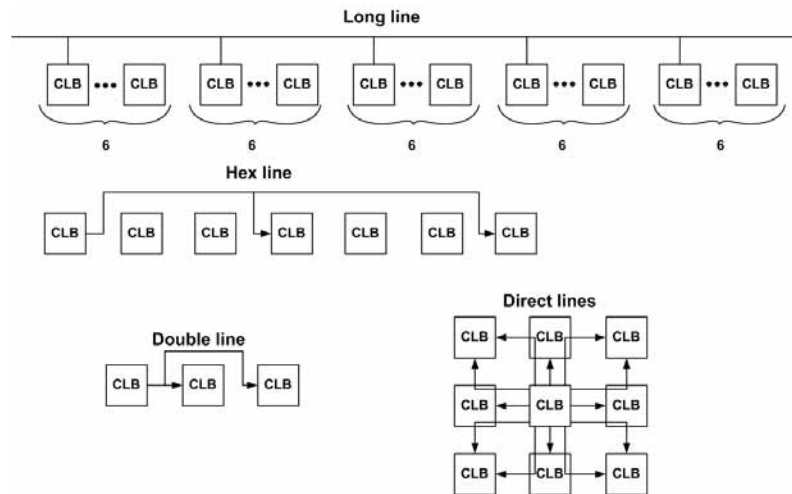


Figure 1.2: Routing resources in Xilinx Virtex devices [PHB06]

1 bit to 512×36 bits, in various depth and width configurations. The multiplier block is a dedicated 18×18 bit multiplier, and optimized for MAC and DSP filter applications. Both, the SelectRAM memory and the embedded multiplier, are connected to four switch matrices in order to access the general routing resources.

PowerPC blocks are IBM RISC processors whose number varies from 0 to 2. They will not be considered in this work, as their structure corresponds to that of the microprocessors, and the estimation of microprocessor's power consumption is a well-researched topic [AR05, ALV03, ALV04, AAR⁺07, dHAJW⁺07].

All the IOs, CLBs, SelectRAMs and embedded multipliers use the same interconnect scheme and the same access to the global routing matrix. Interconnections are buffered and relatively unaffected by the fanout. Various global hierarchical routing resources can be identified based on long, hex, double and direct wires (see Fig. 1.2) [Xil]. The long wires span the full height and width of the device. The hex wires route signals to every third or sixth CLB in all directions. The double wires span up to two blocks in each direction and the single wires route signals to neighbouring blocks. Clock signals are routed via special-purpose dedicated routing networks.

1.1.2. Power consumption

As FPGAs are aimed for implementation of many different designs, a large number of routing switches is used in order to obtain flexible interconnections, whereas LUTs are used for logic, as they are capable of implementing any given logic function for the corresponding number of inputs. However, this type of chip architecture prevents an optimal implementation of a design because it utilizes an excessive number of additional transistors and routing resources, which in turn, contribute to a significant increase in the power consumption of the design. Consequently, FPGA power consumption has become a major concern along with the traditional objectives of

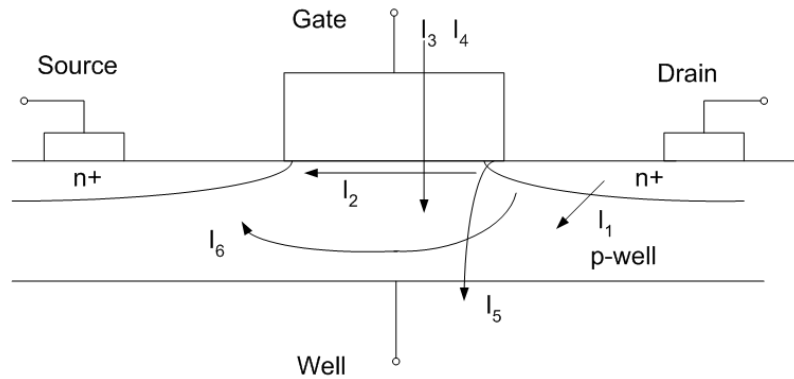


Figure 1.3: Leakage current components [RMMM03]

circuit performance and area efficiency.

Power consumption in FPGAs has two main components: static and dynamic power. Static power occurs due to the leakage current, while dynamic power results from switching of the capacitive load voltage between different voltage levels. Since FPGAs are made in CMOS technology, the power components will be explained next considering circuits built in this technology.

Static Power

In order to achieve higher density and performance and lower power consumption, CMOS devices have been continuously scaled down for more than 30 years [RMMM03]. However, when the transistor technology has an order of magnitude around or below 100 nanometers, the advantages gained by the size shrink disappear due to the quantum effects which can not be neglected. These effects produce an excessive increment in the leakage current.

Leakage current consists of two main components (see Fig. 1.3): subthreshold leakage current (I_2) and gate leakage current (I_3). There are some other leakage current components that have started to gain interest recently due to an excessive scaling of the transistor dimensions. They occur due to the shorter channel-length: injection of hot carriers from substrate to gate oxide (I_4) and punchthrough leakage (I_6), due to the thinner oxide thickness: gate-induced drain leakage (I_5), and, due to high doping concentrations: junction reverse-biased current (I_1) [RMMM03].

However, the largest amount of static power is still owed to a subthreshold current. It is the most temperature-dependent leakage component [AMK⁺05, RMMM03], and thus, every increase in dynamic power, produces an increment of the chip temperature, which in turn, increases the leakage current. This leakage component is also one of the main reasons why the scaling process is facing difficulties as it is explained next.

Decreasing transistor sizes enable higher densities of transistors on a chip. In order to con-

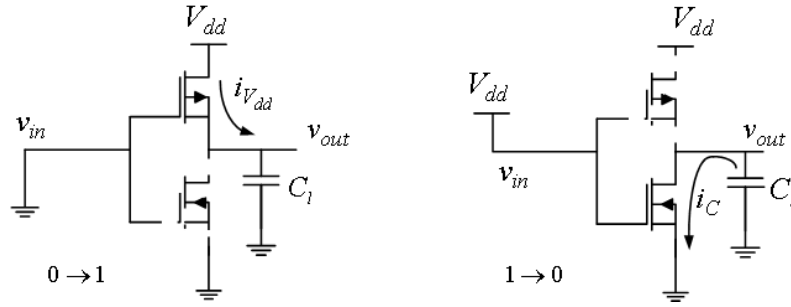


Figure 1.4: Charging and discharging of a load capacitance

control the power of the circuit, the power supply voltage is also reduced with each transistor scaling. For CMOS circuits, a lower supply voltage means lower performance [GGH97]. This problem is solved by reducing the threshold voltage (V_{th}) of a transistor. V_{th} is defined as a gate-source voltage of a MOSFET transistor, above which, the transistor is turned on. Ideally, if the gate voltage is below the threshold voltage, the transistor is not conducting any current. However, in practice there is still some current flowing from the drain to the source of a transistor. This is the subthreshold current. Its most important feature is that it increases exponentially with any V_{th} decrease, thus, limiting significantly the scaling process.

Consequently, the leakage current has become an important source of power dissipation. Therefore, as the CMOS technology process becomes smaller, a trade-off has to be made between the static power and operating circuit speed [AMK⁺05, RP00].

FPGAs have similar leakage problems as other CMOS-based circuits, but their structure and functionality require different solutions. A leakage power analysis in 90-nm FPGAs is given in [TL03], while the leakage power optimization in FPGAs has been addressed in [LLHC04] through dual- V_{dd} techniques, in [GTV⁺04] by creating coarse-grained "sleep regions", in [AN06] through the selection of input signal polarities entering LUTs and leakage-aware routing etc.

Dynamic Power

Dynamic power occurs due to the charging of the load capacitance when the transistors change their state from logic '0' to logic '1'. A higher frequency leads to more signal transitions, which in turn, increase the circuit power dissipation. This work focuses on the dynamic power in FPGAs.

For example, consider the inverter circuit in Fig. 1.4. The transistors marked with dot-line are turned off during the corresponding transition. Every time there is a transition from logic '1' to logic '0' at the inputs of the inverter, the load capacitance C_l , is charged from 0 to V_{dd} . The energy needed for capacitance charging is provided by the power supply V_{dd} . Additionally, part of the energy provided by the power supply is dissipated on the PMOS transistor. During the input transition from '0' to '1', C_l is discharged through the NMOS transistor, and the

accumulated energy in C_l is dissipated by this transistor.

While the energy dissipated by the transistors is important for locating the so-called "hot-spots" of the design, the power consumed by the design strictly refers to the power provided by the power supply. Thus, power is consumed only during the $0 \rightarrow 1$ transitions at the output.

The power consumed while charging the load capacitance during the time interval T is computed as (details can be found in [NM97]):

$$P = C_l \cdot V_{dd}^2 \cdot \frac{n_t(T)}{T} \quad (1.1)$$

where $n_t(T)$ is the number of $0 \rightarrow 1$ transitions during a time period T . Therefore, in synchronous circuits, the average power consumption of a single gate is computed as:

$$P = \alpha \cdot C_l \cdot V_{dd}^2 \cdot f \quad (1.2)$$

where α is the average number of $0 \rightarrow 1$ transitions in one clock-cycle, and f is the clock frequency. As the capacitance has to be discharged in order to be charged again, the number of $0 \rightarrow 1$ transitions is equal to the number of $1 \rightarrow 0$ transitions, and the parameter α is often replaced with the expression $0.5 \cdot sw$, where sw is the average number of all transitions during one clock-cycle (referred to as switching activity). The total dynamic power of a design is obtained as the sum of power consumptions over all gates in the design.

The value of the power supply is usually fixed and constant, and the clock frequency is defined for each specified design. Therefore, two parameters remain unknown for power estimation: load capacitance and switching activity. There are many different ways to address the problem of finding these two parameters, and they will be discussed in detail in the next chapter.

According to the features of these two parameters, dynamic power can be further divided into three components: the power of the clock circuitry (with a fixed switching activity equal to two), the logic power consumed in the functional units and memories (where the load capacitances correspond to the loads driven by the outputs of the logic gates), and the power of the interconnects that are used between the units (where the load capacitance depends on the type and length of a wire used for the connection). While each of these power components contributes to the power of the design, when an FPGA architecture is considered, the interconnect power represents the dominant portion of the total power [SKB02,DT05] due to relatively large capacitive loadings of the programmable switch matrices. It is followed by the power of the clock circuitry, while the logic power normally has a lower impact on the total power. However, in data-dominated systems, such as DSP circuits, the logic power component can not be neglected. Therefore, in this work, power estimation of both, logic and interconnects, will be addressed.

A third source of power dissipation in CMOS circuits can be identified: the short-circuit power which occurs when both transistors in an inverter are turned on at the same time while the input switches. It can be considered to be a part of the dynamic power as it depends on the switching activity of the gate inputs. However, this component is much smaller than the first two [NS00], and with the scaling of the transistors it becomes even less significant.

1.1.3. Power optimization

In order to cope with the complexity of modern applications, Computer-Aided Design (CAD) tools must be used to approach the power reduction problem. At first, these tools began to tackle the power problem at the gate-level of abstraction. However, by the time the design has been specified at the gate level, it is often too late or too expensive to go back to the architectural level to fix the serious power problems. Architectural changes are much easier at higher levels of abstraction. A designer can investigate a higher number of different design architectures at these levels, so the possibilities of finding candidates with low power consumption are extended. Fig. 1.5 shows the percentage of power that can be saved during each of the phases of the design flow in VLSI circuits [RJD98, Kri04]. It also shows the iteration times required for power optimization at each stage of the design flow. As can be seen, the largest savings (between 10 and 20 times) are achieved through the new system and architecture implementations. These iterations have the shortest duration, thus enabling the evaluation of a large number of low-power candidate architectures in a short time. The influence on power without significant sacrifice on the performance becomes harder as the design goes through stages of realization. Power savings around only 10 - 20 % are achieved at the lowest levels of abstraction where the architecture has already been defined. Therefore, it is critical to address the power issue at the earliest stages of the design flow.

The input to the system-level phase of a design flow is a set of abstract communicating processes or tasks, with no knowledge of whether the tasks are implemented in hardware or compiled into software. In this phase, the most efficient algorithm in terms of area, power and performance is chosen depending on the values of system-level estimates, different architectures are explored in order to find the best one for the given power, area and time constraints, and hardware-software partitioning is performed. The result of this phase is a behavioral or algorithmic description of the design mapped on to HW and SW components, where the detailed cycle-by-cycle behaviour of the design and its structure are not yet defined.

High-level synthesis converts the functional description (in the behavioral domain) into a structural RTL implementation (register-transfer level) that consists of macromodules and random logic together with their connections to the other modules. This phase consists of scheduling, binding, allocation and resource sharing tasks. The clock boundaries of a design are also determined during this phase. Power optimizations at this level deal with the design of the

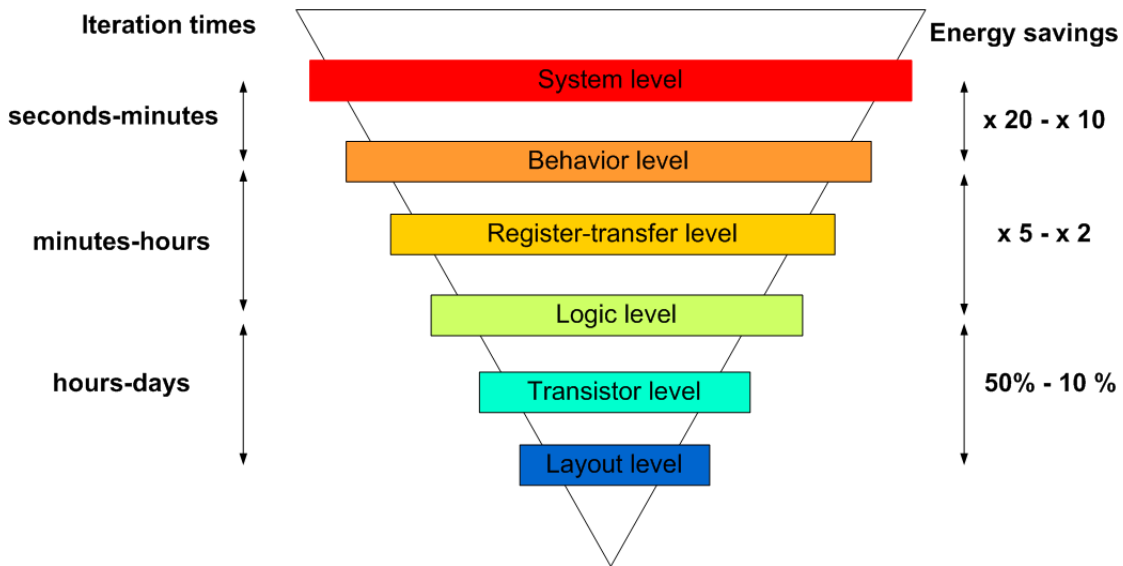


Figure 1.5: Power savings and iteration times at different levels of a design flow [RJD98]

circuit architecture.

Next, logic synthesis maps a technology-independent representation of a combinational and sequential logic (RTL) into a semi-custom technology library (logic level). As a result of the logic design phase, a netlist is created that describes the required logic gates and their connections. Power optimization techniques at these levels include clock-gating, glitch reduction, optimization of the clock tree connections, etc.

Finally, at the layout phase, the mapped circuit is placed and routed, and the technology and the size of the transistors are chosen in order to reduce the leakage current, and to improve the performance of the design.

FPGA design flow

ASICs are designed from the behavioral description to the physical layout. Unlike ASICs, FPGA design ends with a bitstream used to configure the device. This means that there is no physical layout design (the layout phase ends with routing).

When FPGAs are considered, the energy savings in the logic and layout phase (i.e. obtained by power-aware technology mapping, clustering, placement and routing) have been reported in [LW03]. The majority of the savings are achieved at the logic phase through mapping and clustering. In addition, the savings were cumulative and, when applied concurrently, an energy reduction of around 20% was achieved. In [WFDA06], a methodology for power minimization through placement and routing constraints has been applied to FPGAs and an average power reduction of 10% was achieved. In [AN02], a power aware technology mapping has been implemented, achieving also around 10% of the power reduction, while in [Ale97], power improvements of 26% have been achieved by rearranging the input signals of LUTs in order to

minimize the total switching activity. Consequently, FPGAs follow the same trend as shown in Fig. 1.5 regarding the influence of power optimization techniques on the design power at the implementation levels of the design flow.

High-level optimization techniques

Since the largest savings are achieved at the highest levels of abstraction, a number of Register-Transfer Level (RTL) and high-level optimization methodologies have been proposed for both ASICs and FPGAs. They tackle the power optimization problem through RTL glitch minimization [RJD98, RDJ99], high-level synthesis combined with: floorplanning [Neb04, SHS⁺03a, SHS⁺03b, BRS00], dual voltage supplies [LHW97, CP97, CCLH04, CCX05], multiplexer optimization [CCF03, CX08], interconnect power minimization [ZJ05, SK08], partially guarded computation [CJC00], loop folding [KC97], management of the transient power such as cycle-by-cycle peak power and peak power differential [RRRL01, MRC03] etc., or through the dynamic power management at the system level [BBPDM99] by selectively turning off the system components when they are idle. In all cases, the key elements for design space exploration is the availability of accurate high-level estimation models. These models need to be integrated into the high-level optimization techniques, as to avoid time-consuming low-level implementations of each possible design architecture.

The main goal of this project is the construction of an efficient technology for high-level power estimation that can be successfully used in high-level algorithms aimed at power optimization.

1.1.4. Power estimation

Existing high-level power estimation techniques for FPGAs aim to represent power consumption in the form of an equation. Variable parameters in the equation depend on the various factors, from the simple ones, such as input and output signal statistics, operand word-lengths, circuit fanout, etc., to the more complex ones, such as the component structure, signal statistics propagated through the component, etc. The coefficients standing by the variables are obtained through extensive transistor or gate level circuit simulations as a step prior to high-level synthesis. As it is not possible to cover all the possibilities for these variables in a reasonable time, a solution is sought in numerical methods, thus often resulting in not so accurate estimates. Another critical parameter is the word-length of the operands. Since word-length optimization of DSP algorithms has proven to provide significant cost savings ([CCC⁺06], [KS01]), it is very important to have fast power estimates for components of any operand word-length in order to see if the power constraints are met during architectural synthesis. However, as the number of combinations for input word-lengths is extremely high, a new set of simulations for module

characterization is necessary every time the module parameters change.

Furthermore, the accuracy of the high-level estimation models depends on the accuracy of the low level simulation tools used for their characterization. There are a few such tools designed for commercial FPGAs, and the most widely used are XPower from Xilinx [Xil] and PowerPlay from Altera [Alt]. These tools provide a detailed power breakdown of a design based on the resource capacitance, resource utilization and data switching activity. Nevertheless, large errors are detected when the estimates obtained from these tools have been compared to physical measurements [Alt05, ESJ06, LLCC05]. Additional problems are encountered when complex designs with many signals are to be modelled, as these tools require large amounts of memory and long execution times. As a result, it is preferred that the power estimation models are characterized through on-board measurements.

1.2. Objectives

The main objectives of this PhD work are the development of fast and flexible RTL and high-level power estimation models for DSP circuits implemented in FPGAs and the verification of the proposed models through on-board measurements.

It is clear that the switching activity computation is necessary for estimating logic power. Still, it is also essential for estimating interconnect power, as the activities of the lines that connect the modules are directly related to the switching activities of the outputs of the modules. As a result, the switching activity can be considered to be the power term that, once computed, can be used together with a proper signal model for the power estimation of both, logic and interconnects. However, the load capacitance is modelled differently in these two power groups: it depends on the logic function of a gate in the logic power, and on the types and lengths of wires in the interconnect power. Therefore, two different high-level estimation models have to be developed, one for each of these two groups. Consequently, the objectives of this thesis project can be summarized as follows:

- The analysis of the most-suitable signal model for DSP applications that enables fast and easy switching activity computation
- The development of a methodology for switching activity computation
- The development of RTL and high-level power estimation models for logic in DSP circuits implemented in FPGAs, which includes:
 - a) The modelling of the load capacitance for DSP modules implemented in FPGAs at high level of abstraction (while the design is still at the algorithm level).
 - b) The development of a methodology for high-level logic power estimation of DSP circuits implemented in standard FPGA fabric based on the results of the previous objectives.

- c) The development of a high-level power estimation model for DSP blocks embedded in FPGA architectures.
- The development of RTL and high-level power estimation models for interconnections in DSP circuits implemented in FPGAs, which includes:
 - a) The analysis of the FPGA interconnect power and the study of the FPGA routing properties, wire-lengths and on-chip wire-types.
 - b) The development of a high-level interconnect model that is suitable for the integration with high-level optimization techniques (i.e. based on floorplanning algorithms).
- The verification of the proposed models through on-board measurements which includes:
 - a) The development of a measurement system that provides accurate power values for each power group separately.
 - b) The validation and characterization of each of the developed power estimation models for various signal statistics and input word sizes.

1.3. Book organization

This thesis tackles many different areas, such as signal model representation for DSP circuits, switching activity computation, logic power estimation, power estimation in interconnections, measurement methodology for model evaluation, etc. For this reason, the state of the art for each of these areas will be analyzed separately at the beginning of the corresponding chapters which are organized as follows (see Fig. 1.6).

Chapter 2 gives a description of the measurement technology that will be used in the rest of the chapters for model characterization and evaluation. It focuses on a methodology for wire capacitance extraction, since this information is essential for determining measured interconnect and logic power components. A brief overview of XPower, one of the low-level commercial tools for power estimation provided by the chip vendors, is also given in this chapter, as it will be used later for comparison with some other estimation models found in the literature.

Chapter 3 describes a novel approach for power estimation of global interconnects. The model takes advantage of the router properties to minimize the delay in the circuit. A characterization procedure for the power estimation model is described by using some circuits designed for this purpose. It is followed by the evaluation of the model performance for various different DSP test circuits. Additionally, an analysis of routing power in FPGAs is given in detail.

In chapter 4, logic power estimation is presented in detail. First, the signal model used for the purposes of switching activity computation is presented. The model was adapted to describe the characteristics of DSP signals with a small number of high-level parameters. Next,

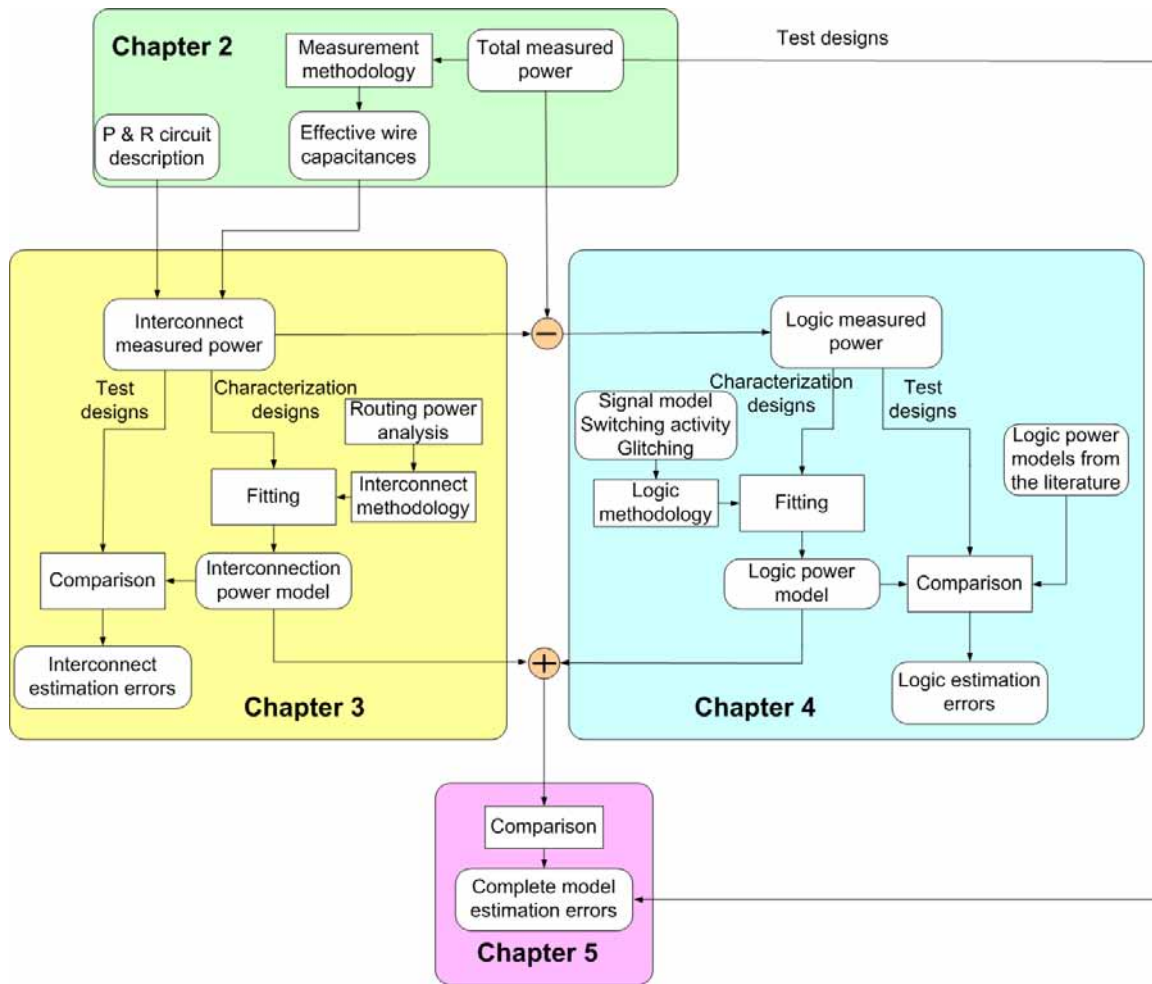


Figure 1.6: Thesis organization

the methodology for switching activity propagation throughout the component and a novel approach for estimating the amount of glitching generated inside the component are described. The methodology is adapted for various DSP component structures that correspond to the implementation structures of IP cores in Xilinx devices. The chapter ends with the experimental results that present error performance of the logic estimation model for different arithmetic component types, compared against both XPower and physical on-board measurements. Also, a comparison with some other logic estimation models found in the literature is included.

Chapter 5 focuses most of all on the analysis of the accuracy when both models from chapters 4 and 3 are joined in a complete model for power estimation. The error performance for the final model is analyzed for a wide range of input parameters and different DSP benchmarks.

Finally, chapter 6 contains the conclusions of this work.

1.4. Publications

This thesis' contributions along with other related research performed, have been published in the following international conferences and journals:

- R. Jevtic, C. Carreras and G. Caffarena, "High-level Switching Activity Models for Multipliers in FPGAs", *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, pp. 224-225, February 2007.
- R. Jevtic and G. Carreras, C. Caffarena, "Switching Activity Models for Power Estimation in FPGA Multipliers", *Int. Workshop on Applied Reconfigurable Computing*, LNCS, vol. 4419, pp. 201-213, March 2007.
- R. Jevtic, C. Carreras and G. Caffarena, "Fast and Accurate Power Estimation of FPGA DSP Components Based on High-level Switching Activity Models", *Int. Journal of Electronics*, vol. 95, no. 7, pp. 653-668, July 2008., Impact factor: 0.567
- R. Jevtic and C. Carreras, "Analytical High-level Power Model for LUT-based Components", *PATMOS'08*, LNCS, vol. 5349, pp. 369-378, Sept. 2008.
- R. Jevtic, C. Carreras and D. Helms, "A Comparison of Approaches for High-level Power Estimation of LUT-based DSP Components", *ReConFig'08*, pp. 361-366, December 2008.
- R. Jevtic and C. Carreras, "Power Estimation of Embedded Multipliers in FPGAs", *IEEE Trans. on VLSI*, to be published, 2009., Impact factor: 1.373
- R. Jevtic, C. Carreras and V. Pejovic, "Floorplan-based FPGA Interconnect Power Estimation in DSP circuits", *SLIP'09*, July, 2009.

The research work was partially developed through the participation in the following projects:

- AMEBA 2 (National Research & Development Plan)
- AMEBA 3 (National Research & Development Plan)
- DOVRES (Airbus)
- SINTEI (Universidad Politécnica de Madrid)

CHAPTER 2

Model characterization and verification

The accuracy of the high-level estimation models is highly dependent on the accuracy of the method used for their characterization. Low-level simulation tools need to make some approximations when they try to model the simulated system and the target architecture in order to obtain the estimates in reasonable times. Hence, the highest precision is achieved by using physical on-board measurements.

Due to different load capacitance characteristics in interconnections and logic, their power estimation is achieved through different models. Consequently, the models need to be characterized separately, so separate logic and interconnect measured power values are required.

This seems to be a problem when the power of any chip (ASIC, FPGA) is measured, as the chips are encapsulated and we can not access the power supply of the elements of interest. The only way is to measure the total power of the chip and then find a way to separate the different power components by carefully designing the circuits to be measured and post-processing the results afterwards.

In this chapter, we focus on the description of the measurement technology that will be used in the rest of the chapters for model characterization and evaluation. First we give a brief overview of the measurement methodologies that have been used so far for FPGAs. Next, we present a methodology that includes a measurement setup, a tool developed for gathering the information about the placed-and-routed design, and the extraction of wire capacitances. Finally, at the end of this chapter, we give a description of the XPower tool.

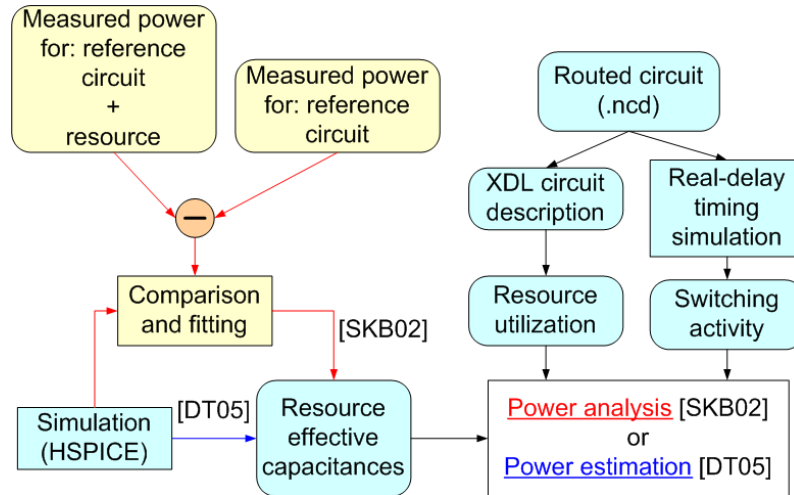


Figure 2.1: Methodology used in [SKB02] and [DT05]

2.1. FPGA Power estimation based on on-board measurements: background

Measurements on-board have been used for many different purposes, from the analysis of power distribution over different elements such as logic, clock circuitry and interconnects [SKB02, DT05], to the influence of the design architecture features on power [WAL04,LLCC05], and the characterization and verification of the power estimation models [EJH⁺04,ESJ06,dHAJW⁺07].

An analysis of dynamic power consumption in Virtex II is carried out in [SKB02], by extracting the effective capacitance of all resources through simulation and measurement, and obtaining the design resource utilization and switching activity after place and route (see Fig. 2.1). Since the resource utilization varies with the design, they use a large number of real circuits in order to obtain statistically valid results. The aim is to better understand where power is consumed in FPGAs. This analysis can identify the cost of using each resource, expressed through the value of its effective capacitance. Still, the results for power dissipation distribution can not be generalized to any design, as the utilization of resources and the switching activity strongly depend on each particular design and input data set, respectively.

The work in [DT05] presents a methodology for pre-silicon dynamic power estimation of FPGA-based designs. Similarly to [SKB02], the effective capacitance of all resources is extracted through transistor-level simulation, and the design resource utilization and switching activity are obtained from the simulation of a placed-and-routed design (see Fig. 2.1). A large number of benchmarks is used and the estimation model is validated against physical measurements. However, the design has to be placed-and-routed in order to obtain power estimates, which increases the design time significantly.

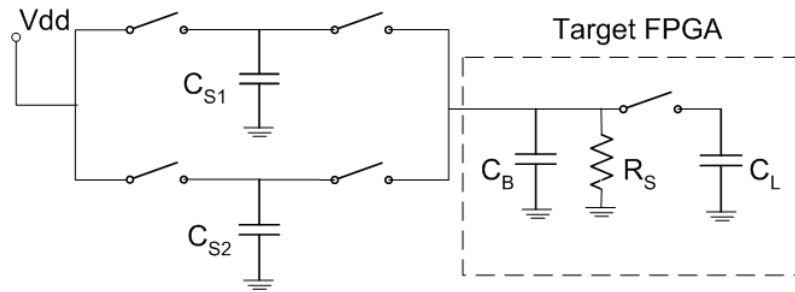


Figure 2.2: Cycle-by-cycle energy measurement

The work in [BHU03] measures the power consumed in Virtex FPGA chips, but focuses most of all on the power consumed during dynamic reconfiguration.

A high-level power estimation model of Xilinx FPGA embedded memories and hardware functional blocks has been presented in [EJH⁺04,ESJ06]. The model uses a set of high level parameters, divided into architectural and algorithmic parameters, where the coefficients standing by the parameters are obtained through curve fitting over power values gathered from measurements. For this purpose, the measurement system contains two FPGA chips, one where the characterization designs are implemented and the second one which provides simulation vectors to the first one, similar to the work presented here. This enables the power consumed by input vector generation to be separated from the design power consumption. Significantly, results in [ESJ06] point out that a maximum error of 132% was reported in the estimates provided by XPower, for the implementation of the FIR filter in Virtex II Pro and Virtex E.

The work in [WAL04] analyzes the impact of pipelining on power consumption in both Xilinx and Altera FPGAs by varying the number of pipeline stages and detecting the power difference. However, as they measure the total board consumption, they have no way of isolating dynamic or interconnect power consumption of the FPGA in order to guide other architectural decisions, apart from the number of pipeline stages.

Cycle-by-cycle energy measurement in Xilinx FPGAs is presented in [LLCC05]. The measurement is based on switched capacitors, similar to [RPM⁺03], which allow determining the static and dynamic energy per clock cycle. The setup is presented in Fig. 2.2. The switches are alternatively turned on-off, and each capacitor (C_{S1} and C_{S2}) charges for one clock cycle and discharges for another. The energy is computed from the final voltage values over the capacitors (i.e. when the capacitor is fully charged and fully discharged). Beside the total energy, the system is able to compute static and dynamic power separately by using the resistance R_S , and obtaining the value of the on-chip by-pass capacitor C_B through the charge sharing rule. Additionally, the authors compute the average power value and report high overestimation errors when these values are compared to XPower estimates. Although they are able to identify the difference in the interconnect power by applying area constraints or to investigate the effects of

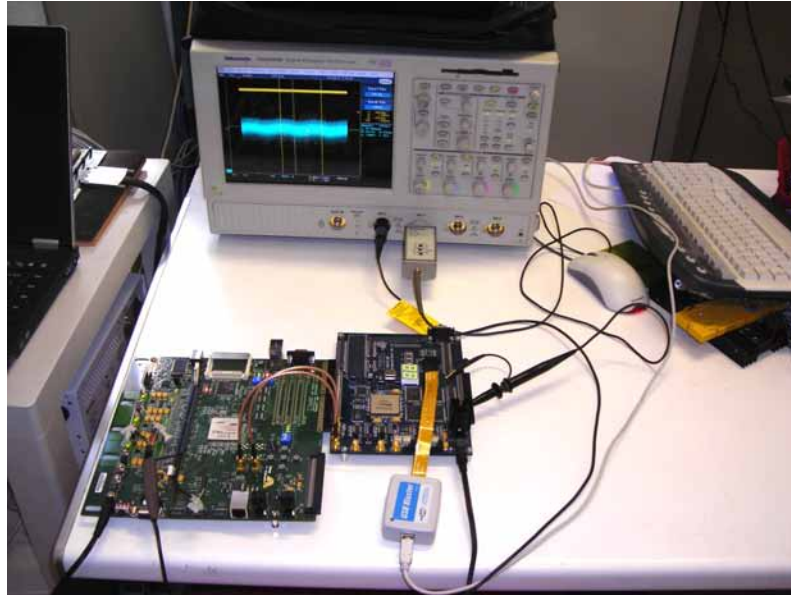


Figure 2.3: Measurement setup

loop-unrolling and pipelining on a design, the logic and interconnect power components are not identified separately from the energy consumption of the whole FPGA core.

The approach described in [dHAJW⁺07] uses measurements for the characterization of high-level power estimation models. This work is focused on the development of instruction-level power estimates in soft-core processors implemented in FPGAs and falls out of the scope of this study.

2.2. Measurement setup

The first goal of the measurement methodology is to obtain the capacitance values of the global wires used for routing in Xilinx Virtex-II Pro devices, as this information contains proprietary technology details and it is unavailable to us. These values will be later used for obtaining the value of the interconnect measured power.

In order to accomplish this goal, we use a common method to extract the effective capacitance [SKB02, DT05, EJH⁺04].

The measurement setup is presented in Fig. 2.3 and its block scheme in Fig. 2.4. The system contains two FPGA boards: a XUP board from Xilinx and a Stratix DSP Development board from Altera. The board from Altera is used for loading the simulation vectors to the XUP board. The XUP board serves for measuring the power of a specific design. As the power supplies for the core, I/O pins and auxiliary power supply are separated on the XUP board, we measure directly only the core power of the FPGA. The 1.5V power supply for the core voltage is provided by a synchronous buck-switching regulator connected to the 4.5V-5.5V external

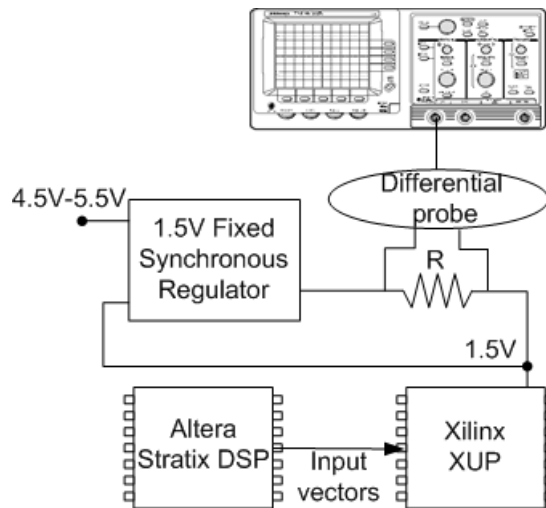


Figure 2.4: Block scheme of the measurement setup

power input [Xil05].

We use a resistance at the entrance of the core power supply to the chip, and for each test design, we measure the voltage over this resistor, which enables us to calculate the current provided by the supply. The value of this resistance is set to 10 ohms as this value provides maximum measurement precision and ensures the correct functionality of the buck-switching PWM regulator on the XUP board. The tolerance of the resistance is $\pm 5\%$ and it indicates the measurement systematic error. An interested reader can find additional information on how this particular resistance value was chosen in Appendix A. The functionality of the chip itself is guaranteed by a direct feedback from the chip power supply to the input of the regulator (see Fig. 2.4). The voltage over the resistance is measured by using a differential probe Tektronix P6248, and the measured value is the average of 750000 voltage values recorded in the oscilloscope (75 values for each of the 10000 loaded input vector pairs). An additional signal is generated on the Altera board that indicates the beginning and the end of the loaded input vector sequence. The power is then obtained as the product of the power supply voltage and the current going through the resistance.

The power is measured for simple designs that consist of a multiplier or adder core (in further text both referred to as modules), that is replicated several times in the design (between one and four times), in order to improve the accuracy of the measurements. The number of replications is limited by the number of I/O pins on the board that are aimed at user purposes.

As the designs are stimulated externally, they do not contain extra blocks like memory arrays, control logic etc., that would contribute to the total power and thus, prevent any possible separation of the module power from the global interconnects. Hence, each design consists of several identical modules and the lines that connect the module pins to the I/O pins (see Fig. 2.6).

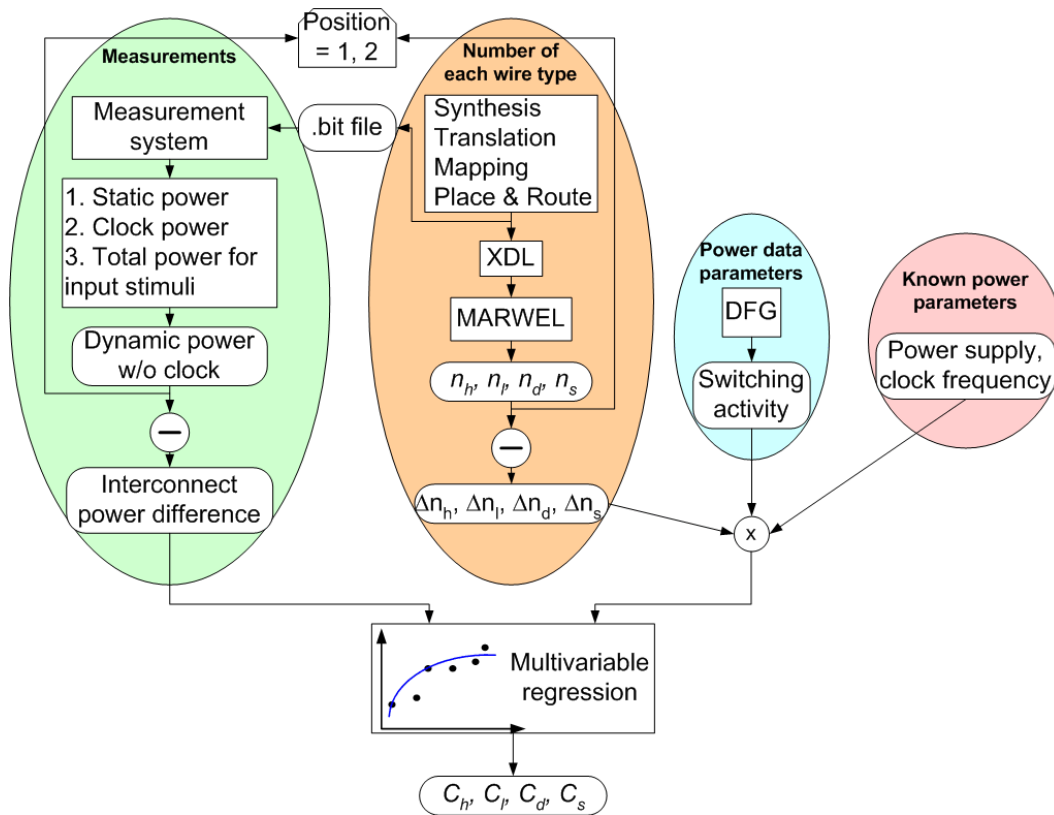


Figure 2.5: Methodology for effective capacitance extraction

For each design, we repeat the following steps:

- 1) The static power is measured when no input vectors and clock signal are injected.
- 2) The clock circuitry power is measured when the design is stimulated only with the clock signal, while the inputs are set to '0'.
- 3) Various measurements are performed for sets of 10000 input signal vectors with gaussian distributions and different autocorrelation coefficients.
- 4) The power of the clock circuitry together with the static power is subtracted from the power values obtained in 3), as to isolate the dynamic power of the logic and the interconnects for each input stimuli set.

The static power varies with the state of logic signals during design operation, and also with the way a design utilizes the FPGA hardware. The activity of the logic signals increases the chip temperature, which in turn, increases the static power as well. However, the designs we have used are extremely small, so it has been assumed that the static power increase would be negligible. In order to confirm this assumption, we have repeated steps 1) - 4) at two different frequencies (50MHz and 100MHz) for several of the most power-consuming designs in the set (containing multipliers implemented in LUTs), and indeed the relationship between two obtained values for isolated dynamic power for each design, corresponded to the relationship between these two frequencies (i.e. two).

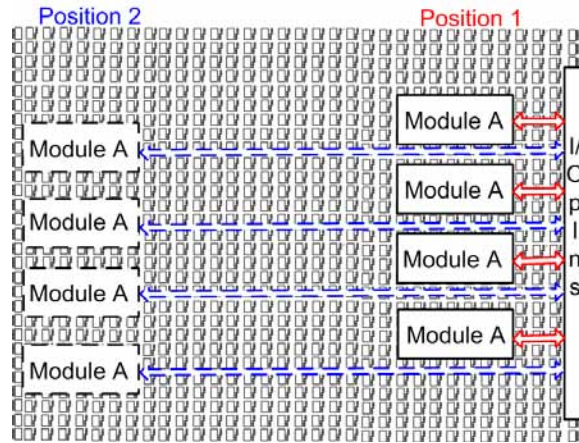


Figure 2.6: Two different module positions in FPGA

As the circuits contain only synchronized combinatorial logic without any feedback loops, it can be considered that there will be no toggling on logic signals when all the inputs are set to '0' and that the clock power is measured properly.

In order to separate the power of a component from the power of the global routing we use a method similar to [SKB02,DT05], where the effective capacitance of each of the global routing resources is obtained through the measurements. The methodology is presented in Fig. 2.5.

First, we repeat the set of measurements 1) - 4) for two different positions of the modules on the chip: one where the modules are placed very close to the I/O pins, and the other where they are placed far from them (see Fig. 2.6). We use area constraints in order to accomplish the wanted module positions. By subtracting the two values obtained for the dynamic power consumption of these two designs, we are able to obtain the value that corresponds to the power consumption of the interconnect difference between them, as it is explained next.

It is important to note, that the modules considered here have registered inputs and outputs. Inserting registers at the inputs and outputs is necessary in order to eliminate the glitching that might occur inside the module due to the different paths from the I/O pins and thus, we ensure that, as a result of the subtraction, the module power is completely cancelled.

2.2.1. Wire capacitance extraction

In commercial FPGAs, routing is accomplished through the hierarchy of segmented routing resources in order to achieve high speed. The most power consuming are the long lines, followed by the hex and double lines, while the least consuming are the single lines [SKB02, AN04a, PHB06].

We model the effective capacitance of each resource (long, hex, double and single) as the capacitance of the routing wire together with the programmable switch that drives the wire, as in [DT05]. After placement and routing of a design, the Xilinx tool ISE creates a native

circuit description file (.ncd) which represents the physical circuit description of the input design as it applies to a specific device [Xil]. We have developed a tool in C++ called MARWEL (Measurement of ARchitectural WirE Lengths), based on the Graph Template Library (GTL) [GTL], which is capable of obtaining the length and the number of the different wires used from the XDL file. This file is the text version of the placed and routed circuit description (.ncd) and is created by the Xilinx Design Language (XDL) tool. An interested reader is referred to Appendix A for more information on MARWEL.

Therefore, for each interconnect i that goes from or to I/O pins in the design, MARWEL provides the number of hex, long, double and single wires used for its routing: n_{hi} , n_{li} , n_{di} and n_{si} .

As the inputs and outputs are registered and there is no glitching in the wires that connect I/O pins with inputs and outputs of the modules, we are able to obtain the switching activities sw_i , of the routing wires from simple data flow graph simulations. The value of the switching activity for each interconnect is then multiplied by the corresponding number of wires of each type used for its routing.

According to (1.2), we need four parameters in order to calculate the power of the interconnects. Two of them are known, as the power supply has a value of 1.5V for Virtex-II Pro devices, and the clock frequency is fixed to the value used in our measurements (50MHz or 100MHz).

As we can not obtain the measured interconnect power value separately from the rest of the design power, we subtract the obtained dynamic power for two different positions of modules on the chip. Thus, we eliminate the logic power that is the same for both implementations. This allows us to break-out the power consumed in the interconnects. Therefore, we can express the power difference of a design in the two measured positions as:

$$\begin{aligned}
P_1 - P_2 = & 0.5 \cdot V_{dd}^2 \cdot f \cdot (C_h \cdot \sum_{i=1}^{I_1+I_2+O} [(n_{hi}^1 - n_{hi}^2) * sw_i] \\
& + C_l \cdot \sum_{i=1}^{I_1+I_2+O} [(n_{li}^1 - n_{li}^2) * sw_i] \\
& + C_d \cdot \sum_{i=1}^{I_1+I_2+O} [(n_{di}^1 - n_{di}^2) * sw_i] \\
& + C_s \cdot \sum_{i=1}^{I_1+I_2+O} [(n_{si}^1 - n_{si}^2) * sw_i])
\end{aligned} \tag{2.1}$$

where P_1 and P_2 are the measured dynamic power of the design with the modules in the positions far from and near to the I/O pins respectively, C_h , C_l , C_d , C_s are the variables representing the effective capacitance of the hex, long, double and single wires respectively, I_1 , I_2 are the word-lengths of the two input operands and O is the word-length of the output. The design position is identified through superscripts 1 (far) and 2 (near). A multivariable regression over

a number of measurements for modules with various operand word-lengths is applied, as to obtain the effective capacitance for all types of wires.

Once we have these values, we can obtain the power consumption of any interconnect, by using the information about the number of different wire types used for its routing. Finally, the power is computed as:

$$P = 0.5 \cdot V_{dd}^2 \cdot f \cdot sw \cdot (n_h \cdot C_h + n_l \cdot C_l + n_d \cdot C_d + n_s \cdot C_s) \quad (2.2)$$

2.2.2. Logic and Input buffer power

Beside the interconnect power, logic power of the modules can be obtained as follows.

The remaining power obtained by subtracting the power of all the interconnects from the dynamic power of the design contains two power components: the module power and the input buffer power (as the input buffers are also power supplied by the FPGA core voltage). In order to obtain module power, we need to compute the effective capacitance of input buffers as well. This capacitance is computed by measuring the power of two designs: one containing three multipliers implemented in LUTs, and the other containing only one multiplier implemented in LUTs. First, we subtract the corresponding interconnect power from each of the designs. Thus, we obtain the following logic power values:

$$P_{log,1} = 3 * P_{mult} + P_{in_buf} \quad (2.3)$$

$$P_{log,2} = P_{mult} + P_{in_buf} \quad (2.4)$$

where $P_{log,1}$ and $P_{log,2}$ are the logic power values of the first and the second design, respectively, P_{mult} is the logic power of the multiplier and P_{in_buf} is the power of the input buffers. From these two equations we are able to extract the power of the input buffers. The effective capacitance of a single input buffer is then obtained by dividing P_{in_buf} by the sum of the switching activities of the inputs, square of the power supply and design frequency:

$$C_{in_buf} = \frac{P_{in_buf}}{0.5 \cdot V_{dd}^2 \cdot f \cdot SW_{in_buf}} \quad (2.5)$$

$$SW_{in_buf} = \sum_{i=1}^{N_{in_buf}} sw_in_i \quad (2.6)$$

where N_{in_buf} is the total number of inputs, and sw_in_i is the switching activity of the i -th input. As a result, the measured effective capacitance of a single input buffer is found to be 3.52 pF.

The module power can now be easily obtained by subtracting the interconnect power and the power of the input buffers from the total dynamic power of the design.

Table 2.1: Effective capacitances for different wire types.

Wire type	Capacitance per unit-length [fF]
Long	178.133
Hex	86.578
Double	71.47
Direct	≈ 0

2.2.3. Effective wire capacitances

In this section, we give the values obtained for the wire capacitances, and also the errors that were obtained when the measured power difference (corresponding to the interconnect power difference) was compared to the estimated interconnect power difference computed by using these capacitance values and the information obtained from MARWEL. In order to ensure correct values, some measurements were repeated several times under different temperature conditions in the laboratory. The static power changed accordingly to the alteration of the temperature, and the maximum relative error between dynamic power values obtained through the repeated measurements was found to be only 3%, thus verifying the correctness of the measurements.

The experiments were performed on four different size multipliers implemented in LUTs, four different size embedded multipliers and five different size adders with operand sizes of 8, 12 and 16 bits. The module input signals had a zero-mean gaussian distribution with autocorrelation coefficients that varied between 0 and 0.9995, meaning that the switching activity of the input bits varied, between 0.00025 and 0.5 (the relationship between switching activities and autocorrelation coefficients will be discussed in detail in chapter 4). The characterization set used for the multivariable regression considered the power values corresponding to the input signals with autocorrelation coefficient equal to 0 (i.e. switching activity of 0.5), as they provided the largest consumption and thus, the best accuracy. Additionally, we have also used the multiplier 16×16 and adder 16×16 and autocorrelation coefficients of 0.5, 0.9 and 0.99 as they are the components with the largest size used for measurements, and thus, have the largest consumption. Although adder consumes less power than a multiplier, we have replicated the adder core three times in order to improve the measurement precision. The measured capacitance values for the different wire types are given in Table 2.1. Their values correspond to the capacitances spanning the distance between two neighbouring CLBs. The total wire capacitance for each wire type is obtained when the corresponding capacitance per unit-length is multiplied by the number of segments that the wire spans.

Furthermore, for each module and each autocorrelation coefficient, we computed two values: δP , that corresponds to the power difference for the module positions 1 and 2, in the

Table 2.2: Error for the interconnect power computed with the effective capacitance values.

Module types	Error [%]				
	$\rho = 0$	$\rho = 0.5$	$\rho = 0.9$	$\rho = 0.99$	$\rho = 0.9995$
mult16x16	4.23	-6.05	10.56	-2.01	-0.81
mult12x12	9.75	6.16	11.10	1.61	6.05
mult8x8	-1.04	-0.35	-0.64	-11.72	-1.81
mult12x8	-3.66	-9.41	-4.18	-11.14	-8.93
emb12x12	5.65	-4.98	-5.51	-12.71	-7.06
emb16x12	13.95	8.16	5.33	5.76	6.22
emb16x8	-6.89	-6.58	-2.16	-7.83	-6.98
emb12x8	-3.04	-13.2	-0.63	-7.18	-0.97
add16x16	-0.95	2.11	2.50	1.03	-1.60
add12x12	4.27	0.84	4.79	-6.00	-0.93
add8x8	7.76	6.51	2.01	3.28	12.39
add16x8	6.03	5.84	6.42	4.52	12.09
add12x8	-3.84	-2.17	3.96	3.91	5.77

left-hand side of equation (2.1), and P_{cap} , that corresponds to the right-hand side of the same equation, computed from the obtained effective capacitance values. Table 2.2 shows the relative errors when the computed P_{cap} is compared to the measured δP . Shading is used to differentiate the characterization set. It can be observed that, the resulting discrepancy is always smaller than 13.95 % and probably occurs due to the local wire parasitics as explained in [DT05].

2.3. XPower

XPower is a Xilinx low-level tool used for power estimation. It allows a user to analyze total dynamic power, and power per-net, of routed, partially routed or unrouted designs. However, it is recommended to fully place and route a design in order to obtain the most accurate estimates.

The typical design flow for XPower is given in Fig. 2.7. First, the gate-level timing simulation of the placed-and-routed design is run, and as a result, a VCD file is obtained. We have used the Mentor Graphics ModelSim simulator for this purpose, which is one of the simulators supported by XPower. The VCD file contains detailed information on the toggling rates and frequencies of all the signals in the design, and it is used as the input simulation file for XPower. Beside this file and the .ncd file which contains the physical information of the placed-and-routed design, a user can optionally load the .pcf file with information about user constraints and the .xml file that contains settings for the design saved after some previous XPower analysis.

The output file of the tool is a power report. A user can specify if the report should be

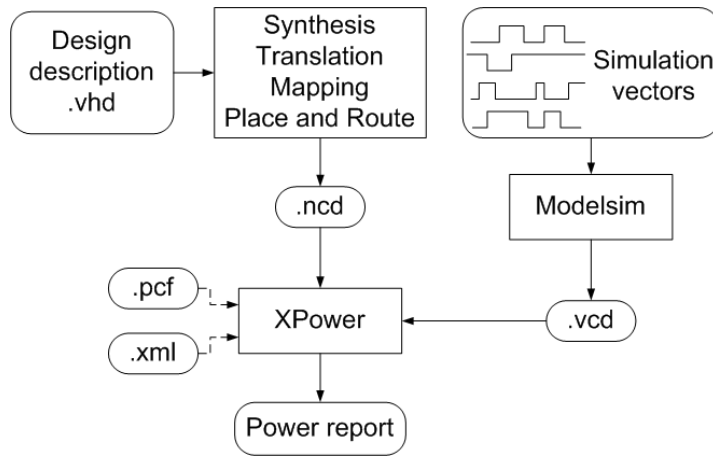


Figure 2.7: XPower design flow

"Standard", "Detailed" or "Advanced", depending on how much information does he or she need. The report that contains the most detailed information on design power is the "Advanced" report, and we have always used this option in our experiments. Information about the power of each individual element in the design is listed and sorted by type into the following four groups:

- 1) The power of the clock tree including both, the power of clock nets and the power of all clock buffers, except the input clock buffer (Clock power group);
- 2) The power of logic considering only the power inside Configurable Logic Blocks (CLBs) and embedded blocks (Logic power group);
- 3) The power of signals including both, local connections inside the component, like the connections between the CLBs that form the component, and global connections used between I/O pins and component input and output registers (Signals power group);
- 4) The power of input buffers (Inputs power group)

In the following chapters, we have used XPower for two different goals. The first goal is the comparison of the estimation models developed here with some other estimation models that were characterized with XPower, and/or that needed a very high number of measurements for their characterization, so that it was difficult to characterize them through on-board measurements. The second goal is the exploration of XPower accuracy against the on-board measurements.

CHAPTER 3

Power estimation models for interconnections

In this chapter, we present a high-level interconnect power model that is capable of giving fast and accurate estimates for any given distance between the connected components. There are two input parameters to the model: the relative position of the components and the ordering of the pins on the components' boundaries. Hence, the only information that is needed for the usage of the model is the design floorplan, meaning that it can be used at higher levels of abstraction. Consequently, the proposed model is suitable for integration with floorplan-aware high-level synthesis aimed at power optimization, where accurate estimates are needed in the shortest possible time. In further text, this model will be referred to as High-Level Interconnect Model (HLIM).

This chapter is organized as follows. First a brief overview of the previous work on interconnect power estimation is given in section 3.1. It is followed by the interconnect power analysis for Virtex II Pro devices in section 3.2, where some router properties that are important for modelling interconnection power consumption are stressed out. Next, a model for interconnection power between two modules is given in section 3.3, and it is extended for the case of n modules in section 3.4. Experimental results are given in section 3.5.

3.1. Interconnect power estimation: background

The problem of interconnect power estimation is to estimate the capacitance and activity of each wire in a design. Because the activity can be derived from the activity data of the modules, the

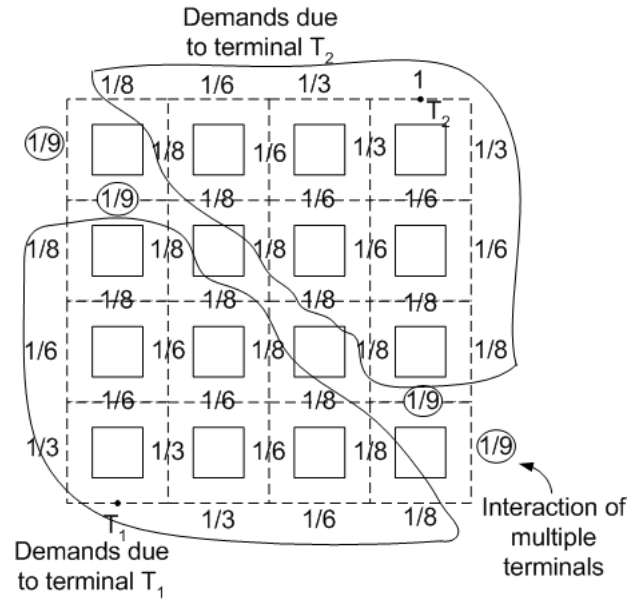


Figure 3.1: fGREP: net demands for 2-terminal net

major problem is to estimate the wire capacitance, which is primarily determined by the wire's length.

Interconnect wire length estimation has been previously studied in the literature due to its high importance for circuit delay estimation. As both delay and power of interconnects depend on the wire length, an overview of the wire length estimation techniques developed for the purposes of timing analysis will be also given in this section.

Most of the existing interconnect estimation techniques are applied at the post-placement design level, as the information on global routes is extremely scarce at the higher levels of abstraction. Furthermore, in [MCSB06], it was noted that by changing the seed of the placement algorithm for generic logic in FPGAs, the delay of some interconnects can change significantly (up to 2 or 3 times). In the continuation, a description of several post-placement techniques is given, followed by a description of the techniques that attempt to model the interconnections at pre-placement stages of the design flow. Finally, wire length estimation techniques based on Rent's rule are explained at the end of this subsection.

3.1.1. Post-placement techniques

The methodology presented in [KBB01, KBB06] estimates only interconnect routing demand, while power is not analyzed. The design routing demand is defined for each channel in an island-style FPGA and represents the number of routes that are used in the channel. They predict the channel width and the routing demand as follows. First, for all routing channels in the given net the terminal demand is computed, as the inverse of the total number of the

routing channels at the same distance from the terminal. Then, the net-demand for each routing channel is computed as the maximum of all terminal demands in a net. Finally, the routing demand is obtained by adding up all net-demands over all nets in the design. This enables to find out if the routing is feasible or not. In Fig. 3.1 two net terminals are presented together with the net demands of the channels between them. In this particular case, the net demands of the both terminals are the same due to the symmetry of their locations in the FPGA. The routing analysis is performed by using Versatile Place-and-Route tool (VPR [BR97]), a non-commercial tool used to study FPGA architectures. This tool enables the user to explore different FPGA structures, by changing the number of LUTs per slice, the number of slices per CLB, the number of routing wires between each two CLBs, the channel width, etc.

The same authors compare their methodology with several other methodologies that estimate interconnect routing demand in [KBB04]. These methodologies were originally proposed for ASIC flows, but they were readily adapted for FPGAs in [KBB04]. One of them is RISA [Che94], where the routing demand of a net depends on the net's bounding box size and the net weight which is computed empirically by generating a wire distribution map (WDM). For each M -point net, a WDM is built from a large number of optimal Steiner trees [BOI94] for randomly distributed M points, while the net weight is obtained by normalising the sum of the demands due to each steiner route. Another methodology used for comparison is Lou's method [LKS01]. It is based on the ratio of the number of the paths that use a specific routing region to the total number of paths possible. This method operates on two-terminal nets, while multiterminal nets have to be decomposed in two terminal segments. It has been adapted for congestion estimation in FPGAs also in [AN04a]. According to the comparison, the methodology presented in [KBB01] has the most accurate estimate of the routing demand (average error less than 1%), followed by [Che94] (average error of 3.77%), and [LKS01] (4.12% average error). However, the results are compared to the values obtained from VPR, so the methodologies are not tested against commercial tools which have a more faithful representation of the real design implementation.

In [AN04a], it was shown that changing the order of nets before routing the exact same design, can lead up to 20% of power variations on average, due to the variation of the interconnect capacitance. This capacitance noise occurs due to the trade-offs that the router makes between the FPGA resources allocated to each net, which are resolved arbitrarily in different cases. Thus, the accuracy of pre-routing power estimation models is limited by this inherent capacitance noise.

The methodology for interconnect power estimation presented there, is similar to the one that is used for power macro-models. Power is represented in the form of an equation, but the variables in the equation depend on the routing properties of the design and the underlying FPGA architecture. They include pre-layout parameters such as fan-out and half perimeter of

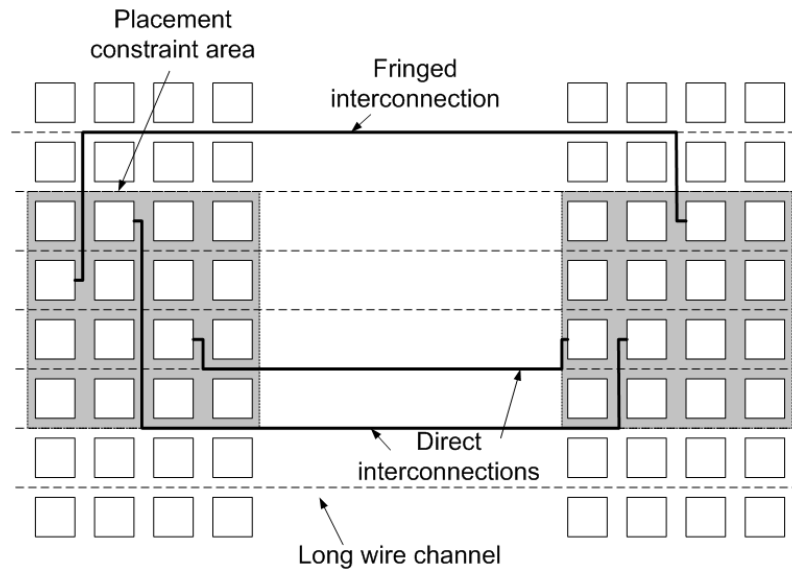


Figure 3.2: Communication link between modules and their different types of connections

the bounding-box of a net, and architecture-specific parameters such as the number of CLB tiles in which a net has pins, the pin types on a net, congestion estimated as in [LKS01] etc. The influence of each of these parameters on interconnect power is studied in detail in order to find the best function for the estimation model. Considerable benefits are achieved by introducing model parameters of the underlying FPGA interconnect architecture in addition to the pre-layout parameters (the estimation error for the model with architecture parameters decreases more than 20% compared to the pre-layout parameter model). Furthermore, an analysis is performed in order to choose the lowest order of the polynomial function that models the interconnect power, without significantly sacrificing accuracy. The final model has an average error of approximately 35% while taking into account power variations due to different interconnect capacitance in the routing solutions.

3.1.2. Pre-placement techniques

The previously described techniques are very time-consuming, as the design needs to be mapped and placed in order to obtain architecture-specific parameters.

The methodology described in [FWAW05], presents power estimation at the post-synthesis level. They use the information only about the fanout of the design in order to predict interconnect power. The main goal is not to achieve precise power estimation, but to guide the user to design-level hot-spots in the circuit. The interconnect power prediction errors are not reported separate from the rest of design power. However, these errors can be deduced from the plot given in their work which presents the nets' capacitances values versus nets' fanouts of one analyzed circuit. The errors go over 400% for some of the design nets. Besides, the results are

given only for the circuit which was used as part of the characterization set and therefore, the applicability of the model remains unclear.

A method for estimating the bounding box of the connections between the modules was proposed in [CCC07]. It uses the circuit topology and pre-characterized component fan-out and area values to calculate routing demands before RTL-synthesis. Still, it seems that the estimation errors are quite high, since the authors state that their values are similar to the errors presented in [FWAW05]. The errors themselves are not presented in their work.

A stochastic approach was used in order to predict interconnection lengths of communication links in FPGAs in [MSCL07] (see Fig. 3.2). The model is applicable to floorplanning, as it depends on the parameters such as area dimensions of the connected regions, the Manhattan distance between the regions, the number of connections and the number of available long lines in a channel. The model also takes into account the routes that use long lines in channels that are further away when all direct connections are occupied (fringed connections in Fig. 3.2). The average interconnection length depends on the distance between the modules and the average local route length inside the placement area. The model is used in order to determine the delay of the connection, which is found to have a linear dependence on the wire-length. This approach has some similarity to the approach for interconnect power estimation presented in this work regarding the modelling of the component by its area constraint, while accounting for the local routes separately (it will be explained in detail in section 3.3). However, they assume that no more than two regions are connected, only long lines are used for the routing, and the connected regions are separated by a significant distance on the chip, whereas in the work presented here there are no such limitations. The average error for predicting interconnection length is found to be 7% for the benchmarks that consist of only two FIFOs with a variable number of the connections between them.

A pre-placement methodology for predicting individual wire length and routing demand of each net in designs implemented in FPGAs is presented in [BB03]. A circuit is presented as a set of nodes, and a heuristic approach is applied in order to simulate the placement of the circuit and determine the bounding box for every single net. The design has to be mapped before applying this methodology. The average error reported for bounding box span estimation is 11.6%. The resulting bounding boxes are further used in order to predict the channel width for the routability of the design, and the reported average error is 6.1% for the peak routing demand. The estimates are only compared to the values obtained from VPR.

Some high-level techniques have been proposed for estimating interconnect power in ASICs [GZJ03, ZJ05]. They develop a statistical model that describes the characteristics of a single interconnect, given its initial and terminal coordinates in the floorplan. The primary purpose is to determine the segment length distribution of an interconnect and the number of vias and buffers on this interconnect in order to obtain all interconnect power components: switching

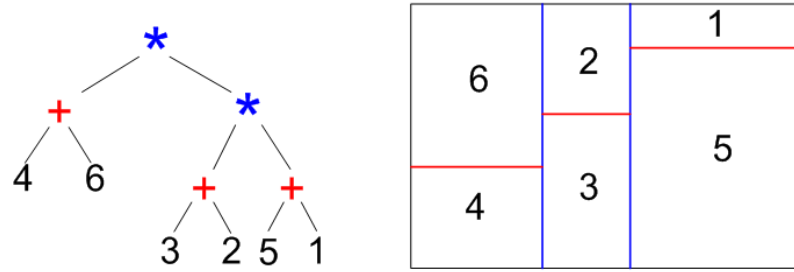


Figure 3.3: Slice floorplan: binary tree and rectangular representation

power, via power and repeater power. For this purpose, they assume that the route between two points has to go via intermediate points where the route changes its direction. The occurrence of intermediate points is modelled as a Poisson process. However, the model includes an empirical constant that has to be determined per design basis, which represents a serious limitation for the model utilization. There are no reported errors for the overall interconnect power model, due to the lack of interconnect power values to be compared against. Instead, the authors validate their approach for several benchmarks by plotting the real and estimated segment length distribution in order to explore if the intermediate points can be modelled as a Poisson process. They also give the relative errors for the estimated number of vias in the interconnections (around 3% on average), and investigate if the interconnection lengths follow the Manhattan distance (only 2% violate this distance). However, a complete interconnection power model validation is missing.

Another popular wire estimation approach for ASICs is performed by generating a slicing tree floorplan [SHS⁺03a]. The wire capacitance is derived using a capacitance model which is based on wire length, number of pins and number of branch points. Pins are the connecting points to RT-resources and their number can be extracted from the corresponding RT-model. The number of branch points and the wire length is extracted from a floorplan using Steiner trees. For interconnect length estimation SA (simulated annealing) based floorplanner is used. It performs the algorithm for slicing floorplans. A slicing floorplan is a floorplan that can be obtained by recursively partitioning a rectangle by either a vertical line or a horizontal line into smaller rectangular regions. It can be represented by an oriented rooted binary tree, where each internal node of the tree is labelled either * or +, corresponding to either a vertical cut or a horizontal cut (see Fig. 3.3). This floorplan is further optimized through special "moves" between the leafs and nodes of a binary tree, thus optimizing the overall interconnect power consumption. The reported interconnect power savings are 24% on average, while the interconnection power and wire length estimation are not verified separately. Apart from the wire length and the number of branch points, this technique should also be able to estimate the number of switch matrices and the type of wires used for routing in order to be applied to FPGAs.

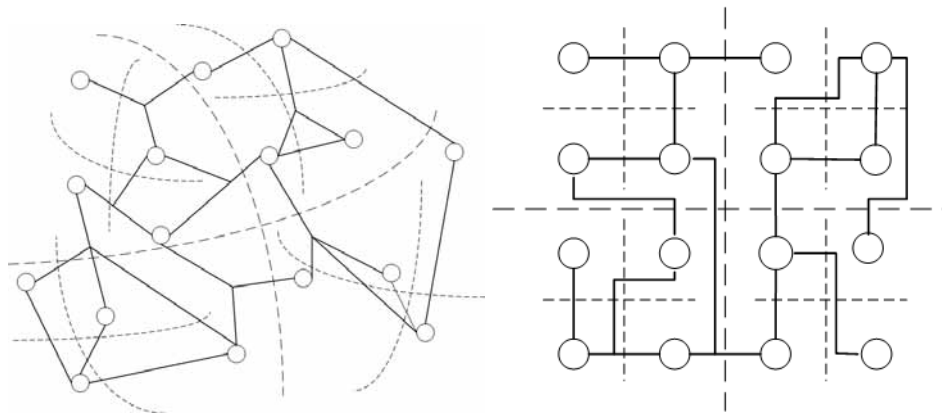


Figure 3.4: Recursive partition scheme of a design and the physical architecture

3.1.3. Rent's rule

Some efforts have been made to use Rent's rule to estimate the average wire length needed for the calculation of dynamic power consumption [CCF03] and delay estimation [NHCB02]. Rent's Rule is an empirical metric used to quantify circuit complexity. In the 1960's several researchers independently found that the relationship between the number of signal input and output (I/O) terminals T and the number of gates N of a circuit can be expressed as a simple power law expression known as Rent's Rule:

$$T = k \cdot N^p \quad (3.1)$$

where parameter p is an empirical constant, known as Rent's parameter and k is the average number of connections per gate. Since then, this empirical measure has been applied in a variety of areas, including studies of new computer architectures, synthetic benchmark generation and circuit routability.

Rent's rule was first used for wire-length estimation by Donath [Don79]. The circuit and its layout are hierarchically partitioned (see Fig. 3.4) and Rent's rule is used to determine the average number of connections between the partitions, assuming they have the same routing properties (i.e. the same Rent's parameter). The method presented there provides only the upper bound on the average interconnection length, and in some cases the estimated wire-length differs from the real wire-length by a factor of two. It was further improved in [vMSvC96] by assuming a non-random cell distribution in a layout region and including the limitations in the number of wires between the partitions expressed as the probability for each particular wire length. Both methodologies were used for predicting average wire length in ASICs.

The methodology presented in [SMS02a,SMS02b], utilizes Rent's rule as an empirical measure for efficient clustering and placement of circuits in clustered FPGAs by minimizing the number of external nets that need to be routed, and matching the Rent's exponent of the clus-

tered design to that of the underlying FPGA architecture. Their placement technique tends to cluster the circuits by absorbing as many small nets into clusters as possible and by depopulating clusters according to Rent's rule in order to achieve spatial uniformity in the clustered netlist.

In [SWD09], an analytical model that relates architectural parameters such as: look-up table size, cluster size, and number of inputs per cluster to the average post-placement wirelength of implementations in FPGAs, is presented. The aim is to understand the impact of architectural decisions on the expected wirelength.

The work presented in [DDM98a, DDM98b] introduces an improved wire length estimation based on a complete description of local, semi-global, and global wires for targeted microprocessor architectures. It models the architecture as homogeneous arrays of gates evenly distributed in a square die. This architecture model closely reflects the characteristics of an island-style FPGA architecture, where each logic block can be treated as a gate. Consequently, it has been used in [CCF03] for the purposes of wire-length estimation in FPGAs. They use the Rent's exponent extracted from [SMS02a] as it corresponds to the underlying FPGA architecture and is design-independent.

However, all these equations for wire length estimation are derived under the assumption that for any two blocks exists a number of connections sufficient enough to cover all the interconnections between those two blocks. Therefore, it is necessary to define a new wire length estimation model for designs implemented in FPGAs which would include the existence of their routing resource limitations. Furthermore, in non-hierarchical designs where different parts of a design exhibit different routing congestion, wire length and number of interconnections, the errors introduced by methods based on Rent's rule applied to FPGAs can be significant [CS00, SMS02a]. In these cases, the characterization of the routing structure of the whole design with only one parameter (i.e. Rent's parameter) is hardly possible.

3.1.4. Summary of the previous work on interconnect estimation

In Table 3.1 we present the summary of features of some of the described interconnect models and the model presented here: HLIM. The models that are not listed in the table are either used for some specific designs as is the case in [Don79, vMSvC96, DDM98a], where the design has to be hierchically "self-similar" (i.e. all the submodels of the design need to have the same interconnect complexity), or the errors are not stated and seem to be too large for the purposes of power estimation as in [CCC07].

The target technology is listed first. It is followed by the method and its primary goal (interconnection power estimation, wire-length and/or routing demand estimation). Next, the tool used for obtaining "real" interconnect values for model validation is listed for each of the methods, together with the last phase in the design flow that needs to be performed in order to

Table 3.1: Summary of features of the interconnection models.

Technology	Method	Estimation	Tool	Design phase
ASIC	Princeton Univ. [GZJ03, ZJ05]	Power	Synopsys	floorplan
	Univ. of Oldenburg [SHS+03a]	Power/ Wire-length	Synopsys	floorplan
ASIC, FPGA	Cadence: RISA [Che94]	Routing demand	Synopsis, VPR	post-place
	Synopsys: Lou [LKS01]	Routing demand	Synopsys VPR, ISE	post-place
FPGA	UT Dallas: fGREG [KBB01, KBB06]	Routing demand	VPR	post-place
	UT Dallas [BB03]	Wire-length/ Rout. demand	VPR	post-map
	Imp. Coll. of London [MSCL07]	Delay/ Wire-length	ISE	floorplan
	Univ. of Toronto [AN04a]	Power	ISE	post-place
	Univ. of South. Calif. [FWAW05]	Power	ISE	post-synthesis
	UCLA: Rent [CCF03]	Power/ Wire-length	VPR	post-synthesis
	Tech. Univ. of Madrid: HLIM	Power	Measurement	floorplan

apply the corresponding method.

It can be seen that the method proposed here needs very few design phases to be completed, as it depends only on the floorplan. Floorplanning can be considered to be at the same level of abstraction as a post-synthesis design phase, since it takes an RTL design description as input. Furthermore, as it is not necessary to have a complete RTL description (including all the control signals, etc.), floorplanning could be categorized as a design phase between high-level and RTL synthesis. Another important observation is that the model presented here is the only interconnect model that has been verified with on-board measurements, thus resulting in the most confident estimate values.

The accuracy of the models has not been presented in the table since the models have different estimation goals (power, delay, wire-length, routing demand), and different benchmarks were used for their validation. Among the five models from the literature that target intercon-

nection power estimation, we have been able to compare the accuracy of our model with only two of them: [AN04a] and [FWAW05]. [GZJ03, ZJ05, SHS⁺03a] are oriented to ASICs and their adaptation to FPGAs is not so straightforward; and [CCF03] is based on non-commercial tools and the reported errors are given only for the total wire-length and not for the interconnect power estimation.

The model presented here achieves similar accuracy as the model presented in [AN04a], while it is capable of obtaining estimates at an earlier phase in the design flow. On the other hand, it provides the estimates at the same level of abstraction as the model presented in [FWAW05], but it achieves far better accuracy, as it is shown later.

3.2. Interconnection Power Analysis

In this section we will analyze the interconnection power and try to establish its dependence on the wire length. We always consider the connections between arithmetic blocks, as this thesis focuses on DSP circuits that are primarily data-path oriented.

The smallest segment that a global wire can span inside an FPGA corresponds to the distance between two neighbouring Configurable Logic Blocks (CLBs), and is referred here to as one unit-length.

During the routing phase, the minimum cost paths are selected to implement the connections. The cost consists of two parts: the one that accounts for the competition between different nets for the same wiring segments, and the one that reflects the routing delay associated with the routing segment [She99]. If there is no congestion in the circuit, the first cost function can be neglected. Therefore, the router is timing driven in this case and we assume that the interconnect power per unit-length tends to be constant.

The router will try to minimize the total interconnect delay, which would reflect in the minimization of the interconnect capacitance, resulting also in reduced interconnect power. Furthermore, it was already shown that, as a result of the router's minimal delay path search, the increase in the interconnect routing delay can be modelled with a linear approximation [Smi06]. Consequently, the wire capacitance increases almost linearly with the module distance, so we conclude that the interconnect power also tends to be homogeneously distributed over distances between the modules.

In order to validate this assumption, Fig. 3.5 shows the measured power per interconnect between two modules A and B (in this case two multipliers), versus their distance. The outputs of module B are connected to the inputs of module A as shown in Fig. 3.6. The position of module A is fixed near the I/O pins on the right-hand side of the chip. The position of module B is varied from the position nearest to module A, to the position near the I/O pins on the left-hand side of the chip, opposite to module A, and further up, along the I/O pins on the left-hand side

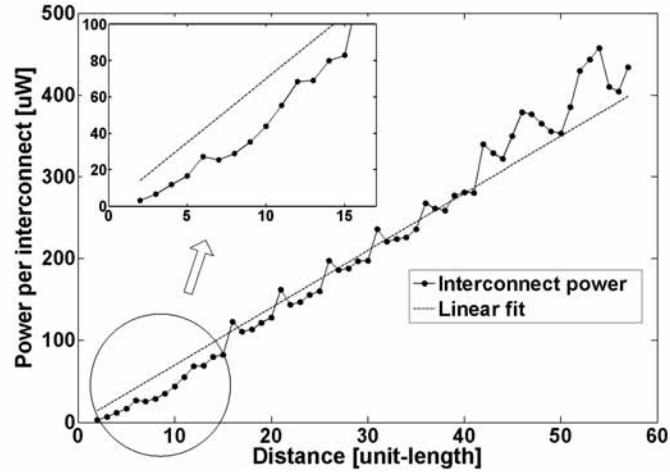


Figure 3.5: Power per interconnect between the modules A and B

of the chip. The distance between the modules is computed as Manhattan distance between the centers of the connected pins, which are marked in Fig. 3.6. The module pin center is defined as the center of the minimal bounding box that includes all of the module pins connected to the other module. Thus, if x_1 and y_1 are the coordinates of the module pin center of module A, and x_2 and y_2 are the coordinates of the module pin center of module B, the distance between them is computed as:

$$d = |x_2 - x_1| + |y_2 - y_1| \quad (3.2)$$

The power per interconnect is computed after the place-and-route of the design for each position by using the effective capacitance of the routing wires, obtained as explained in the previous chapter, and using the information about the length and the number of different wire types used for the interconnections obtained from the tool MARWEL. First, the total power of the interconnects is computed and then, it is divided by the number of interconnects to obtain the power per interconnect. The power values are normalized with the switching activity, because data dependencies are not significant for the purpose of this analysis.

It can be seen in Fig. 3.5, that the power per interconnect has almost a linear dependency on the distance between the modules. This confirms the assumption that the power per unit-length tends to be constant.

Two additional effects can be noted in Fig. 3.5. First, the power of the interconnects for larger distances follows some kind of pattern. In this case, it has a peak value every five unit-lengths. We believe that this is due to the limited connectivity of the switch matrices used for routing. At these specific distances there are no available combinations of connections between the wires, which would connect directly the input of one module to the output of another. As a solution, the routing tool searches for the connections in the neighbour CLBs around the

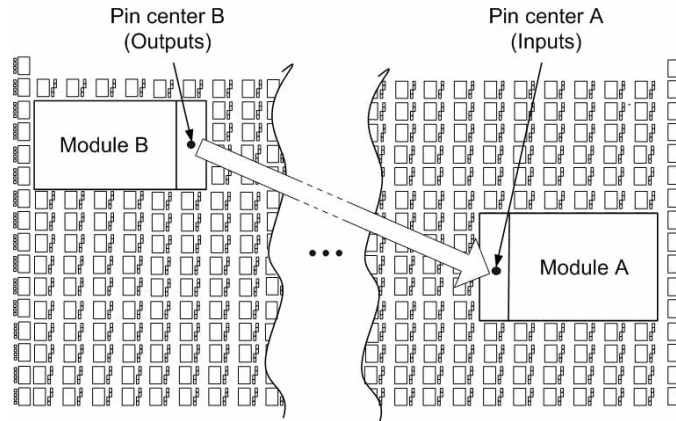


Figure 3.6: Simulation setup when the interconnects between two modules are considered

destination CLB and thus, increases the total wire-length.

This pattern seems to break at the largest distances. However, these are the distances where module B changes its path direction and continues to move along y-axis. The router has more choices for the shortest connection path and thus, the interconnect power does not necessarily follow any pattern.

The second effect is that, although the dependence seems linear, the linear fit does not describe the interconnect power accurately for the smallest distances. The upper left corner of Fig. 3.5 shows the power corresponding to the shortest distances. It can be seen that the linear fit overestimates the interconnect power, resulting in increased estimation errors. Further analysis of the type of wires used for global routing, has demonstrated that, beyond some specific distance d_l , the router uses all 4 types of wires: hex, long, double and direct. However, when the distance is smaller than d_l , it ceases to use long lines for routing. As the capacity of the long lines is the highest, this results in a significant decrease in the power consumption of the interconnects. This explains the overestimates of the linear fit. Moreover, for the shortest distances, it is observed that the router not only avoids using long lines, but hex lines as well.

As a result, three different routing zones can be identified. Fig. 3.7 shows the routing zones when the module is simplified and consists of only one CLB. The first routing zone corresponds to the minimal distance of the modules, where only direct and double lines are used. The second one, corresponds to distances smaller than some specific distance d_l , where three types of wires are used for routing: direct, double and hex (since we are dealing with Manhattan distances this zone will have the shape shown in Fig. 3.7). Finally, the third zone, corresponds to distances larger than d_l , where all four types of wires are used. The last two zones obviously depend on the distance between the modules. The first one, however, corresponds to the smallest distance between the modules, where there is a significant effect of the physical properties of the modules on the interconnection path. For example, in Fig. 3.7 it is assumed that the other connecting module is also a CLB, and thus the minimal distance between the modules would correspond

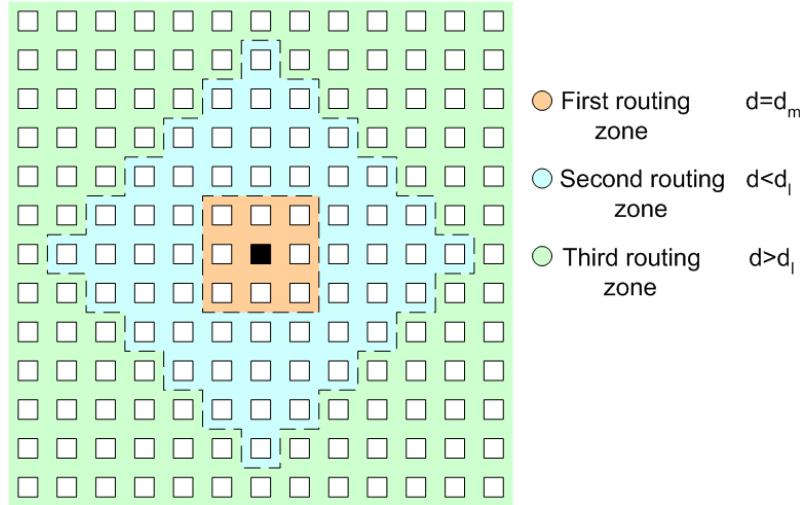


Figure 3.7: Routing zones

to one unit-length. As the CLBs placed diagonally from the center CLB (distance of two unit-lengths) can be reached by one direct wire, these CLBs are also included in the first routing zone. However, when the modules consist of a number of CLBs and have their own particular shape, the computation of the minimal distance is not so straightforward. This effect will be explained in detail in the next section.

3.3. High-level Point-to-point Interconnect Power Model

Since three different routing zones can be clearly distinguished, we propose the following power model for the interconnects:

$$P_{int} = \begin{cases} k_3 \cdot L, & d = d_m \\ k_2 \cdot (d - d_m) + k_3 \cdot L, & d_m < d < d_l \\ k_1 \cdot (d - d_l) + k_2 \cdot (d_l - d_m) + k_3 \cdot L, & d > d_l \end{cases} \quad (3.3)$$

where P_{int} is the power per interconnect, d_l is the specific distance beyond which the router starts using long lines, d_m is the minimal distance between the module pin centers, L corresponds to the distance between the module pins and their pin center as will be explained next, d is the distance between the modules, and k_1, k_2, k_3 are the coefficients calibrated by multiple regression analysis over measured power values for different distances between the modules.

Distance L is defined to model a number of different scenarios where the pin ordering between modules is not aligned.

In the case of the two modules described in the previous section, at the minimal distance of one unit-length, the B outputs and A inputs were completely aligned. In real-case designs, the connected inputs and outputs may not be necessarily placed in the same order, specially when

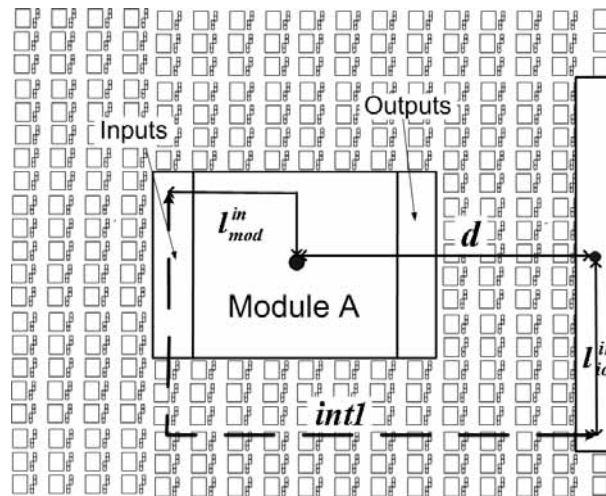


Figure 3.8: Simulation setup when interconnects between a module and I/O pins are considered

considering connections from or to I/O pins, as the I/O pin location also relates to the board design. Besides, in designs with feedback loops, it may occur that both inputs and outputs of a module are connected to only one specific module. This may result in a different minimal distance between the modules and, therefore, different interconnect power.

Such scenario is presented in Fig. 3.8. The design consists of the module A with both, inputs and outputs, connected to I/O pins which are placed in a column on the right-hand side of the chip. The module positions under consideration vary from the position nearest to the pins, to the position on the left-hand side of the chip, opposite to the pins. The inputs of the module A are placed on its left-hand side, while the outputs are placed on the opposite side. As both inputs and outputs are connected to the I/O pins, it is possible to see the effect of the physical dimensions of the module itself on the routing of the interconnects.

The first observation is that the center of the connected module pins is no longer situated on one side of the module, as there are pins on both sides of the module connected to the I/O pins. Thus, in general, the minimal distance between two module centers depends on the shape of both modules. The second observation is that, due to the larger distance from the I/O pins to the inputs on the opposite side of the module, the router uses the hex lines even if the module is next to the I/O pins (i.e. at the shortest distance). As a consequence, we consider that the power of the interconnects for the minimal distance depends entirely on the shape of the modules. This power is modelled as follows.

As the module pins are situated on the boundaries of the module, the distance between the module pin centers d , does not correspond exactly to the length of the interconnects. For example, consider the interconnect marked as *int1* in Fig. 3.8, that goes from one input pin to the corresponding I/O pin.

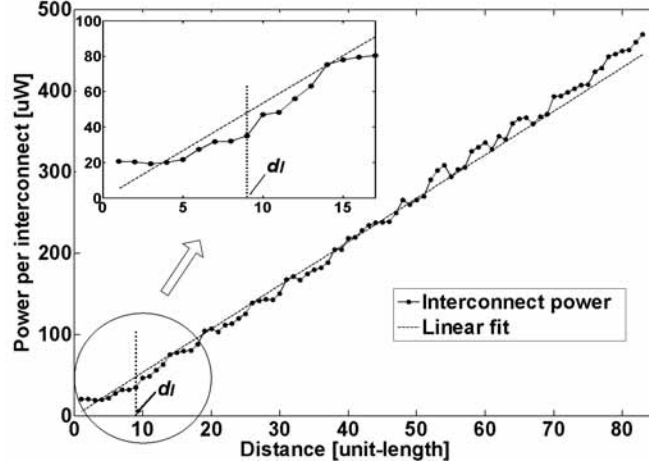


Figure 3.9: Power per interconnect between the module A and the I/O pins

It can be seen that:

$$l_{io}^{in} + d + l_{mod}^{in} = int_1 \quad (3.4)$$

where l_{mod}^{in} is the Manhattan distance from the module pin center to one input pin, and l_{io}^{in} is the distance between the I/O pin center and the I/O pin which is connected to the given input pin.

As a result, we model the limitations that occur due to the module shape and size by parameter L . For each module (I/O pin group), L_{mod} (L_{io}) is computed as the sum of local routes from the module pin center to the module pins that are connected to the other module. Consequently, the parameter L is obtained as follows:

$$\begin{aligned} L &= L_{io} + L_{mod} \\ L_{io} &= \frac{\sum_{i=1}^{I_1+I_2} l_{io,i}^{in} + \sum_{j=1}^O l_{io,j}^{out}}{I_1+I_2+O} \\ L_{mod} &= \frac{\sum_{i=1}^{I_1+I_2} l_{mod,i}^{in} + \sum_{j=1}^O l_{mod,j}^{out}}{I_1+I_2+O} \end{aligned} \quad (3.5)$$

where $l_{mod,i}^{in}$, $l_{mod,j}^{out}$ are the distances from the module pin center to the input pin i and output module pin j , respectively, and $l_{io,i}^{in}$, $l_{io,j}^{out}$ are the Manhattan distances from the I/O pin center to the input pin i and output I/O pin j , respectively.

The example presented in Fig. 3.8 describes the situation when the upper pins of the module are connected to I/O pins in the bottom part, and vice versa. Although, it does not correspond exactly to the situation where the module and the I/O pins are connected from top to bottom in consecutive order, we choose this scenario to obtain L , the parameter that models the interconnection length corresponding to the minimal distance, as it simplifies further computations.

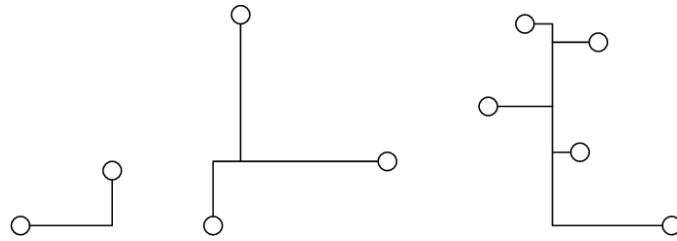


Figure 3.10: Rectilinear Steiner trees

In order to prove its applicability, we have replaced L with the real interconnect length for minimal distance between the modules, computed separately for each possible scenario used in the experiments, and there has been no significant impact on the results.

It should be emphasized that, as the wire length at the minimal distance d_m is accounted for through parameter L , the distance d_m has to be subtracted from the total distance in the other two routing regions (see equation 3.3).

In Fig. 3.9 the measured power per interconnect between the module A and the I/O pins is plotted versus the distance between them. Once again, the area around smallest distances is zoomed in the upper left corner of the figure. It can be seen that, again, the linear fit overestimates the interconnect power below the specific distance d_l , because of the absence of the long lines. However, when approaching to the smallest distances the power per interconnect is underestimated, due to the shape of the module.

The exact value of the d_l is obtained empirically and will be reported in section 3.5.

3.4. High-level Multi-point Interconnect Power Model

So far, we have considered only connections between a pair of modules. However, it is more often the case that the output of one module is connected to several inputs belonging to other modules. In this section, we present a multi-point interconnect power model.

As the goal of global routing is to connect the pins of a signal net using minimal wirelength, the connection pattern for each net can be represented as a Rectilinear Steiner Tree (RST). RST is defined as a tree with minimum total edge length in Manhattan distance that connects a given set of nodes possibly through some extra (i.e. Steiner) nodes [CW08]. Some examples of rectilinear Steiner trees are presented in Fig. 3.10. In this situation we have modified the power model for point-to-point connections to consider multi-point connections in the following way.

First, distance d is calculated as the length of the RST, with nodes defined as the module pin centers. Although, the problem of finding the RST is NP-complete [GJ77] and is often computationally very expensive, it is important to note that, we do not apply this algorithm on a pin-to-pin basis. Instead, the Steiner tree connects the module pin centers, so the algorithm does not depend on the word-length of the module's operands, but only on the number of con-

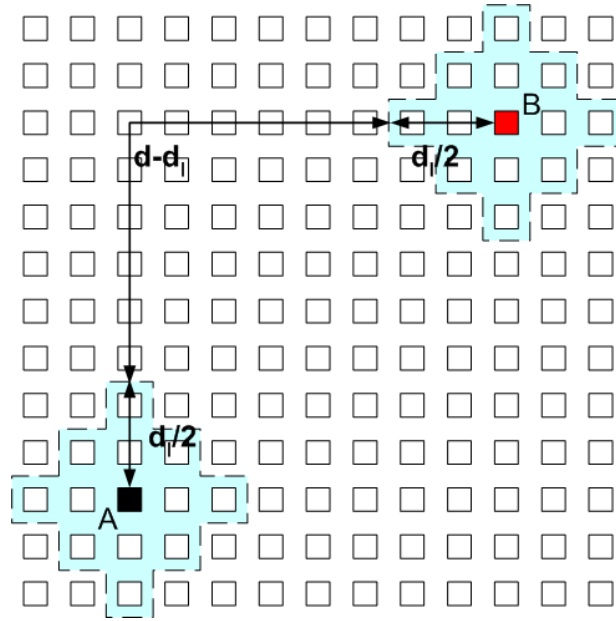


Figure 3.11: Second routing zone divided equally between the modules

nected modules. As this number is relatively small compared to the number of routed nets, the computation time to obtain the RST is highly reduced.

Second, we assume that the second routing zone applies to the lines in the proximity of both modules and thus, it can be divided into two parts, each part belonging to the proximity of one module (see Fig. 3.11: the simplified modules A and B are separated by the distance d). For the distances d greater than d_l , these two parts are separated by the remaining distance $d - d_l$, while for the smaller distances, there is no physical separation limit. Furthermore, we assume that the parts are equivalent, meaning that the zone is equally shared by the both modules. In other words, half of the zone belongs to one module, and the other half to the other (see Fig. 3.11: the second routing zone is equally distributed between the modules). When n modules are considered, the zone is also equally shared by them. Thus, for n modules we obtain a new distance limit d_l^{RST} as follows:

$$d_l^{RST} = n \cdot \frac{d_l}{2} \quad (3.6)$$

Third, the parameter L in the model is computed as the sum of the L_i values, obtained for the n modules using equation 3.5, as all modules are subject to short distance lengths.

Finally, equation 3.3 is applied in order to obtain the power estimate.

It can be seen that high-level point-to-point interconnection model (HLPM) and high-level multi-point interconnection model (HLMM) are actually the same, as HLPM is obtained from HLMM when n is equal to two. However, in the next section they will be first considered separately, as HLPM has been used for the model calibration, and HLMM has been used for its

evaluation. At the end of the next section as well as in the following chapters, both models will be unified and referred to as high-level interconnection model (HLIM). HLIM was presented in [JCP09].

3.5. Experimental results

As previously mentioned, the interconnect power model needs two parameters: the distance between the modules (i.e. the length of the RST), and the ordering of the pins on the component boundaries (in order to compute parameter L). We have built a library containing parameter L for the arithmetic modules. For the components not belonging to this library, L has to be computed from the component's layout. The length of the RST was obtained by using Geosteiner [Geo], a free software for computing Steiner Trees. The coordinates of the module pin centers that are needed for the RST computation were obtained from the floorplans of the placed designs. As the model uses only these coordinates and does not require any other placement information, it can be easily integrated into power optimization techniques that perform high-level synthesis combined with floorplanning. In these cases, the accuracy of the model will depend on the accuracy of the floorplan estimate.

The experiments are divided into three sets. The first one considers the connections between all combinations of two different modules; an adder and a multiplier. This is the characterization set used to obtain the coefficients k_i of the proposed model. Beside the connections between two modules, we consider the connections between a module and the I/O pins separately, as they have some special properties due to different router bounds. Furthermore, the amount of capacitance variation is determined in order to characterize the error intrinsic to the model. This analysis is based on moving one of the modules in only one direction.

The second set of experiments focuses on the accuracy of the model when more than two modules are connected. The coefficients k_{mod} , obtained from the experiments considering two modules, are used here in order to obtain power estimates.

The third set of experiments was designed in order to explore the accuracy of the power estimates for various DSP circuits, with different input signal statistics and module positions on a chip. Beside the estimates obtained by the proposed model, these experiments also include the error of XPower with respect to the reference power values.

Reference power values for all experiments are computed by using the effective capacitance values and the tool MARWEL after place-and-route, since we can not obtain the interconnect power by directly measuring power on-board. In this chapter, reference power values will be referred to as measured power values. The accuracy of this approach is evaluated as a part of the third set of experiments.

All experiments were performed in the bottom half of the FPGA chip, in order to avoid

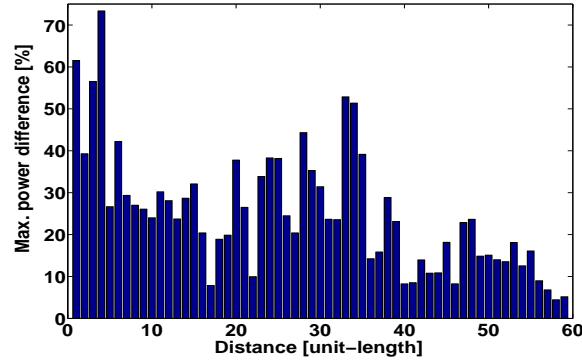


Figure 3.12: Maximum power difference for different placements

the impact of the Power PCs on global routing, as the work presented here refers only to the standard FPGA architecture.

3.5.1. Point-to-point interconnect model evaluation

It has been already mentioned that, in [AN04a] it was shown that changing the order of nets before routing the exact same design, can lead on average up to 20% of power variations, which are caused by the variation of the interconnect capacitance. This capacitance noise occurs due to the trade-offs that the router makes between the FPGA resources allocated to each net, which are resolved arbitrarily in different cases.

Thus, we divide the HLPM evaluation into two parts. First, we evaluate the model under the capacitance noise by considering several different placements for each distance between the modules. Second, we focus on determining the amount of capacitance variation in order to characterize the error intrinsic to the model.

HLPM evaluation under capacitance noise

The critical distance d_l was obtained empirically through MARWEL for all the combinations of the two modules, and a unique value of d_l was used no matter which of the two modules were connected. Furthermore, we also obtained the value of d_l when the module was connected to the I/O pins. In this case, d_l depended on the type and the size of the module that was connected to the I/O pins, but the difference between d_l and the minimal distance between the modules, d_m , was constant for all cases. The following equation describes the empirical relationship between the minimal distance between the modules d_m and the critical distance d_l :

$$d_l = d_m + 8 \quad (3.7)$$

Once we have d_m , d_l and L , there are three unknown coefficients left to complete the power

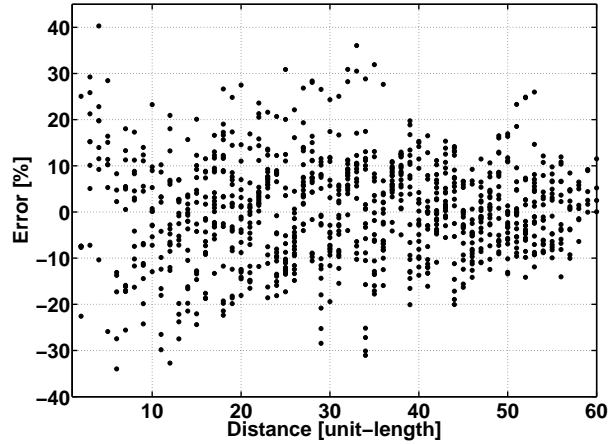


Figure 3.13: HLPM performance when applied to the connections between modules A and B

Table 3.2: Coefficient values for connections between modules, k^{mod} .

k_1^{mod}	k_2^{mod}	k_3^{mod}
8.5502	5.0511	-1.1889

model: k_1 , k_2 and k_3 . Their computation is explained next.

In order to account for the interconnect capacitance noise, five different placements were generated for each distance between the modules. Fig. 3.12 shows the maximum relative difference between the measured power values obtained at each distance, computed as:

$$\delta = \frac{(P_{max} - P_{min})}{P_{min}} (\%) \quad (3.8)$$

where P_{max} and P_{min} are respectively, the maximum and the minimum of five power values obtained for different placements at the same distance. For the sake of clarity, the figure is plotted only for the connections from an adder to a multiplier. The results were practically the same when other combinations of modules were considered. It can be seen that, as the distance grows, the difference tends to be smaller, given that there is no congestion in the circuit. The average relative difference was found to be 25%, so the results are consistent with the variations reported in [AN04a]. It is important to note that the figure presents maximum of all possible errors. If the relative difference between the measured power values was to be computed by using the average power P_{avg} (computed as the mean of five power values) instead of P_{min} , the difference would be smaller. Consequently, if the model is calibrated with the average power, we do not expect such high power difference to occur between the power predicted by the model and the actual power corresponding to some particular placement.

Thus, for each distance, a mean power value was computed. Finally, the coefficients k_i were obtained by using multivariable regression over the mean power values for various distances.

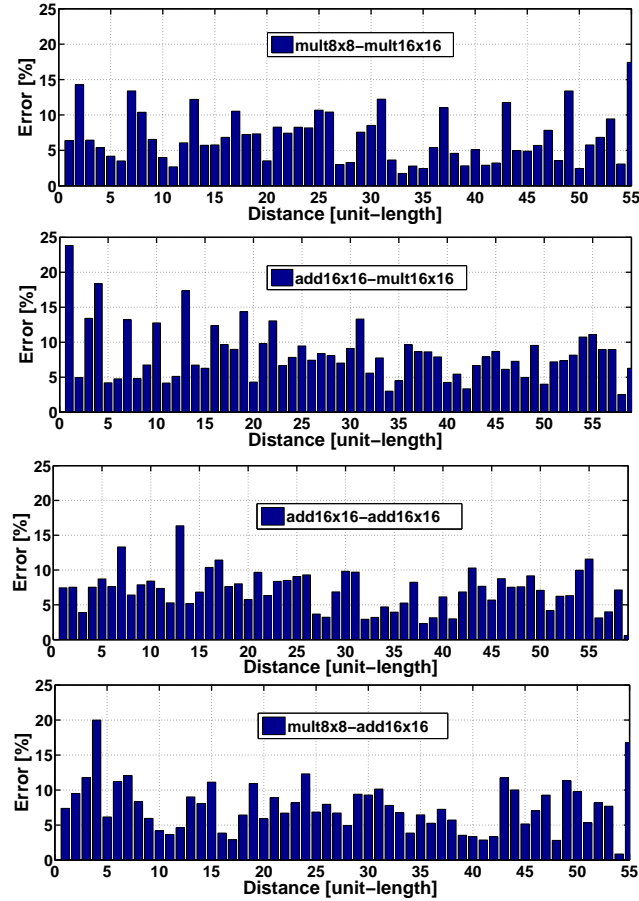


Figure 3.14: HLP average relative error when applied to the connections between modules A and B

Their values are given in Table 3.2. They are marked with a superscript *mod* in order to differentiate them from the coefficients that will be computed for the connections from/to I/O pins as it will be explained later.

Fig. 3.13 shows the relative errors for all the placements in the characterization set versus the distance between the two modules. The estimates obtained after placement, are compared to the reference power values computed by using MARWEL after place-and-route and the effective capacitance values. It can be seen that, in most cases, the error lies in the range $[-20\%, +20\%]$, with an absolute maximum error of 40%. In Fig. 3.14 we have plotted the average error in power estimate versus the distance between the two modules. It was computed by averaging the absolute values of estimation errors obtained for the five different placements at each distance. It can be seen that the error of the model is always smaller than 25% with most cases below 10%.

Furthermore, the largest average errors are obtained for the smallest distances, meaning that their impact on the error of the absolute total power estimate of the design would be very small, as the shortest interconnects represent a small portion of the total power.

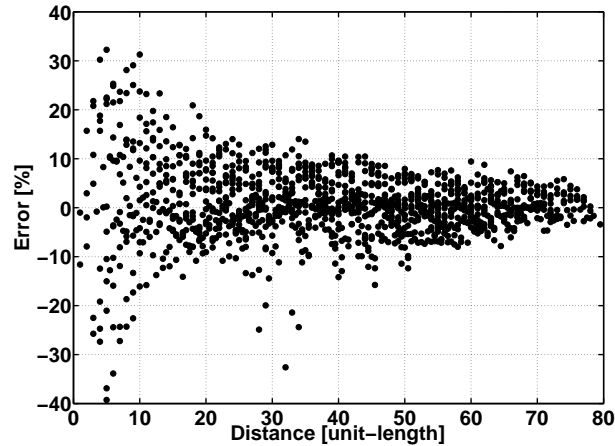


Figure 3.15: HLPM performance when unique set of coefficients is applied to the connections between the module and I/O pins

Table 3.3: Coefficient values for connections between modules and I/O pins, k_i^{io} .

k_1^{io}	k_2^{io}	k_3^{io}
5.852	2.6024	3.9014

Another important observation is that the coefficients k_1^{mod} , k_2^{mod} and k_3^{mod} obtained for the different combinations of the two modules were practically identical. This means that the same three coefficients can be applied to any combination of modules. It was confirmed that the errors for HLPM when the same coefficients are applied to all combinations of modules were practically the same as in Fig. 3.13 and Fig. 3.14, except for the smallest distances, where they raised between 5% and 10%.

Apart from the connections between two modules, we have also considered the connections between a module and the I/O pins. The experiments were performed on four different modules: 16×16 adder, 8×8 adder, 12×12 multiplier and 8×8 multiplier. The relative error for all combinations of modules is plotted in Fig. 3.15. It can be seen that the variation in power between different placements is much smaller than the power variance for connections between two modules. We believe that this effect occurs because the router uses much tighter bounds when routing the connections from or to I/O pins, compared to the routing inside the chip core.

Another observation from the experiments, that leads to the same conclusion about the different routing bounds, comes from the comparison of the k_i values obtained when connecting two modules (k_i^{mod}) and the coefficients obtained when the module is connected to the I/O pins (k_i^{io} , shown in Table 3.3). If we apply k_1^{io} , k_2^{io} and k_3^{io} to estimate the connections between two modules (i.e. instead of k_1^{mod} , k_2^{mod} and k_3^{mod}), there will be significant underestimation errors. It seems that, indeed, the router tool has tighter bounds when routing connections from or to the I/O pins, compared to the routing inside the chip core. This is probably due to the timing of the

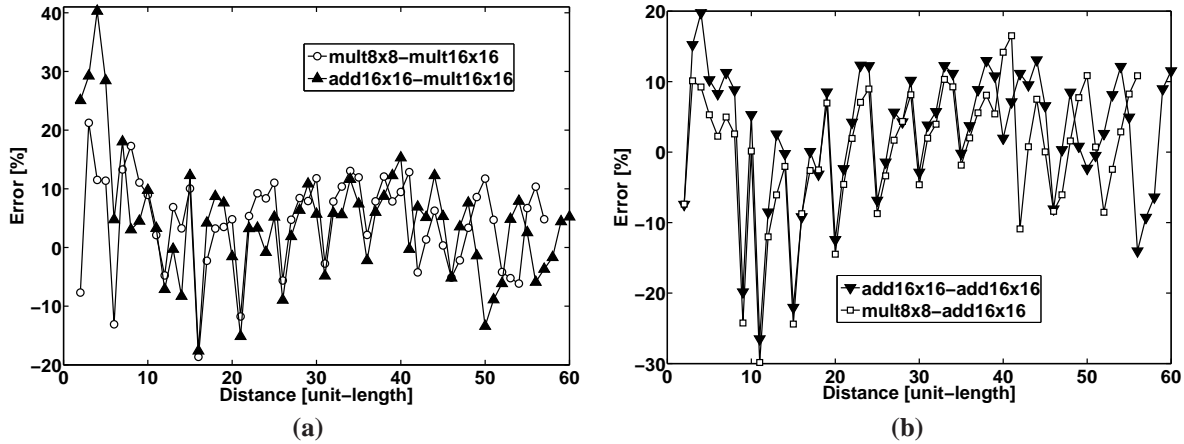


Figure 3.16: Errors for the HLPM when applied to the connections between the modules A and B by varying their distance in only one direction

signals coming from outside the board which depends on the elements connected to the board and can not be controlled by the chip itself, while the signals inside the chip have known timing properties that the router can adapt to.

The values of the first two coefficients, k_1^{mod} and k_2^{mod} , are clearly higher than the values of k_1^{io} and k_2^{io} . However, the third coefficient is smaller. This is due to the shape of the module and the influence of the direct wires used for the minimal distance routing. When two modules are separated by the unit-distance, they are completely aligned, and thus, all the lines can be routed as direct lines. The capacitance of the direct wires is close to zero, which leads to the small value for the coefficient k_3^{mod} . However, in the case of the interconnections between a module and the I/O pins, many lines are routed around the module, leading to a higher number of double and hex lines, which results in a higher value for the coefficient k_3^{io} .

As a conclusion, we have two sets of coefficients used for the interconnection estimation. One is to be applied when the connections lie inside the core of the chip, and the other, when connections to or from the I/O pins are considered.

Capacitance noise exploration

Although the maximum variation of the capacitance registered in our experiments goes up to 70%, we expect it to be significantly smaller when limiting the module's placement along one direction. This can enable us to eliminate most of the variations created by the router, and thus, determine the errors intrinsic to the presented estimation model. For this reason, we have varied the position of one module first along the x-axis, and after reaching the other side of the FPGA, along the y-axis, while the position of the other module remained fixed.

Fig. 3.16 shows the error performance of HLPM for this experiment. There are four different combinations of the modules. We have separated these combinations into two groups, one

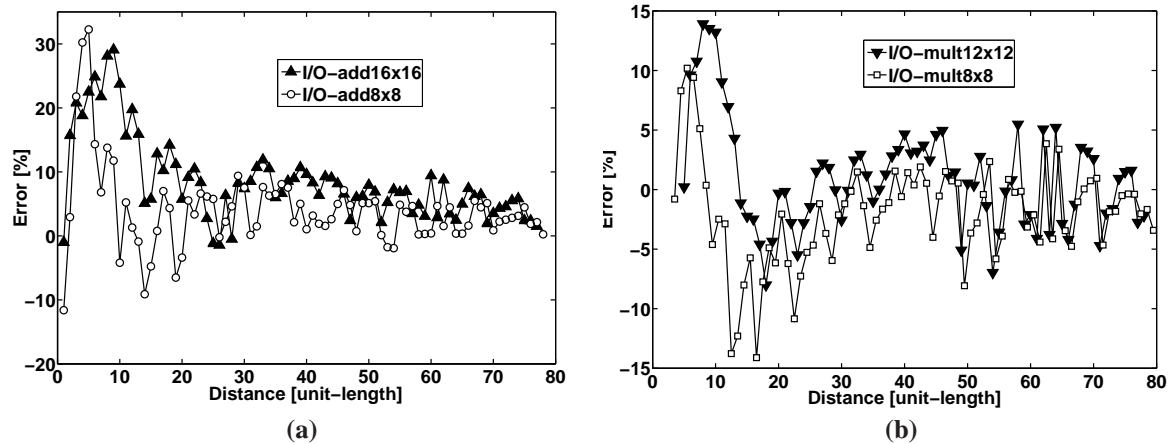


Figure 3.17: Errors for the HLPM when applied to the connections between the module A and I/O pins by varying their distance in only one direction

where the destination module was the multiplier, and the other group, where the destination module was the adder.

First, as expected, errors do not display such a great variability in the power values for the neighbouring distances as in Fig. 3.12, and most of them lie in the range $[-15\%, +15\%]$. This error variation can be considered to be the intrinsic error of the model.

Second, the error behaviour is very similar for the combinations of modules which share the same destination module, with peaks and valleys occurring at the same distances. The reason for this is that they have the same routing structure and resource occupation around the destination pins.

Third, all the minimum values (which are underestimates as they relate to negative errors), exactly correspond to the power peaks that were noticed in Fig. 3.5 and were explained by the limited connectivity of the switch matrices used for routing.

Finally, Fig. 3.17 shows the error performance when a module is connected to the I/O pins. In this case, there is no destination module, as some connections go from the I/O pins and some toward them. Thus, the error behaviours, although have some similarities for the same type of the modules, do not follow the same pattern.

3.5.2. Multi-point interconnect power model evaluation

We consider two different designs. One is composed of three modules, where the output of one module is connected to the inputs of the other two modules. The other is composed of five modules where, again, the output of one module is connected to the inputs of the other four modules. The positions of the source module and all except one destination module are fixed, while the position of the remaining destination module is varied throughout the chip. The coefficients k_1^{mod} , k_2^{mod} and k_3^{mod} , obtained from the experiments in the previous subsection, are

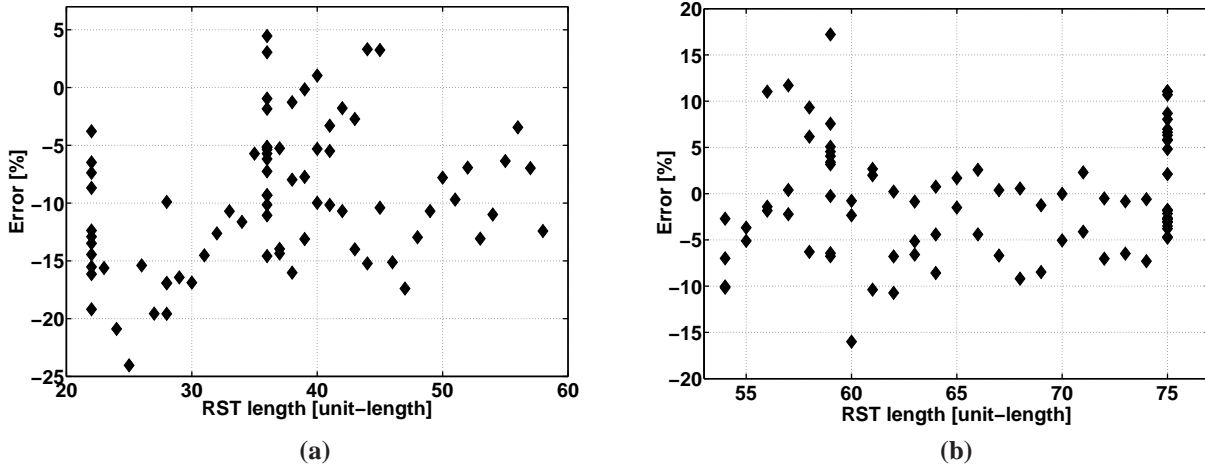


Figure 3.18: Errors for the HLMP when applied to a) 3-module design, and b) 5-module design

used here in order to obtain power estimates. Fig. 3.18 presents relative errors of the multi-point interconnect power model compared to the reference power values computed by using MARWEL and the effective capacitance values.

It can be seen that, for both designs, the model provides very good estimates. For some RST lengths, there are a lot of different positions which result in the same interconnect length (for example, lengths 22 and 36 in Fig. 3.18a and lengths 59 and 75 in Fig. 3.18b). This effect occurs because the Steiner trees corresponding to these positions, belong to the set of equivalent trees [Hwa76]. The equivalent trees are obtained by shifting a branch of the tree that contains a node (i.e. a module) between two parallel lines, resulting in an unchanged RST length.

The limited connectivity of the switch matrices creates congestion, as the number of interconnections is bigger than in the experiments for point-to-point model evaluation. This is probably the reason for the larger number of HLMP underestimates than overestimates. The analysis and modelling of the congestion in the interconnects is planned as part of our future work.

3.5.3. Model evaluation for DSP test designs

The third set of experiments was designed to test DSP circuits. The evaluation set consists of four DSP designs: three of them implement different arithmetic expressions and the fourth one is *CORDIC*, a design taken from [Ope] representing industrial application. The DSP designs implement the following functions:

$$\begin{aligned}
 DSP_1 &= (x_1x_2 + 1)x_3x_4 + (256x_1 + x_2) \\
 DSP_2 &= ((x_1 + x_2)(x_3 + x_4) + x_1x_2)x_2(x_3 + x_4) \\
 DSP_3 &= (x_2x_3)x_2 + (x_1 + x_3)x_2
 \end{aligned} \tag{3.9}$$

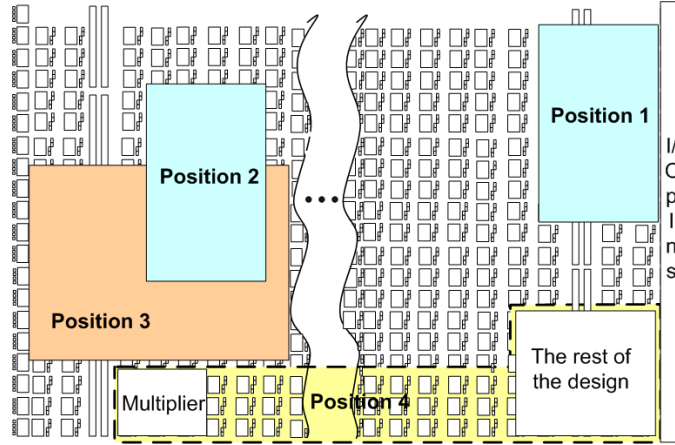


Figure 3.19: Design positions for DSP_1

In this analysis, we have included the switching activity obtained from bit-level DFG simulations. For each net, the switching activity was computed separately as the average number of transitions during one clock cycle, and then it was summed over all bits in a signal word in order to obtain the total switching activity of the particular connection between the modules. The final power estimate is computed as:

$$P_{total}^{int} = P_{int} \cdot \sum sw_i \quad (3.10)$$

For example, for the 16-bit connections between two multipliers, first the power per interconnect was computed by using 3.3, and then it was multiplied by the sum of the switching activities of all 16 bits as to obtain total interconnect power estimate.

Table 3.4 shows the results for each benchmark when data with different autocorrelation coefficients, ρ , are applied to its inputs. Both point-to-point and multi-point interconnection models are marked as HLIM, as they actually present the same interconnection power model. The table also includes the number of slices and embedded multipliers used by each design. The results for DSP_1 are obtained for four different placements (see Fig. 3.19).

The first placement (i.e. Position 1 in Fig. 3.19) is achieved without using any area constraints. For the second one (i.e. Position 2), the relative positions of the modules are kept as in the first placement, but all the modules are placed far from the I/O pins. Third, in Position 3, a bounding box with the size of a quarter of the FPGA surface is applied as an area constraint, and it is placed on the opposite side of the pins. In the fourth position (i.e. Position 4), an area constraint for only one of the multipliers is created by placing it far from the I/O pins and the rest of the design.

Evaluating power in four different positions also enabled us to confirm that the interconnect power values computed using MARWEL and the effective capacitances could serve as a fair substitute for direct power measurements. After measuring the dynamic power consumption of

Table 3.4: Relative errors for the proposed model (HLIM) and XPower (XP), for different auto-correlation coefficients.

Benchmark	Slices	Emb. Mult.	Position	Comp. time [s]	ρ	Er(HLIM) [%]	Er(XP) [%]
<i>DSP1</i>	290	0	1	1.08	0	-19.9	174.45
					0.9	-19.4	167.56
					0.99	-20.3	164.21
					0.9995	-22.54	157.06
			2	1.02	0	-5.97	38.48
					0.9	-5.55	35.47
					0.99	-4.75	36.25
					0.9995	-3.62	39.48
			3	1.14	0	-1.7	38.95
					0.9	-1.07	33.84
					0.99	-0.37	32.71
					0.9995	2.86	33.80
			4	0.83	0	6.51	87.27
					0.9	8.06	87.16
					0.99	10.59	89.33
					0.9995	15.3	99.97
<i>DSP2</i>	192	2	-	1.1	0	-8.09	258.45
					0.9	-13.63	233.50
					0.99	-18.73	216.23
					0.9995	-5.6	245.27
<i>DSP3</i>	212	2	-	0.92	0	6.32	328.79
					0.9	4.5	316.48
					0.99	-1.93	281.70
					0.9995	-9	246.91
<i>CORDIC</i>	591	0	-	0.3	NA	-9.22	NA

the design in the four positions, for each design position, we subtracted the computed interconnect power from the measured dynamic power. The results, which represent the logic power, should be the same in all four positions. Indeed, the maximum relative difference between these logic power values was found to be 2.05%.

Computation times for each benchmark when HLIM is applied, are listed in the next column of Table 3.4. The experiments were performed on a Pentium 4, at 3.00 GHz with 1GB of RAM. It can be seen that all the times lie around one second, and therefore satisfy the short timing demands imposed by the high-level phase of design flow. It is also important to note that DFG simulations are responsible for more than 50% of the computation time. They were not performed for benchmark *CORDIC* and consequently, the time needed for its interconnect estimation was more than two times smaller when compared to the rest of the circuits.

Beside the estimates obtained from the proposed model, Table 3.4 also includes the error of XPower obtained with respect to the reference power values, as described next. We have

used the advanced power reports provided by latest available version of XPower (from ISE 10.1), and the same design flow as presented in Fig. 2.7. First, we ran the Modelsim gate-level timing simulation of the placed-and-routed design, and as a result obtained a VCD file. This file contains detailed information on the toggling rates and frequencies of all the signals in the design, and it was used as the input simulation file for XPower. We were unable to use the new tool, XPower Analyzer, because the power values are displayed in milliwatts for now, and as such, it can only be used for large designs where this precision does not have a significant impact on the accuracy.

The information about the power of each individual element is listed in the XPower advanced report. We generated a script that parses the XPower report and extracts the information on power of the global nets in the circuit. Then, the power values for all global nets were added to obtain the total interconnect power value. The only benchmark that was not evaluated by XPower is the one taken from [Ope], as the input data are unavailable to us. In this case, for the sake of comparison with the power model presented here, we have assumed 0.5 switching activity on all nets in the design.

The results are presented in Table 3.4. For the HLIM, we observed that the highest underestimates were reported for the designs where the modules were placed tightly next to each other, and thus, generated congestion in the routing lines. Still, the highest detected error was -22.54%, with almost all of the errors lying in the range $[-20\%, +20\%]$, thus, proving the applicability of the model in a power optimization process.

On the other hand, the XPower tool shows large overestimate errors. We believe that this is due to the fact that the static power reported by XPower is a constant for the Virtex II Pro device, and that the tool is calibrated to estimate the power of large designs. The power values for interconnects are higher than their real values in order to compensate for the increase in static power due to a higher temperature generated by the activity of the large designs. Consequently, it seems that XPower is aimed at coarse architecture optimization (order of watts), while the HLIM is also aimed at detailed architecture refinement (order of milliwatts).

3.6. Conclusions

A high-level approach to estimate power consumption of interconnections in FPGA designs has been presented in this chapter. The approach exploits the goal of the router tool to minimize wire delay, in order to determine the associated wire power. The model has been first applied to point-to-point connections. It only depends on the modules' distance and some information about the modules' shapes including the position of the pins on the modules' boundaries, and can be used for any module distance. Three different coefficients are needed for the model calibration and they are obtained from the measured power corresponding to the connections

between all combinations of two different module types. Once obtained, this set of coefficients remains unchanged and can be applied to any other connection between the modules. However, a different set of coefficients was obtained for the connections between modules and I/O pins due to tighter bounds imposed upon the router when routing the connections that come from or go to the outside of a chip. Consequently, these coefficients are smaller than the ones used for module connections.

The model was further extended in order to consider interconnect power between any given number of modules, by applying a Rectilinear Steiner Tree algorithm to compute the length of the nets, and by adapting the model parameters to the case of n connected modules.

Many different placements were applied in the experiments in order to account for the variability of the net capacitance due to the different router solutions (the variability is found to be approximately 20%). The results show that the accuracy of the models, in most cases, lies within 20% of the power measurements. The model performance has been explored over a wide range of input parameters, signal components and module positions on a chip. The accuracy of the model has also been verified through on-board measurements of some test DSP designs. The results are the most accurate reported so far in pre-place interconnect power estimation for FPGAs, and clearly suggest the applicability of the estimation model in high-level power optimization techniques combined with floorplanning.

3.6.1. Future work

The work presented here considers only modules with registered inputs and outputs, as is the case in pipelined designs. However, in non-pipelined designs, the amount of glitching can represent a high percentage of the total power. Our future work is oriented toward extending the models to include glitching effects. Besides, the results also suggested the importance of the routing congestion, since the minimum steiner tree underestimates the wire length in the cases with more than two modules. Finally, one of the goals of the future work is to develop an efficient floorplanning algorithm that will enable successful integration of the interconnection power model into high-level power optimization techniques, together with the estimation models for logic power that will be described in the following chapter.

CHAPTER 4

Power estimation models for logic

In this chapter, we present a novel high-level analytical approach to estimate logic power consumption of arithmetic components implemented in FPGAs in the presence of glitching and correlation. In particular, models of adders, multipliers implemented in LUTs and embedded multipliers are presented in detail. All the components are considered to have registered inputs and outputs. The proposed methodology is based on: 1) an analytical model for the switching activity of the component, and 2) a structural analysis of the FPGA implementation of the component (in this work we consider the component implementations found in Virtex-2 and Virtex-2 Pro devices). The complete model is parameterized in terms of complexity factors such as word-lengths and signal statistics of the operands. It also accounts for the glitching introduced by the component. Compared to the other power estimation methods, the number of circuit simulations needed for characterizing the power model of the component is highly reduced.

Although we use the same approach for all arithmetic components, there are some slight differences depending on whether the components are implemented in LUTs or in embedded blocks. The use of embedded multiplier blocks has become a norm in DSP applications, due to their high performance and low power consumption. They are not built from standard programmable FPGA fabric. Instead, their design corresponds to that of an ASIC, as they are specialized for some chosen arithmetic functions and optimized to achieve the highest performance. Since only the required transistors and routing resources are used for the implementation of embedded blocks, their power is reduced as well. However, as their implementation details in commercial FPGAs are not available to the users, and the power estimates given by the tested low-level tool are not accurate enough to validate high-level models, the work on power estimation of these blocks is very limited. Hence, the novel methodology presented here

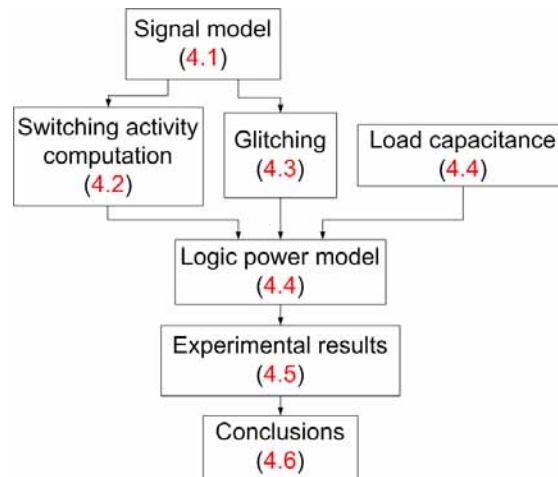


Figure 4.1: Chapter 4 organization

for dynamic power estimation in embedded multipliers is an adaptation of the power estimation method for LUT-based components, while it also takes into account the particularities of their implementation.

According to equation 1.2, four parameters needed for the power estimation: V_{dd} , f , SW and C_l . Two of them, V_{dd} and f , are fixed for some given FPGA architecture (V_{dd} is 1.5V for Virtex II and Virtex II Pro devices) and clock period of the design. Therefore, we need to know the other two parameters in order to obtain the power estimate. In this work, the load capacitance will be modelled relying on the special features of the DSP components implemented in FPGAs, while the switching activity will be analytically computed by applying a probability-based methodology.

The organization of this chapter is shown in Fig. 4.1. The presented approach for logic power belongs to word-level power models, so the input signals are modelled through their statistics. In section 4.1, we establish a signal model that can reflect the influence of the whole input data set on the dynamic power by using as few parameters as possible. In section 4.2, we present a methodology for switching activity computation, specially adapted to the propagation of the signal transitions throughout arithmetic components. Each different component structure is analyzed separately. In section 4.3, we include the model developed for the estimation of the glitching generated inside the components. Section 4.4 presents the logic power model based on the switching activity computation presented in section 4.2 and the glitch model presented in section 4.3. It also describes the approach we have used in order to model the load capacitance in DSP components. Experimental results that include the model accuracy exploration, the comparison with several other approaches in the literature, and the comparison with XPower estimates are presented in section 4.5. Finally, the conclusions are drawn in section 4.6.

4.1. Signal model

Power dissipation is a direct function of the number of the signal transitions occurring at the capacitive nodes under consideration. The switching activity and the transition probability are metrics used to determine the average number of signal transitions at a node per clock cycle. Input signals at each node determine the amount of its switching activity. Hence, the FPGA power consumption can vary significantly with the statistical distribution of the input data.

The switching activity depends on the present and immediately past value of the input signals. The most accurate method for the switching activity computation is the simulation of the placed-and-routed design with real input data vectors. However, as our final goal is power modelling at the highest levels of abstraction, it is necessary to establish a signal model with a small number of parameters, but still able to model the influence of the whole input data set on dynamic power. Instead of considering the switching activity analysis at the bit level, many power techniques use input word-level statistics in order to estimate the switching activity of a circuit. The latter approach has been also adopted in this work. We have chosen the word-level signal model where the input data sets are represented by their statistical parameters, such as mean, variance, autocorrelation coefficient etc.

Since we consider DSP components, the data signals can be assumed to be stationary and to have Gaussian distributions. This assumption can be considered to be valid as this distribution is often used for modelling DSP signals accurately [LR95, CGC05, JKSN99]. In subsection 4.1.1, we present a signal model for zero-mean gaussian distribution. This model has been extended in order to consider non-zero mean signals in subsection 4.1.2.

4.1.1. Zero-mean signals

First we will describe the gaussian signal generation model given in [CGC05], as it was later used for the computation of some important characteristics of the signal model used in this work, and also for investigating the effect of different input signal statistics on the component power. The signal generation model is capable of generating two signal input vectors with some chosen statistics as it is explained next.

The switching activity in the synchronous circuit is defined entirely by the present and immediately past value of the signal. Thus, for two-input arithmetic component with inputs x and y and assuming they are stationary with zero-mean gaussian distribution, the switching activity will depend on their joint probability density function (PDF) $p(x_0, x_1, y_0, y_1)$. The joint PDF for zero-mean multi-variate gaussian distribution is given by [Pap91]:

$$p(x) = \frac{1}{(2\pi)^{n/2} \cdot \det^{1/2}(C)} \cdot \exp\left[-\frac{1}{2} \cdot x^T \cdot C^{-1} \cdot x\right] \quad (4.1)$$

where $x = [x_1, x_2, \dots, x_n]$ is a signal vector and C is $n \times n$ symmetric matrix with $C_{i,j} = E\{x_i x_j\}$. For two input signals we obtain the following expression for their joint PDF:

$$p(x_0, x_1, y_0, y_1) = \frac{1}{(4\pi^2) \cdot \det^{1/2}(C)} \cdot \exp\left[-\frac{1}{2} \cdot [x_0 x_1 y_0 y_1] \cdot C^{-1} \cdot [x_0 x_1 y_0 y_1]^T\right] \quad (4.2)$$

where C is equal to:

$$C = \begin{bmatrix} r_{xx0} & r_{xx1} & r_{xy0} & r_{xy1} \\ r_{xx1} & r_{xx0} & r_{yx1} & r_{xy0} \\ r_{xy0} & r_{yx1} & r_{yy0} & r_{yy1} \\ r_{xy1} & r_{xy0} & r_{yy1} & r_{yy0} \end{bmatrix}. \quad (4.3)$$

Therefore, all the information required for the switching activity characterization is contained in the following seven statistical parameters:

- 1) r_{xx0} , the variance of signal x
- 2) r_{xx1} , the autocorrelation coefficient of signal x
- 3) r_{xy0} , the cross-correlation of the two signals
- 4) r_{xy1} , the cross-correlation of the two signals with unit-time lag in y
- 5) r_{yx1} , the cross-correlation of the two signals with unit-time lag in x
- 6) r_{yy0} , the variance of signal y
- 7) r_{yy1} , the autocorrelation coefficient of signal y

Two signals x and y with gaussian distribution and seven chosen statistical parameters can be generated from two temporally and spatially uncorrelated zero-mean gaussian signals u and v in the following way. First, the auxiliary signals p and q are formed as:

$$p = \beta \cdot u \quad (4.4)$$

$$q = \delta \cdot u + \gamma \cdot v \quad (4.5)$$

After that, the elements of the x and y array are obtained as follows:

$$x_i = p_i + a_1 \cdot x_{i-1} + a_3 \cdot y_{i-1} \quad (4.6)$$

$$y_i = q_i + a_2 \cdot x_{i-1} + a_4 \cdot y_{i-1} \quad (4.7)$$

where $a_1, a_2, a_3, a_4, \gamma, \delta$, and β are the coefficients obtained from the following expressions:

$$\begin{bmatrix} r_{xx1} \\ r_{xy1} \end{bmatrix} = \begin{bmatrix} r_{xx0} & r_{xy0} \\ r_{xy0} & r_{yy0} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_3 \end{bmatrix} \quad (4.8)$$

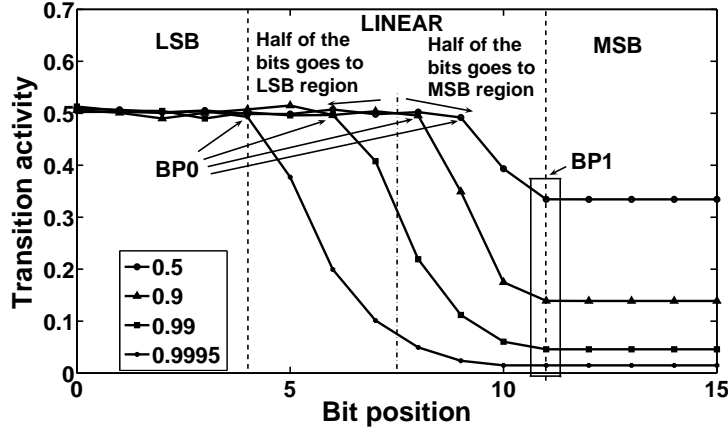


Figure 4.2: Bit transition activity vs. bit position in a word for zero-mean signals

$$\begin{bmatrix} r_{yx1} \\ r_{yy1} \end{bmatrix} = \begin{bmatrix} r_{xx0} & r_{xy0} \\ r_{xy0} & r_{yy0} \end{bmatrix} \cdot \begin{bmatrix} a_2 \\ a_4 \end{bmatrix} \quad (4.9)$$

$$r_{xx0} = r_{pp0} + a_1 \cdot r_{xx1} + a_3 \cdot r_{xy1} \quad (4.10)$$

$$r_{yy0} = r_{qq0} + a_2 \cdot r_{yx1} + a_4 \cdot r_{yy1} \quad (4.11)$$

$$r_{xy0} = r_{pq0} + a_1 \cdot r_{yx1} + a_3 \cdot r_{yy1} \quad (4.12)$$

$$r_{pp0} = \beta^2, r_{qq0} = \gamma^2 + \delta^2, r_{pq0} = \beta \cdot \delta \quad (4.13)$$

It has been shown that dynamic power consumption in arithmetic components is affected to a greater extent by autocorrelation than by cross-correlation [CGC05]. Therefore, we will consider only the effects of signals variances and autocorrelations on the power consumption models, thus reducing the number of statistical parameters needed for determining the power consumption to only four: r_{xx0} , r_{yy0} , r_{xx1} , r_{yy1} .

By using the expression 4.6, and assuming that the cross-correlation coefficients can be neglected, it can be seen that a signal with a gaussian distribution and some determined autocorrelation coefficient can be represented as:

$$x(n) = \beta \cdot u(n) + a_1 \cdot x(n-1) \quad (4.14)$$

We will use this expression in order to determine some important characteristics of the signal model used in this work.

Word-level signal model

In [LR95], the authors first noted that a signal word with zero-mean gaussian distribution can be divided into three regions according to its word-level signal statistics. In Fig. 4.2 we have plotted the bit transition activity in signal word versus their bit position in the word for different

autocorrelations (all the signals have a Gaussian distribution of the same variance σ^2). The three regions that can be identified are: 1) LSB uncorrelated bits (in Fig. 4.2 they are marked for $\rho = 0.9995$), 2) linear region with correlated data bits and 3) MSB sign bits. The uncorrelated data bits have a fixed transition activity of 0.5 and include the bits from the least significant bit (LSB) up to a certain breakpoint BP0. The highly correlated bits on MSB positions are placed from the most significant bit (MSB) to another breakpoint BP1. A linear model is then employed for the switching activity of the data bits, which lie between the MSB bits and the uncorrelated data bits. Equations defining BP0 and BP1 are presented in terms of word-level statistics.

We have chosen this method (known as dual-bit type method, DBT) in order to model the input signal words. DBT method has been extended and improved in [RSH97] and [SP00]. More accurate expressions for breakpoints BP0 and BP1 which separate the LSB from the linear, and the linear from the MSB region respectively, are given. The signals under consideration in their work correspond to autoregressive moving-average (ARMA) signal models. As gaussian signals belong to this subclass of signals, it is possible to use the given expressions to obtain the breakpoints as it will be explained next. Additionally, instead of measuring the transition density of the MSB bits as in [LR95], it is estimated from the word-level statistics. In [RSH97] the transition densities at the outputs of an adder, a multiplier and a register are analytically computed from the input parameters, assuming a zero-delay model for these components.

Hence, instead of a word division into regions based upon the transition activity (as presented in [LR95]), we will divide it based upon their bit-level correlation ρ_i following the methodology described in [RSH97].

By definition $\rho_i = 0$ for $i < BP0$. It is considered that for the MSB sign bits the following assumption is valid : $\rho_{BP1} = \rho$, where ρ represents the word-level temporal correlation.

The expressions for the autocorrelation coefficient of all three regions are given by:

$$\rho_i = \begin{cases} 0 & i < BP0 \\ \frac{(i-BP0+1) \cdot \rho_{BP1}}{BP1-BP0} & BP0 \leq i < BP1 \\ \rho_{BP1} & i \geq BP1 \end{cases} \quad (4.15)$$

In order to calculate the exact transition activity t_i we use the following relationship between the bit-level probability p_i and the bit-level autocorrelation ρ_i ([RSH97]):

$$t_i = 2 \cdot p_i \cdot (1 - p_i) \cdot (1 - \rho_i) \quad (4.16)$$

The bit-level probability p_i can be calculated as:

$$p_i = \sum_{j \in \Psi} \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-(j-\mu)^2/2\sigma^2} \quad (4.17)$$

where Ψ is the set of all elements that a signal can take such that the i -th bit is 1.

Since the values of the signal lie between values $x_{max} = x + 3\sigma$ and $x_{min} = x - 3\sigma$, the signal dynamic range is equal to $x_{max} - x_{min}$ and $\log_2(x_{max} - x_{min})$ bits are required to represent this range [RSH97]. Hence, the final expression for breakpoint BP1 is given as follows:

$$BP1 = \lceil \log_2(6\sigma) \rceil \quad (4.18)$$

It is important to note, that in the expression for the bit-level autocorrelation coefficient 4.15, the MSB region starts from the bit on position $BP1$, meaning that this bit also belongs to the sign region. Thus, the signal variation is actually represented with the following number of bits:

$$BP1 - 1 = \lceil \log_2(3\sigma) \rceil \quad (4.19)$$

We will return to this expression when considering non-zero mean regional decomposition as it will be explained in the next subsection.

The expression derived for breakpoint BP0 depends on the coefficients of the ARMA (autoregressive moving average) signal model and is to be computed here for signals that have zero-mean Gaussian distributions.

An (N,M)-order ARMA model can be represented as

$$x(n) = \sum_{i=0}^N d_i \gamma(n-i) + \sum_{i=1}^M a_i x(n-i) \quad (4.20)$$

where the signal $\gamma(n)$ is a white (uncorrelated) noise source with zero mean, and $x(n)$ is the signal being generated. This model is an infinite-impulse response (IIR) filter with coefficients a_i and d_i , that takes as input zero-mean white noise. It is also possible to transform this IIR model into one that depends only on the inputs as shown below

$$x(n) = \sum_{i=0}^{\infty} h_i \gamma(n-i) \quad (4.21)$$

where h_i can be computed according to the following recursion:

$$h_k = d_k + \sum_{i=1}^N a_i h_{k-i} \quad (4.22)$$

where $h_k = 0$ for $k < 0$, and $h_0 = d_0$. The breakpoint BP0 for signal $x(n)$ presented in 4.21 is estimated as the maximum of the BP0's of the signals $h_i \gamma(n-i)$, as pointed out in [RSH97].

Hence,

$$BP0 = \lceil \log_2(h_{\max}\sigma_\gamma) \rceil \quad (4.23)$$

where $h_{\max} = \max(|h_i|)$. We will show the method for computing the maximum of ARMA-model coefficients in 4.23 when a signal has a zero-mean Gaussian distribution. Equations 4.21 and 4.22 have been used as our starting point in finding a relation between the signal statistics and the coefficients of the ARMA signal model. As already seen in the expression 4.14, any given signal with a zero-mean Gaussian distribution of variance σ^2 and autocorrelation coefficient ρ can be represented as an ARMA(0,1) model:

$$x(n) = d_0\gamma(n) + a_1x(n-1) \quad (4.24)$$

Computing the variance and autocorrelation of a signal represented as in 4.24 leads to a system of two equations:

$$\sigma_x^2 = E\{d_0\gamma(n)^2\} + E\{(a_1x(n-1))^2\} \quad (4.25)$$

$$\rho = \frac{E\{x(n)x(n-1)\}}{\sigma^2} = \frac{E\{d_0\gamma(n)x(n-1) + a_1x(n-1)^2\}}{\sigma^2} \quad (4.26)$$

For the sake of simplicity, equation 4.25 is presented here after the elimination of the terms equal to zero (due to the zero value of the means of both signals: $\gamma(n)$ and $x(n-1)$, and also due to their mutual independence: although the signals $\gamma(n)$ and $x(n)$ are mutually dependent, the signals $\gamma(n)$ and $x(n-1)$ are mutually independent). The coefficients d_0 and a_1 are then obtained by solving this system and the resulting expressions are:

$$\sqrt{(1-a_1^2)} \cdot \sigma_x = d_0 \cdot \sigma_\gamma \quad (4.27)$$

$$\rho = a_1 \quad (4.28)$$

Next, we have computed the coefficients for the ARMA(0,1) model. Combining 4.22 and 4.28 we obtain that

$$\begin{aligned} h_0 &= d_0 \\ h_1 &= a_1d_0 = \rho \cdot d_0 \\ h_2 &= a_1^2d_0 = \rho^2d_0 \\ &\dots \end{aligned} \quad (4.29)$$

As the autocorrelation coefficient is always less than or equal to one, we obtain that the maxi-

mum of all h_i is h_0 , i.e. d_0 . Finally, by replacing expressions 4.27 and 4.28 in 4.23, it becomes:

$$BP0 = \left\lceil \log_2(\sqrt{1 - \rho^2} \cdot \sigma_x) \right\rceil \quad (4.30)$$

We have used this expression for the breakpoint $BP0$ in our approach as it gives better results than the one proposed in [LR95]. The comparison of the two approaches will be shown in detail in section 4.5.

Finally, for the sake of simplicity, the linear region is divided in two equal parts, where the upper half of the bits is attributed to the MSB region, while the bottom half of the bits in linear region are attributed to the LSB region (see Fig. 4.2). This approximation simplifies the further computations, without introducing a significant error in the switching activity estimation. It has been also used in [LR95, CGCC06].

4.1.2. Non-zero mean signals

Many DSP signals take only positive values, as for example, non-compressed image signals, where each pixel is described with a positive integer. Consequently, we consider gaussian signals with mean μ , variance σ^2 and autocorrelation coefficient ρ .

In Fig. 4.3 we have plotted the bit transition activity in a signal word versus the bit position in the word for different autocorrelations and some chosen value for μ . It can be seen that the signal-word can be also divided into different activity regions as explained next.

The two breakpoints, $BP0$ and $BP1$, that divided a signal word for zero-mean signals are also present in the non-zero mean signal decomposition. Furthermore, for a zero-mean signal these two breakpoints are sufficient in order to account for the contribution of each of the regions to the total switching activity of the component. However, in the case of non-zero mean signals with a gaussian distribution, the previous signal model does not account for some important effects.

The region beyond breakpoint $BP1$ is transformed into two subregions with mean and sign bits (see Fig. 4.3). The activity of these regions is zero, regardless of the autocorrelation coefficient, but the values of these bits depend on the value and the sign of the mean respectively, and as such, have a great impact on power consumption. This effect is especially important when considering power consumption in multipliers. The MSB bit (sign bit) of one of the multiplier's inputs is extended when summing the partial products of the two operands. This means that if the mean is negative, this bit will be equal to logic '1' and it will cause the switching activity propagation of the other operand throughout the component. However, if the mean is positive (i.e. the MSB bit equal to '0'), the switching activity at the outputs of the corresponding elements will be zero regardless of the switching activity of the other operand. The same conclusion stands for the values of the rest of the mean bits. Hence, it is clear that the values of

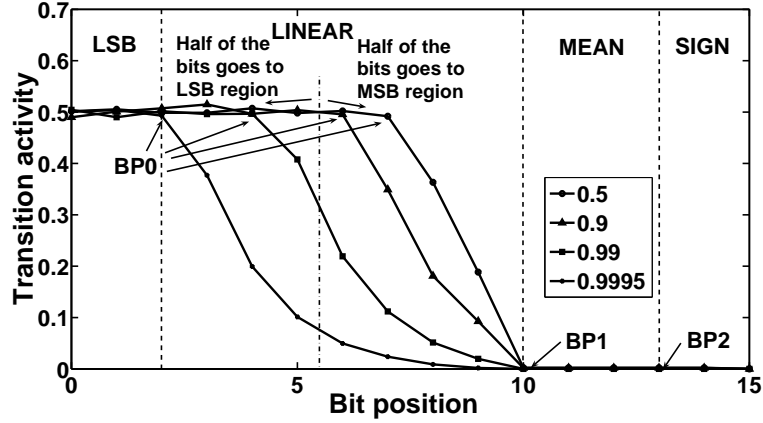


Figure 4.3: Bit transition activity vs. bit position in a word for non-zero mean signals

the mean and sign bits contribute significantly to the component's total power, although they do not exhibit any switching activity.

The values of the two breakpoints $BP0$ and $BP1$ are obtained as it was explained in the previous subsection. The new breakpoint, $BP2$, is obtained as follows. The maximum value of non-zero mean gaussian signals is $\mu + 3\sigma$ and $m = \log_2(\mu + 3\sigma)$ bits are needed for its binary representation. As previously mentioned, $v = \log_2(3\sigma)$ bits are needed for the representation of the signal variation around the mean (see the equation 4.19 in the previous subsection). Thus, the number of bits that are not changing in a data word and whose values correspond to the K upper bits of the mean are computed as follows:

$$K = m - v = \log_2(\mu + 3\sigma) - \log_2(3\sigma) \quad (4.31)$$

If a signal word has N bits, the number of bits in the sign region that are all taking value '0' or '1' depending on the sign of the mean, can be computed as:

$$S = N - m = N - \log_2(\mu + 3\sigma) \quad (4.32)$$

As a consequence, the MSB region of the signal-word when considering non-zero mean signals, is composed only of half of the bits belonging to the linear region (see Fig. 4.3). Hence, the third breakpoint $BP2$, which separates the mean region from the sign region, is calculated as follows:

$$BP2 = \lceil \log_2(\mu + 3\sigma) \rceil \quad (4.33)$$

The switching activities of the bits in each region are computed as in 4.16. The bit-level autocorrelation coefficient has a value 0 for the LSB bits, can be approximated by ρ for the

MSB bits, and has a value 1 for the mean and sign bits.

4.2. Switching activity computation

Once we have the switching activities of the input bits which are computed according to the signal model presented in the previous section, we can compute the switching activities of the component's internal nodes. In this section, a brief overview of the switching activity techniques will be given in 4.2.1, followed by the description of the methodology that has been used in this work in 4.2.2. Next, in 4.2.3, we will apply this methodology to all arithmetic component structures considered in this thesis.

4.2.1. Switching activity: background

Many accurate techniques for power estimation already exist at the logic and circuit levels. At these levels, the load capacitance is usually found from the gate-level information on the design. The main problem remains with the switching activity. The switching activity depends on the input data values and logic function of a gate, but also includes glitches, spurious activity produced by the different signal delays entering the same logic component.

The accuracy of the switching activity computation depends on the timing model that is chosen to model delays in a circuit. There are three different timing models: zero-delay, variable-delay and real-delay model.

The zero-delay model considers that a delay of any logic gate is equal to zero, and that the signals arriving at the inputs of the gate immediately produce the output signal. This model has the lowest accuracy, but it has two important advantages. First, it can be used at the higher levels of abstraction (i.e. pre-placement and pre-routing), since the delay of the interconnections is not taken into account. Second, glitching is not included in this model and thus, the switching activity falls into the range $[0,1]$ and can be modelled as a probability that a signal will change its logic state during one clock cycle. This allows for the probability methodology to be applied for the switching activity computation as it will be explained later.

The second timing model is the variable-delay model. In this model, the delays of the logic gates are taken into account, but the delays of the interconnections are not. The accuracy of this model is higher, while it can be still applied at the higher levels of abstraction. The main disadvantages of the model are that it is necessary to know the implementation and technology details of the circuit in order to be able to extract the timing parameters, the use of the probability-based methodology is not so straightforward any more, and the computation times increase significantly.

The third timing model is the real-delay timing model which takes into account all the delays in the circuit. It can be only used after the routing is completed, as this phase is necessary in

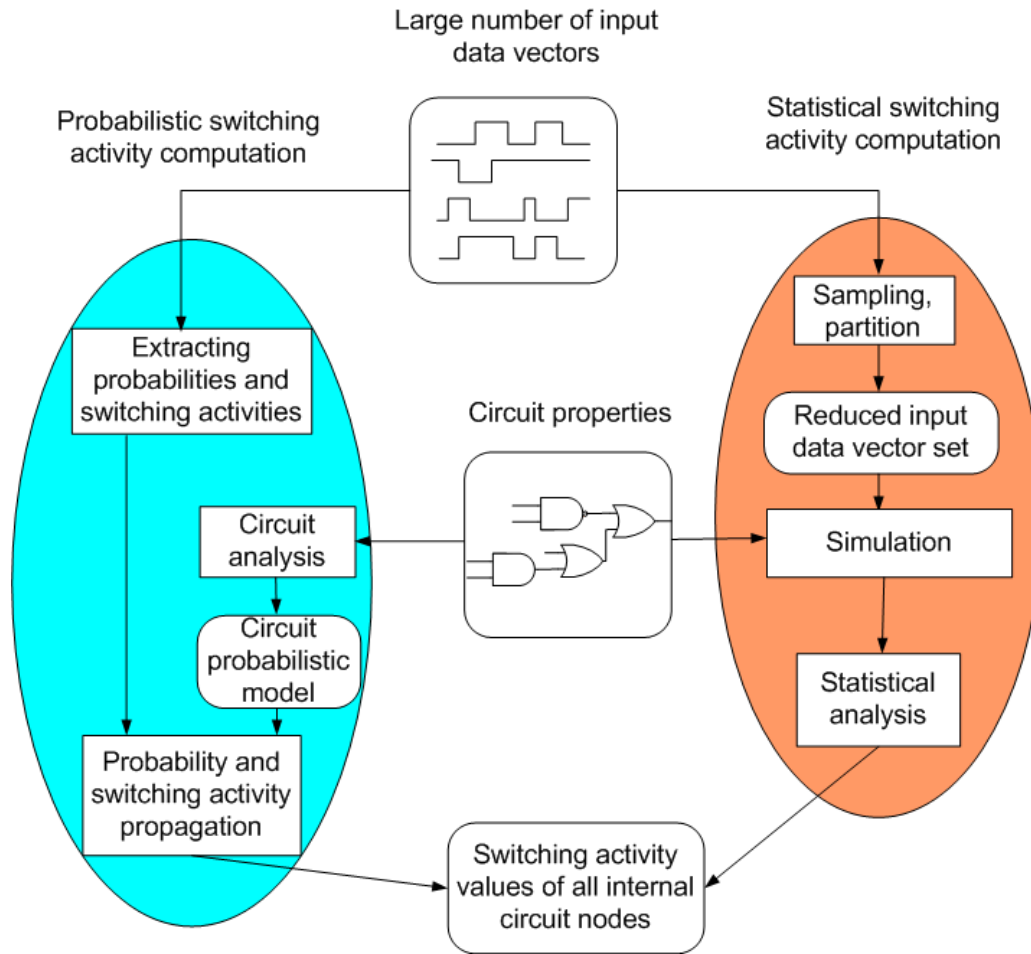



Figure 4.4: Statistical and probabilistic switching activity computation

order to extract the delays of the interconnect wires. The application of this model is hardly possible in large designs, as the computation times become extremely long.

There are two different approaches that try to address the problem of estimating the switching activity propagation: statistical and probabilistic (see Fig. 4.4 [MS08]). The statistical approaches simulate the circuit with input vectors and collect the statistical data for each node in the circuit. The problem lies in very long execution times for large circuits and large input data sets. On the other hand, probabilistic methods analyze the circuit and generate the expressions for the switching activity propagated throughout the circuit. Hence, they do not depend on the number of the input data vectors, but only on their statistics. These methods also have problems when analyzing large circuits, as the complexity of the analytical expressions depends on the number of the inputs and logic depth of a circuit.

In [TGS⁺02, TB05], a statistical approach is used to estimate total and individual-node average power consumption for combinational FPGA circuits. They use detailed information on the placed-and-routed design, and by applying Monte-Carlo simulations, they monitor the activity per clock period for each node in the design when randomly generated patterns are applied to

Table 4.1: Probabilistic method for AND-gate


A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

A ₀	B ₀	A ₁	B ₁	C ₀	C ₁
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

$$P_{C=1} = P_{A=1} \cdot P_{B=1}$$

$$\begin{aligned}
 sw_C = & P(A_{0 \rightarrow 1}) \cdot P(B_{0 \rightarrow 1}) + P(A_{0 \rightarrow 1}) \cdot P(B_{1 \rightarrow 1}) \\
 & + P(A_{1 \rightarrow 1}) \cdot P(B_{0 \rightarrow 1}) + P(A_{1 \rightarrow 0}) \cdot P(B_{1 \rightarrow 0}) \\
 & + P(A_{1 \rightarrow 0}) \cdot P(B_{1 \rightarrow 1}) + P(A_{1 \rightarrow 1}) \cdot P(B_{1 \rightarrow 0})
 \end{aligned}$$

the inputs. Short glitches are filtered as to avoid overestimation and the user can specify the tolerated error and confidence level of the estimation. The number of the input vectors applied to the inputs depends on the wanted confidence level of the estimation: larger numbers of input vectors corresponds to higher confidence levels.

Probabilistic methods represent the switching activity as a probability that a signal will change its logic state during one clock-cycle. Apart from the switching activity, they also include a probability that a signal has a value of logic '0' or logic '1'. They represent each logic gate with an equation that describes the dependency of the switching activity and the probability of the output bit on the switching activities and the probabilities of the gate input bits. For example, for an AND gate (see Tab. 4.1), the output will have the value of '1' only if both inputs are also '1's. Thus, the probability of the output bit being '1' is equal to the product of the probabilities of both inputs being '1' if the inputs are independent. The computation of the switching activity is similar, except that now, we have to take into account two successive time moments. For all combinations of input bits, where the output has a different value in these two time moments, we say that a signal switched (the combinations are marked with yellow fields in Fig. 4.1). Thus, the switching activity of the output bit will depend on the probabilities of the input bits and also on their switching activities. We have chosen a probability method for switching activity computation in this work and it will be described in detail in the next subsection.

The methodology in [Naj91] is based on the propagation of probabilistic parameters from the primary inputs. Transition density, $D(x)$, corresponds to the average switching activity at

the circuit node, while considering zero-delay timing model. Based on an stochastic model of the signals, an algorithm is presented that propagates the transition density to internal and output nodes in the following way. Given a logic function $f(x_1, x_2, \dots, x_n)$, the dependence of the output of f on the input x_i is given through their boolean difference defined as:

$$\frac{\partial f}{\partial x_i} = f|_{x_i=0} \oplus f|_{x_i=1} \quad (4.34)$$

If this difference is equal to 1, then every transition on x will cause a transition on f . Hence, the final transition activity of f is the sum of the activities contributed by each of the inputs:

$$D(f) = \sum_{i=1}^n P\left(\frac{\partial f}{\partial x_i}\right) D(x_i) \quad (4.35)$$

However, the influence of the each input signal on the output density is taken into account separately from the rest of the input signals, so the simultaneous switching at the inputs of a logic gate is ignored during the propagation. This problem is solved in [CRP94] where a zero-delay model is used for logic gates and the switching activities at the outputs are computed through input signal probabilities and activities by using the probabilistic method, similar to the one that was used in this work. Since the exact calculation of signal probabilities is NP-hard [Naj91], in the case of very large circuits, a partitioning algorithm that limits the number of independent inputs to a module has to be employed. However, at the gate level independent signals are not so easy to find. In order to reduce the complexity and to accelerate the partitioning of the circuit, an RTL circuit partitioning approach based on disjoint signal detection has been proposed recently in [MRT08, MS08].

4.2.2. Methodology for switching activity computation

The basic cell of all DSP components is a full-adder cell (see Fig. 4.5). We will show the methodology for computing the switching activities at the outputs of the full-adder cell. In particular, the methodology employed for computing the switching probability of the carry bit will be presented here, as it is the most complex case. The calculation of the rest of the probabilities is obvious and only their expressions are provided. The list of notations used in the equations is given below:

- p is the transition probability of one of the inputs of the full-adder cell
- q is the transition probability of the other input of the full-adder cell
- c_{in} is the transition probability of the incoming carry bit
- c_{out} is the transition probability of the outgoing carry bit
- s is the transition probability of the output of the full-adder cell
- p^0 and p^1 are the probabilities of input p being '0' and '1' respectively

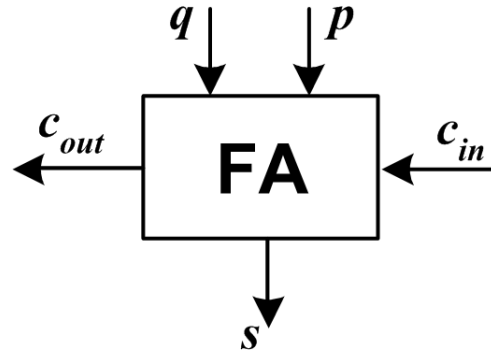


Figure 4.5: Full-adder cell

Time moment t_0					Time moment t_0+1				
p	q	C_{in}	C_{out}		p	q	C_{in}	C_{out}	
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
0	1	0	0		0	1	0	0	
0	1	1	1	←	0	1	1	1	Second case
1	0	0	0		1	0	0	0	
1	0	1	1	←	1	0	1	1	Second case
1	1	0	1	←	1	1	0	1	Fourth case
1	1	1	1	←	1	1	1	1	Third case

Figure 4.6: Probability methodology

- q^0 and q^1 are the probabilities of input q being '0' and '1' respectively
- c^0 and c^1 are the probabilities of the incoming carry bit being '0' and '1' respectively
- s^0 and s^1 are the probabilities of the output bit being '0' and '1' respectively

As mentioned before, the switching activity depends on two consecutive signal values. The output bits of the full-adder cell depend on three input signals. There are 2^3 combinations for the inputs in one clock cycle. For each combination, there are also 2^3 combinations for the inputs in the following clock cycle. For example, in Fig. 4.6 we assumed that the input signals at time t_0 are "001". At time, $t_0 + 1$, their values could be any of the eight combinations listed in the table of Fig. 4.6. However, only four combinations will produce a transition at the carry bit. The bits belonging to these combinations that have changed their value during two consecutive time moments are marked with green fields, while the bits that stayed the same are marked with yellow fields. Each of these combinations can occur with a certain probability. Thus, in general, we should consider a total of 64 combinations of input bits at two consecutive times, and look for the ones that cause the output bit to change. In order to reduce this number, we consider all the combinations for input bit values in one clock cycle, but divide the events which could occur in the following clock cycle into four possible cases. In the first case, neither of the inputs

changes. Hence, there will be no transition in the carry. In the second case, only one of the inputs makes a transition. In this case, for the combinations "000" and "111" at time $t_0 + 1$ there will be no change at the outgoing carry bit. For the combinations "001", "010" and "100", if any of the zeros changes, the transition will occur at the carry too. Hence, the transition activity for this case is:

$$\begin{aligned} c_{out}^{A1} = & p^0 \cdot q^0 \cdot c_{in}^1 \cdot (1 - c_{in}) \cdot (p + q - 2 \cdot p \cdot q) + \\ & p^1 \cdot q^0 \cdot c_{in}^0 \cdot (1 - p) \cdot (q + c_{in} - 2 \cdot q \cdot c_{in}) + \\ & p^0 \cdot q^1 \cdot c_{in}^0 \cdot (1 - q) \cdot (p + c_{in} - 2 \cdot c_{in} \cdot p) \end{aligned} \quad (4.36)$$

For the rest of the combinations "011", "110" and "101" a change in the any of the ones, will produce a transition in the carry. Hence, the switching activity is

$$\begin{aligned} c_{out}^{A2} = & p^0 \cdot q^1 \cdot c_{in}^1 \cdot (1 - p) \cdot (q + c_{in} - 2 \cdot q \cdot c_{in}) + \\ & p^1 \cdot q^0 \cdot c_{in}^1 \cdot (1 - q) \cdot (p + c_{in} - 2 \cdot c_{in} \cdot p) + \\ & p^1 \cdot q^1 \cdot c_{in}^0 \cdot (1 - c_{in}) \cdot (p + q - 2 \cdot q \cdot p) \end{aligned} \quad (4.37)$$

In the third case, two inputs change and one remains the same. In this case, for the combinations of bits "000" and "111", any two operands will produce the change at the carry. Hence, the transition activity for these combinations is

$$c_{out}^{B1} = (p^0 \cdot q^0 \cdot c_{in}^0 + p^1 \cdot q^1 \cdot c_{in}^1) \cdot (p \cdot q \cdot (1 - c_{in}) + (1 - p) \cdot q \cdot c_{in}) \quad (4.38)$$

The combinations containing two zeros and a single one, such as "001", "010", "100", will make the carry to change by changing the zeros into ones. The produced activity is

$$\begin{aligned} c_{out}^{B2} = & p^0 \cdot q^0 \cdot c_{in}^1 \cdot p \cdot q \cdot (1 - c_{in}) + \\ & p^0 \cdot q^1 \cdot c_{in}^0 \cdot (1 - q) \cdot p \cdot c_{in} + \\ & p^1 \cdot q^0 \cdot c_{in}^0 \cdot (1 - p) \cdot q \cdot c_{in} \end{aligned} \quad (4.39)$$

Finally, the combinations "011", "110" and "101", will generate the transition in the carry by changing both ones into zeros. Hence, we have

$$\begin{aligned} c_{out}^{B3} = & p^0 \cdot q^1 \cdot c_{in}^1 \cdot (1 - p) \cdot q \cdot c_{in} + \\ & p^1 \cdot q^0 \cdot c_{in}^1 \cdot p \cdot (1 - q) \cdot c_{in} + \\ & p^1 \cdot q^1 \cdot c_{in}^0 \cdot p \cdot q \cdot (1 - c_{in}) \end{aligned} \quad (4.40)$$

In the fourth case, all input bits change. It can be easily seen that in this case, the carry at the output will make a transition regardless of the combination of zeros and ones at the inputs.

Hence, we have that the switching activity of the carry bit for this case is

$$c_{out}^C = p \cdot q \cdot c_{in} \quad (4.41)$$

The final expression for the carry is obtained by adding up the transition probabilities of all three cases:

$$c_{out} = c_{out}^{A1} + c_{out}^{A2} + c_{out}^{B1} + c_{out}^{B2} + c_{out}^{B3} + c_{out}^C \quad (4.42)$$

The probabilities of the carry bit and the output bit of the full-adder cell being '0' and '1' are computed as:

$$\begin{aligned} c_{out}^0 &= p^0 \cdot q^0 + c_{in}^0 \cdot (p^0 \cdot q^1 + p^1 \cdot q^0) \\ c_{out}^1 &= 1 - c_{out}^0 \\ s^0 &= (p^0 \cdot q^0 + p^1 \cdot q^1) \cdot c_{in}^0 + (p^0 \cdot q^1 + p^1 \cdot q^0) \cdot c_{in}^1 \\ s^1 &= 1 - s^0 \end{aligned} \quad (4.43)$$

Now, the only probability missing for the computation of the total switching activity is the probability at the output of the full-adder cell and it is given by:

$$s = (p \cdot q + (1 - p) \cdot (1 - q)) \cdot c_{in} + (p \cdot (1 - q) + q \cdot (1 - p)) \cdot (1 - c_{in}) \quad (4.44)$$

This methodology is similar to the methodology described in [CRP94]. However, in [CRP94], the method is applied to the whole module. Since the calculation of signal probability is NP-hard, in large circuits they employ a partitioning algorithm that limits the number of inputs to a module. The approach proposed here applies the method for estimating switching activity only to a basic cell of the DSP component, thus providing a simple expression for the total switching activity of the module, no matter its size, by propagating the switching activity throughout the module. The drawback of this approach is that the input spatial dependencies when two or more full-adder cells are connected are not taken into account. However, we consider that the error introduced by spatial correlations of the signals inside the DSP component can be neglected for the purposes of the high-level switching activity analysis presented here.

4.2.3. Structural model

Based on the regional decomposition of the input words according to their switching activity, a whole component is divided into activity regions in [LR95]. A switched capacitance model is generated (i.e. capacitance multiplied by switching activity) for every component region, and the power consumed by a given module is obtained through the summation of power consump-

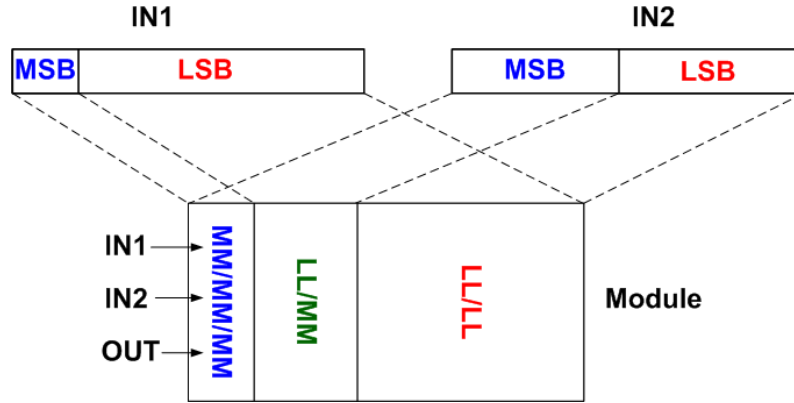


Figure 4.7: Ripple-carry subtracter with misaligned breakpoints

Table 4.2: Capacitance coefficients for a ripple-carry subtracter [LR95]

Transition templates	Capacitive coefficients			
LL/LL	$C_{LL/LL}$			
MM/MM/MM	$C_{++/++/++}$	$C_{++/++/+}$	$C_{++/++/-}$	$C_{++/++/--}$
	$C_{++/+-/++}$	$C_{++/+-/+}$	$C_{++/+-/-}$	$C_{++/+-/--}$

	$C_{--/-+/++}$	$C_{--/-+/-}$	$C_{--/-+/-}$	$C_{--/-+/-}$
	$C_{--/--/++}$	$C_{--/--/+}$	$C_{--/--/-}$	$C_{--/--/--}$
LL/MM	$C_{LL/++}$	$C_{LL/+}$	$C_{LL/-}$	$C_{LL/--}$
MM/LL	$C_{++/LL}$	$C_{+-/LL}$	$C_{-+/LL}$	$C_{--/LL}$

tions from all the activity regions.

A disadvantage of this model lies in the fact that the number of required simulations increases rapidly with the complexity of the component structure. For example, consider a ripple-carry subtracter (see Fig. 4.7). Each of the two inputs can be divided into two signal regions: LSB and MSB. This leads to two component regions: LSB-LSB and MSB-MSB if the breakpoints of the two inputs are aligned, with an additional LSB-MSB or MSB-LSB region if they are not aligned. Switched capacitance possibilities are shown in Table 4.2 for different component regions. There is only one switched capacitance for the LSB-LSB region, as the output bits will also be uncorrelated. However, in the MSB-MSB region, the switched capacitance can significantly depend not only on the values of the MSB sign bits but also on the values of the output bits, which can not always be determined based on the input MSB bits (e.g. the subtraction of two positive numbers can lead to a negative number as well as to a positive one). Hence, distinct capacitive coefficients are derived for this region for all possible combinations of pairs of sign bit values at both, the input bits and the output bit. For example, $++/++/+$ represents two consecutive cycles such that in the first cycle one positive number is subtracted from another and the result is positive, while in the second cycle for the same signs of inputs, the output is

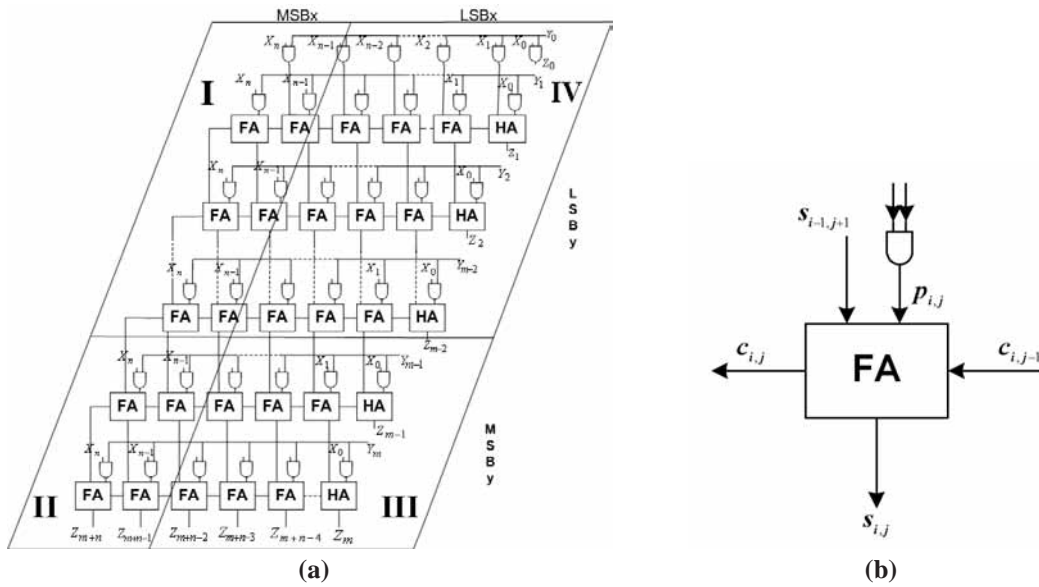


Figure 4.8: a) Regionally decomposed array multiplier, b) Full-adder cell

negative. In all, there are a total of 73 capacitive coefficients for a ripple-carry subtracter. As the complexity of the component grows, this number tends to be extremely large.

We have used a similar approach here, but instead of the summation of power consumptions from all the activity regions, we sum only their switching activities in order to obtain total switching activity of the component. Thus, we avoid the large number for different switched capacitances, and obtain the switching activities at the outputs of the basic cells very fast through the described probability method. The problem of estimating the load capacitance separately from the switching activity when considering DSP basic cells will be addressed in the next section.

In the following, we give the expressions for the total switching activity in array multipliers, row adder tree multipliers, booth multipliers and ripple-carry adders.

Array Multiplier

The structure of the standard array multiplier is shown in Fig. 4.8a. Operand x has N bits and y has M bits. The multiplier consists of basic elements, namely two-bit multipliers and half-adder and full-adder cells. According to the division of the inputs into LSB and MSB regions, the whole component is also divided into activity regions. The input signals in the Fig. 4.8a are considered to have zero-mean gaussian distribution, as only two input activity regions are identified (LSB and MSB). However, this has no relevance for the purposes of the analysis presented here, since the methodology is the same for the non-zero mean signals.

In Fig. 4.8b, a basic cell of an array multiplier is presented. It consists of an AND gate together with a full-adder cell. Index i is the row number and j is the column number of the

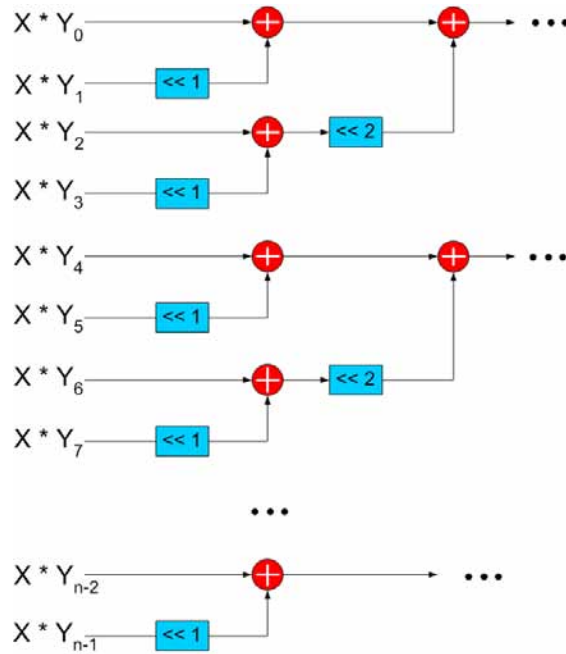


Figure 4.9: Row-adder tree multiplier

full-adder cell in the array. The carry bit from a previous cell is the third input to a full-adder cell. As q represents the transition probability at the output of the full-adder cell from the previous level, it is clear that $q_{i,j} = s_{i-1,j+1}$.

The total switching activity of the array multiplier consists of the switching activities of the carry-bits and outputs of the adder and multiplier cells. Hence, the final result for the switching activity is obtained from the following expression:

$$SW = \sum_{i=1}^{M-1} \sum_{j=1}^N (s_{i,j} + c_{i,j} + p_{i,j}) \quad (4.45)$$

where $s_{i,j}$, $c_{i,j}$ and $p_{i,j}$ are computed as explained in the previous section.

Row-adder Tree Multiplier

There are many different ways to implement a multiplier into an FPGA. In particular, we consider the Virtex 2 and Virtex-2 Pro family of FPGAs from Xilinx [Xil]. Xilinx IP Cores optimize the implementation of the array multiplier by transforming it into a row adder tree multiplier. This type of multiplier rearranges the adders of the array multiplier to equalize the number of adders that the result from each partial product must pass through [Gro] (see Fig. 4.9). The worst case path is through $\log_2(n)$ adders instead of $2n$ adders as is the case in array multiplier.

As Virtex-2 and Virtex-2 Pro devices use 4-input LUTs, in the first optimization level the partial sum of two products is implemented into one LUT. This procedure optimizes two rows

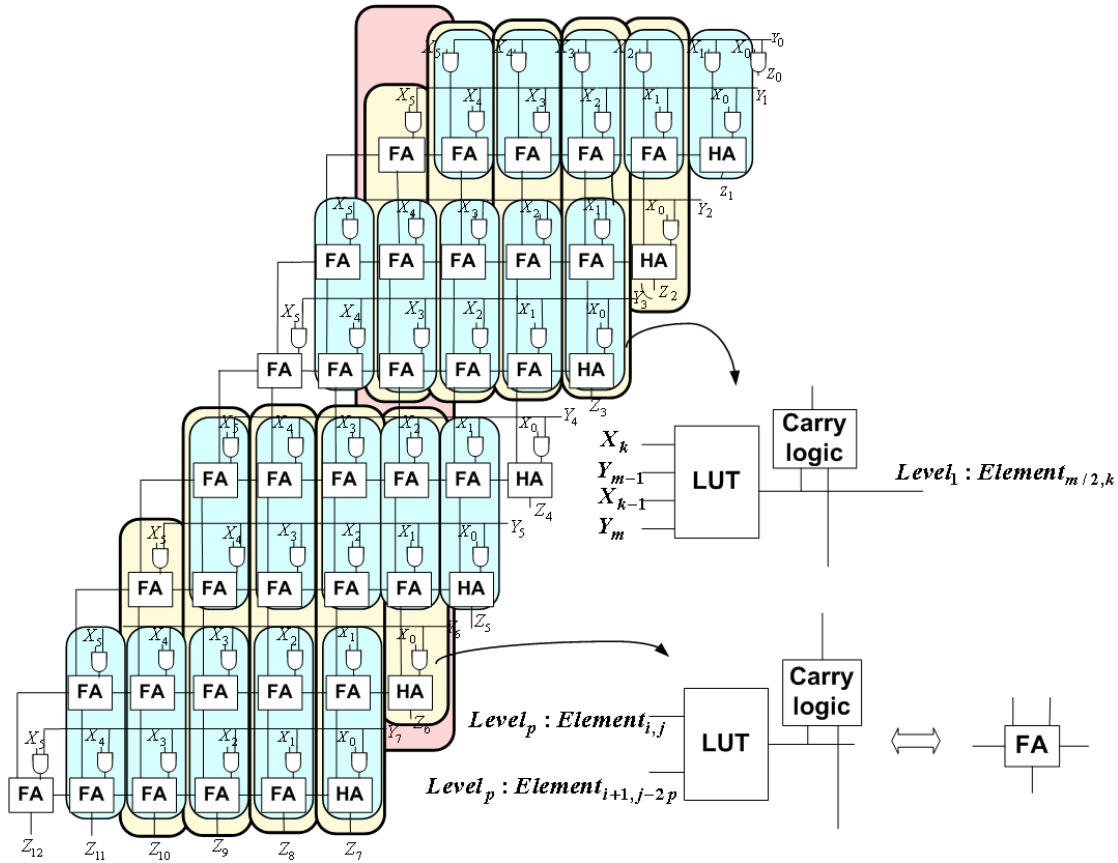


Figure 4.10: Row-adder tree multiplier implemented in FPGAs

of the array multiplier by using only one row comprised of LUTs. Fig. 4.10 shows how a 6×8 array multiplier is transformed into a row adder tree multiplier and also how the optimization levels are implemented in the FPGA. The blue rectangles represent the first optimization level and their implementation is drawn just beside the multiplier. The next level of the optimization compresses two LUT rows from the first optimization level into one using a similar methodology. This level is represented with yellow rectangles. For the sake of the clarity, the last optimization level is represented just for one column of full-adder cells with a pink rectangle. Taking into account all these specific details of the multiplier implementation into the FPGA leads to a new expression for its switching activity. The methodology used for computing the switching activity is the same as in subsection 4.2.2. We give the final expression for the total switching activity when the word-length of operand y is a power of two as:

$$SW = \sum_{i=1}^{\lceil \log_2 M \rceil} \sum_{j=1}^{l_j} \sum_{k=2^i-1}^{N+2^i-1} (s_{i,j,k} + c_{i,j,k}) \quad (4.46)$$

In other cases, the counters i and j of the two inner summations have slightly different values due to the parity of the number of LUT rows in each optimization level.

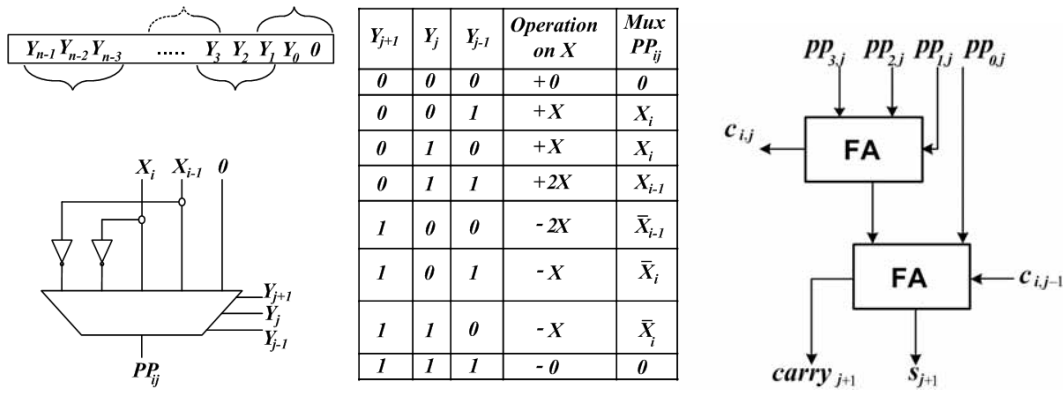


Figure 4.11: Modified Booth Algorithm

Modified Booth Multiplier

Next, we adapt the switching activity methodology to the embedded multiplier block structure.

Multiplication is one of the basic and most commonly used operations in DSP algorithms, so a lot of effort is made to speed-up its computation and to avoid excessive power consumption. Among the many proposed multiplication algorithms, the Modified Booth algorithm is predominantly used for multiplier design in VLSI systems, as it is energy-efficient and has an operating speed close to the Wallace tree architecture, which is considered to be the fastest architecture for multipliers [ZA95].

The dedicated 18×18 bit multipliers in Spartan 3 FPGAs use a Modified Booth Algorithm [Xil03]. Since Spartan 3 and Virtex II Pro reached the market at almost the same time, we assume that the embedded multiplier architectures in these two families are based on the same algorithm.

The Booth algorithm for multiplying two numbers, multiplier (Y) and multiplicand (X), encodes the two's complement multiplier in order to reduce the number of partial products to be added. The partial products are formed within 2 steps: encoding and selection. During encoding, the multiplier Y is divided into overlapping groups of 3 bits as shown in the upper left corner in Fig. 4.11. Each group is encoded in parallel in order to define the operation to be performed on the multiplicand according to the rules in the table of Fig. 4.11. All of the entries in the column *Operation on X* can be obtained through shifting and complementation of the multiplicand. Thus, each partial product is selected through a multiplexer instead of produced by an AND gate as in direct multipliers, as shown in the bottom left corner in Fig. 4.11. As the X bits are only inverted through the multiplexer, bit Y_{j+1} is added later together with the corresponding partial products, as this bit determines if X_i or X_{i-1} needs to be complemented.

Using this algorithm, the number of summands is $\lceil \frac{n+1}{2} \rceil$. The extra 1 in the expression comes from the need to ensure that the last summand is a positive multiple of the multiplicand [HTF95]. This is achieved by adding an extra zero to the left of the multiplier, and an extra sign

bit to its right only if the bit-width of the multiplier is an odd number.

Partial products are added using different size compressors. These structures contain special combinations of full-adder cells which yield a faster partial product compression. The commonly used 4-2 compressor realized with full-adder cells is shown at the right-hand side of Fig. 4.11. It compresses five partial bits into three, at the expense of more complex interconnections. Four input bits come from the same bit position of weight j , while one bit is fed from the neighbouring $j - 1$ (known as carry-in). The output of such a 4:2 module consists of one bit in position j and two bits in position $j + 1$. This structure has improved efficiency as it reduces the number of partial product bits by one half at each stage [VO93]. In order to improve the critical path and to optimize sign extension and the number of full-adder cells, we assume that apart from the 4-2 compressor, and another standard compressor size of 9-2, non-conventional compressors of different sizes described in [CPHC03] are also used.

The result of the compression are two rows: one with the partial sum bits, and the other with the partial carry bits. An adder module is used to add these two rows and obtain the final result.

The total switching activity is obtained as the addition of the switching activities of the outputs and carry bits of all the basic cells in the component. As the structure of the Booth encoding procedure is quite complicated, instead of the probability method used described in the previous subsection, we have used zero-delay signal simulations in order to obtain the switching activities at the outputs of the compressors and the final adder. They are summed to obtain the parameter SW :

$$SW = \sum_{i=1}^{compr} \sum_{j=1}^{N_i} (s_{i,j}^{compr} + c_{i,j}^{compr}) + \sum_{i=1}^{36} (s_i^{add} + c_i^{add}) \quad (4.47)$$

where $compr$ is the total number of compressors, N_i is the number of full-adder cells in the i -th compressor, and $s_{i,j}^{compr}$, $c_{i,j}^{compr}$, s_i^{add} and c_i^{add} are the switching activities at the outputs and carry bits of the full-adder cells in compressors and the final adder, respectively. The number of full-adder cells in the adder is fixed to 36, as the multiplier operands are always sign extended to 18 bits.

Ripple-carry Adder

The adder considered in this subsection is a ripple-carry adder consisting of full-adder cells as presented in Fig. 4.12. Module decomposition in the case of an adder depends on the breakpoint position of its input operands. There are two different cases as shown in Fig. 4.13. In both cases, operands have the same length as the shorter operand is always sign-extended until it reaches the length of the longer operand. One case corresponds to the situation where the MSB-LSB breakpoint of operand X lies in the MSB area of Y and another where the situation is vice versa.

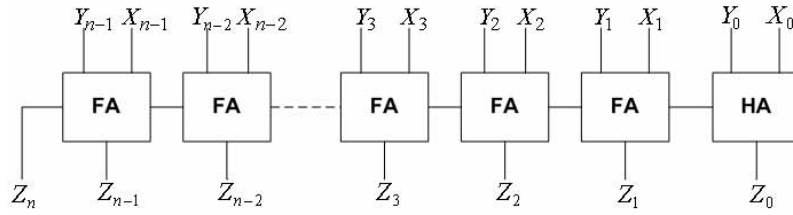


Figure 4.12: Ripple-carry adder

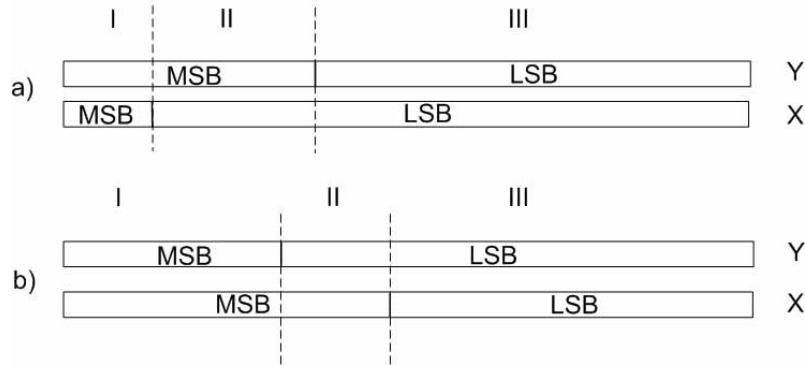


Figure 4.13: Module decomposition in function of MSB-LSB breakpoint of the longer operand

Thus, the adder decomposition consists of three parts: LSB-LSB, LSB-MSB and MSB-MSB. However, special care has to be taken when calculating the switching activity of the middle part, LSB-MSB, as in the first case the LSB part belongs to one operand and in second to the other. There is a third special case when both breakpoints coincide, which results in only two adder regions.

As the basic cell of the adder is a full-adder cell, the computation of the transition probabilities of the carry bit and the output bit has already been explained in section 4.2. The total switching activity of the adder is obtained by summing the switching activities of the carry bits and output bits:

$$SW = \sum_{i=1}^N (s_i + c_i) \quad (4.48)$$

4.3. Glitching model

As can be seen, the switching activity method used in this work is based on a zero-delay timing model, and as such, it does not take into account glitches. Glitching is a spurious activity caused by different logic or interconnect delays. For example, consider a circuit in Fig. 4.14. If both signals A and C , are equal to logic '1', then the output should also be '1' regardless of the value of the signal B . However, due to the delay of the inverter, it can be seen that there is a period of

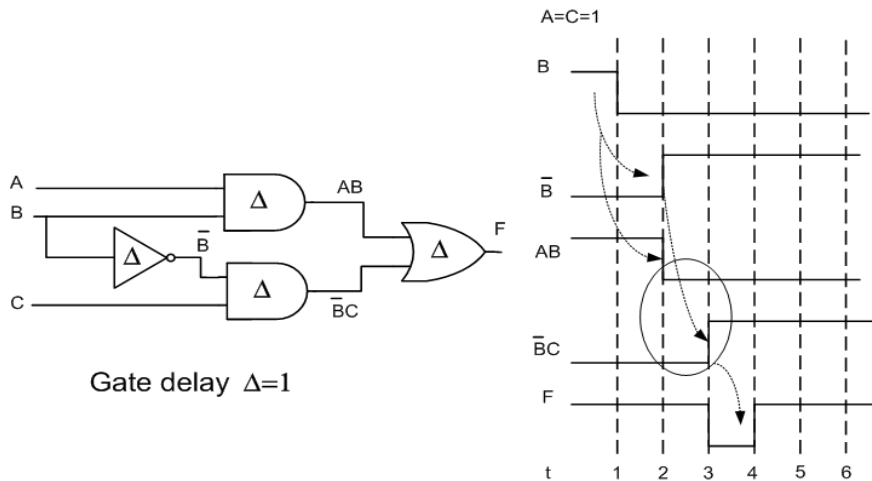


Figure 4.14: Glitch generation in a logic circuit

time where both products (AB and \overline{BC}) are logic '0', which will provoke a glitch at the output (in this case the so-called false '0'). However, if the delay of the logic gate was to be greater than the time difference of the arriving input signals, the glitch would not have appeared at the output of the gate.

Glitching activity can form a high percentage of the total power consumption (up to 80%) [LLW07, LLW08]. Still, tools used for circuit simulation usually report very short glitches that can lead to activity overestimation [LCHC03] if they are not filtered when their duration is shorter than the delay of the logic components they pass through.

In the following section, first we will give a brief overview of the techniques for glitching minimization, and then, we will describe the methodology used in this work for estimating the amount of glitching generated inside the DSP components.

4.3.1. Glitching: background

As the amount of glitching depends on the delays of logic gates, as well as the delays of interconnects, it is hardly predictable in the early phases of a design flow. On the other hand, accurate estimations of glitching at the gate-level are very time-consuming for large designs. In [MDG⁺97], similar to [Naj91], switching activity estimation is equivalent to the problem of computing signal probabilities of a multilevel circuit derived from the original circuit by a process of symbolic simulation. During such process, they use a variable-delay model. This means that the glitches are included into the switching activity computation. All the glitches shorter than the delay of the logic they pass through are filtered, and the correlations between the internal signals are also taken into account. The problem lies in the extremely long execution times.

In [RW05], the amount of glitching was estimated by using gate-level Modelsim simulations

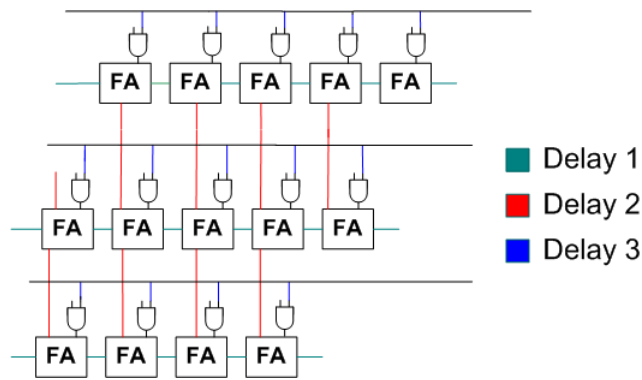


Figure 4.15: Repetitive delays at the entrance of the basic cells

of the place-and-routed design, and adjusting the time resolution of the simulator in order to capture the major amount of generated glitches, and still to be able to simulate large design in a reasonable time. The accuracy of the estimation, beside the simulator time resolution, also depends on the delay models for all FPGA resources. These models are provided by the commercial tools from chip vendors, as they have the access to the proprietary technology which is necessary for delay computation. Still, some approximations have to be made while building these models in order to reduce the simulation time. For example, it has been noted that short glitches are often not filtered when they pass through interconnections, leading to glitch overestimation [TGS⁺02].

In [CCCS08], an attempt was made to model the glitch power at higher levels of abstraction. They have used the switching activity computation presented in [Naj91], where the final value for the switching activity at the output of a gate is obtained as if the arrival times of all input signals were separated by a time difference larger than the delay of a gate (i.e. the influence of each signal is accounted for separately). Since this assumption is not valid in most of the cases, this methodology produces large overestimates. Thus, in [CCCS08], they try to avoid the overestimates by introducing several coefficients to the model that have to be obtained empirically. Besides, the results are given only for the adder, and it seems that applying the same methodology to larger circuits such as multipliers is hardly possible due to the extremely large computational effort.

As glitching is hard to estimate, some efforts have been made in order to minimize it. The most widely used technique for FPGAs is inserting pipeline stages [RW05, LLW07, WAL04]. It requires little additional cost since often many of the flip flops within the design's CLBs go unused.

4.3.2. Glitching methodology

The switching activity at a node increases with glitching which is produced by the different signal delays entering the same logic component. As already seen in the last section, the structure of DSP components is quite repetitive (array of full-adder cells in array multiplier, row of full-adder cells in adder...). All the components can be built by repeating logic blocks (together with its connections to the neighbouring cells) throughout arrays. Thus, although glitches propagate through logic depending on the logic function they pass through, we can still make the following assumption.

As the repeated cell always has the same logic function, and practically, the same delay for all input signals, the difference in glitching at the outputs of two logic blocks will depend on the difference in transition activities at their inputs. For example, Fig. 4.15 shows a part of the array multiplier. Due to the repetitiveness and regular location of full-adder cells, the delay of the input wire entering the left side of each full-adder cell is probably the same for all full-adder cells (marked with red colour in the figure). The same statement can be made for the inputs on the right (marked with blue) and the carry bits (marked with green). Hence, each full-adder cell has approximately the same input signal delays as any other cell in the component. It is also obvious that it performs the same logic function. The only variable that differs from one full-adder cell to another is the transition activity at its inputs. As a consequence, it is considered that glitching is directly related to the input transition activities and thus, the most significant amount of glitching produced inside the component is generated from the most active regions of its inputs. In Fig. 4.16, the distribution of the glitching activity in the array multiplier is presented (signals are assumed to have non-zero mean distributions and positive autocorrelation coefficients).

When considering zero-mean gaussian signals and only positive autocorrelation coefficients, the LSB input regions exhibit the highest switching activity. This can be deduced directly from equation (4.16). In this context, glitching has been modelled as the sum of the average equivalent glitching produced by each cell belonging to the LSBx-LSBy region of the component (see Fig. 4.8).

The expression for the glitching when considering only the LSBx-LSBy region for the implementation of the multiplier (i.e. IP core implementing a row adder tree multiplier), is given as:

$$\begin{aligned}
 G &= k \cdot l_{s_x} \cdot \sum_{i=1}^{\lceil \log_2(l_{s_y}) \rceil} m_i = k \cdot G' \\
 m_i &= \lceil m_{i-1}/2 \rceil \\
 m_1 &= \lceil l_{s_y}/2 \rceil
 \end{aligned} \tag{4.49}$$

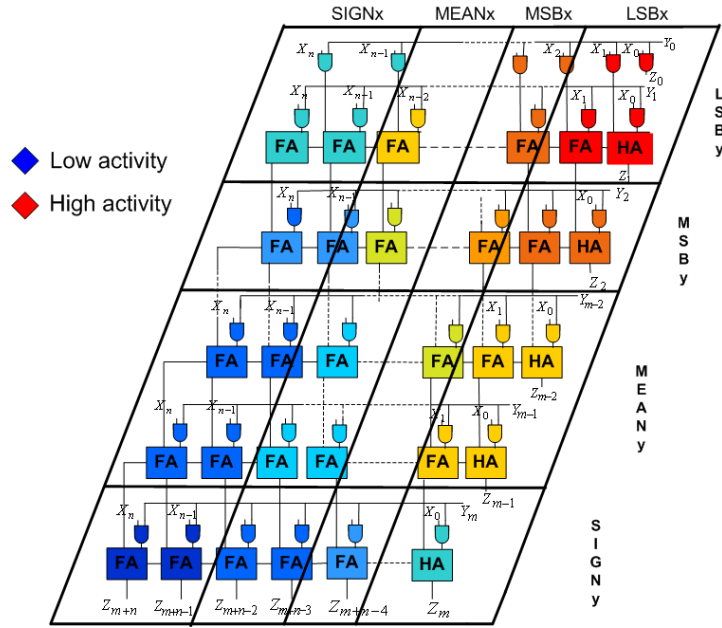


Figure 4.16: The distribution of glitching generated inside the array multiplier

In the case of the adder, the following expression was used:

$$G = k \cdot \min(ls_x, ls_y) \quad (4.50)$$

where G is the amount of glitching, ls_x and ls_y are the number of bits in the LSB region of the input signal words and k is an empirically derived constant which represents the average glitching at the output of one LUT. When the position of the MSB-LSB breakpoint is known, the number of LSB bits in the operands is easily obtained.

This glitching model was presented in [JCC07a, JCC07b, JCC08].

However, for negative autocorrelation coefficients, it is clear that MSB regions exhibit a higher switching activity than all neighbouring regions. Therefore, the glitching model has to be modified to account for the contribution of these regions as well. This is achieved by introducing a new factor into the expression for glitching that depends on the value of the autocorrelation coefficient, as follows.

Assuming that the MSB bits have an equal probability of being '0' or '1', according to 4.16 the expression for their switching activity becomes:

$$t_i = 0.5 \cdot (1 - \rho) \quad (4.51)$$

As the LSB bits have a switching activity of 0.5, the relationship between the switching activities of these two regions can be expressed as a coefficient $l = 1 - \rho$. At the same time, this is the relationship we expect between the average glitching produced in the MSB and LSB parts,

as the amount of glitching is proportional to the transition activity of the input bits. Hence, the extended glitching model which is the sum of glitching in the four component's regions (LSBx-LSBy, LSBx-MSBy, MSBx-LSBy and MSBx-MSBy in Fig. 4.8) is expressed as:

$$G = k \cdot \sum_{i=1}^4 (1 - \rho_{1i}) \cdot (1 - \rho_{2i}) \cdot FA_i = k \cdot G' \quad (4.52)$$

where G is the amount of glitching, k is an empirically derived constant which represents the average glitching at the output of one LUT in the LSBx-LSBy part of the component, ρ_{1i} and ρ_{2i} are the bit-level autocorrelation coefficients of the LSB/MSB regions of inputs (depending on the particular region of the component), and FA_i is the number of full-adder cells in the corresponding component's region.

The extended glitching model was presented in [JC08].

Glitching in embedded blocks

Dedicated multipliers are highly optimized for performance and power, so we assume that the delays of local wires connecting multiplier's basic elements are quite balanced. This allows us to simplify their power model by neglecting the glitching generated inside the component.

4.4. Logic power estimation

In this section we present the high-level logic power model for both, LUT-based components and embedded blocks. First, we give an overview of the previous work on logic power estimation and a classification of the power models according to the chosen signal model. Then, we model the load capacitance in DSP components, as this is the final parameter needed for the power estimation. Next, we describe the power model for LUT-based components, followed by the model for embedded blocks. Finally, we end this section by describing an optimization which can be applied to the logic power model for LUT-based components in order to improve its accuracy.

4.4.1. High-level logic power estimation: background

Beside the long switching activity computation time, all low-level estimation techniques need transistor or gate level circuit descriptions, so power estimation occurs late in the design process, leading to severe penalties in design time when constraints are not met. Some methods to estimate power consumption at higher levels have been proposed in order to reduce this time.

When only the relative design power increase and decrease is needed, some techniques *a priori* make an assumption about the input data and therefore, use the area of the design

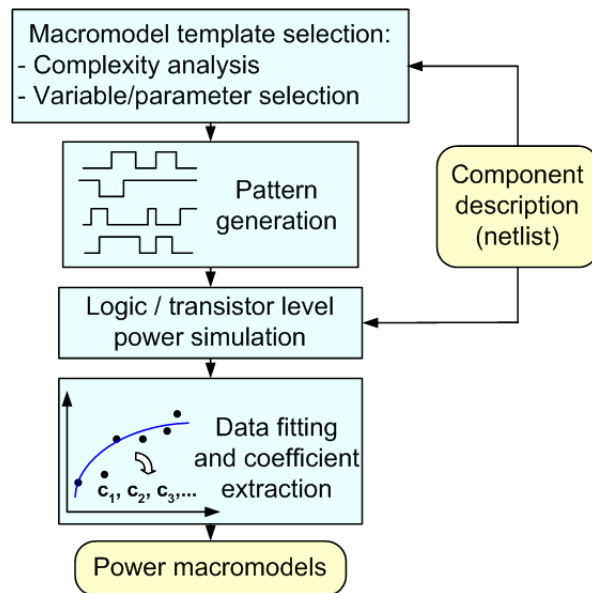


Figure 4.17: Power macromodelling

as a relative ratio for power variability [SGLVLB06]. However, when a more detailed power breakdown is needed, high-level methodologies that include the dependency of power on input data are required.

The most common technique is based on power macromodels ([AN04a, GN00, JTB04, HSS⁺02, JKSN99, CGC05]), where power is presented as an equation (very often polynomial) with variable parameters depending on the input and output signal statistics, input word-lengths, etc. Apart from the type of the variables, the order of the polynomial equation needs to be determined, and the appropriate combinations of the parameters have to be chosen as terms in the equation, in order to faithfully model power consumption. The designer's insight and interaction are required in order to come up with accurate complexity parameters that have the largest impact on power. Coefficients standing by the variables are then found through data-fitting of the power values obtained from extensive low-level simulations (see Fig. 4.17).

All power macromodels can be further divided into three groups according to the characterization of the input data set: bit-level, Hamming distance and word-level power macromodels.

There is an extensive ongoing research work on the underlying FPGA architecture. The number of Look-Up Tables per cluster, the number of clusters per Configurable Logic Block [AR04], the most efficient routing structures [LEG07], and many other parameters are being explored, in order to find the best trade-off between design area, performance and power consumption. One of the promising architecture modifications are special-purpose blocks, such as Embedded Multipliers and DSP Blocks, that are used to accelerate arithmetic intensive applications.

Standard FPGA power estimation techniques have not been used, so far, for the power

estimation of special-purpose blocks, as their architecture differs from the programmable fabric structure. For this reason, an overview of the embedded power estimation is given separately here, in addition to overviews of the three types of power macromodels.

Bit-level power macromodels

The first group of power macromodels is based on bit-level input signal statistics ([JTB04], [GN00], [SJ01], [DR06], [DAR07]). It considers average bit level statistics which are found to be in direct relationship with power consumption. Four of these statistics can be considered as the most important ones: 1) the input signal probability P_{in} , defined as the average fraction of clock cycles in which the final value of a bit is '1'; 2) the input transition density D_{in} , defined as the average fraction of cycles in which the node makes a logic transition; 3) the input spatial correlation coefficient S_{in} , that represents the correlation between the bits inside a signal word; and 4) the output transition density D_{out} , defined as the average fraction of cycles in which the output node makes a logic transition. All these statistics appear in the model as average values obtained from circuit functional simulations. They are introduced as variables in an equation which estimates the average power consumed by the module:

$$P = f(D_{in}, P_{in}, S_{in}, D_{out}) \quad (4.53)$$

Coefficients standing by the variables are found through extensive simulations, which are listed into an n-dimension array in look-up table models [GN00, SJ01]. The number of simulations is reduced in equation-based macro-models [JTB04, DR06, DAR07], but is still quite high, as for some components the number of coefficients that needs to be calibrated goes up to 20. The reported computation time for model calibration is measured in hours in [GN00, SJ01], and for some of the circuits goes up to 50 [GN00]. Other methods report the number of simulation vector sets needed for model characterization ([DR06, DAR07]). This number lies between 1000 and 2000 for small components such as comparators and 4×4 and 8×8 multipliers. Both, large computational time and effort, represent serious limitations when using bit-level model.

The techniques that use equation 4.53 for power estimation report 3% average error and below 15% maximum error for small components such as multipliers, comparators, adders, etc. ([SJ01, DR06]), and 6% average error and below 25% maximum error for larger circuits [GN00]. However, these models are not parameterized in terms of component size, so the calibration of the model needs to be repeated whenever a component with different word-length inputs is used in the design. As already seen, this is highly time-consuming and it should be avoided whenever possible. The only model that includes a unique variable for component size is [JTB04]. The reported average error is 4% while the maximum error goes up to 33%. Still, this model is not capable of producing estimates for components with different operand sizes.

Hamming-distance power macromodels

The second power estimation group is based on power macro-models built by using the spatio-temporal correlation previously defined as Hamming distance [HSS⁺02,JKSN99,RSN06]. Hamming distance models represent 'black box models' that do not use any knowledge of the components internal structure, but instead, abstract to input data statistics.

The variables used in Hd-models are Hamming distance, Signal distance and Zero distance. The Hamming distance is defined as the number of transitions between two consecutive input vectors:

$$Hd = \frac{t_{0 \rightarrow 1} + t_{1 \rightarrow 0}}{\sum_{i,j} t_{i \rightarrow j}} \quad (4.54)$$

where $t_{i \rightarrow j}$ is the number of bit transitions from i to j within two consecutive input bit-vectors.

The Signal distance is the number of input bits that are fixed to logic one in two consecutive input vectors.

$$Sd = \frac{t_{1 \rightarrow 1}}{\sum_{i,j} t_{i \rightarrow j}} \quad (4.55)$$

This number increases the probability that the switching activity of the inputs is propagated through the component [HSS⁺02].

The Zero distance is the number of input bits that are fixed to logic zero in two consecutive input vectors and is obtained from the following equation:

$$Hd + Sd + Zd = 1 \quad (4.56)$$

These three variables are used to classify different input streams. Normally, the characterization set includes every possible combination of input-streams for $Hd = \{0, 0.25, 0.5, 0.75, 1\}$ and $Sd = \{0, 0.25, 0.5, 0.75, 1\}$. The combinations are limited by (4.56). For a component with inputs A and B, the number of possible sets of Hd_A , Hd_B , Sd_A and Sd_B is:

$$M = (1 + 2 + \dots + n_A) * (1 + 2 + \dots + n_B) \quad (4.57)$$

where n_A and n_B are the number of different values in the sets Hd_A and Hd_B respectively. Hence, for the most common characterization set, the number of low-level simulations equals to 225. Consequently, the characterization effort is smaller than the effort used for bit-level model characterization.

The components' input word-lengths or/and their combination, are chosen as the last model variables in order to make the model scalable. In this case, the characterization process has

to be repeated for every size of every component, with the input sizes taken from the input word-length set bw .

Without a significant sacrifice of the accuracy of the model, the Hd-model can be expressed as a product of two separate functions: one for the dependency on the input word-lengths and the other for the dependency on the normalized Hd and Sd values [JKSN00]. The word-length function is obtained by regression over the power consumption of the component for different input word-lengths, while maintaining the signal statistics fixed. The signal statistics function is sampled at a fixed word-length for special combinations of Hd and Sd . An interpolation is applied as to obtain a power value for statistics not belonging to the characterization set.

The average reported error lies around 16% for FPGAs (12% for ASICs) with the maximum errors reaching 40% (22% for ASICs). It can be seen that the smaller characterization effort is accomplished on the account of higher estimation errors. Besides, the power macro-models based on Hamming distance tend to give large errors when two different input signals that result in different output statistics, are characterized with the same signal parameters [KAA⁺07].

A comparison of the approach proposed here with the Hd-approach will be presented in section 4.5.

Word-level power macromodels

Approaches based on word-level signal statistics ([LR95, CGC05, CGCC06]) constitute the third power estimation group. They consider the variation in power consumption caused by the variation of the input signal variance σ^2 , mean μ and correlation coefficients ρ . Input signals are approximated with a gaussian distribution.

The power model presented in [LR95] has been the principal motivation for the logic power model developed in this work. As it was previously explained, we use the component decomposition into activity regions in order to compute the total switching activity, while the model presented in [LR95] uses this decomposition in order to compute the total power. Their average reported error is less than 10% with the maximum error being around 20%. The main drawback of their approach is the large number of coefficients that have to be obtained through extensive simulations.

Some other word-level models have been proposed in the literature. In [CGC05], characterized equations for arithmetic components implemented in FPGAs are presented. In particular, this approach generates first-order and second-order equations for adders and multipliers respectively. The following equations are used for power estimation in these components:

$$P_{add} = C_0 \cdot W + C_1 \quad (4.58)$$

$$P_{mult} = C_0 \cdot W^2 + C_1 \quad (4.59)$$

The coefficients C_0 and C_1 are obtained from extensive simulations and depend on word-level signal statistics of both operands. Pre-made tables of coefficients are constructed as a result of these simulations. The only variable introduced in the equation is the operands' word-length W . Therefore, this methodology is not capable of producing estimates for components where both operands have different sizes. This work was further extended in [CGCC06] in order to consider non-zero mean input signals, by including additional coefficients in the model that depend on the number of the mean bits and the sign of the mean. The average error for both zero-mean and non-zero mean signals is around 7%. However, the maximum error according to the power plot presented in their work, can reach up to 100% with most errors lying below 55%. The number of simulation vector sets used for model characterization is 1500. We will compare their methodology for zero-mean gaussian signals with the methodology presented here in section 4.5.

The authors in [LMR01] consider the switching activity estimation in DSP architectures in order to obtain power estimates. The approach is similar to the one described in [GN00], but instead of using bit-statistics, the resulting models depend only on word-level statistics. They model components such as array multipliers, adders and delays. In order to generate transition activity models for different components, a least square approximation is applied to the results obtained from a very large number of measurements. However, it has been pointed out that the purpose of their work is only to detect the relative changes of power-dissipation at the system level and, therefore, cannot be used for accurate power estimation.

Embedded power estimation

The power estimation of embedded blocks is relatively new, so the work on this topic is quite limited. In [EJH⁺04], a high-level power estimation model of Xilinx FPGA embedded memories has been presented. The model uses a set of high level parameters, divided into architectural parameters such as the number of bits in the address and data buses, and algorithmic parameters such as the communication rates of the address and data buses. The coefficients standing by the parameters are obtained through curve fitting over power values gathered from measurements when each of the parameters varies independently. However, as the interconnect power is expressed only through the number of interconnections, and not their length, the accuracy of the model can be guaranteed only when applied to the characterization set.

The work presented in [CLW06] evaluates the impact on power consumption when logic is implemented in unused embedded memory arrays in FPGAs. Although, the circuit density is improved and there are fewer LUTs and connections between them, it was shown that this solution results in additional power consumption due to long bit and word lines, amplifiers, and decoders in embedded memory arrays. Additionally, they investigate the memory flexibility, size and shape in order to find the most power-efficient solutions.

The so-called virtual embedded blocks (VEBs) are described in [HLL⁺06]. They represent dummy elements created in order to determine the impact of embedded multipliers on area and delay of a circuit implemented in FPGAs. The area of an embedded block (EB) is approximated as the area of the EB implementation on a silicon chip divided by the area of a logic cell. This encourages the implementation tools to think about EBs in terms of FPGA resources. Logic delay is modelled through a use of adder carry chains, as it is relatively easy to adjust its delay by changing its length. However, the power consumption of embedded blocks is not considered.

An activity-based strategy for estimating the average power dissipation of hard DSP and multiplier blocks embedded in FPGAs was presented in [CKCW06]. The authors compared several methods for activity estimation of all nodes in the circuit and chose the one that gives the best results. The average reported error for [CKCW06] is 8% with the maximum error going up to 50% according to the power plot presented in their work. However, as their final power estimation model is based on the activity computation at the gate-level implementation of the embedded blocks, it requires information on proprietary technology and implementation details which are unavailable to most users.

Summary of the previous work on logic power estimation

In Table 4.3 we present the summary of features of the described high-level logic model and the model presented here: HLLM. First, approaches are clarified according to the target technology. Then, after identifying the approach, the methodology used for characterizing input data set and the tool used for obtaining "real" logic values for model validation are listed. The next columns include the maximum and the average value of the relative errors reported in each work. If there is more than one error reported, the table lists the worst case (for example, if there are two errors; one reported for the adders and the other for the multipliers, we choose the worse error and present it in the table).

In the next column we present the effort for the model characterization. High characterization effort represents one of the three following cases: the reported time for model characterization goes over 10 hours, or the number of coefficients needed and obtained by multivariable regression goes over 20, or the number of different input data sets needed for model calibration is more than 1000. Medium characterization effort requires between 100 and 1000 different input data sequences for model calibration. Low characterization effort requires less than 10 different input data sequences for model calibration.

The symbol \sim indicates that a very large number of input vectors were tested (16500) and that more than 95% of the power estimation errors lie below this value. The errors with the sign \approx beside them were extracted from the plots provided in the papers.

It can be seen that, the model presented here is the only logic model that has been verified with on-board measurements, thus, resulting in the most confident estimate values. Further-

Table 4.3: Summary of features of the logic power models.

Technology	Institution	Method	Tool	Max. error [%]	Avg. error [%]	Charact. effort	Different comp. size	Different oper. size
ASIC	Univ. of Toronto [GN00]	Bit-level	Synopsys	23%	6%	HIGH	NO	NO
	Tech. Univ. of Madrid [DR06, DAR07]	Bit-level	Synopsys	12%	2%	HIGH	NO	NO
	Univ. of Oldenburg [HSS ⁺ 02, JKSN99]	Hamming distance	Synopsys	22%	12%	MEDIUM	NO	NO
	Berkeley [LR95]	Word-level	Irsim	≈ 20%	<10%	HIGH	YES	YES
FPGA	Princ. Univ. [SJ01]	Bit-level	XPower	8%	3%	HIGH	NO	NO
	Northwest. Univ. [JTB04]	Bit-level	XPower	33%	4%	HIGH	YES	NO
	Univ. of Oldenburg [RSN06]	Hamming distance	XPower	≈ 40%	16%	MEDIUM	NO	NO
	Imp. Col. of London [CGC05, CGCC06]	Word-level	XPower	~ 55%	7%	HIGH	YES	NO
	Tech. Univ. of Madrid: HLLM	Word-level	Measurement XPower	23% 33%	9% 11%	LOW LOW	YES YES	YES YES

Table 4.4: Summary of features of the embedded power models.

Method	Tool	Technology	Max. error [%]	Avg. error [%]	Required prop. details
Univ. of Toronto [CKCW06]	PrimePower	FPGA	50%	8%	YES
Univ. of Madrid HLLM	Measurement	FPGA	21%	8%	NO

more, the characterization effort is the lowest compared to the other estimation errors (since it needs less than 10 measurements for model characterization), and it is capable of estimating the power of components with different size operands. The latter feature was only found in another methodology [LR95]. However, this methodology has an extremely high characterization effort, since the number of coefficients needed for modelling the most simple component is 73.

The accuracy of the models that are parameterizable in terms of the component and/or the operand sizes is lower than the accuracy of the models that do not take the size of the component into consideration, which is to be expected. HLLM supports components with different operand word-lengths and still achieves good accuracy for the purposes of high-level power estimation. Such accuracy has been evaluated through both, measurements and XPower. The latter tool was required for model characterization in order to perform a fair comparison with some other methods proposed in the literature. The results of the comparison are presented in the next section.

Due to the lack of power estimation models for embedded blocks, Table 4.4 only includes two power embedded models: the one presented in [CKCW06], and HLLM adapted for power estimation of embedded blocks. The main difference between the models is that [CKCW06] uses proprietary information, while HLLM does not. As such, HLLM is available to any user. Besides, the maximum error is much smaller than the one observed in [CKCW06]. The reason for this lies in the fact that, in [CKCW06], only the average input switching activity of all input bits is used as a power parameter. One average input switching activity value can correspond to various input vector sets, and thus different power values. On the other hand, HLLM accounts for the switching activities of all bits resulting in more accurate estimates for different input signal statistics.

4.4.2. Logic power model

The logic power model presented here uses a very small number of coefficients, and is parameterized in terms of clock frequency, input signal word-lengths and input signal statistics.

The embedded power model is considered separately due to its special implementation fea-

tures. It is also parameterizable in terms of input parameters and clock frequency. It should be pointed out that the main advantage of the power model for embedded blocks is that it provides accurate estimates by only using general information about the architecture of the multipliers. As it does not enter into implementation details, it is available to any user.

Load capacitance

DSP blocks are built from a number of basic elements, namely two-bit multipliers and half-adder and full-adder cells. As previously explained, we consider that each type of element is implemented into the slice section composed of one LUT and logic gates. As the LUT has the same structure regardless of the function performed in it, we assume that the capacitance being switched per each basic element is the same.

Although, the carry wires have a lower capacitance than other data wires as they are directly connected to the next adder cell via dedicated routing, the assumption referring to constant C_l can be considered valid for the purposes of high-level estimation. Arithmetic components exhibit a regular, repetitive structure composed of full-adder cells, implemented in programmable elements (LUTs and AND gates) that also have a regular structure and thus, C_l can be regarded as an effective capacitance when both types of wires are accounted for.

This allows us to divide the dynamic power consumption of each DSP component into four separate terms: V_{dd}^2 , f , C_l and α . As the first three terms remain the same for each basic element of the component, we can simply calculate the total switching activity as explained in section 4.2, then sum it with the amount of glitching estimated as explained in section 4.3, and with two measurements of the logic power of the component, obtain the two constants needed for the calibration of the power model: one that represents the product of the first three terms multiplying the expression obtained for the sum of the switching activity and glitching, and the other that represents the average glitching at the output of a LUT in the LSBx-LSBy part of the component, since this information has to be obtained empirically.

Logic power model for LUT-based components

Hence, the final model for estimating the power consumption in the presence of glitching and autocorrelation is given as follows:

$$P = b \cdot (SW + k \cdot G') \quad (4.60)$$

Constant b can be obtained together with constant k which has been introduced into the expression for glitching through two logic power measurements.

Thus, the complete power estimation characterization procedure consists of the following steps:

- 1) The input signal words are divided into activity regions by computing the positions of the breakpoints BP_0 , BP_1 and BP_2 .
- 2) The transition probabilities of all bits in the inputs are set to the values defined by 4.16;
- 3) The switching activity is calculated on the outputs of the basic cells (partial sum generators for row adder tree multiplier, full-adder cells for array multiplier and adder,...) and is added as in the corresponding equation (4.45, 4.46, 4.48);
- 4) Glitching presented in 4.52 is introduced into a final power model;
- 5) Two low-level power measurements for different component sizes using the same ρ are sufficient in order to determine coefficients b and k . As the factors SW , P and G' are known, the coefficients can be easily obtained. However, in order to increase the accuracy of the model, we use a multivariable regression approach with more than two measurements for obtaining these two coefficients. The number of measurements is still significantly smaller than any other existing high-level approach for building power macro-modules. It is also clear that the model is parameterizable in terms of the operands word-lengths and the input signal statistics.

The model represented by (4.60) has been used only for the power estimation of the logic elements in a component in [JCC07b,JCC08,JC08]. However, it was noted that it can be applied to the whole component together with its local routes as follows. It is important to note that, including the local routes has not increased nor characterization, nor computation time of the model.

The components considered here are arithmetic IP cores that are implemented as Relatively Placed Macros. It means that the position of each LUT relative to the position of any other LUT inside the core stays the same, regardless of the global position of the complete module on the chip. The LUTs are tightly packed as to achieve maximum performance, and minimum occupied area. As such, the local interconnections between the LUTs, are routed mostly with direct and double lines, as they are the shortest and the fastest connections. The LUT's propagation delay is greater than the propagation delay of a direct or double line, so the transitions can not be further filtered. Thus, the switching activity of each local line is equal to the activity generated at the output bit of the corresponding LUT (where the line begins). If we assume a unique value for the line capacitance C_{line} , equivalent to the effective capacitance when both types of wires are accounted for, the power of the local routes is:

$$P_{line} = 0.5 \cdot V_{dd}^2 \cdot f \cdot C_{line} \cdot SW = a_{line} \cdot SW \quad (4.61)$$

Similarly, when glitching effects are taken into account, an expression like (4.60) is obtained. Thus, it is assumed that the local interconnect power is proportional to the logic power, and the logic model can be applied to the whole component.

Logic power model for embedded blocks

The assumption about the regularity of the programmable elements can no longer be made for the purpose of the analysis presented here, as the embedded blocks are not implemented into standard FPGA fabric. However, as the compressor blocks are composed of repeated full-adder cells and the structure of embedded blocks is fixed and independent of its location in the FPGA, the approximation of a single effective capacitance for all the wires in the module is maintained in the model.

The total switching activity is obtained as the addition of the switching activities of the outputs and carry bits of all the basic cells in the component as explained in subsection 4.2.3.

Furthermore, dedicated multipliers are highly optimized for performance and power, so we assume that the amount of glitching inside the component can be neglected.

As a consequence, we use a single capacitance value for each element inside the embedded block, leading to a single coefficient a . However, as the inputs and outputs of the multiplier are registered, we use another capacitance value (i.e. another coefficient) for the outputs of the registers, since they are implemented in standard FPGA fabric. Both capacitance values are obtained from the multivariable regression over different power measurements for various signal statistics and multiplier sizes. Thus, the final model can be presented as:

$$P = a_e \cdot SW_e + a_r \cdot SW_r \quad (4.62)$$

where a_e and a_r are the coefficients representing the product of three power terms (V_{dd}^2, f, C_l) for the elements inside the embedded block and registers, respectively, and SW_e and SW_r are the total switching activities generated inside the embedded block and at the outputs of the registers, respectively.

The power model for embedded blocks has been presented in [JC09].

Cycle-by-cycle accuracy

In [JKSN99], it was demonstrated that the application of the Hamming distance distribution, rather than average values, increases the estimation accuracy when power has a non-linear dependency on the Hd . This is precisely the case in many DSP data-streams and data modules.

The Hamming distance distribution is obtained in the following way. For each two consecutive input vectors of both operands A and B, the Hamming and Signal distances are calculated. Hence, the number of appearances of each combination of Hd_A, Sd_A, Hd_B and Sd_B is available for a given data set. The products of the probabilities and the corresponding power values are added to form a new and more accurate power estimate.

We have applied the same methodology on the logic power model for LUT-based components, but instead of computing Hd and Sd , we have classified each two consecutive input

vectors as belonging to different Gaussian distributions, depending on the number and the value of the MSB bits that are the same in both vectors. For example, if we have the following three vectors:

$$\begin{array}{l} 010010110 \\ 010011010 \\ 010110100 \end{array} \quad (4.63)$$

then we say that the first two belong to the Gaussian distribution with the most significant mean bits equal to 01001, and the second and the third belong to the Gaussian distribution with the most significant mean bits equal to 010. In both cases, the first bit that stands immediately after the mean bits, changes with a switching activity of 1, and the rest of the bits behave as uncorrelated, random bits with a switching activity of 0.5. Based on this classification, we have applied the power model described in this section to each Gaussian distribution detected in the input data set and the corresponding power value was summed to the expression for the final power estimate, according to the number of input vectors associated to it:

$$P = \sum_i P_i \cdot \frac{n_i}{N - 1} \quad (4.64)$$

where N is the total number of vectors in the input data set, n_i is the corresponding number of vector pairs belonging to the particular Gaussian distribution, and P_i is the corresponding power value computed as in (4.60).

This expression enabled us to improve the accuracy of the logic power model for LUT-based components at the cost of increased computation time as it will be explained in detail in the next section.

4.5. Experimental results

Experimental results are divided into three sets. In the first set, we explore the accuracy of the expressions given for the breakpoints according to the ARMA signal model and according to the Dual-bit type method. We test the two models for various input signal statistics. In the second set, we evaluate the logic power model developed for LUT-based components against XPower low-level estimates. Additionally, we compare this model to the two other proposed models in the literature [CGC05] and [HSS⁺02]. Finally, in the third set, we compare both, the logic power model for LUT-based components and the power model for embedded blocks, against on-board measurements.

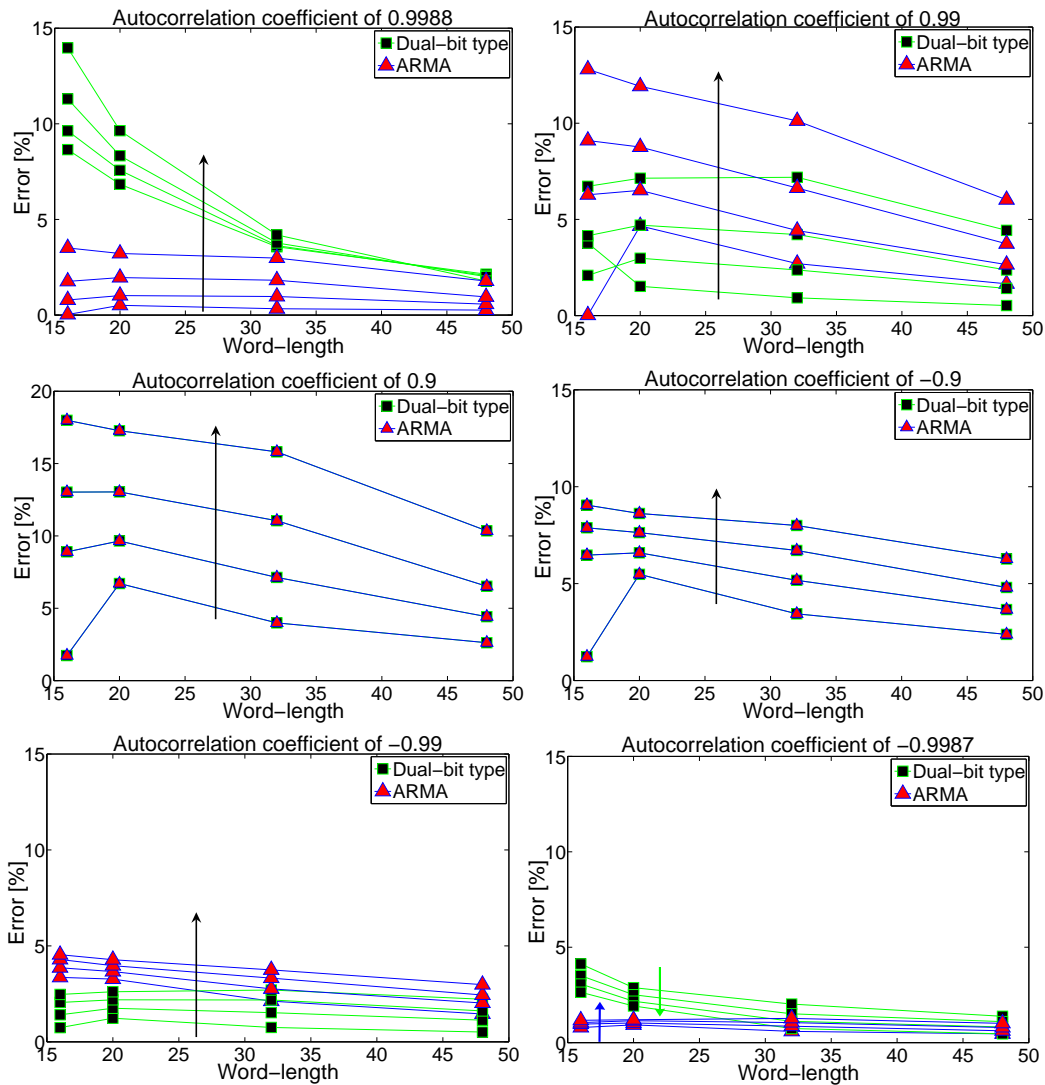


Figure 4.18: Errors for two signal models: ARMA and Dual-bit type, when considering zero-mean gaussian signals and various sizes of the MSB activity region. The arrow indicates how errors behave when the number of sign bits increases

4.5.1. Signal model accuracy

In this work we focus on gaussian signals as they can faithfully represent the input signals to DSP systems. We use the methodology for computing the breakpoints presented in [RSH97]. However, as they derive the breakpoints for ARMA(N,M) signal generation model, the results presented in their work include various signals that can be obtained by varying parameters M and N . Among the considered signals in their work, only two correspond to the gaussian distribution. Furthermore, although they point out the drawback of the expression given in [LR95] for the breakpoint $BP1$ when non-zero mean signals are considered, they do not compare their results with the dual-bit type method [LR95].

We divide the experiment into two subsets. In the first subset we consider zero-mean, and

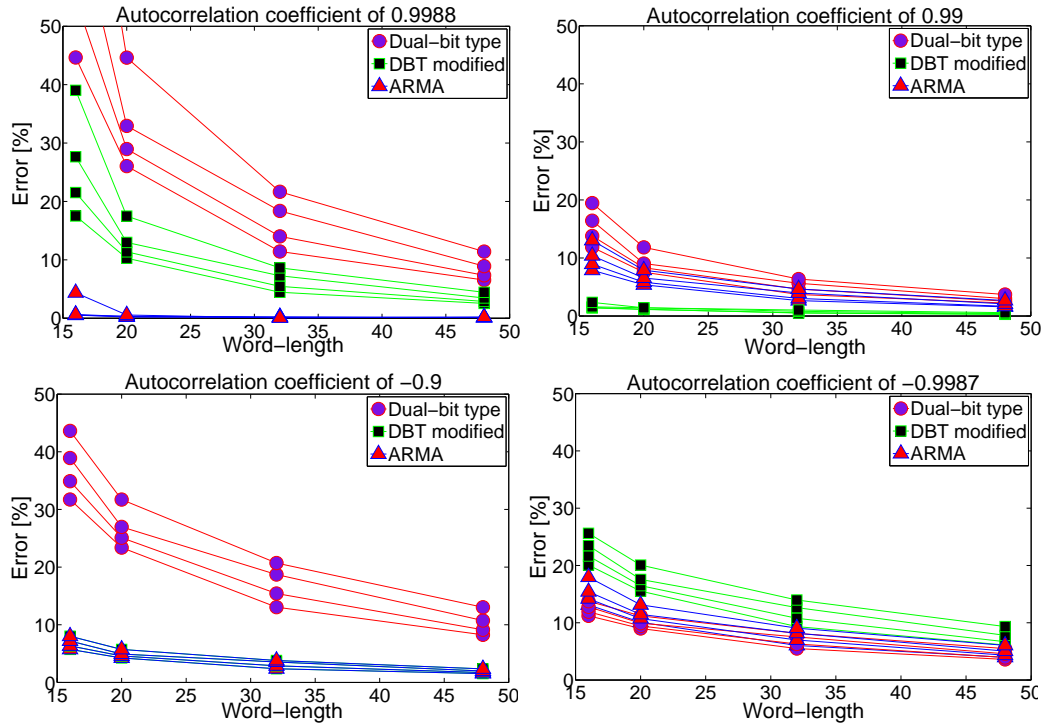


Figure 4.19: Errors for three signal models for gaussian signals with mean equal to 10

in the second, non-zero mean signals. In both subsets, signals with various autocorrelation coefficients (between -0.998 and +0.998) and word-lengths (16, 20, 32 and 48 bits) are explored. Additionally, we also vary the number of the sign bits in a signal word (i.e. the size of the MSB activity region) between 8 and 24 depending on the word-length. For each autocorrelation coefficient and word-length four different MSB sizes are chosen.

For each signal we compute the total switching activity as the sum of the switching activities of the signal bits obtained according to both models and compare them to the total switching activity measured through the simulation of the input vectors.

In the continuation, we list the expressions for the breakpoints computed in [LR95].

The expression given in [LR95] for the breakpoint $BP0$ is:

$$BP0 = \left\lceil \log_2 \left(\sigma \cdot \left(\sqrt{1 - \rho^2} + \frac{|\rho|}{8} \right) \right) \right\rceil \quad (4.65)$$

The expression given in [LR95] for the breakpoint $BP1$ is:

$$BP1 = \lceil \log_2(\mu + 3\sigma) \rceil \quad (4.66)$$

In Fig. 4.18 we present the comparison of the dual-bit type method and the method presented in [RSH97] (referred to as "ARMA") after it was adapted to gaussian distributions as explained in section 4.1. The results are given for the first experimental subset where we consider zero-

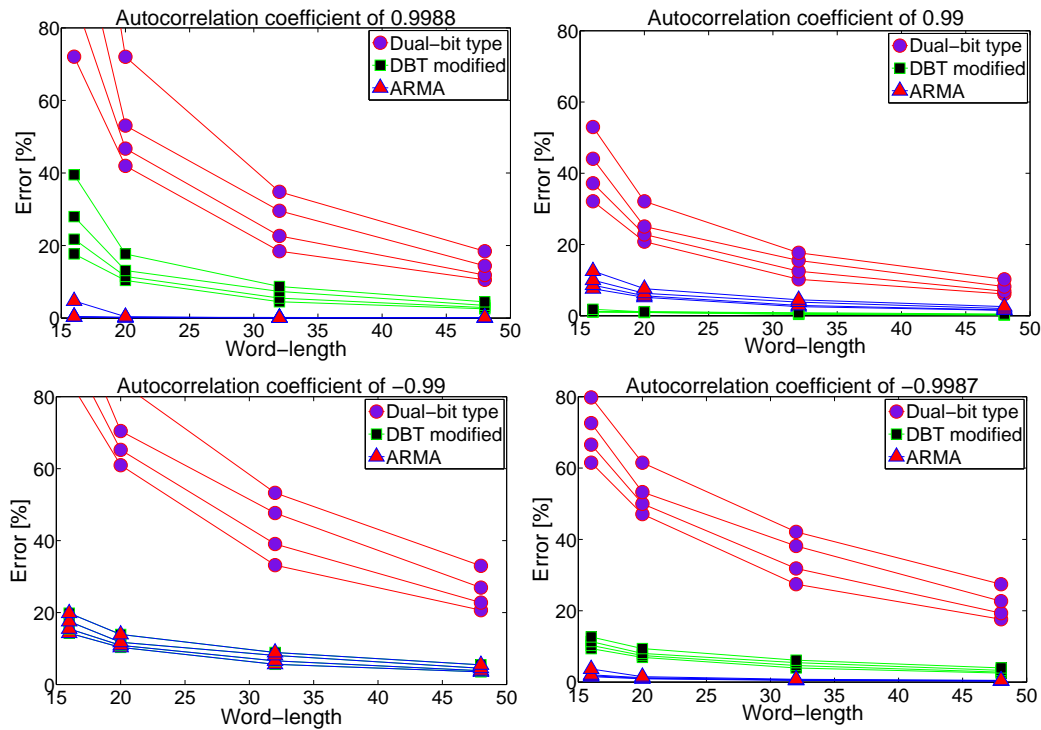


Figure 4.20: Errors for three signal models for gaussian signals with mean equal to 125

mean gaussian distribution. All the errors are presented with their absolute values for the sake of clarity. It can be seen that the models have a similar accuracy, while the ARMA model achieves better accuracy for the highest autocorrelation coefficient. This value corresponds to the autocorrelation coefficients of many audio and image signals as it will be presented later. It is also observed that both models achieved the same accuracy for the values of the autocorrelation coefficients of 0.9 and -0.9. This is because the term equal to $|\rho|/8$ in the expression for the breakpoint $BP0$ in (4.65) is too small for small values of autocorrelation coefficients, leading to the same expressions for $BP0$ in both models. Additionally, the accuracy of the models seems to worsen with larger number of sign bits in most cases. For the cases where the accuracy does not follow this pattern, the arrows are given separately for each model and have a colour corresponding to the colour of their graphs. The direction of the arrows in the figures indicates the direction of the increase in sign bit number.

It is important to note here that it was also observed that ARMA signal model tends to underestimate the total switching activity rather than overestimate (this effect is not shown in the figures, since absolute values of the relative errors were used for the sake of clarity). This is significant for the analysis of the power model behaviour, and it will be explained in the next chapter.

The second subset is designed in order to explore the accuracy of the two models when the input signals have non-zero mean gaussian distribution. We include a third model in the analysis

presented in this subset, which is derived from the Dual-bit type method as follows.

As already seen, the expression given in [LR95] for the breakpoint $BP1$ depends on the mean of the signal. However, the breakpoint $BP1$ is determined by the signal variation around the mean and thus, is independent of the value of the mean [RSH97]. For this reason, we have included the so-called "Modified Dual-bit type model" in the comparison for non-zero mean signals, where the value of the $BP1$ is computed as:

$$BP1 = \lceil \log_2(3\sigma) \rceil \quad (4.67)$$

This expression provides the same value for zero-mean gaussian signals as 4.66, so only one of these models is included in the results for the first experimental subset.

In Figs. 4.19 and 4.20 we give the error performance of the three models: dual-bit type, the modified DBT, and ARMA model, for two different values of mean: 10 and 125, respectively. The directions of the sign bit number increase/decrease are not marked in the figures, as it was noted that the accuracy of the models worsens with a larger number of sign bits in all cases. It can be seen that the Dual-bit type performs poorly, as it should be expected since breakpoint $BP1$ takes a wrong value due to the value of the mean. However, the modified DBT and ARMA models achieve similar accuracy. Still, the maximum error for the ARMA model does not surpass 20 % while the Modified DBT goes up to 40 % in some cases.

Consequently, we have chosen the ARMA signal model for the purposes of word-level power estimation presented in this thesis.

4.5.2. Logic power model for LUT-based components : XPower

In this experimental set, the accuracy of the logic power model for LUT-based components is compared to the accuracy of two other logic power estimation models found in the literature [CGC05] and [HSS⁺02]. The table-based model given in [CGC05] has been calibrated by using low-level power estimates given by XPower, and the coefficients used in their model were downloaded from the web address provided in their paper. The calibration of the Hamming distance model presented in [HSS⁺02] was also performed with the estimates given by XPower since it is extremely difficult to use on-board measurements for its characterization, as it will be explained later. Hence, in this subsection we also calibrate the logic power model for LUT-based components presented in this thesis (HLLM) by using XPower estimates for the purposes of the comparison with the other two power models.

The authors in [WFDA06] used both, XPower estimates and hardware measurements, in order to identify the power difference between designs with different placement and routing constraints. They refer to a design without any constraints applied to it as a baseline version, while the constrained designs represent optimized versions. Although they did not report the

results obtained from the measurements due to variables such as room temperature and device fabrication variances, they assure that the percentage power reduction between the optimized and baseline versions remained constant between XPower software reports and hardware measurements in the experimental testing.

First, we give the comparison of the HLLM with the table-based method presented in [CGC05]. For this purpose, we use three different models for multipliers implemented in LUTs, in order to investigate the grade of accuracy improvement when glitching generated inside the component is included into the model. Next, we present the error performance for the logic power model proposed here when considering non-zero mean signals, as this analysis is necessary for the third set of experiments, where HLLM is compared to the Hamming distance power model for various real-world data signals.

Comparison with [CGC05]

The following experiments have been performed to verify the proposed logic power model for arithmetic components implemented in FPGAs. Two types of experiments have been considered: one where the word-lengths of both input signals are the same while the size of the component is varied together with the input signal statistics, and the other where one of the input word-lengths is varied while the signal statistics remain fixed. The experiments have been performed on multipliers and adders implemented as IP Cores in Xilinx Virtex-2 XC2V2000-5 devices, as these devices have been also used in [CGC05]. The design frequency used in all experiments was set to 100 MHz. Different autocorrelation values between 0 and 0.9995 were used in the comparison. The signals used for experiments had zero-mean Gaussian distributions. The test-benches for arithmetic components with input word-lengths smaller than 55 bits were generated using Matlab, whereas the 64-bit numbers were generated using functions provided in [lib] Multiple Precision Arithmetic Library (GMP). The signal generation model described in 4.1.1 was used in order to generate the input signals with the specified autocorrelation coefficients. Each operand signal word was divided into regions according to equations 4.18 and 4.23.

All the estimated values have been compared to low level power estimated values obtained from the XPower tool (from ISE 7.1 as this version was also used in the work presented in [CGC05]). Power estimation values are given for the Xilinx cores and thus, refer only to the DSP component structure which is used for the implementation of cores in Xilinx FPGAs (for example, the row adder tree multiplier for the multipliers).

The first set of experiments assumes multipliers and adders with operands with the same word-lengths. The results in [CGC05] relate to tables of coefficients used to obtain the power consumption, and a clock frequency is required to perform the appropriate computations. Since such information is not included in their work, we assume a frequency of 25MHz for multi-

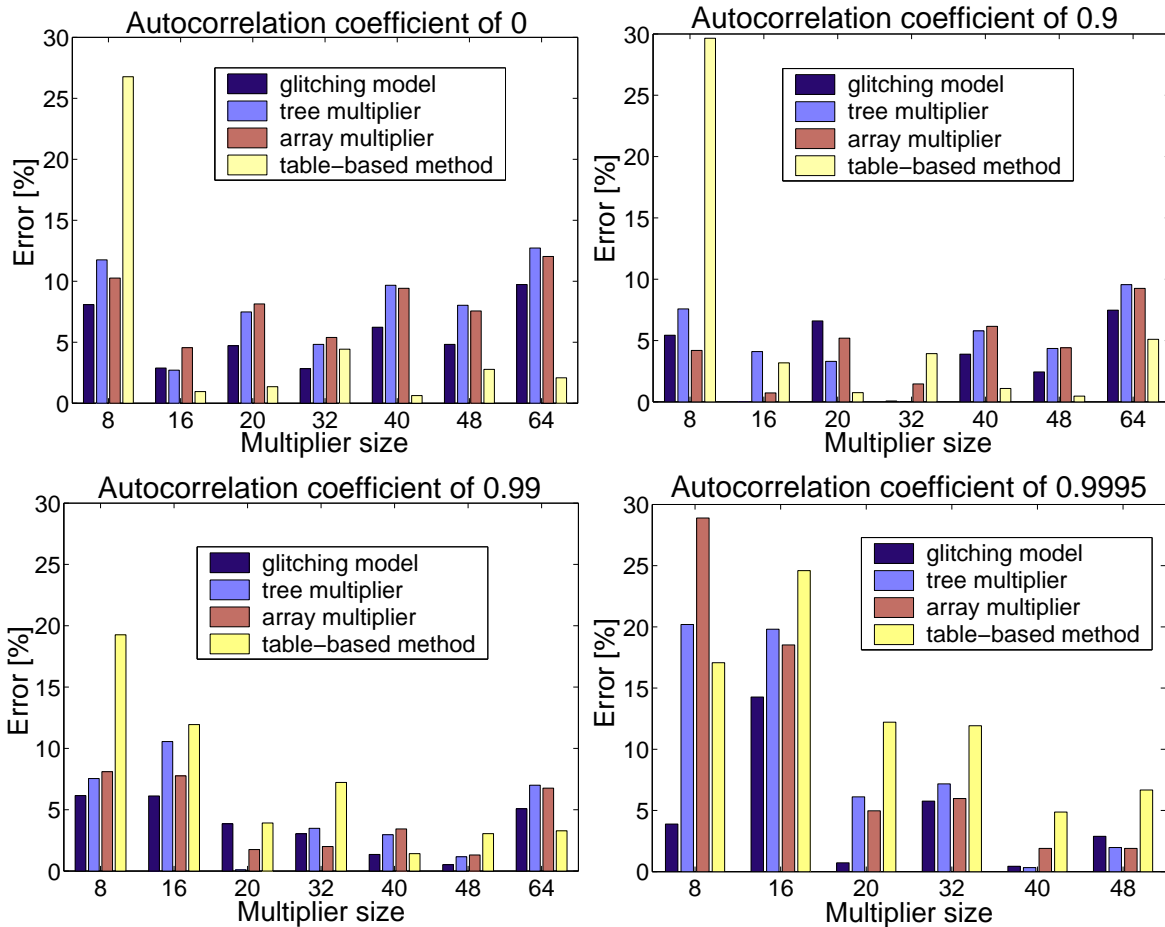


Figure 4.21: Error performance of multipliers for various autocorrelation coefficients

pliers and 100MHz for adders, as these are the ones providing the results that are closer to the values obtained from XPower. Input word-lengths are varied between 8 and 64 bits. The errors obtained for multipliers are given in Fig. 4.21. Four models are taken into consideration: 1) the model that considers both glitching effects and implementation details (presented by 4.60); 2) the model for a row adder tree multiplier (the switching activity presented by 4.46 is multiplied by a constant representing the product of three power terms); 3) the model for an array multiplier (the switching activity presented by 4.45 is multiplied by a constant representing the product of three power terms); and 4) the table-based model described in [CGC05] (presented by 4.59). The expressions given in 4.49 and 4.50 are used for estimating the amount of glitching in the multiplier and adder respectively, as they refer only to zero-mean signals. The improvement rate of this glitch model by using the equation 4.52 is given in the next subsection.

It can be observed that in most cases the estimate provided by the model with glitching effects is accurate up to 10% of the value obtained by XPower. Besides, it clearly outperforms the models that do not consider glitching effects. The table-based method gives good results when considering large word-lengths, while the error is greater than 20% for 8-bit multipliers.

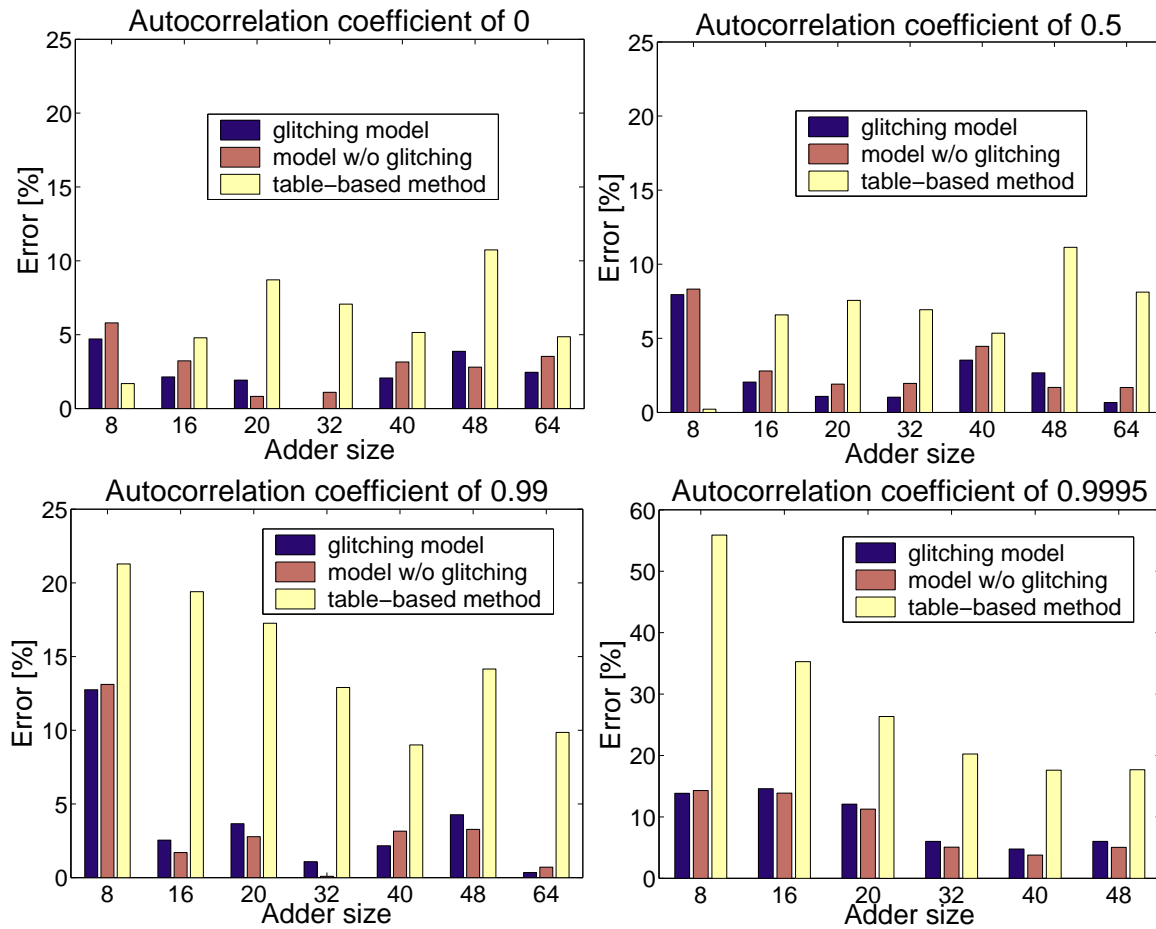


Figure 4.22: Error performance of adders for various autocorrelation coefficients

We believe that this is due to the quadratic nature of this model which is too simple for modelling the power consumption of multipliers.

Next, in Fig. 4.22 we present the errors obtained for adders. The results are given for three different power consumption models: 1) the model that considers glitching effects (presented by 4.60); 2) the model for the ripple-carry adder (the switching activity presented by 4.48 is multiplied by a constant representing the product of three power terms) and 3) the table-based method (presented by 4.58). Again, it can be observed that in most cases the estimate provided by the model with glitching effects is accurate up to 10% of the measured value and that the error of the table-based method increases substantially for small word-lengths.

The second set of experiments evaluates the error performance for DSP components where one of the inputs is first fixed to 48 bits while varying the other from 8 to 40 bits and then changed to 32 bits while varying the other from 8 to 20 bits. This set is designed to prove the advantage of HLLM in providing accurate estimations for DSP components with different operand sizes, without the need of a different power model.

First, the error performance is given for the three proposed models for multipliers. The

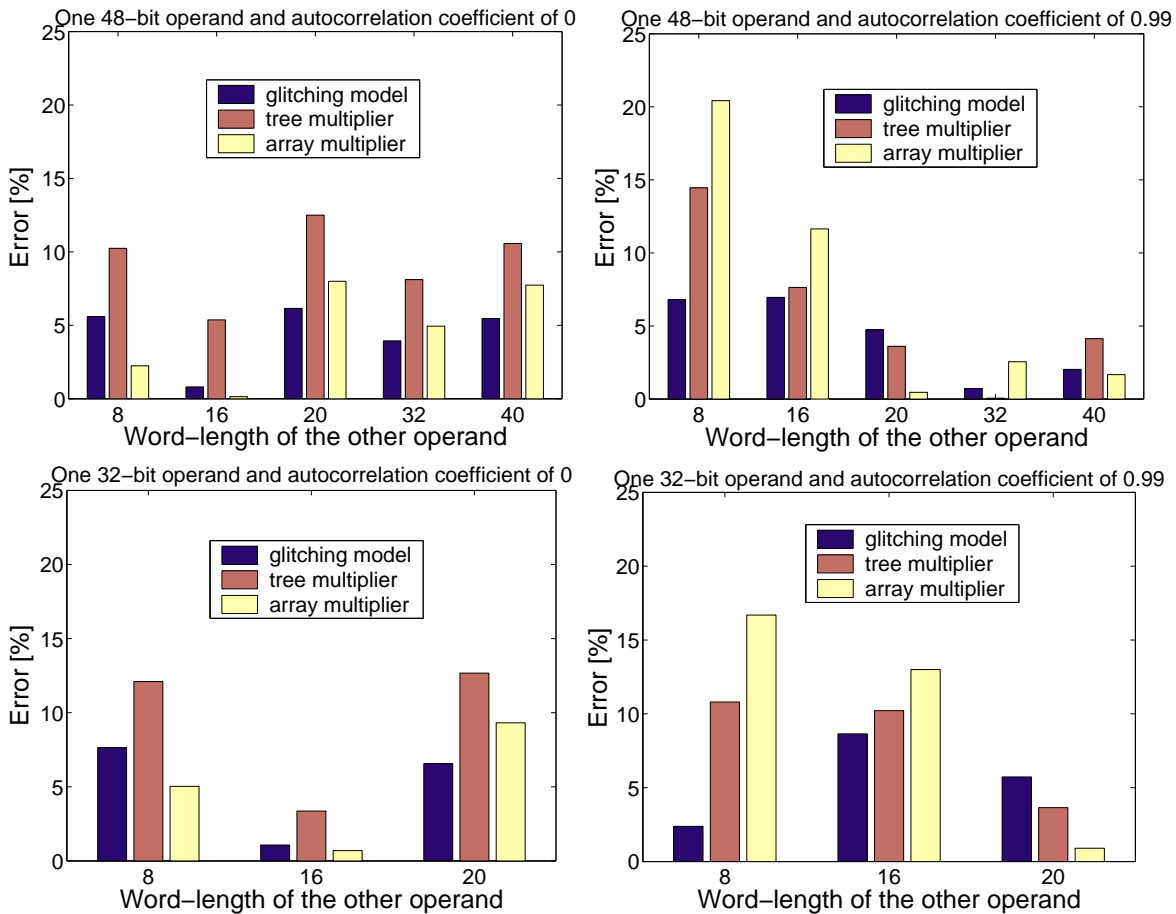


Figure 4.23: Error performance for multipliers with operands of different sizes

results are shown in Fig. 4.23. It can be seen again that the model considering glitching effects outperforms the other models. When comparing the "tree multiplier" and the "array multiplier" models, it can be noted that the latter has a larger maximum error, but better error performance. This can be explained by the fact that this model considers separately the transition activities at the outputs of the AND gates and the outputs of the full-adder cells. The transition activities of these elements in turn contribute to the glitching activity of the multiplier, so the "array multiplier" model indirectly takes into account some of the glitching effects, whereas this does not occur in the "tree multiplier" model.

Fig. 4.24 represents the errors obtained for adders. We consider the same power models as in Fig. 4.22. It can be seen that the models proposed here are highly accurate, but behave similarly in many cases. This is probably due to the existence of only one logic level in the case of adder components, so glitching has a smaller effect on the total power consumption. On the other hand, the table-based method gives large errors, especially for the adders where the other operand is sign-extended for a large number of bits. This is due to the simplicity of the equations described in the table-based method. As they depend only on the autocorrelation

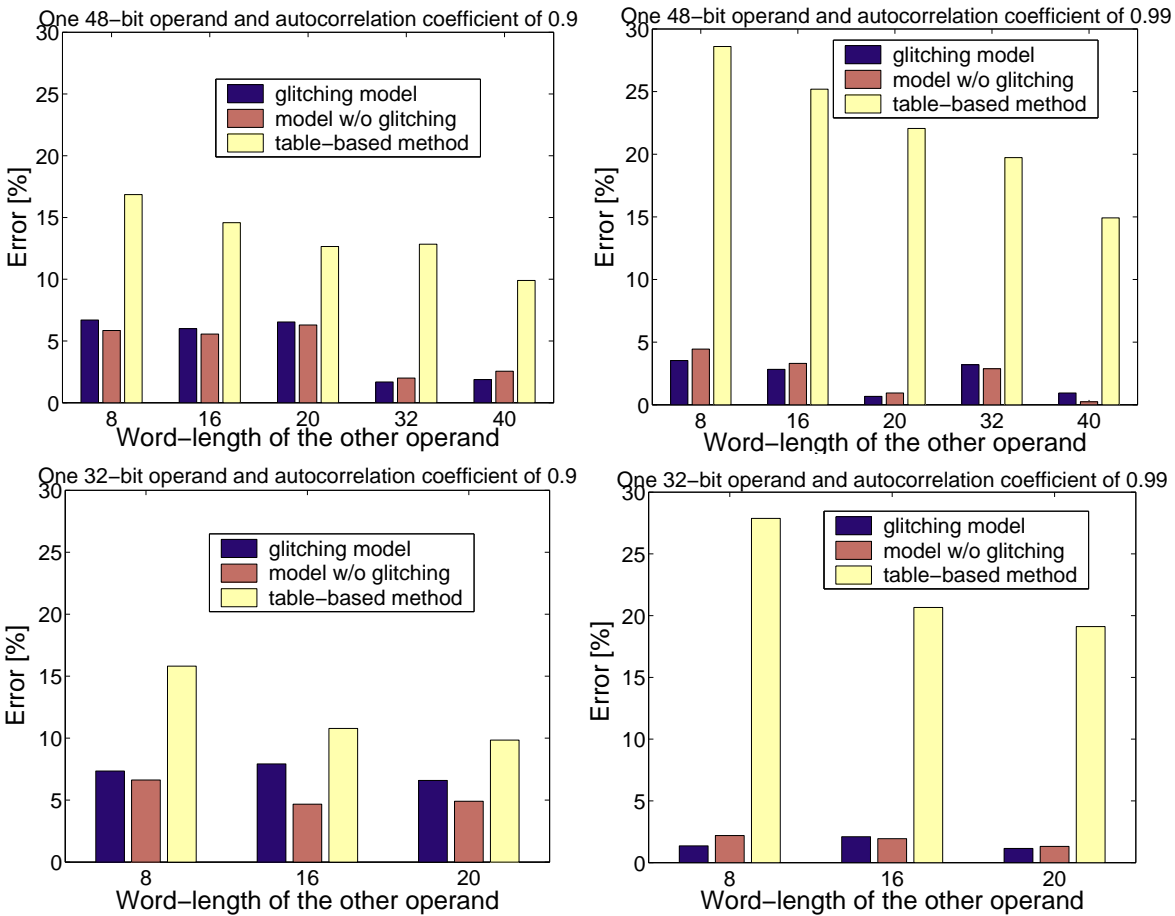


Figure 4.24: Error performance for adders with operands of different sizes

coefficient and the size of the adder, the same power value is obtained for any adder of a specific size regardless of the number of the sign bits of its operands.

High-level logic model accuracy for non-zero mean signals

We split the model evaluation into two sets of experiments. Both sets of experiments have been performed on multipliers and adders implemented as Xilinx IP Cores in Virtex II devices. All the estimated values have been compared to low level power estimates provided by the Xilinx tool XPower [Xil]. The signals used as input stimuli had Gaussian distributions with means equal to 0, 10 and 125 respectively. We have chosen these values for the mean in order to see the difference between the power values obtained for signals with many and with a few '1's in their mean. We have used 16×16 and 32×32 multipliers and signals with zero-mean gaussian distributions and autocorrelation coefficients of 0, 0.9, -0.9 and -0.99 for our characterization set. Multiple regression over the relative errors was performed for obtaining the constants b and k .

In the first set we evaluate the accuracy of the power estimation model for non-zero mean

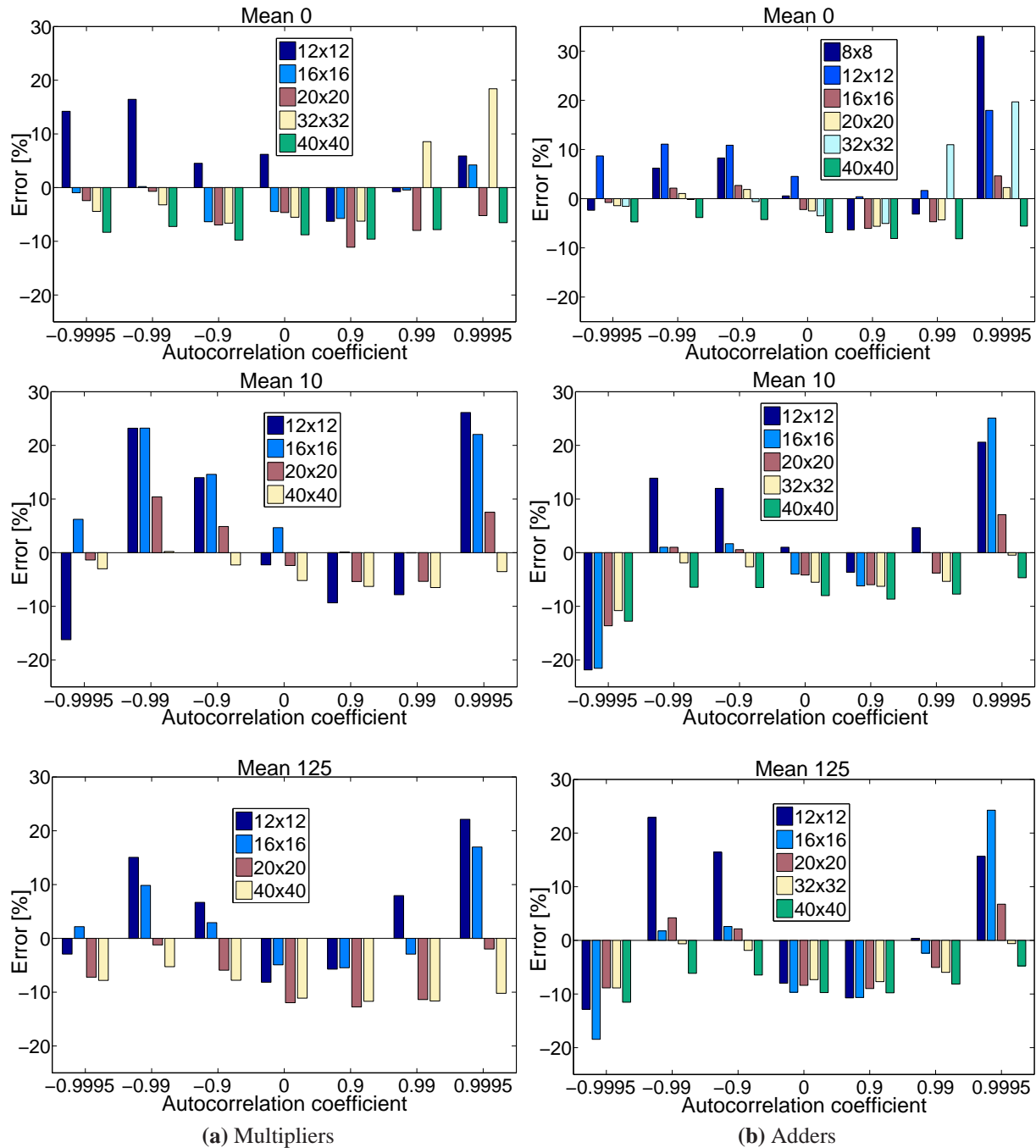


Figure 4.25: Error performance for multipliers and adders with operands of the same size

signals presented in 4.60 (where the expression 4.52 is used for glitching estimation) considering both, components with inputs of the same size, and components where the input bit-widths differ. Fig. 4.25 presents the estimation errors for multipliers and adders when operands have the same word-lengths. Input word-lengths varied between 8 and 40 bits, and autocorrelation coefficients varied between -0.9995 and 0.9995 . It can be noted that a similar error performance is obtained for all adders when signals with mean 10 and mean 125 are applied to its inputs.

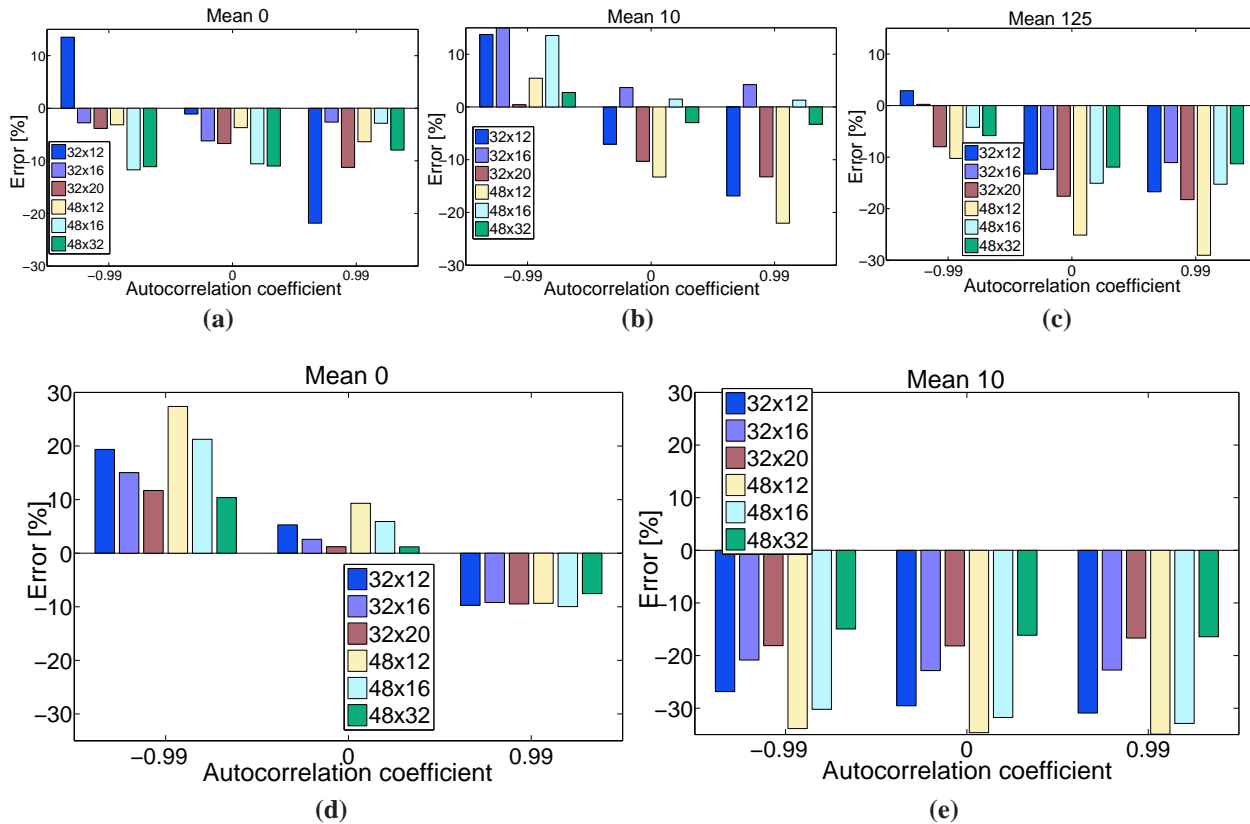


Figure 4.26: Error performance for multipliers - a), b), c) and adders - d), e) with different size operands for various signal statistics

This is a direct consequence of a feature of the operation *add*. As the mean bits do not change their value, when added, they will give the same result at the outputs of the full-adder cells, and hence, there will be no switching activity in this part of the adder. This result has also been confirmed by the identical XPower values.

Next, Fig. 4.26 shows the errors obtained for multipliers and adders with different operand sizes. The experimental set includes three autocorrelation coefficients of -0.99, 0 and 0.99. Again, the adder errors for mean values of 10 and 125 were almost the same, so we have included only one of the plots (Fig. 4.26e). Another important observation from this figure is that the power model for adders clearly underestimates when non-zero mean signals are applied to its different-size inputs. This is due to the fact that the glitching model does not consider the amount of glitching produced in the part of the adder where the input bits of one operand belong to the mean region, while the input bits of the other operand belong to the LSB or MSB region. In reality, there will be glitching in this part of the adder due to the different arrival times of the carry bit and the input bits of the other operand.

Overall, the mean relative error for multipliers is 8.34% and for adders 11.14%. It can be seen that the models are capable of providing quite accurate results over a wide range of operand

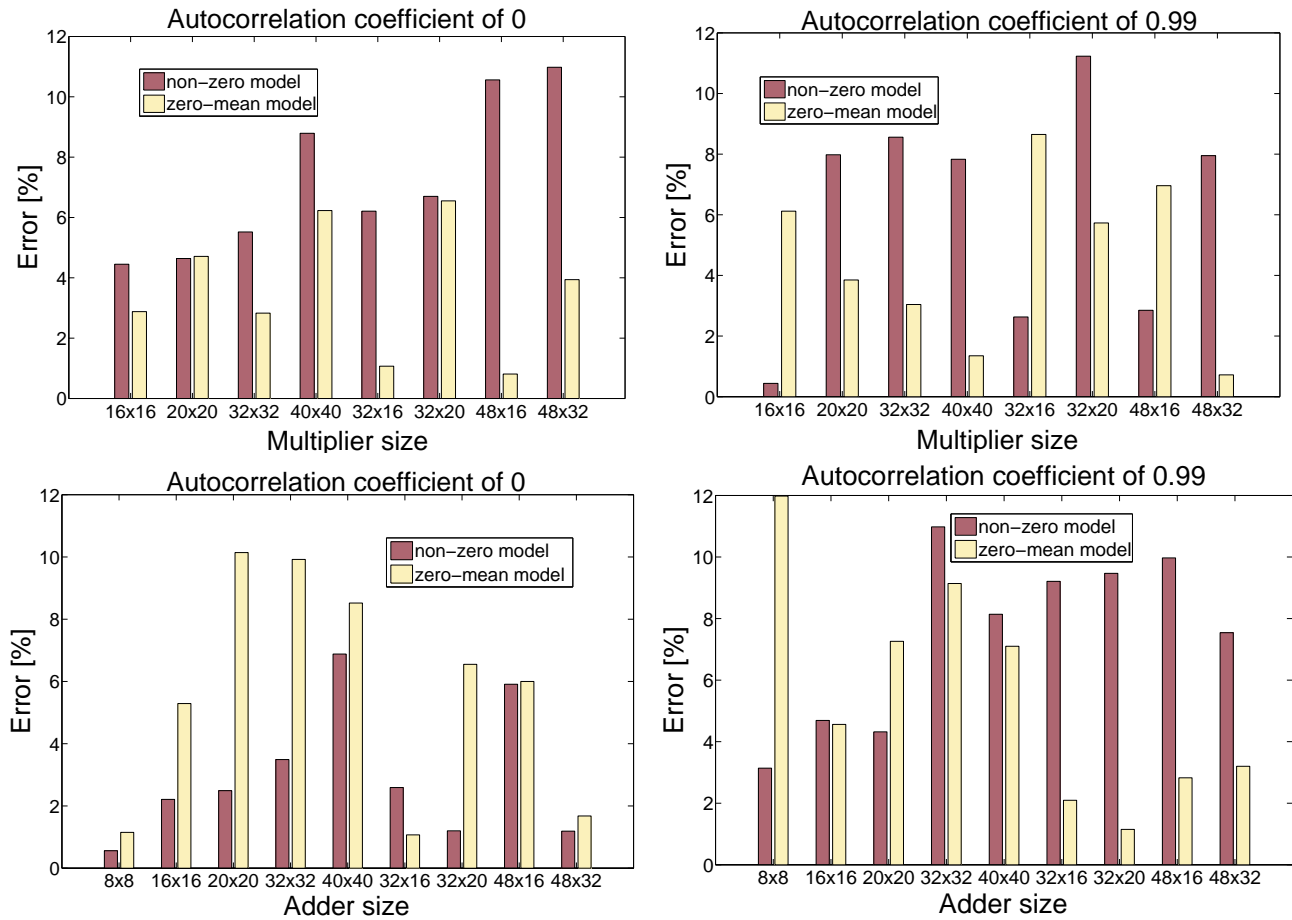


Figure 4.27: Error performance for multipliers and adders for various signal statistics considering two approaches: the model presented here and the model presented in [JCC07b]

sizes, signal autocorrelation coefficients and mean values.

In the second set, we focus on evaluating the same power model (presented in 4.60 with the expression 4.52 for glitching estimation), against the estimates obtained by the power model used for the comparison with [CGC05] where the simplified expressions are used for glitching (4.49 and 4.50). This experimental set evaluates adders and multipliers with input word-lengths between 8 and 40 bits. The signals are assumed to have zero-mean and the autocorrelation coefficient values are only positive and vary between 0 and 0.9995 to allow fair comparison with the model with simplified expressions for glitching.

For an easier comparison, we give the absolute value of the relative errors in Fig. 4.27. It can be seen that the accuracy of the new model does not worsen when positive autocorrelation coefficients are considered, while providing power estimates for a significantly larger set of input signal parameters.

Hd and HLLM model comparison

The experiments have been designed in order to present the comparison of the HLLM and the Hd-model for signals taken from real-world applications. They have been applied to multipliers and adders implemented as Xilinx IP Cores in Virtex II devices. All the estimated values have been compared to low level power estimated values obtained from the Xilinx tool XPower [Xil].

The characterization set for the construction of the logic power model presented here is based on 16×16 and 32×32 components and signals with zero-mean gaussian distributions and autocorrelation coefficients of 0, 0.9, -0.9 and -0.99. A typical characterization set mentioned in section 4.4, has been used for the Hd-model construction.

The experiments have been carried out for five different types of input stimuli. The pattern set includes:

- 1) row speech signal
- 2) image signal
- 3) memory access index (counter-like signal)
- 4) randomly chosen signal variable in a C-code FDCT
- 5) uniform white noise

Each of the power models has been used in two different ways in order to obtain the estimation errors for the given input data. The first one takes average values of the signal statistics for the whole input data set (marked as Average in Tables 4.5, 4.6 and 4.7), while the second one considers cycle-by-cycle input signal characteristics (marked as Cycle), as explained in section 4.4.

Tables 4.5 and 4.6 present the estimation errors obtained for multipliers and adders respectively. It can be seen that the Hd-model gives good estimates for all signals, except for type III. As this type represents a counter-like signal, it strongly differs from the characterization patterns that are normally composed of '1's and '0's randomly distributed in a signal-word with the bits that are switching also located at randomly distributed bit-positions. On the other hand, a counter-like signal has established bit positions of '1's and '0's and the bits that are switching are determined. Thus, applying the cycle-by-cycle power computation, barely improves the accuracy of the model.

The HLLM with cycle-by-cycle computation gives good results for all signals except when considering multipliers for signals of type IV. We have observed that this is entirely due to the nature of the signals in the FDCT. As the bit switching activity is distributed over the bit-positions in a random fashion, signal-word decomposition explained in section 4.1, can not be performed in this case. This is the reason for equally poor performance when considering average and cycle-by-cycle signal distributions. It can be also noted that cycle-by-cycle computation improves the accuracy of the logic power model up to 60% for type III.

In the continuation, we extend the comparative analysis of the two high-level power estima-

Table 4.5: Comparison of two models for multipliers.

Data types	Hd-model error [%]		HLLM error [%]	
	Average	Cycle	Average	Cycle
I	7.41	4.25	12.65	-8.8
II	2.97	-3.96	-10.03	-10.23
III	43.75	35.33	47.71	-11.87
IV	11.07	0.63	31.03	33.1
V	-5.25	-8.23	-10	-0.29

Table 4.6: Comparison of two models for adders

Data types	Hd-model error [%]		HLLM error [%]	
	Average	Cycle	Average	Cycle
I	3.98	1.58	11.87	4.88
II	-12.03	-14.75	-4.47	1.3
III	22.74	18.62	63.8	-0.21
IV	5.64	-0.49	12.73	3.46
V	-3.56	-3.42	-4.81	0.59

Table 4.7: Comparison of the two models when component size differs from the input signal bit-width

Data types	Hd-model error [%]		HLLM error [%]	
	Average	Cycle	Average	Cycle
Mul.-II	56.73	55.18	3.24	-13.67
Mul.-III	41.83	33.94	47	-12.58
Add.-II	10.5	9.36	-4.44	1.33
Add.-III	36.88	34.2	63.9	-0.15

tion models with four additional aspects that try to establish the applicability of the approaches in real world situations.

The first aspect is the computational effort used for the model characterization and utilization. The number of simulations needed for the Hd-model construction was 225 for each component with specified operand sizes (according to (4.57)), while only 8 simulations were needed for the construction of HLLM. It can be seen that the Hd-model is extremely dependent on the accuracy and time performance of the low-level simulation tool as it requires a large number of low-level simulations. The best accuracy is achieved when the model is characterized with on-board power measurements. FPGA power measurements need to be carefully prepared

Table 4.8: Computation times for the two models applied to 16×16 multiplier

Data types	Number of vectors	HLLM model		Hd-model	
		Average	Cycle	Average	Cycle
II	15100	0.25 s	10 s	0.00005 s	0.055 s
IV	297	0.25 s	0.85 s	0.00005 s	0.013 s
V	10000	0.25 s	4.32 s	0.00005 s	0.061 s

and processed in order to obtain separate interconnect and logic power values (as explained in chapter 2), thus, making the automatization of the measurement process extremely difficult. In the case of the logic power model presented here, the number of simulations needed for model characterization is highly reduced. Hence, it can be directly based on power measurements leading to better accuracy.

There is also a difference in the computational effort required by each model when cycle-by-cycle signal statistics are taken into account. The experiments were performed on a Pentium 4, at 3.00 GHz with 1GB of RAM. In the Hd-model, the parameters Hd and Sd have to be computed for each pair of consecutive vectors, meaning that the computation includes all the bits in each signal-word. In HLLM, this number is reduced to the number of the most-significant bits that have the same value in both vectors. However, when equation (4.64) is applied, the values of P_i are taken directly from the table or interpolated from the neighbouring table values in the case of Hd-model, while they need to be computed for HLLM. When these two effects are taken into account, the Hd-model has some advantage over HLLM, although the difference is barely noticeable when average signal statistics are taken into account as each estimate P_i only takes a few milliseconds in the worst case. The final computation time for cycle-by-cycle computation depends on the number of consecutive input vector-pairs in the input data set. Table 4.8 presents computation times for both, HLLM and Hd, when both, average and cycle-by-cycle, signal statistics are applied to a 16×16 multiplier with several different real-world input signals. When considering cycle-by-cycle statistics, HLLM computation time is signal dependent as the difference in times between type II and V is not proportional to the difference in the corresponding number of input vectors. It can be also seen that it is necessary to take into account the trade-off between accuracy and computational time when cycle-by-cycle methodology is applied, as the computation times increase significantly with a larger number of consecutive vector pairs.

The second aspect to be considered is the model accuracy. Tables 4.5 and 4.6 confirm that the Hd-model gives better estimates than the logic power model presented here for most real-world applications, specially in the cases where the cycle-by-cycle accuracy is exploited. The mean relative error for the Hd-model excluding data type III, is 4.66% for the cycle-by-cycle

Table 4.9: Summary of features for both models when applied to real-world applications.

Metrics	Hd-model		HLLM	
	Average	Cycle	Average	Cycle
Accuracy	6.5 %	4.66 %	20 %	4.77 %
Characterization effort	225	225	8	8
Computational effort	$\sim 100 \mu s$	$\sim 100 \mu s + 100 \mu s$	$\sim 10 ms$	$\sim 1 ms + 100 \mu s$
Resource sharing	NO	NO	YES	YES
Component structure	YES	YES	NO	NO

and 6.5% for the average model. On the other hand, HLLM shows a 4.77% mean relative error for the cycle-by-cycle and a 20% error for the average model excluding data type IV.

The results in Tables 4.5 and 4.6 are given for components where the size of the input operand was adjusted to the input signal-word size. However, when resources are shared, it is often the case that smaller word-length input signals enter larger word-length component inputs. Thus, the third aspect is the model accuracy when resource sharing is considered. In this case, HLLM will still provide the correct power estimate (i.e. the parts of the component that are not exhibiting any switching activity will not contribute to the total power). The Hd-model will also account for this difference through the Hd and Sd , as they will decrease with respect to the full input length. However, the Hd-model is characterized assuming that the bits that are switching are located at randomly distributed bit positions, but in this case, the bits that are switching are all located at the LSB positions in the signal word. Hence, the Hd-model will tend to overestimate the power consumption. Table 4.7 shows the errors for the Hd-model and the logic power model presented here when 8-bit signals of data type II and 13-bit signals of data type III are used as inputs of 16×16 multipliers and 16×16 adders. It can be seen that the Hd-model error in most cases increases significantly with respect to the values in Tables 4.5 and 4.6, while HLLM maintains its accuracy.

The final aspect is the model construction for different component structures. The models used for the comparison presented here had the internal architecture of a row-adder tree multiplier and a carry-skip adder, as these structures are used for the implementation of cores in Xilinx FPGAs. The Hd-model methodology does not depend on the component structure, and as such can be easily adjusted to any given component. On the other hand, the component structure is taken into account for the switching activity computation in HLLM. Thus, every time some component is replaced by a module with the same functionality, but different structure, the analytical computation method has to be specially adapted to the new features of the component's internal architecture.

Table 4.9 summarizes the comparison between the two high-level logic power models. The results of the comparison were presented in [JCH08].

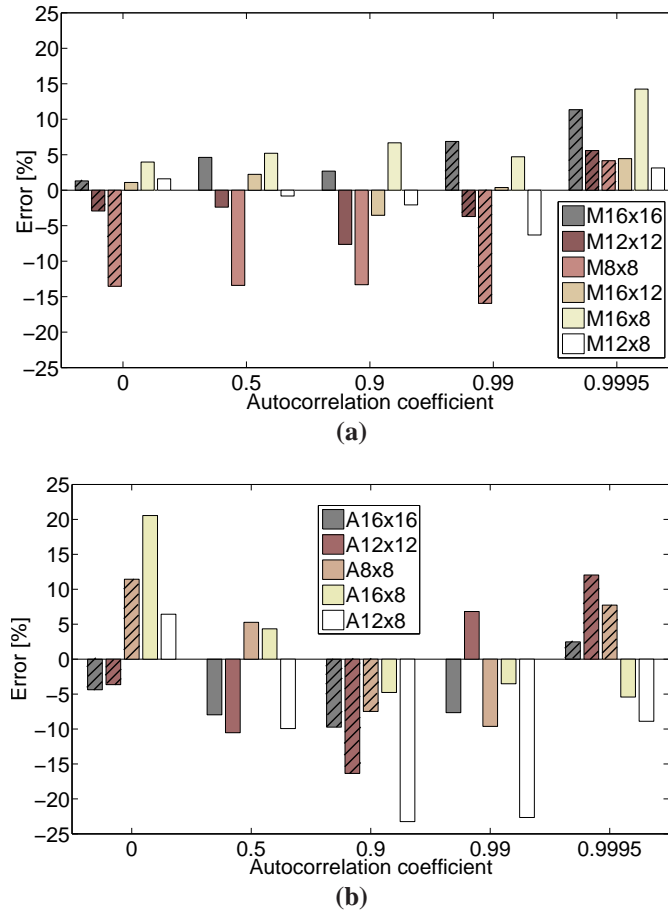


Figure 4.28: Errors with respect to measurements of the logic model for a) multipliers, b) adders

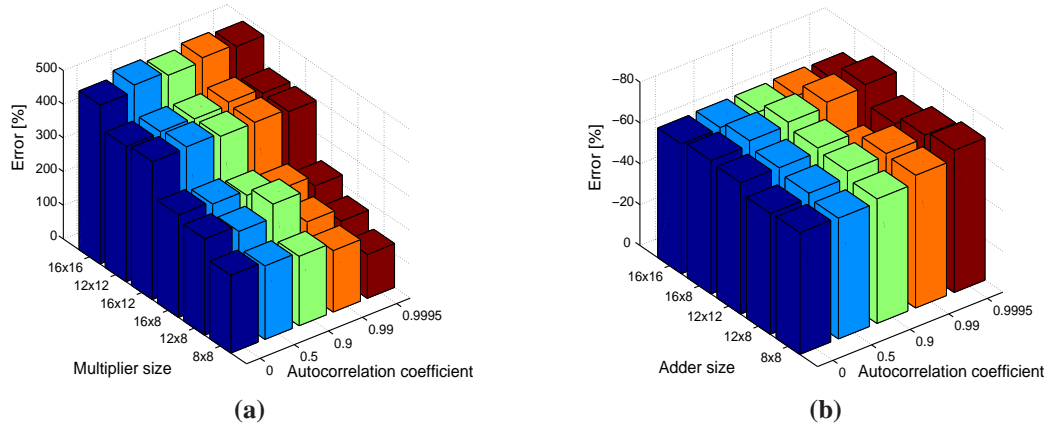
4.5.3. Logic power model for LUT-based components: on-board measurements

So far, the logic power model has been applied only to the power consumed in the logic, without considering the power of the local routes. Calibration and comparison with XPower has been easy, since all power values were taken from the Logic power group in XPower report.

Here, we have applied the logic power model to the estimation of the power consumed in the whole component together with its local routes as it was explained in 4.4. The components were implemented in the Virtex-2 Pro devices that were used for the on-board measurements (XC2VP30). All the estimates were compared to the measured logic power values, that were obtained by subtracting the power of the clock circuitry, and the interconnect power (computed by using the effective capacitances and values provided by MARWEL) and the power of the input buffers from the total measured power of the design. The input signals had zero-mean gaussian distributions with autocorrelation coefficients that varied between 0 and 0.9995. The operand sizes were 8, 12 or 16 bits, as the number of input bits was limited by the number of pins connecting the two boards.

Table 4.10: Computational times for logic power models.

Component type	Add	Multiplier
Time [s]	0.12	0.24

**Figure 4.29:** Errors for a) multiplier, and b) adder logic power given by XPower

The results are represented for five different size adders and six different size multipliers in Fig. 4.28. The bars with black strips are used to differentiate the characterization input stimuli. It can be seen that the model achieves high accuracy with an average relative error of 9.32% for adders and 5.67% for multipliers. This validates the assumption that the logic power model can be successfully used for power estimation in both, logic and local interconnects, of the component.

Average computational times for both, adder and multiplier, power models are presented in Table 4.10. The deviation around this average value lies in the range of $\pm 2\%$ depending on the size of the component. It can be seen that the models satisfy the critical timing condition that is imposed at high levels of abstraction.

Additionally, we explore the accuracy of XPower for the designs we have used in the measurements. For the estimates given by the low-level tool, we have used advanced power reports provided by the XPower (from ISE 10.1). We were unable to use the new tool, XPower Analyzer, because at the moment the power values are displayed in milliwatts, so this tool can only be used for large designs where this precision does not have a significant impact on the accuracy.

Logic power values obtained from the measurements include both, power in the logic elements and power in the local interconnections. In order to compare the power values given by XPower to the measured ones, we have generated a script that parses the XPower report, and separates the power of local interconnections from the power consumed in connections that go to/from I/O pins. These connections are identified through special names assigned to the nets that contain them. Finally, we add the value of the logic power to the power of the local

Table 4.11: Autocorrelation coefficients for real-world data.

Signal type	ρ_{in1}	ρ_{in2}
Speech	0.9976	0.9979
Image	0.9879	0.9972
FDCT	0.7642	0.7562
Uniform noise	0.0117	-0.0016

connections in order to obtain the equivalent of the measured total logic power.

It can be seen that XPower gives large underestimate errors for logic power in adders, while it overestimates multiplier logic power. This is probably due to the high percentage of the local connections inside multipliers and the low percentage of these connections in adders, since it seems that XPower tends to overestimate short lines. A more detailed explanation of this effect together with the analysis of XPower accuracy will be given in the next chapter.

4.5.4. Logic power model for embedded blocks: on-board measurements

The experiments were performed on the following multiplication sizes implemented in a Virtex II Pro embedded multiplier: 16×16 , 12×12 , 8×8 , 16×12 , 16×8 and 12×8 . The design frequency used in all experiments was set to 50 MHz. The signals used in the experiments had zero-mean Gaussian distributions. The autocorrelation coefficient varied between 0 to 0.9995, and its values were chosen so as to model real-world applications, as shown in Table 4.11. All the experiments were performed on Pentium 4, with 3.00 GHz speed, and 1GB of RAM.

The estimates obtained from the model proposed here, and the ones given by the tool XPower, were compared to the values obtained from on-board measurements previously processed as explained in chapter 2. We have used multivariable regression over power values for two different operand size multiplications (in this case 12×12 and 8×8) and signals with autocorrelation coefficients of 0.9 and 0.9995 in order to obtain the two coefficients needed for the proposed model: a_e equal to 24.37 and a_r equal to 168.85. As it was expected, the value of the register capacitance was proved to be much higher than the embedded capacitance. The registers are located inside the programmable fabric, while the architecture of the embedded multipliers can not be reprogrammed and therefore, does not contain additional transistors that would increase the load capacitance value.

As previously mentioned, the low-level tool estimates are obtained from advanced power reports provided by XPower (from ISE 10.1). We apply the same methodology that was used for the comparison of logic power values of LUT-based components. We add the value of the embedded block power to the power of the local connections in order to obtain the equivalent

Table 4.12: Power values and relative errors for the proposed high-level model (HLLM) and the low-level tool XPower (XPwr).

ρ	Mult. size	Meas. [μ W]	HLLM [μ W]	XPwr [μ W]	Err [%] HLLM	Err [%] XPwr
0	16x16	11389	9408	13254	-17.39	16.37
	12x12	8015	7927	9284	-1.1	15.83
	8x8	6164	6465	7135	4.89	15.75
	16x12	8712	8622	10106	-1.03	16
	16x8	7136	7845	9595	9.94	34.47
	12x8	6222	7146	7999	14.84	28.55
0.9	16x16	10335	8103	10911	-21.6	6.76
	12x12	6601	6138	6677	-3.53	3.74
	8x8	4439	4452	4277	0.29	-3.64
	16x12	7555	6880	7581	-8.94	0.34
	16x8	5605	5794	6363	3.37	13.53
	12x8	4716	5047	5031	7.03	6.69
0.99	16x16	9331	7337	9401	-21.37	6.1
	12x12	5291	5141	5278	-2.85	-0.25
	8x8	3311	3356	2971	1.35	-10.25
	16x12	6290	5941	6309	-5.54	0.31
	16x8	4630	4743	4797	2.44	3.6
	12x8	3560	3953	3675	11.04	3.25
0.9995	16x16	8114	6527	7471	-19.56	2.49
	12x12	4055	4131	3853	1.86	-4.98
	8x8	2111	2137	1796	1.2	-14.93
	16x12	5333	4972	4869	-6.78	-8.7
	16x8	3390	3665	3319	8.11	-2.09
	12x8	2459	2790	2490	13.49	1.26

of the total measured logic power.

The values of the autocorrelation coefficient and the multiplier size are listed in the first two columns of Table 4.12. They are followed by the measured power value, the power estimate according to the high-level model and the XPower estimate. It can be seen that the variation in measured power for the different bit-widths and auto-correlation coefficients clearly surpasses the measurement error margin, thus confirming the validity of the measured power values. The relative errors for estimates provided by the proposed model and for estimates provided by XPower are given in the last two columns of Table 4.12. The shading differentiates the selected characterization designs that were used to determine the a_e and a_r coefficients.

It can be seen that the accuracy of the high-level model is quite high, lying within [-20%,+20%]. The larger errors are underestimates obtained for the 16×16 multiplier. Further analysis has shown a considerable amount of local connections between the registers and the embedded block. As they are not taken into account in our analysis, they may be the source of

Table 4.13: Computational times in seconds for embedded power models.

Mult. size	HLLM	XPower
16x16	2.18 s	212 s
12x12	2.13 s	141 s
8x8	2.12 s	102 s
16x12	2.15 s	157 s
16x8	2.13 s	137 s
12x8	2.13 s	127 s

this discrepancy.

On the other hand, the XPower tool achieves higher accuracy in this particular case, as it considers the connections between the registers and the embedded block. However, the accuracy of this tool, in general, varies depending on the value of the autocorrelation coefficient. Errors detected for the uncorrelated signals are large overestimates (with a maximum error of 35%), while they even become underestimates for the highest autocorrelation values. It has been noted that the power values given by this tool were identical for all embedded blocks (without registers), regardless of the bit-width of the operands and input signal statistics. This is a shortcoming that has been also confirmed by the Xilinx support. As the power of the registers gets smaller with higher autocorrelation coefficients, and the power estimate of the embedded block, instead of decreasing, remains the same, the sign nature of the errors changes. Low level tools also need to make some approximations in order to model the real power. The calibration of XPower involves silicon measurements on a small set of designs aimed at representing a wide range of applications [DT05]. Also, the power of the programmable logic is much larger than the embedded block power. Hence, there is a smaller influence of the embedded block power (compared to the programmable logic power) on the total design power. We assume that these could be the reasons for the less accurate embedded power estimates given by XPower.

When comparing the two power estimation models, it can be seen that, even though the low-level tool has detailed design and data information at the gate level, the high-level model is more accurate in many cases. Besides, the computation time required for HLLM was found to be around two seconds, while XPower with Modelsim simulation needed several minutes to finish (see Table 4.13).

4.5.5. Correlation factor for logic power estimates

In this subsection we present the fidelity analysis for the estimation models developed for multipliers implemented in LUTs, adders and embedded multipliers. We use two metrics for model evaluation: correlation coefficient and coefficient of determination (r-squared).

The first term demonstrates the dependence of two variables and shows if the increase/decrease in one variable corresponds to the increase/decrease in the other. For the logic power model presented here, the correlation coefficient shows if the model is capable of following the changes in the measured power. For example, if we increase the size of the multiplier, its measured power value will also increase, and we expect that the model will also predict a higher power value. The correlation coefficient is computed as:

$$\rho = \frac{\sum_{i=1}^m (y_i^e - \bar{y}^e) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^m (y_i^e - \bar{y}^e)^2} \cdot \sqrt{\sum_{i=1}^m (y_i - \bar{y})^2}} \quad (4.68)$$

where y_i^e is the i -th estimated value, y_i is the i -th measured value, and \bar{y}^e and \bar{y} are their means, respectively.

The coefficient of determination demonstrates how well a regression model fits a particular data set. It is compared to the most simple model where the mean of the measured values is used to predict power. Thus, we define the following terms:

1. Total sum of squares, SST :

$$SST = \sum_{i=1}^m (y_i - \bar{y})^2 \quad (4.69)$$

It represents the sum of squared deviations of the individual measurements from their mean.

2. Residual sum of squares, SSE :

$$SSE = \sum_{i=1}^m (y_i^e - y_i)^2 \quad (4.70)$$

It represents the sum of squared deviations of estimated values from measured values.

3. Coefficient of determination, R^2 , is:

$$R^2 = 1 - \frac{SSE}{SST} \quad (4.71)$$

This coefficient demonstrates the quality of the regression model, since estimated values are more accurate as R^2 gets closer to the value of one.

In Fig. 4.30, the measured and estimated power values for all three types of components are presented over a wide range of operand sizes. The power values for adders were obtained by dividing the measured power values by 3, as the designs containing adders were composed of three adder modules in order to improve the accuracy of the measurements. It can be seen that the estimated values and measured values are highly correlated. The correlation factor was

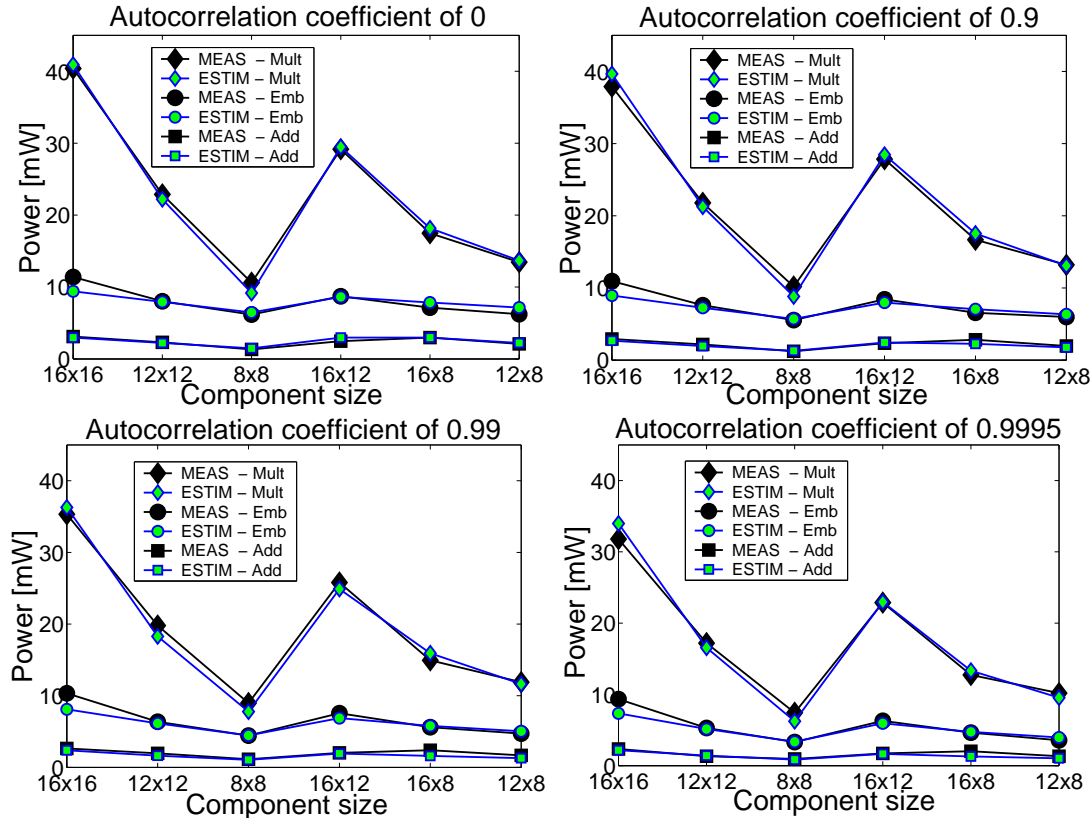


Figure 4.30: Measured and estimated power for various component sizes

found to be 0.99 for multipliers implemented in LUTs, 0.92 for adders and 0.95 for embedded multipliers. We have also computed the coefficient of determination and it was found to be 0.987 for multipliers implemented in LUTs, 0.955 for adders and 0.86 for embedded multipliers.

4.6. Conclusions

We have presented a high-level analytical approach to estimate logic power consumption of adders, multipliers implemented in LUTs, and embedded multipliers. The proposed methodology is based on an analytical model for the switching activity of the component and its structural description. For this purpose, a word-level signal model has been used to model the input signals. The model includes a partition of the signal word according to the activity of its bits. The activities of the input bits have been propagated throughout the component by using the probability method for switching activity computation. Additionally, glitching effects were accounted for as additional switching activity generated inside the component. The methodology has been applied to different component structures such as: array multiplier, row adder tree multiplier, modified booth multiplier and ripple-carry adder.

These logic models can estimate the power consumption for any given clock frequency, sig-

nal statistics and operands' word-lengths. They also account for the different power behaviour observed when considering zero-mean and non-zero mean input signals. The number of simulations needed for model characterization is highly reduced compared to other high level power models. Power estimation is also faster. Power estimates require times in the order of milliseconds. It has been shown that the accuracy of the proposed models is within 10% of low-level power estimates given by the tool XPower over a wide range of these parameters. The model that accounts for the glitching activity clearly outperforms the models that do not consider this effect.

Additionally, a comparison between the logic power model presented here for LUT-based components (HLLM) and two other high-level power estimation models; the Hamming distance model and the word-level power macromodel, has been carried out. When comparing HLLM with the word-level table-based macromodel, it was shown that the HLLM is more accurate, specially for the smaller-size input operands. Furthermore, HLLM is parameterized in terms of word-lengths of both inputs and thus, is capable of producing accurate estimates for components with operands of different sizes, whereas the table-based method assumes that the operands are of the same length and only one component size is introduced as a variable in the equation.

An extended comparison has been carried out with Hamming distance model, considering both, average values of input data set statistics and cycle-by-cycle accuracy. The experiments were performed on real-data applications and the results show that the accuracy of the proposed model is improved up to 60% when cycle-by-cycle signal statistics are taken into account. When comparing the two models, HLLM achieves better accuracy when considering highly-correlated signals, while the Hd-model gives better results when the switching activity of the input bits is distributed in a random fashion over the bit positions. We have also presented a comparative analysis between the two models for different model aspects which allowed us to identify their limitations and advantages. As a result, the proposed model needs significantly smaller number of low-level simulations than the Hd-model for its characterization, and achieves better accuracy when resource sharing is used. Still, when the operand word-length is adjusted to the input word-length, the Hd-model is slightly more accurate for most of the applications, and it does not require any changes in its model characterization method when using different component structures.

A methodology for power estimation of embedded multipliers in FPGAs has been presented separately, as they are not implemented in standard FPGA fabric. The high-level analytical model used for power estimation of LUT-based components has been adapted to consider embedded block implementation. The main change in the model was introduced due to the different multiplier structure, which is based on the Modified Booth algorithm. The logic power model for embedded blocks has been characterized and verified with on-board power measurements, instead of using low-level estimation tools which often lack the required accuracy.

The experimental results have shown that the accuracy of the model lies within $[-20\%, 20\%]$, with a global average of 7.9%, which has been demonstrated to be better than the accuracy of a low level commercial tool (global average of 9.7%). The presented model can give fast and accurate estimates at high and RTL-levels in the design flow, and the only information needed for the power estimate are the input signal statistics and operand word-lengths. Consequently, it is adequate for further integration with high-level power optimization techniques.

4.6.1. Future work

The logic power model described here can be used only for components with registered inputs and outputs. However, as previously mentioned, glitching can be the source of a high percentage of the total power. Thus, the next step would be to develop a model that would account for the power of non-registered components, by including the propagation of glitching between and throughout the components and considering the glitches at the component inputs while computing the component's power.

Apart from the component types described in this work, constant multipliers are also widely used in DSP applications. Therefore, an adaptation of the presented logic model to the constant multiplier structure used in Xilinx IP cores is planned for future work.

The embedded model described here considers individual embedded blocks with operand sizes up to 18 bits and registered inputs and outputs. However, when multipliers larger than 18×18 are used, several embedded blocks are combined to perform the multiplication. Therefore, an extension of the presented model to estimate the power of larger operand size multiplications and to include the power of the connections between the registers and the embedded block is also planned.

CHAPTER 5

Complete power estimation flow

In this chapter, we present the complete FPGA power estimation flow performed at high level of abstraction. The basic components of the estimation flow are the two power models that were presented in chapters 3 and 4. The model presented in chapter 3 is used for power estimation of the global routing employed for interconnections between the components and depends on their mutual distance and shape. The model presented in chapter 4 is used for both, local interconnect and logic, power estimation of the components and is based on the analytical computation of the switching activity produced inside the component. The complete model is obtained when the two models are applied together in order to estimate the design dynamic power. The model has been verified by on-board power measurements and the results demonstrate that it is capable of giving fast and highly accurate estimates for DSP-oriented designs.

This chapter is organized as follows. First, a brief overview of the previous work on power estimation of total dynamic power of designs is given in section 5.1. It is followed by a description of the complete power estimation flow together with an overview of the two power models (interconnection and logic) in section 5.2. The experimental results are given in section 5.3. The error performance of both, the model presented here and XPower, is given for some test DSP circuits similar to large real-world applications regarding their size and the number of arithmetic components. Additionally, an analysis of signal statistics when they pass through a chain of multipliers is presented, and the improvement of the complete model performance is demonstrated when these effects are taken into account. Finally, we conclude this chapter in section 5.5.

5.1. High-level FPGA estimation flow: background

In this section, we present high-level FPGA estimation techniques that have been used so far for estimating the total power consumption of a design.

In [CCF03], they use Rent's rule for wire length estimation, zero-delay model for switching activity computation and pre-characterization-based macro-modelling for average LUT and register power. As in the work presented here, they assume that the load capacitance can be considered to be constant. Thus, SPICE simulations with randomly generated input vectors are employed in order to obtain average dynamic power per access to a LUT. The reported average error of the whole presented estimator is 16.2%, and the maximum error is 27.5% for five chosen benchmarks. However, estimated values are compared to low-level estimates obtained from their tool that is presented in [LHC03] and based on the VPR design flow, while there is no comparison with the real measured power values.

In [CJMP03] a domain-specific macromodeling for kernel design is proposed. A domain corresponds to a family of architectures and algorithms that implements a given kernel. First, the power dissipation of all component types is described as a power function of a set of parameters, and then the power models are characterized by applying multivariable regression over a large number of low-level power estimates. Interconnect power is obtained by subtracting the power of all modules connected through the candidate interconnects from the total power of a module cluster measured at a low level in the design flow. This limits the given approach to be applied only with power optimization techniques aimed at logic power reduction, since the interconnect power is assumed to be the same for all the different architectures and thus, the location of the modules has to be fixed. The reported errors lie between 7% and 15% for the circuits which have regular architecture and only differ in the memory depth and/or number of components. The average error is not reported. The reported time for model characterization is approximately 12 hours.

An approach that is also based on domain-specific macromodeling has been proposed in [ESJ06, EJM06, AFJS07]. The whole domain is modelled as an IP core. The logic power parameters correspond to the number of basic operators in a core, the frequency of the core and the activity rates of the basic operators. The interconnect power parameters include the number of the connections between the basic operators, and their average switching activity. However, as the final model does not depend on the interconnect length, but only on the interconnect number, the location of the modules is limited to the one used in the characterization set. The maximum reported error is 31.8%, while the average error is 13.7%. The reported number of measurements needed for model characterization is 250,000 [AFJS07].

In [AN04b] the authors present a pre-routing prediction of two important parameters for power consumption estimation: switching activity (including glitches) and interconnect capac-

itance. Both parameters are modelled by using the methodology for power macro-model construction that was described in chapter 4. They choose the variables for the model that have the biggest influence on the design power and apply multivariable regression in order to obtain the coefficients standing by the variables. The average reported errors for switching activity lie in a range between 14% and 30%. The average error for interconnect capacitance prediction is around 35% while taking into account inherent capacitance noise. However, the circuit has to be fully placed in order to obtain estimates, thus, increasing the design time significantly.

The authors in [DT05] have classified their power estimation approach as a high-level method. However, the power estimates are obtained after the place-and-route of a design, and thus, the design time is extremely large.

5.1.1. Summary of the previous work on power estimation flow

In Table 5.1 we present the main features of high-level power estimation flows proposed in the literature, and the method presented here (HLM). All techniques are meant to be used during high-level synthesis, where the partial RTL description of the circuit is known. The tool used for model characterization and validation is listed in the second column of the table. It is followed by the maximum and average reported errors in the third and the fourth columns, respectively. The fifth column specifies the characterization effort whose definition was already presented in the previous chapter, in section 4.4. The last column marked as "Flexibility" determines the method capability to estimate a wide variety of circuits. The methods with low flexibility do not consider interconnection length and contain power models for large domains (coarse-grain). For example, in [CJMP03], for an FFT they include power models for Radix-4, twiddle factor computation and RAM memories, so the architecture of each of these blocks needs to remain fixed. The methods with medium flexibility do not consider interconnection length, but contain a library with smaller components such as adders and multipliers (fine-grain). It is possible to cover much more possibilities for different design architectures by using smaller components. The methods with high flexibility take into account interconnection length and also contain models for arithmetic components.

It can be seen that, the model presented here has the lowest characterization time, while achieving high flexibility. It has been characterized and validated by on-board measurements, just as the methodology presented in [ESJ06, EJM06, AFJS07], thus resulting in most confident estimation values. The average error is in the same range as the errors of the other high-level power methodologies.

5.2. Power estimation flow

A complete estimation flow is presented in Fig. 5.1. The design is first described at the algorithm

Table 5.1: Summary of features of the high-level power estimation.

Institution	Tool	Max. error [%]	Avg. error [%]	Charact. time	Flexibility
Univ.of UCLA [CCF03]	VPR	27.5%	16.2%	HIGH	HIGH
Univ. of South. California [CJMP03]	XPower	15%	-	HIGH	LOW
Univ. of Bretagne [ESJ06,EJH06]	Measurements	31.8%	13.7%	HIGH	MEDIUM
Tech. Univ. of Madrid: HLM	Measurements	27.41%	13.5%	LOW	HIGH

or RTL level. Based on this description, a DFG is constructed, the number and the size of all arithmetic components is extracted, and the floorplanning of the design at component level is performed. In this work we have used placement data instead of floorplanning data. However, the use of the model is not limited to the placement phase of the design flow, since the only parameter that is extracted is the interconnection length, and thus, it can be also easily obtained from high-level floorplan estimates. The only drawback of the high-level floorplan approach is the accuracy of the wire length estimate, which depends on the accuracy of the floorplan estimate.

The data flow graph analysis provides three different outcomes: the signal vectors at the inputs of all embedded multipliers, the word-level signal statistics at the inputs of all LUT-based components, and the bit-level switching activities of the connection lines between the modules.

The power of adders and multipliers implemented in LUTs is computed by knowing their operand sizes, and the input signal statistics for each of the operands. Therefore, the functions g for adders and h for multipliers in Fig. 5.1 are given as:

$$g(N_{ADD}, S_{ADD}, \mu_A, \rho_A, \sigma_A) = \sum_{i=1}^{N_{ADD}} P_{i,logic}^{add}(N_i, M_i, \mu_{A_i}^{op1}, \rho_{A_i}^{op1}, \sigma_{A_i}^{op1}, \mu_{A_i}^{op2}, \rho_{A_i}^{op2}, \sigma_{A_i}^{op2}) \quad (5.1)$$

$$h(N_{MULT}, S_{MULT}, \mu_M, \rho_M, \sigma_M) = \sum_{i=1}^{N_{MULT}} P_{i,logic}^{mul}(K_i, L_i, \mu_{M_i}^{op1}, \rho_{M_i}^{op1}, \sigma_{M_i}^{op1}, \mu_{M_i}^{op2}, \rho_{M_i}^{op2}, \sigma_{M_i}^{op2}) \quad (5.2)$$

where N_{ADD} and N_{MULT} are the total number of adders and multipliers implemented in LUTs, respectively, S_{ADD} and S_{MULT} are the sizes of the adders and multipliers expressed as arrays of operand word-lengths N_i and M_i (corresponding to the i -th adder), and K_i and L_i (corresponding to the i -th multiplier), and μ, ρ, σ are the input signal statistics for multipliers with a

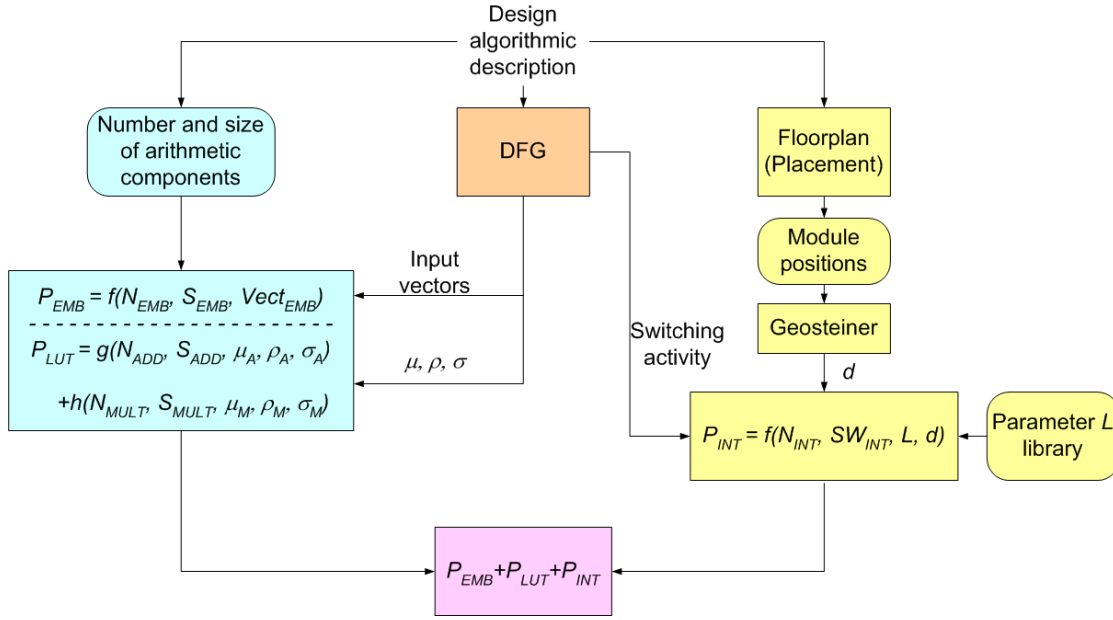


Figure 5.1: Power estimation flow

subscript M and adders with a subscript A . The superscripts $op1$ and $op2$ refer to the component operands.

Function P_{logic} is described by equation 4.60 in chapter 4, and represents the final model for estimating the logic power consumption in the presence of glitching and autocorrelation. It is given as:

$$P_{logic} = b \cdot (SW + k \cdot G') \quad (5.3)$$

where k and b are empirically derived constants which represent the average glitching at the output of one LUT in the LSBx-LSBy part of the component, and the product of the three power terms (f , V_{dd}^2 , C_l), respectively. SW is the total switching activity of the component, and G' is the number of basic cells in the four different component activity regions properly scaled by the corresponding coefficients that reflect the activity of each region.

The total power of LUT-based components is obtained by summing the power estimates for all adders and multipliers implemented in LUTs:

$$P_{LUT} = g(N_{ADD}, S_{ADD}, \mu_A, \rho_A, \sigma_A) + h(N_{MUL}, S_{MUL}, \mu_M, \rho_M, \sigma_M) \quad (5.4)$$

The total power of the embedded multipliers (function f in Fig. 5.1) is computed by knowing their operand sizes, and input data vectors. Therefore, the function f in Fig. 5.1 is expressed as:

$$P_{EMB} = f(N_{EMB}, S_{EMB}, V_{EMB}) = \sum_{i=1}^{N_{EMB}} P_{i,emb}(N_i, M_i, V_{E_i}^{op1}, V_{E_i}^{op2}) \quad (5.5)$$

N_{EMB} is the total number of embedded multipliers used in the design, S_{EMB} is the array of embedded block sizes expressed through operand word-lengths N_i and M_i (corresponding to the i -th embedded block), and V_{EMB} is an array of data signal vectors at the inputs of the embedded blocks.

The model for an individual embedded block power is described as (see equation 4.62):

$$P_{emb} = a_e \cdot SW_e + a_r \cdot SW_r \quad (5.6)$$

where a_e and a_r are the coefficients representing the product of the three power terms for the elements inside the embedded block and registers, respectively, and SW_e and SW_r are the total switching activities generated inside the embedded block and at the outputs of the registers, respectively.

The function e in Fig. 5.1 represents total interconnect power and is expressed as:

$$P_{INT} = e(N_{INT}, SW_{INT}, L_i, d_i) = \sum_{i=1}^{N_{INT}} P_{i,interc} \cdot SW_{i,interc} \quad (5.7)$$

where N_{INT} is the number of connections between the modules, $P_{i,interc}$ is the power estimate of the average power of each net in the i -th connection, and $SW_{i,interc}$ is the sum of the switching activity over all bits in a signal word belonging to the i -th connection. The connection between the modules refers here to the communication link between the modules that is equivalent to the connection between signal words, rather than to the connections on pin-to-pin basis.

The average power per one interconnect is estimated by using expression 3.3 from chapter 3 which is repeated here:

$$P_{intecon} = \begin{cases} k_3 \cdot L, & d = d_m \\ k_2 \cdot (d - d_m) + k_3 \cdot L, & d_m < d < d_l \\ k_1 \cdot (d - d_l) + k_2 \cdot (d_l - d_m) + k_3 \cdot L, & d > d_l \end{cases} \quad (5.8)$$

where d_l is the distance beyond which the router starts using long lines, d_m is the minimal distance between the module pin centers, L corresponds to the distance between the module pins and their pin center, d is the distance between the modules (the length of the RST), and k_1, k_2, k_3 are the coefficients calibrated by multiple regression analysis over measured power values for different distances between the modules.

The distance between the modules is obtained by applying a rectilinear Steiner tree algorithm to the centers of the module pins, by using *Geosteiner* software tool after the placement of the modules.

The parameter L models the limitations that occur due to the shape and size of the modules

and is computed as (see equation 3.5):

$$L = \sum_{k=1}^n L_k \quad (5.9)$$

$$L_k = \frac{\sum_{i=1}^{I_{1k}+I_{2k}} l_{k,i}^{in} + \sum_{j=1}^{O_k} l_{k,j}^{out}}{I_{1k}+I_{2k}+O_k}$$

where $l_{k,i}^{in}$, $l_{k,j}^{out}$ are the Manhattan distances from the module pin center of the k^{th} module to its input pin i and its output pin j , respectively. I_{1k} and I_{2k} are the number of the k^{th} module input pins used for the connection, and O_k is the number of its output pins used for the connection. We have created a pre-characterized library of several L values corresponding to the arithmetic blocks used in this work.

The total dynamic power consumption of the design is obtained by summing the power consumptions of all interconnects in the design and the power consumptions of all design modules:

$$P_{total} = P_{LUT} + P_{EMB} + P_{INT} \quad (5.10)$$

5.3. Experimental results

We divide the experiments in two sets. In the first set, we evaluate some test DSP circuits that consist of a small number of arithmetic modules. This is useful to establish the errors that can be expected when power estimation is applied in the fine-grain optimization process (i.e. sensitive to changes in the position or the word-length of one component). In the second set, we evaluate several different configurations of a large test DSP circuit that is similar to real-world applications in terms of the occupied area and the number of components. The results are analyzed in detail, and possible improvements of the model are identified by switching from the word-level to the bit-level signal statistics. Additionally, XPower estimates are compared to the on-board measured values and the error distribution over different resources (logic, interconnections, input buffers) is presented.

5.3.1. Small test DSP circuits

We analyze the same three DSP circuits presented in chapter 3. For each of them we obtain two values: the total power estimate according to 5.10, and the total measured dynamic power, obtained by subtracting the static power and the power of the clock circuitry from the total measured power. The relative errors obtained when the estimates are compared to the measured values are given in the fifth column of Table 5.2. The first three columns indicate the benchmark name and its size, expressed as the number of slices and embedded blocks respectively. The fourth column lists the autocorrelation coefficients of the input data. The last column represents

Table 5.2: Relative errors for the proposed model (HLM) and XPower (XP), for different auto-correlation coefficients.

Benchmark	Slices	Emb. Mult.	Position	ρ	Er(HLM)[%]	Er(XP)[%]
<i>DSP1</i>	290	0	1	0	10.63	345.17
				0.9	5.42	346.10
				0.99	8.50	352.28
				0.9995	25.56	372.04
			2	0	9.88	292.95
				0.9	5.02	291.76
				0.99	7.25	296.47
				0.9995	23.00	317.36
			3	0	8.15	284.09
				0.9	3.10	281.55
				0.99	6.17	289.74
				0.9995	21.60	310.05
			4	0	13.94	334.05
				0.9	8.53	333.68
				0.99	10.09	336.14
				0.9995	27.41	362.47
<i>DSP2</i>	192	2	-	0	-1.23	162.13
				0.9	-0.11	147.87
				0.99	5.91	138.69
				0.9995	17.83	115.90
<i>DSP3</i>	212	2	-	0	17.48	294.80
				0.9	6.62	295.89
				0.99	9.59	302.30
				0.9995	13.68	320.35

the relative error of XPower estimates for the design dynamic power when compared to the measured power values. The XPower design dynamic power estimate is obtained from the XPower advanced report, by summing the power of the Interconnect, Logic and Input buffer power groups.

It can be seen that the errors given by the power model are always below 30%, with the maximum errors obtained for the highest autocorrelation coefficient. A detailed analysis of the estimation errors, their nature and distribution among the different power components will be given later. Additionally, in Fig. 5.2 we present the estimated power values obtained from HLM versus measured power values for both, DSP circuits and individual components presented in the previous chapter. The solid line represents the case when both values are equal, and the dotted line in the figure presents $\pm 20\%$ deviation from the linear fit. As already mentioned, most of the estimated values are accurate to within 20% of the measured value.

On the other hand, it can be seen from Table 5.2, that XPower clearly overestimates the design power. In order to understand better the error distribution among different power groups

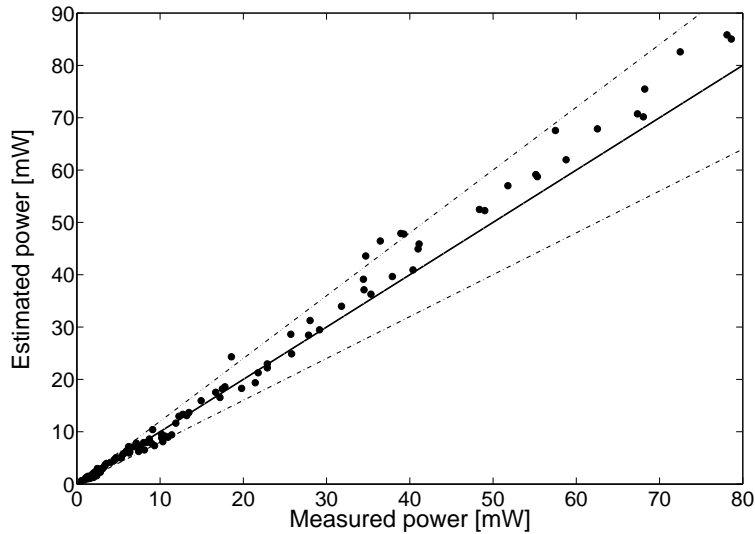


Figure 5.2: Estimated vs. measured power values for DSP circuits and individual components

given by XPower, we present a power distribution pie chart for both, measurements and XPower, in Fig. 5.3. Beside the three power groups that form the power estimates, we have also included the Clock power group so as to analyze the total dynamic power consumption of the design. Fig. 5.3a corresponds to the power distribution of the benchmark DSP_1 when it is located in the position 1 (i.e. near the I/O pins), while Fig. 5.3b corresponds to the power distribution of the same benchmark located in the position 2 (i.e. far from the I/O pins). It can be seen that the percentage of each power group obtained from the measurements compared to the percentage of the same power groups obtained from XPower do not match in neither of the design positions. Furthermore, XPower fails to account properly for the significant increase in the interconnection power when placing the design further away from the pins. In both cases, the logic power is a dominant power component, as it is expected (due to high amount of logic in data-path oriented designs).

Fig. 5.4 and Fig. 5.5 represent the error distribution for HLM and XPower estimates of the design dynamic power after the clock power has been subtracted from the total dynamic power, as this work does not consider power estimation of a clock net. On the left-hand sides of the figures, we give the errors of the total power estimates when compared to the measured dynamic power. In the middle column of the figure we give the errors for each of the presented power models separately: logic and interconnect. We have omitted the input buffer error, as it is only symbolic for HLM and equal to zero. We use the effective capacitance of the input buffers obtained through the measurement experiments, which is then multiplied by the input buffer switching activity, square of the power supply and frequency, in order to obtain both, the measured value and HLM estimate. Furthermore, the XPower error for this power group is found to be negligible (approximately 3.5%).

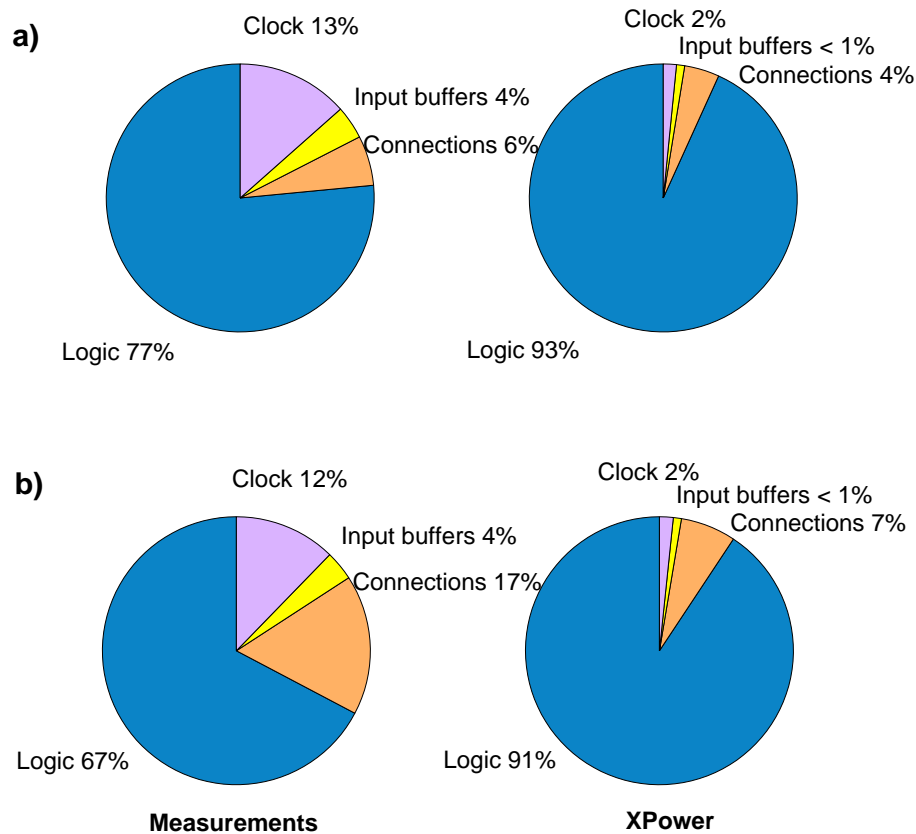


Figure 5.3: Power distribution according to the measurements and XPower for DSP_1 design in a) position 1, and b) position 2

It can be seen that the dynamic power in DSP circuits is dominated by the logic power due to the large number of arithmetic components, in particular multipliers implemented in LUTs which consume a great deal of power. The column on the right-hand side of both figures shows piecharts with the power distribution among different power components (piecharts correspond to DSP_1 in position 2; other DSP circuits and positions produce almost identical piecharts). As already mentioned, the logic power is the dominant power component, so the error performance of the total power estimate is quite similar to the error performance of the logic power estimate. Another observation from Fig. 5.4 is that the model presented here slightly underestimates interconnect power and overestimates logic power. We believe that the interconnect underestimates occur due to the lack of a congestion parameter in the interconnect power model as already commented in chapter 3. The overestimates of the logic power occur due to the signal distribution at the outputs of the multipliers that differs from the gaussian distribution. This effect will be explained in detail in the next section.

On the other hand, in Fig. 5.5 it can be seen that XPower has large overestimates (over

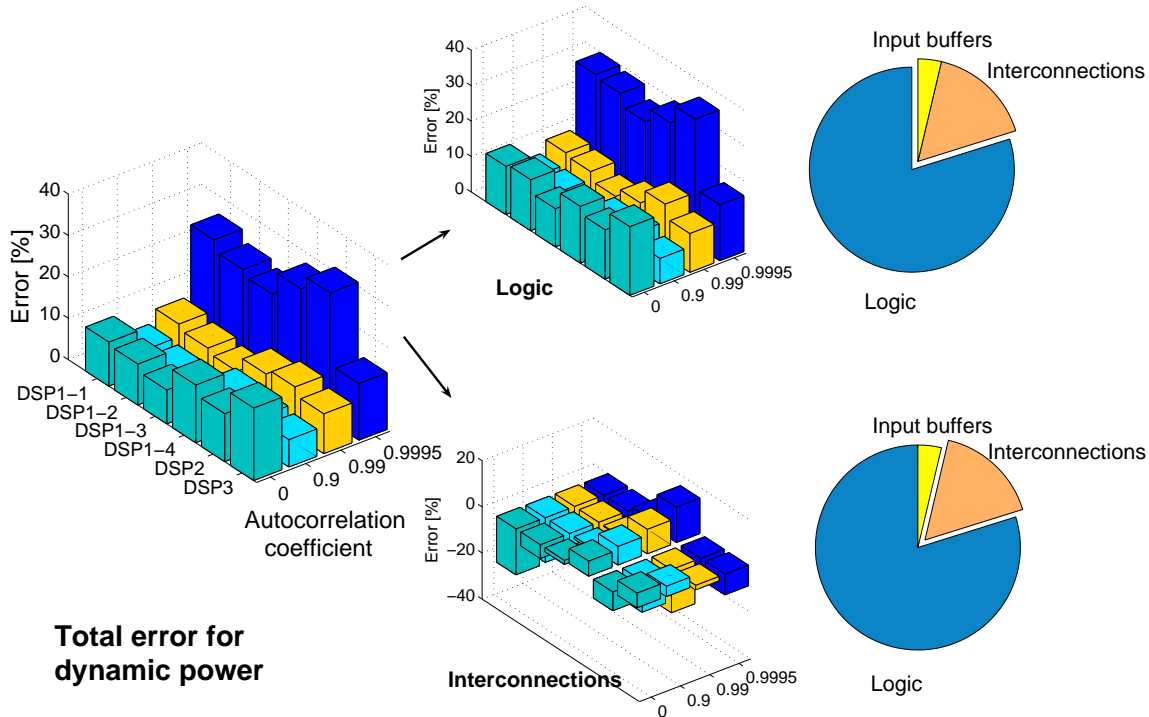


Figure 5.4: Error distribution for HLM considering total dynamic power, and its components: logic and interconnect power

300%) for all power components except for the input buffer power. The dominance of the logic power is even more stressed out for XPower estimates, as it constitutes over 90% of the total power in these designs according to the XPower reports (see Fig. 5.3a and Fig. 5.3b). It can be also seen that XPower tends to overestimate interconnect power less when the interconnect length is longer. For example, when considering the DSP_1 test design in positions 2 and 3 (modules far from I/O pins), the errors drop drastically from 200%, obtained in position 1, to below 50%. It seems that XPower tends to overestimate the short connections more than the long ones. This may be the reason for large logic power overestimates, as they are formed by summing the power consumed in logic and the power of the local connections. Additionally, it can be noted that the presence of embedded multipliers in the DSP_2 test design seems to decrease the total error in Fig. 5.5. This is to be expected as XPower errors for embedded logic are much smaller than errors for arithmetic components implemented in LUTs (see chapter 4, section 4.5). Although the DSP_3 design also contains embedded multipliers, the largest contribution to its power comes from 20×12 multiplier implemented in LUTs, and thus, the decrease in the total estimation error is not so obvious.

When both models are compared, it can be noted that the estimates from HLM are order of magnitude more accurate than XPower estimates. The HLM errors lie in the range [0%, 30%],

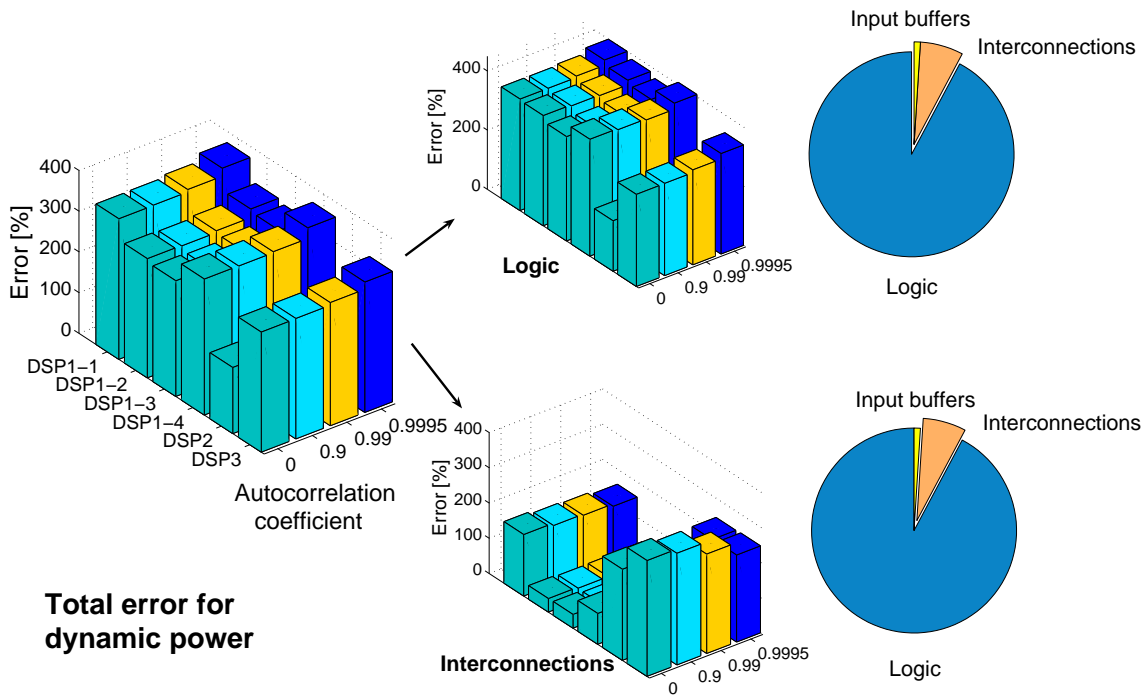


Figure 5.5: Error distribution for XPower considering total dynamic power, and its components: logic and interconnect power

while the XPower errors go over 350%. As explained in chapter 4, this could be due to the size of the designs used in these experiments, as they occupy only 1% of the FPGA chip. As XPower expects designs that occupy a much higher percentage of the chip, and it has to compensate for the assumption that the static power does not vary with the activity of the design, it tends to overestimate all the other dynamic power components.

5.3.2. Word-level statistics: Large DSP designs

In this section, the accuracy of the HLM estimates is explored for three different configurations of a DSP test design (*SYSTEM*) that consists of several DSP_2 and DSP_3 designs connected in a chain-like fashion as shown in Fig. 5.6.

The DSP test design consists of four modules of type DSP_2 and five modules of type DSP_3 . The outputs of both modules are 32-bits wide. Since DSP_2 has four 8-bit inputs and DSP_3 has two 12-bit and one 8-bit inputs, the outputs of the modules are partitioned in the corresponding number of inputs of the next connecting module. For example, consider the connection between the first module (type DSP_2) to the second module (type DSP_3). In this case, the output of the first module will be partitioned into two 12-bit and one 8-bit word in order to enable the

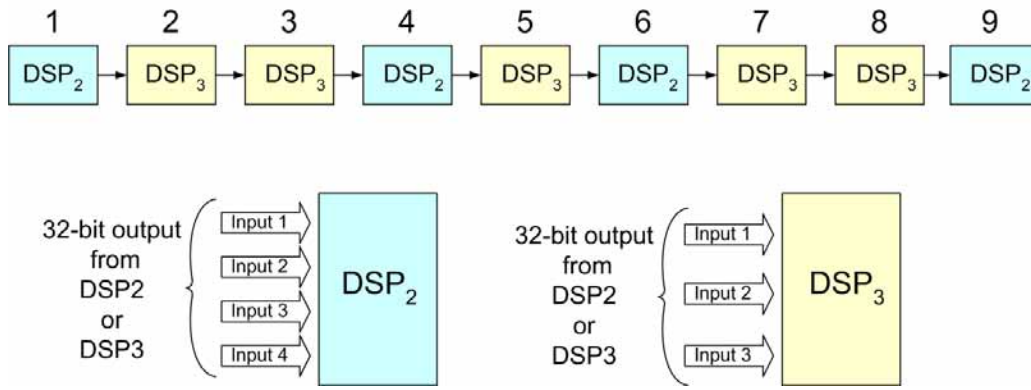


Figure 5.6: SYSTEM block schematic

Table 5.3: The number and size of arithmetic operators in DSP_2 and DSP_3 designs

Benchmark	Operator	Size	Number
DSP_2	Mult	8×8	3
		16×16	1
	Add	8×8	2
		16×16	1
DSP_3	Mult	12×8	1
		12×12	1
		20×12	1
	Add	12×8	2
		32×24	1

connection to the module of type DSP_3 .

The number of operators, their type and size for both designs are presented in Table 5.3.

Three different configurations of the *SYSTEM* design are obtained by varying the number of multipliers implemented in LUTs and embedded multipliers. The characteristics of all three configurations are presented in Table 5.4.

The first configuration uses multipliers implemented in LUTs for the largest multiplications in the design (i.e. 16×16 and 20×12). For the rest of the multiplications, embedded blocks are used. The number of multipliers implemented in LUTs is limited by the measurement setup. In particular, this limit is determined by the value of the maximum power that can be measured for the minimum frequency generated on the Altera board (16 MHz), and the minimum resistance value used for the measurements (1 ohm). The second configuration uses both types of multipliers for various different size multiplications (8×8 , 16×16 , 12×8 , 12×12 for the embedded multipliers, and 8×8 , 20×12 for the multipliers implemented in LUTs). The third configuration was supposed to use all embedded multipliers for multiplications. However, as DSP_3 design has one 20×12 multiplication, it can not fit into only one embedded multiplier.

Table 5.4: The number of different arithmetic module types in the three configurations of *SYSTEM*

Benchmark	Operator-Impl.	Number
<i>SYSTEM_{LUT}</i>	<i>Mult_{LUT}</i>	9
	<i>Mult_{EMB}</i>	22
	<i>Adder</i>	22
<i>SYSTEM_{MIX}</i>	<i>Mult_{LUT}</i>	13
	<i>Mult_{EMB}</i>	18
	<i>Adder</i>	22
<i>SYSTEM_{EMB}</i>	<i>Mult_{LUT}</i>	5
	<i>Mult_{EMB}</i>	26
	<i>Adder</i>	22

Table 5.5: Relative errors for the proposed model (HLM) and XPower (XP), for different auto-correlation coefficients.

Configur.	Slices	Emb. Mult.	ρ	Er(HLM)[%]	Er(XP)[%]
<i>SYSTEM_{LUT}</i>	1972	22	0	39.78	328.94
			0.9	38.50	323.67
			0.99	42.93	331.93
			0.9995	47.67	337.45
<i>SYSTEM_{MIX}</i>	1692	18	0	34.31	246.23
			0.9	35.84	247.44
			0.99	37.88	249.45
			0.9995	45.01	258.60
<i>SYSTEM_{EMB}</i>	1444	26	0	35.97	227.21
			0.9	29.58	224.23
			0.99	31.17	231.42
			0.9995	31.21	235.25

Hence, since the embedded power model developed here supports only up to 18 bit operands for now, we have implemented this multiplication into LUT-based multipliers in all three configurations.

The errors obtained by comparing HLM estimates and XPower estimates to the real measured values are given in Table 5.5. The computation time for XPower estimates was approximately one hour and a half for Modelsim simulation and generation of the XPower report, while the HLM estimates were obtained in a few minutes.

When generating XPower reports, and using the simulator resolution of 1 ps and 10000 input vectors in the input signal set, the .vcd file created according to this resolution was 5 GB large, and the XPower tool was not able to parse it correctly. Therefore, we have parted the input signal set in two halves, and performed XPower analysis for each of them. At the end, we computed the average power value from these two simulations in order to obtain XPower

Table 5.6: Relative HLM errors for the internal modules of $SYSTEM_{MIX}$

Module	1	2	3	4	5	6	7	8	9
Type	DSP_2	DSP_3	DSP_3	DSP_2	DSP_3	DSP_2	DSP_3	DSP_3	DSP_2
Error [%]	11.58	29.53	23.84	16.41	37.17	24.22	57.13	51.68	44.70

estimate. According to the results shown in Table 5.5 it can be concluded that XPower has not increased its accuracy although the test designs are approximately 10 times larger than the test designs used in the previous chapter. It should be noted that, due to the limitations of the measured maximum voltage value on the resistance and in order to avoid the increase in the static power, we have applied to the design the smallest frequency value that can be generated on the Altera board. Thus, the measured power value for this large DSP design is similar to the power values obtained for the small designs, and the accuracy of the XPower stays the same as expected since there are no variations in static power that it can compensate for.

It can be also seen in Table 5.5 that HLM overestimates in all cases. In order to explore the exact source of errors, we have measured the power of each of the DSP_2 and DSP_3 modules separately by applying the corresponding input vectors from the connection points between these modules inside the $SYSTEM$ design. For example, only DSP_2 module was implemented into FPGA, and four power values of the DSP_2 power were measured. The first power value was obtained when global design inputs (i.e. inputs to the $SYSTEM$) were loaded to the DSP_2 inputs. This power value corresponds to the power consumed by the module 1 in Fig. 5.6. The other three power values were obtained when the outputs of the modules 3, 5 and 8 in $SYSTEM$ design were loaded to the DSP_2 inputs. These power values correspond to the power consumed by the modules 4, 6 and 9 in Fig. 5.6. Similarly, five power values of the DSP_3 design power were measured when the outputs of the modules 1, 2, 4, 6 and 7 were loaded to its inputs. We have focused on the logic power estimates, since it determines the total error behaviour.

In Table 5.6 we give separate logic power errors for the nine modules of $SYSTEM_{MIX}$ design for an autocorrelation coefficient of 0. These errors were found to be almost the same for other autocorrelation coefficient values and so they are not repeated here. It is observed that the error seems to decrease for the modules whose inputs are connected to the output of the type DSP_3 , while it increases when the inputs are connected to the output of the type DSP_2 . In general, it seems to increase with the number of connected modules. We believe that the statistics at the outputs of the multipliers are the main cause for these effects and they are analyzed in the next section.

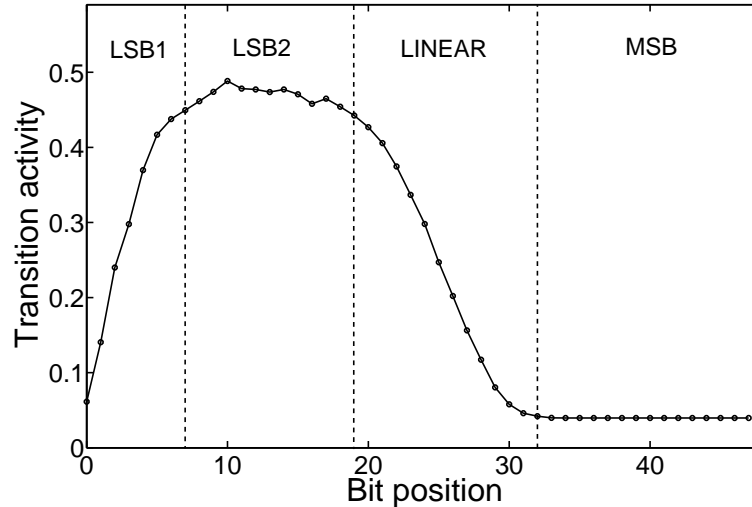


Figure 5.7: Transition activity vs. bit position in the signal word at the output of the multiplier

5.4. Bit-level statistics: Large DSP designs

In [KBN02], the authors have noted that the distribution of the product sequence of two gaussian inputs is symmetrical around the mean value, but it is not a gaussian distribution. The LSB bit of the product exhibits less activity than that of the white noise, because only the product of two odd numbers is odd. It was found that, not only is the LSB bit of the product affected by the multiplication, but there is a region of LSB bits exhibiting lower switching activity. This region tends to be bigger as the number of chained multiplications grows.

In Fig. 5.7 we have plotted the bit transition activity versus bit position in the 48-bit signal word that represents the output of the last multiplier in the chain of three successive multiplications. The inputs to the first multiplier had gaussian distributions with autocorrelation coefficient of 0.9995. It can be seen that a new activity region can be defined in the signal word obtained by Dual-bit type methodology. The LSB region is now divided into two different regions: the lowest bits that exhibit switching activity smaller than 0.5 due to the nature of the multiplication process (*LSB1* in Fig. 5.7), and the rest of the LSB bits that have 0.5 switching activity as before (*LSB2* in the figure).

The DSP_2 output comes from a multiplier, while the output of the DSP_3 comes from an adder. Thus, the signal statistics at the output of DSP_3 are closer to the gaussian distribution than those at the output of DSP_2 . Since the estimation model presented here does not take into account this effect, it represents the output signal of DSP_3 more faithfully than the output of DSP_2 . This is the reason for the error behaviour presented in Table 5.6 for each module power separately.

In order to solve this, we have modified the logic power model presented in chapter 4 in

order to account for bit-level statistics instead of word-level statistics. Consequently, the input parameters to the model are no longer ρ , μ and σ . Instead, they consists of two bit vectors: one containing the switching activity of each bit, and the other containing the probability of each bit being '0' or '1' (these probabilities are necessary since they also change as a consequence of the multiplication process). Both bit vectors are obtained from DFG simulations. The total switching activity generated inside the component is thus computed by using real values of signal probabilities and switching activities of the input bits, rather than using the analytical approach.

The glitching model was also modified in order to account for the bit-level statistics. The scaling factor $l = 1 - \rho$ is replaced by the expression $sw_i/(1 - prob_i)$ for each bit, where sw_i is the switching activity of the i -th bit, while $prob_i$ is its probability of being '1'. Although glitching is proportional to the switching activity of the inputs, the probability of the bit being '1' has an impact on the amount of glitching. As this probability increases, the propagation of glitching is more probable. This effect was not important before, as all the bits had a 0.5 probability of being '1'. Now we have included the divisor $1 - prob_i$ into the expression for the glitching to reflect the dependence of glitching propagation on the logic value of the bit. It is important to note that according to equation 4.16 from chapter 4, the scaling factor $1 - \rho$ is indeed equivalent to the expression $sw/(1 - prob)$ when the signal probability $prob$ is equal to 0.5.

The expression for the glitching in adders becomes:

$$G_{add} = k_{add} \cdot \sum_{i=1}^N \frac{sw_x(i)}{1 - prob_x(i)} \cdot \frac{sw_y(i)}{1 - prob_y(i)} \quad (5.11)$$

where k_{add} is the average glitching per LUT in the adder, N is the word-length of the longer operand, and subscripts x and y correspond to the two component inputs.

The new expression for the glitching in multipliers implemented in LUTs (row adder tree multipliers) is:

$$G_{mul} = k_{mul} \cdot \sum_{i=1}^{\lceil \log_2 M \rceil} \sum_{j=1}^{\lceil \log_2 i \rceil} \frac{sw_{av}(i, j)}{1 - prob_{av}(i, j)} \cdot \left(\sum_{k=1}^N \frac{sw_x(i, k)}{1 - prob_x(i, k)} \right) \quad (5.12)$$

where k_{mul} is the average switching per LUT in the multiplier, N and M are the word-lengths of the operands x and y respectively, and sw_{av} and $prob_{av}$ are the arrays of switching activities and signal probabilities of the other operand in the j -th optimization level, respectively. For the sake of simplicity, the expression for the glitching is provided for the case where the word-length of the operand y is a power of two.

The expressions that are used for obtaining the arrays sw_{av} are given in 5.13. The signal

Table 5.7: Computational times in seconds for bit- and word-level power models.

Config.	Bit-level	Word-level
$SYSTEM_{LUT}$	69.22 s	67.47 s
$SYSTEM_{MIX}$	62.10 s	61.83 s
$SYSTEM_{EMB}$	79.53 s	76.65 s

probability array is obtained in the same way.

$$\begin{aligned}
 sw_{av}(i, j) &= \frac{1}{2} \cdot (sw_{av}(i-1, 2j-1) + sw_{av}(i-1, 2j)) \\
 sw_{av}(1, j) &= \frac{1}{2} \cdot (sw_y(2j-1) + sw_y(2j))
 \end{aligned} \tag{5.13}$$

There are no changes in the embedded power model, since the glitching generated inside the embedded multiplier was assumed to be negligible, and the input parameters to the model were already bit-level signal statistics.

The new models were applied to the three configurations of the $SYSTEM$ design and the errors are listed in Table 5.8. It can be seen that now the errors are between 1.5 and 2 times smaller than those obtained from word-level estimates in the previous section. Consequently, the signal distribution at the outputs of the multipliers is indeed the primary source of the HLM power estimation error.

In Table 5.7 we present the computation times measured in seconds for both, word- and bit-, level models. It can be seen that the configuration with the largest number of embedded multipliers has the longest computation time, since the embedded power model depends on the input signal values, whereas the power models for other logic components need only word/bit level input signal statistics in order to produce an estimate.

Next, in Fig. 5.8 we present the estimated and measured values for all three configurations of the $SYSTEM$ design and four different autocorrelation coefficients applied to each configuration. It can be seen that the values are correlated, with the correlation coefficient of 0.99 for the bit-level models, and 0.97 for the word-level models.

Additionally, we show the errors for the nine modules of the $SYSTEM_{MIX}$ separately in Table 5.9. We have also included the errors obtained from HLM based on word-level statistics for easy comparison. It can be seen that the errors have decreased significantly for all cases except the first one. The reason is given as follows.

The signal statistics at the inputs of all the modules, except the first one, are influenced by the chain of multiplications as already explained. As a consequence, the real values of the bit-level switching activities at the inputs of the components inside these modules are much lower than the bit level switching activities obtained according to the ARMA signal model. This explains the smaller bit-level power model errors in Table 5.9. On the other hand, the inputs to the

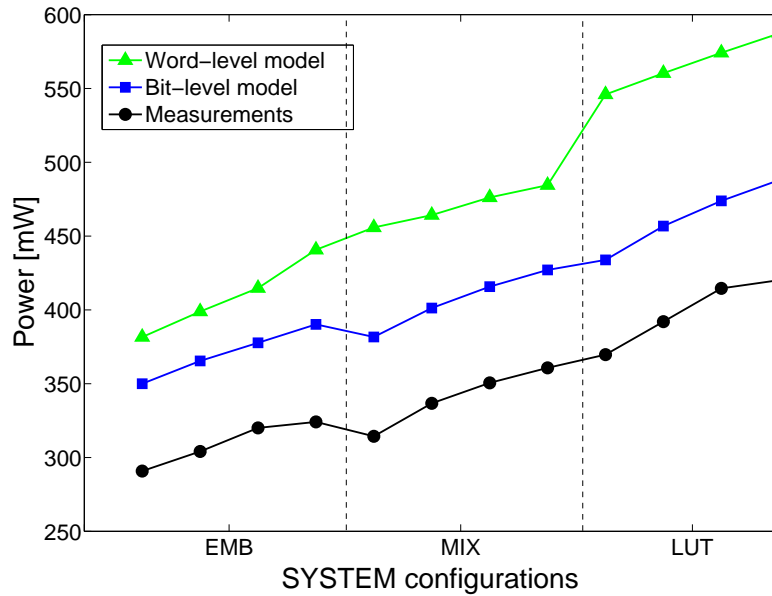


Figure 5.8: Estimated and measured power values for *SYSTEM* designs

Table 5.8: HLM errors for the three *SYSTEM* configurations when bit-level statistics are used

Benchmark	ρ	Error [%]
<i>SYSTEM_{LUT}</i>	0	18.39
	0.9	18.59
	0.99	19.18
	0.9995	21.40
<i>SYSTEM_{MIX}</i>	0	16.14
	0.9	14.30
	0.99	16.52
	0.9995	17.36
<i>SYSTEM_{EMB}</i>	0	20.38
	0.9	18.02
	0.99	20.18
	0.9995	20.34

first module have gaussian distributions and they are not affected by the multiplication process. Additionally, it is important to note that HLM is characterized by considering word-level statistics (i.e. gaussian signals in the characterization set are processed by ARMA signal model in order to obtain estimated bit-level switching activities which are further used for total switching activity computation). It was also observed that the ARMA signal model underestimates the total switching activity of the signal word (see chapter 4, subsection 4.5.1). This means that the HLM calibration was performed for switching activity values smaller than the real ones. Consequently, when real bit-level statistics are applied to the first module, overestimation errors increase.

Table 5.9: Relative HLM errors when applied to each component module in $SYSTEM_{MIX}$ separately for both bit-level and word-level statistics

Module	1	2	3	4	5	6	7	8	9
Type	DSP_2	DSP_3	DSP_3	DSP_2	DSP_3	DSP_2	DSP_3	DSP_3	DSP_2
ErrorW[%]	11.58	29.53	23.84	16.41	37.17	24.22	57.13	51.68	44.70
ErrorB[%]	14.65	19.18	16.65	15.83	18.29	15.16	20.99	19.47	18.90

Table 5.10: HLM errors for the first DSP_2 design when using bit-level instead of word-level statistics

ρ	0	0.9	0.99	0.9995
Error word [%]	11.58	7.60	11.54	30.99
Error bit [%]	14.65	12.76	16.00	24.17

This effect was also observed for almost all other values of autocorrelation as presented in the Table 5.10. The error for an autocorrelation coefficient of 0.9995 is the only error that decreases when bit-level statistics are applied. In this case it is important to note that the DSP_2 module also contains several internal multiplications. Therefore, it seems that the benefits of using bit-level statistics at the outputs of the internal multipliers have outweighed the calibration error. Indeed, the underestimate errors of the ARMA model are below 2% for this highest autocorrelation coefficient, while they are between 5 and 15% for the other autocorrelation values (see Fig. 4.18), so the impact of using real, instead of estimated, bit-level values is negligible in this case.

As a result, we use bit-level instead of word-level statistics for the designs with a large number of multiplications. The average error of HLM is 13.5% when the results for both, small and large, DSP designs (tables 5.2 and 5.8) are taken into account. The maximum error detected is 27.41%.

5.5. Conclusions

In this chapter, a complete estimation flow for DSP circuits has been presented. The analysis of the power distribution among different power components was given for the measurements, XPower estimates and HLM estimates. Since DSP designs are data-path oriented, it was confirmed that the logic power is the dominant power component in these designs.

It was also observed that the accuracy of XPower is quite low, with relative errors going up to 350%. The logic power is overestimated possibly due to the overestimation of the short connections, as errors decrease for designs containing longer connections.

The complete model presented here is highly accurate, with the average error of 13.5%, and the maximum 27.41%. Underestimates of the interconnect power occur due to the lack

of a congestion parameter in the interconnect power model, while the overestimates of the logic power occur due to the signal statistics at the outputs of the multipliers that no longer correspond to the gaussian distribution. It has been demonstrated that by taking into account bit-level statistics instead of word-level ones, the error was reduced as much as two times.

High accuracy and small computation time for both, model characterization and circuit estimation, suggest that the model could be successfully used in high-level power optimization techniques, where accurate estimates are needed in the shortest possible time.

5.5.1. Future work

Future work is oriented towards including a congestion parameter into the interconnect power model, since this parameter can play an important role in today's large industrial designs.

Another important task is modelling word-level statistics at the outputs of multipliers by using the so-called "triple-bit type model" presented in [KBN02]. They present a methodology for dividing a signal word into activity regions according to its signal statistics and the number of multipliers the input signals passed through. The results presented in this chapter indicate that the accuracy of the model could be improved by introducing this methodology. Another future task is oriented towards introducing glitching into the power model and thus, extending the power estimation methodology to non-registered arithmetic components.

The current power model for embedded multipliers uses data values of the input vectors instead of their signal statistics. These vectors can be extremely long, so they should be replaced by word-level statistics in order to accelerate the whole estimation process. Additionally, the embedded power model should be modified in order to accept input signal word-lengths wider than 18 bits.

CHAPTER 6

Conclusions

In chapter 1 we listed the objectives set for this thesis. They were meant to tackle three types of problems in the area of power estimation, completely different in nature: model calibration and verification, where knowledge about actual design power is needed, high-level estimation of interconnect power, where the lengths and the wire types used for routing play an important role, and high-level logic power estimation, where appropriate signal model and switching activity computation methodologies are crucial in order to achieve fast and accurate logic power estimates.

It was confirmed that the main bottlenecks when developing estimation models are their calibration and verification due to the low accuracy of the low-level commercial tool for power estimation. Hence, the most precise models are achieved by using real on-board power measurements.

A brief overview of the measurement system developed in this work for the purposes of power estimation is given in section 6.1. Next, in section 6.2, we point out the main contributions achieved in the area of the interconnect power estimation, and the importance of having such methodology adapted to FPGA devices, where programmable switch matrices have a significant influence on the amount of the total power consumption. In section 6.3, we talk about the main results achieved in the area of logic power estimation starting from the adaptation of the word-level signal model to the purposes of high-level power estimation. Afterwards, we go through the development of the power models for programmable logic. Since the use of embedded multipliers has become a norm in DSP circuits, and the designers are facing many difficulties in modelling embedded power, the contribution of the work presented here regarding embedded power estimation is stressed out separately. Finally, in section 6.4 we give the most important conclusions about the complete power estimation flow when all power models

are applied together in order to estimate the total dynamic power of a DSP circuit.

6.1. Measurement system: conclusions

FPGAs, as well as ASICs, are available only in a closed form to the users. This means that their electrical structure is encapsulated and hidden from the outside world. The only way to separate the power of different elements inside the chip, is to know their capacitances. There are two different ways for obtaining these values: from the low-level tools provided by the chip vendors, or through a methodology based on on-board power measurements.

We have chosen the latter option, since the accuracy of the current low-level tool XPower is proven to be very low for small designs implemented in FPGAs. In the following, we highlight the most important features of the measurement system:

- The measurement system consists of two FPGA boards, one for measuring design power and the other for charging the input vectors to the first one.
- The essential part of the calibration and verification methodology is MARWEL, a tool developed in C++, capable of extracting the exact number and type of the wires used for design interconnections from design files.
- The effective wire capacitances are obtained through measuring power of simple designs containing logic modules in different positions on the chip, and applying the multivariable regression on their power differences.
- Once the effective wire capacitances are known, and the number of each wire type used for design connections is reported by MARWEL, the interconnect power is obtained by summing the power over all different wire types.
- The total logic power is easily obtained by subtracting the interconnect power and the power of the clock circuitry together with the static power (which are measured separately by applying all '0's to the inputs) from the total design power.
- Separated values of logic and interconnect power are used to calibrate each of the high-level power estimation models presented in chapters 3 and 4.
- They are also used for model verification and thus, provide an insight to the error distribution among different power components, which is presented in chapter 5.
- The measurement system provides precise measured power values. In order to ensure correct measured values, many measurements were repeated several times. The maximum relative error between repeated measurements was found to be 3%, thus confirming the validity of the obtained values for the purposes of power estimation.

6.2. Interconnect power estimation: conclusions

Complex DSP systems implemented in FPGAs consist of arithmetic components which are connected by data-path buses. Early prediction of interconnect power is necessary, since the FPGA programmable switches consume a significant amount of power and their number and position need to be optimized. However, the methods found in the literature target power estimation of both, control-oriented and data-oriented, designs. In order to estimate the power of data buses more accurately, a more centralized model is needed.

A high-level approach to estimate power consumption of global interconnections in FPGA DSP designs has been presented in chapter 3. The approach exploits almost linear dependence of the interconnect power on the wire length, which seems to occur due to the router objective to minimize wire delay. The main model characteristics can be summarized as follows:

- A detailed analysis of the connections between all combinations of two different module types (i.e. an adder and a multiplier), has proven that the interconnect power depends mostly on the modules' distance and the positions of the pins on the modules' boundaries. The pin position reflects the influence of the modules' shapes on short connections.
- The Rectilinear Steiner Tree algorithm is used to compute the lengths of the nets that correspond to the distances between the modules.
- The position of the pins on the component boundaries is accounted for through a simple summation of interconnect lengths inside the component.
- Three different routing zones are identified in data-path interconnections, depending on the distance between the modules. A different combination of wire types are used for routing the interconnections inside each zone.
- Only three unknown different coefficients are needed for the calibration of the power estimation model and they are obtained from the measured power corresponding to the analyzed point-to-point connections. Once obtained, this set of coefficients remains unchanged and can be applied to any other connection between the modules.
- A different set of coefficients is obtained for the connections between modules and I/O pins. It seems that the timing constraints are tighter when the connections that come from or go to the outside of a chip are routed. Consequently, these coefficients are smaller than the ones used for module interconnection.
- The results show that the accuracy of the model, lies in most cases, within 20% of the power measurements while taking into account the inherent noise in the nets' capacitances. The model performance has been explored over a wide range of input parameters,

signal components and module positions on the chip. The accuracy of the model has also been verified through on-board measurements of some test DSP designs.

- As the model uses only the relative positions of the modules and does not require any other placement information, it can be easily integrated into power optimization techniques that perform high-level synthesis combined with floorplanning. In these cases, the accuracy of the model will depend on the accuracy of the floorplan estimate.
- The main advantages of the presented model over other interconnection power models found in the literature are:
 - a low number of input parameters which are all available before the placement stage,
 - high accuracy similar to the accuracy achieved by using the post-placement models,
 - fast estimates obtained in the order of microseconds.

6.3. Logic power estimation: conclusions

In data-path oriented designs, power consumption of arithmetic components dominates the power of the other design elements. We have presented a high-level analytical approach to estimate the logic power consumption of adders, multipliers implemented in LUTs, and embedded multipliers in chapter 4. The main conclusions can be expressed as follows:

- A word-level signal model has been used for modelling the input signals. It includes a partition of the signal word according to the activity of its bits.
- Probability method has been developed for switching activity computation. The probability method computes the switching activity and probability at the outputs of the basic elements in the arithmetic components, which are later summed in order to obtain the total component switching activity.
- Glitching has been included into the model. Glitching effects are accounted for as additional switching activity generated inside the component.
- The methodology has been applied to different component structures such as: array multiplier, row adder tree multiplier, modified booth multiplier and ripple-carry adder. For each component structure, the final expression for the total switching activity varied according to the organization of basic elements inside the component.
- Logic power models resulting from the previous methodology (HLLM) are parameterized in terms of the clock frequency, the signal statistics, and the operands' word-lengths. They also account for the different power behaviour observed when considering zero-mean and non-zero mean input signals.

- One of the model's main advantages over other estimation methodologies are small computation times (order of milliseconds) and calibration times (less than 10 low-level measurements).
- A comparison was performed with XPower and two other methods proposed in the literature: the word-level table-based method and the Hamming distance model:
 - HLLM is more accurate than the table-based method specially for the smaller-size input operands. Since it is parameterized in terms of the word-lengths of both inputs, it is capable of producing accurate estimates for components with operands of different sizes, whereas in the table-based method only one component size is introduced as a variable in the equation.
 - HLLM achieves better accuracy than the Hd-model when considering highly-correlated signals, while the Hd-model gives better results when the switching activity of the input bits is distributed in a random fashion over the bit positions. Furthermore, the proposed model needs significantly smaller number of low-level simulations for its characterization than the Hd-model, and achieves better accuracy when resource sharing is used. Still, when the operand word-length is adjusted to the input word-length, for most of the applications the Hd-model is slightly more accurate than the HLLM, and it does not require any changes in its characterization method for different component structures.
 - HLLM is more accurate than XPower estimates when compared to the measured values. The relative error of HLLM lies between -25% and 20% with the average error below 10%, while XPower overestimates the power of the multipliers implemented in LUTs with the relative error going up to 450%, and underestimates the adder power with the relative error going up to 70%.
- Embedded block power estimation. A model used for power estimation of LUT-based components has been adapted to consider the embedded block implementation. The logic power model for embedded blocks has been characterized and verified with on-board power measurements, since it was proven that the low-level tool has a shortcoming when estimating the power of these blocks. The experimental results have shown that the accuracy of the model lies within [-20%, 20%], with a global average of 7.9%, which has been demonstrated to be better than the accuracy of a low level commercial tool (global average of 9.7%).
- The presented high-level logic power estimation model can be successfully used in high-level algorithms that are aimed at power optimization, as the only input information needed for the power estimate are the input signal statistics and the operand word-lengths. Additional features are:
 - Much shorter computation time and higher accuracy clearly give HLLM an advantage

over XPower tool, even though the low-level tool has detailed design and data information at the gate level.

- The model's fast and easy calibration through on-board measurements puts it far ahead of most of the other estimation methods proposed in the literature.

6.4. Complete estimation flow: conclusions

In chapter 5, a complete estimation flow for DSP circuits has been presented. The power models for interconnections and logic were combined in order to estimate total dynamic power of a design. The resulting high-level power estimation tool (HLM) can be used during high-level synthesis in order to select design changes which optimize power consumption.

Additionally, the analysis of the power distribution among the different power components was given for the measurements, XPower estimates and HLM estimates. It was observed that the accuracy of the XPower is quite low, with the relative errors an order of magnitude larger than HLM errors which lie below 30%.

The main characteristics that put HLM above the other existing estimation models are listed as follows:

- High accuracy confirmed by on-board measurements: 13.5% average error, up to 27.41% relative error, with expected improvement to below 20% by switching to bit-level statistics.
- Short computation time for model characterization: less than 10 measurements needed for model calibration.
- Short execution time: order of milliseconds for individual components, seconds for industrial-like designs.
- Small number of parameters needed for power estimation: input signal statistics, operand's word-lengths, pin ordering at the component boundaries, and distance between the modules.
- Easy integration with high-level power optimization techniques.

6.5. Future work

The work presented here considers only modules with registered inputs and outputs, as in pipelined designs. However, in non-pipelined designs, the amount of glitching can represent a high percentage of the total power. Our future work is oriented toward extending the models

to include glitching effects. When considering interconnects, glitching will be reflected directly in the switching activity value of the connections. When considering logic power estimation, the next step will be to develop a model that will account for the power of the non-registered components, by including the propagation of glitching between and throughout the components and considering the glitches in the component inputs while computing the component's power.

The results obtained in chapter 3 also suggest the importance of the routing congestion for the interconnect power estimation, since the minimum steiner tree underestimates the wire length in the connections between more than two modules. This parameter turns out to be even more important in today's large industrial designs where it is likely for a component to have a large fanout, and thus, it should be included in the interconnect power model.

One of the goals regarding interconnect power estimation is to develop an efficient floor-planning algorithm that will enable successful integration of the interconnection power model into high-level power optimization techniques, together with the estimation models for logic power that are already applicable to high-level power estimation.

When considering logic power estimation, another important task is modelling the word-level statistics at the outputs of the multipliers by using the so-called "triple-bit type model" presented in [KBN02]. The results indicate that the accuracy of the model could be significantly improved by introducing this methodology.

The current power model for embedded multipliers uses whole data input vectors as the input. These vectors can be extremely long, so they should be replaced by word-level statistics in order to accelerate the whole estimation process. Additionally, the model described here considers individual embedded blocks with operand size up to 18 bits. However, when multipliers are larger than 18×18 , several embedded blocks must be combined to perform the multiplication. Future work includes the extension of the presented model to estimate the power of larger operand size multiplications and to include the power of the connections between the registers and the embedded block.

Apart from the component types described in this work, a constant multiplier is often used for the implementation of FIR filters which are widely used in DSP applications. Therefore, future work also includes an adaptation of the presented logic model to the constant multiplier structure which is used in Xilinx IP cores.

Appendices

Appendix A

A.1. Resistance value for the measurement system

The measurement system contains two boards: a Xilinx XUP board and an Altera DSP development board (see Fig.1). Measurements of the core power on the XUP board are achieved by measuring the voltage over a resistance that is placed at the entrance of the core power supply of the chip. The resistance value is chosen so as to ensure the correct functionality of the power supply regulator on the XUP board as it is explained next.

The 1.5V power supply for the core voltage, is created by an synchronous buck-switching regulator from the 4.5V-5.5V external power input [Xil05]. The regulator employs a feedback loop in order to maintain a fixed value of the output voltage. The feedback controlling input to the regulator is taken directly from the core power supply pin on the XC2VP30 device and is marked as point B in Fig. 2a. This connection is integrated on the XUP board and is marked with a thicker line in Fig. 2a. Therefore, the voltage at the input of the chip, V_B , is maintained at 1.5V meaning that the functionality of the chip itself is guaranteed.

A simplified block diagram of the PWM buck-switching regulator is given in Fig. 2b. As feedback is obtained through point B in Fig. 2a, the voltage value at the output of the regulator (marked as A in both figures) will have the value:

$$V_A = V_B + R * i \quad (1)$$

The regulator is buck-switching, so it is important to avoid the saturation of the internal coil of the regulator. The saturation will occur when the average voltage value at the output of the buck converter surpasses the value of the average voltage on the other coil end [ED01] (marked as D). The average voltage value at point D equals to:

$$V_D = d * V_{in} \quad (2)$$



Figure 1: Measurement system

where d is the duty cycle of the buck converter. Consequently, the saturation of the coil will not occur as long as voltage V_A is smaller or equal to the maximum voltage V_D . From equations 1 and 2, we obtain the condition that has to be fulfilled:

$$V_B + R * i \leq d_{max} \cdot V_{in} \quad (3)$$

Since the value d_{max} is not provided in the regulator's data sheet, we have obtained it experimentally and it equals to 0.5. Therefore, we obtained that the average voltage over resistance $V_D - V_A$ should not surpass the value of 1V. We have measured power for several different resistance values, starting from 1 ohm, in order to find the largest one that would fulfill the condition of the maximum voltage value. The circuits used in our measurements contained only one to four multiplier or adder components with operand sizes smaller than 17 bits, connected directly to the I/O pins. Thus, their power consumption was always small enough to allow a resistance value of 10 ohms.

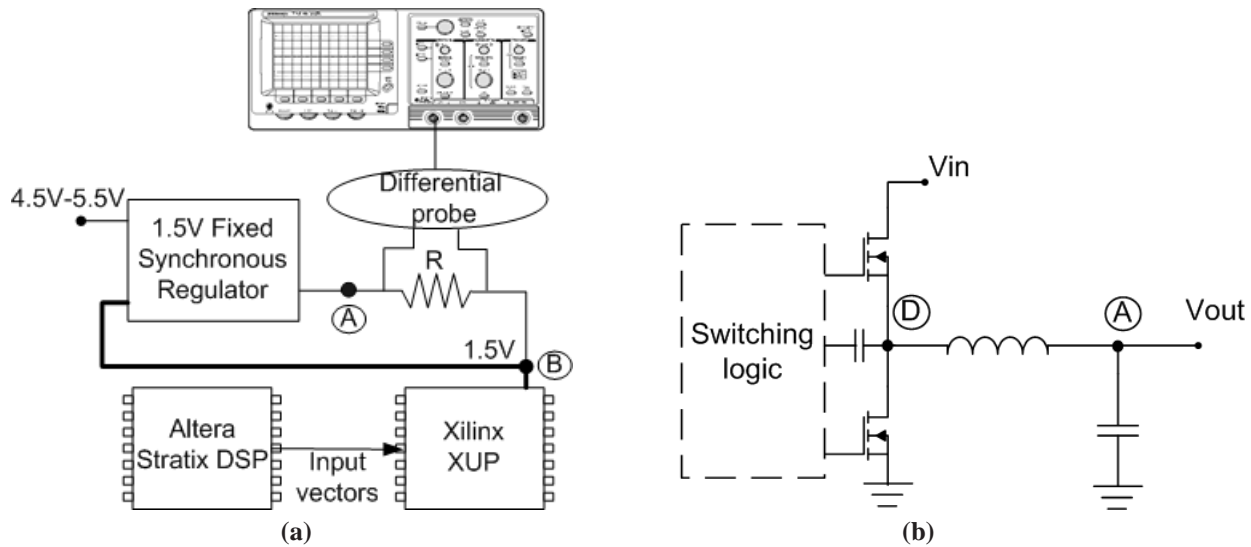


Figure 2: a) Measurement setup b) Buck-switching PWM regulator

A.2. MARWEL

MARWEL is a software tool developed in C++, which is used for extraction of the routing information on the design implemented in Xilinx chips. It takes an input file in .xdl format. This file is a text version of the placed-and-routed design and is created by Xilinx tool XDL. First, we will give an overview of the .xdl file structure, as this information is essential for MARWEL. Then, the structure of MARWEL will be described.

A.2.1. XDL file structure

The .xdl file is obtained through the following command of the Xilinx ISE framework:

$$-ncd2xdl \text{ SourceFile.ncd OutputFile.xdl} \quad (4)$$

An inverse command is also available: *-xdl2ncd*, for converting the .xdl text file to a .ncd graphic representation of the placed-and-routed design.

The .xdl file consists of two parts. In the first part, there is a list of all design instances together with their configuration and location on the FPGA board. Design instances belong to one of the following groups: logic blocks, I/O pins, DCMs, and multiplexers. Each instance description begins with a word "inst" (see Fig. 3). It is followed by the instance name which is later used for describing all the nets where this instance has some of its pins connected. Next to the name there is information on the type of the instance, followed by the position of the instance in the FPGA board. If it is placed inside a CLB, then the slice position inside the CLB is also specified.

Instance name	Logic or I/O	Instance CLB position	Instance slice position
↓	↓	↓	↓
<pre>inst "m/N1479" "SLICE",placed R42C41 SLICE_X81Y77 cfg " BXINV::#OFF BXOUTUSED::#OFF BYINV::#OFF BYINVOUTUSED::#OFF BYOUTUSED::#OFF CEINV::#OFF CLKINV::#OFF COUTUSED::0 CY0F::PROD CY0G::PROD CYINIT::CIN CYSELF::F CYSELG::G DIF_MUX::#OFF DIGUSED::#OFF DIG_MUX::#OFF DXMUX::#OFF DYMUX...</pre>			

Figure 3: XDL file syntax: part I

	Line number
Net name → net "operand1<1>" ,	1
Net end → inpin "m/N214" BX ,	2
Net beginning → outpin "input1<1>" IQ1 ,	3
Switch matrix → pip RIOIR38 I_Q10 -> OMUX5 ,	4
	5
CLB position → pip R39C46 OMUX_SW5 -> S6BEG9 ,	6
	7
	8
	9
	10
	11
	;

Figure 4: XDL file syntax: part II

The basic instance description is followed by its configuration details. Since MARWEL uses only the first line of each instance description (marked with the red circle in Fig. 3), most of the configuration data is not relevant for the extraction of design routing properties.

The second part of the XDL file contains a list of all the nets in the design. An example of a net's description is given in Fig. 4. It always begins with a word "net", followed by the net's name. Next, the name of the pin where the net begins and the names of the pins where it ends are listed. These names correspond to some of the instance names given in the first part of the .xdl file. It is important to note that there is only one "output" pin, while there can be several "input" pins.

The identifier "pip" is used to describe a connection inside the switch matrix. It is followed by the position of the switch matrix (the notations are the same as those used for the CLB position described in the first part of the .xdl file). Finally, a description of the wires that are connected inside that particular switch matrix is provided. The positions of the switch matrices as well as the wire description are essential information for extracting design routing properties.

Hence, we give their more detailed explanation.

CLB position

An FPGA is an array of CLBs, where each CLB position is defined by its row and column number. For example, the CLB position marked with a blue circle in Fig. 4, begins with a letter *R*, followed by a number which represents the row coordinate. The same stands for letter *C* and the column coordinate. Beside this notation, there are several others which are used for CLBs that have some particular locations. For example, in the fourth line of Fig. 4, the CLB location is *RIOIR38*. This is a typical notation for the CLB containing I/O pins, that is placed on the right-hand side of the chip (the first *R* stands for right, and the second one stands for row). As the column number obviously corresponds to the maximum column coordinate (since the CLB is located in the last column on the right-hand side of the chip), the only information that is needed is the row position. Beside this class of CLBs, there are several other types of CLBs: CLBs placed in the corners of the FPGA, CLBs placed next to embedded blocks, CLBs with switch matrices dedicated to clock routing resources etc.

Wire description

The four types of global wires are described in the following way:

- Direct line: it starts with a notation *OMUX* which is then followed by a track number and/or a direction, which depends on whether the notation is used for the beginning or the end of a direct line. For example, line 4 in Fig. 4 marks the beginning of the direct line in track 5, and in line 5 we can see that this direct line ends with a direction south-west (*SW*).
- Double line: it starts with a direction, followed by a number 2 which stands for double, and the abbreviation *BEG*, *MID*, *END* which stand for the beginning, middle and end parts of the wire respectively. At the end of a notation there is a track number (for example, see line 6 in Fig. 4, where *W2BEG8* stands for a beginning of a double line in track 8 that has a direction towards west).
- Hex line: it has the same structure as the double line. For example, the notation marked with a blue circle in line 6 in Fig. 4 marks the end of a hex line in track 9 that had a direction towards south.
- Long line: there are two notations for a long line: *LV* or *LH* depending on whether its direction is horizontal *H*, or vertical *V*.

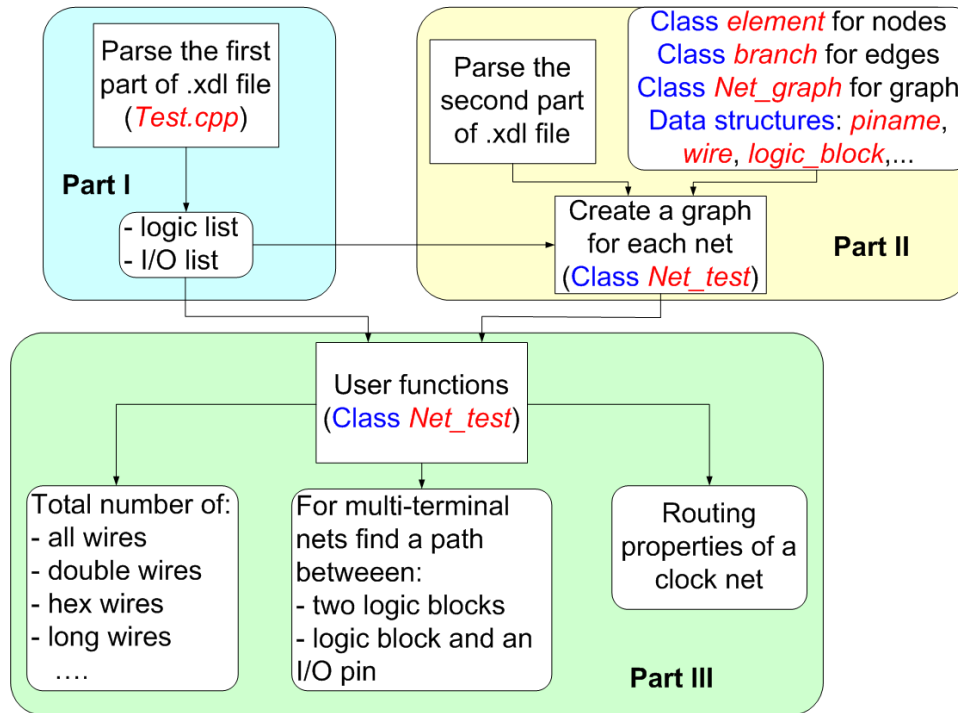


Figure 5: MARWEL structure

A.2.2 MARWEL structure

MARWEL represents nets as graphs, where CLBs are represented by the nodes and wires connecting two CLBs are represented by the edges of the graph. Functions provided in the Graph Template Library [GTL] have been used in order to describe the design nets as graphs. This has facilitated the circuit description and the search algorithms applied in order to find the specific design information.

In the continuation, we give a description of the MARWEL structure. It consists of three parts (see Fig. 5). First, it parses the first part of the .xdl file and gathers the information about the names and the positions of the logic and I/O pins of the design. It separates the list containing all I/O pins from the list that contains logic blocks. This is necessary for the purpose of the work presented here, since we need to identify the connections that go to or from I/O pins separately from the local connections between the CLBs inside an arithmetic component.

Second, each net is transformed into a graph, where nodes represent switch matrices and edges represent wires. There are specially designed classes for these two types of items called *branch* and *element*. Class *element* represents a node in a graph and contains the coordinates of the switch matrix, while class *branch* represents an edge in a graph, and contains a pointer to the previous branch which is connected to the current branch, the type, direction and track of the wire, as well as the positions of the wire's beginning and end. Class *Net_graph* contains the node and edge maps of the graph. Beside these classes, there are some additional class objects

that are used for a better structurization of the tool and basically correspond to C language structures. The functions that are used for creating a graph for each net are located in a class *Net_test* which has an object of the class *Net_graph* as a member. These functions are the most complex within MARWEL, since they have to identify all possible connections inside the net. There is no available published documentation on the XDL tool, so this task is extremely difficult. Failing to identify only one connection, leads to an unfinished graph, as each wire has a single particular predecessor. Furthermore, the order of the connections in the .xdl file does not necessarily correspond to the order of the connections in a net. Thus each time a new edge is added to the graph, the graph has to be searched from the beginning in order to find the correct place for the new edge.

Finally, the third part consists of a large number of functions designed for user purposes. They are all incorporated also into a class *Net_test*. These user functions can be divided into three groups:

- Net functions: for each net there are functions that can compute the total number of all wires, hex wires, double wires, direct wires, long wires, local wires inside a CLB, and switch matrices used for the routing of a net.
- Path finder functions: for multi-terminal nets these functions can find how many routing resources and of which type have been used for routing a part of a net between two specified logic blocks, as well as between a logic block and an I/O pin.
- Clock functions: a clock net is routed via special-purpose wires, and their notation is quite different from that of standard global routing resources. Hence, some special functions are included for analysing the routing properties of the clock nets.

The output data of these functions are recorded in text files. In this work, we have mostly used the function that returns the length and wire types used for a specific interconnection between two module pins or a module pin and an I/O pin.

MARWEL was initially designed for the Virtex-2 XCV26000 family of Xilinx FPGAs. Although, it is based on the structure of these chips, it was then modified to describe some other Virtex-2 families and Virtex-2 Pro families by introducing small changes into the code. The most frequently used version of MARWEL was adapted to the Virtex-2 Pro XC2VP30 family as it corresponds to the family used for the on-board measurements. These modifications were possible due to the fact that the notations for the names of CLBs and types of wires remain the same in all cases, and the only parameters that change are the numbers of rows and columns in the chip and the positions of the embedded blocks on the FPGA board. In addition to the Virtex-2 and Virtex-2 Pro families, MARWEL is able to extract circuit information from the Spartan III and Virtex-4 families. In these cases, the notations for the CLBs are different, but the notations for the types of the wires remain the same. As the code is well-structured,

these adjustments can be easily introduced into the code, by changing only one function which provides the information about the CLB's position according to its notation.

Bibliography

- [AAR⁺07] J. L. Ayala, D. Atienza, P. Raghavan, M. Lopez-Vallejo, F. Catthoor, and D. Verkest. Energy-Aware Compilation and Hardware Design for VLIW Embedded Systems. *Int. Journal of Parallel Programming*, 3(1-2):73–82, 2007.
- [Act] Actel. www.actel.com.
- [AFJS07] N. Abdelli, A.-M. Fouillart, N. Julien, and E. Senn. High-Level Power Estimation of FPGA. In *IEEE International Symposium on Industrial Electronics*, pages 925–930, June 2007.
- [Ale97] M. J. Alexander. Power Optimization for FPGA Look-up Tables. In *Proc. on ISPD*, pages 156–162, 1997.
- [Alt] Altera. www.altera.com.
- [Alt05] Altera. Stratix II vs. Virtex-4 Power Comparison & Estimation Accuracy White Paper. www.altera.com, 2005.
- [ALV03] J. L. Ayala and M. Lopez-Vallejo. Power-Aware Compilation for Register File Energy Reduction. *Int. Journal of Parallel Programming*, 31(6):449–465, December 2003.
- [ALV04] J. L. Ayala and M. Lopez-Vallejo. Improving Register-file Banking with a Power-aware Unroller. In *Workshop on Power-Aware Real-Time Computing*, 2004.
- [AMK⁺05] A. Agarwal, S. Mukhopadhyay, C. H. Kim, A. Raychowdhury, and K. Roy. Leakage Power Analysis and Reduction: Models, Estimation and Tools. *IEE Proceedings - Computers and Digital Techniques*, 152(3):353–368, May 2005.
- [AN02] J. H. Anderson and F. N. Najm. Power-Aware Technology Mapping for LUT-Based FPGAs. In *IEEE Int. Conference on FPT*, pages 211–218, 2002.
- [AN04a] J. H. Anderson and F. N. Najm. Interconnect Capacitance Estimation for FPGAs. In *Proc. of the ASP-DAC 2004*, pages 713–718, January 2004.

- [AN04b] J. H. Anderson and F. N. Najm. Power Estimation Techniques for FPGAs. *IEEE Trans. On VLSI Systems*, 10(12):1015–1027, October 2004.
- [AN06] J. H. Anderson and F. Najm. Active Leakage Power Optimization for FPGAs. *ACM Trans. on Design Automation of Electronic Systems*, 25(3):423–437, March 2006.
- [AR04] E. Ahmed and J. Rose. The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density. *IEEE Trans. on VLSI Systems*, 12(1):288–298, March 2004.
- [AR05] J. L. Ayala Rodrigo. *Power Estimation and Power Optimization Policies for Processor-based Systems*. PhD thesis, Universidad Politécnica de Madrid, 2005.
- [Atm] Atmel. www.atmel.com.
- [BB03] S. Balachandran and D. Bhatia. A-Priori Wirelength and Interconnect Estimation Based on Circuit Characteristics. In *Proc. on SLIP*, pages 77–84, 2003.
- [BBPDM99] L. Benini, A. Bogliolo, B. A. Paleologo, and G. De Micheli. Policy Optimization for Dynamic Power Management. *IEEE Trans. on CAD*, 18(6):813–833, June 1999.
- [BHU03] J. Becker, M. Huebner, and M. Ullmann. Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-offs and Limitations. In *Proc. of the 16th Symp. on Integrated Circuits and Systems Design*, pages 283–288, September 2003.
- [BOI94] M. Borah, R. M. Owens, and M. J. Irwin. An Edge-based Heuristic for Steiner Routing. *IEEE Trans. On Computer-Aided Design for Integrated Circuits and Systems*, 13(12):1563–1568, November 1994.
- [BR97] V. Betz and J. Rose. VPR: A New Packing, Placement and Routing Tool for FPGA Research. In *Proc. of the 7th FPLA*, pages 213–222, August 1997.
- [BRS00] K. Bazargan, A. Ranjan, and M. Sarrafzadeh. Fast and Accurate Estimation of Floorplans in Logic/High-level Synthesis. In *Proc. on Great Lakes Symposium on VLSI*, pages 95–100, 2000.
- [CCC⁺06] G. Caffarena, G. A. Constantinides, P.Y.K. Cheung, C. Carreras, and O. Nieto-Taladriz. Optimal Combined Word-length Allocation and Architectural Synthesis of Digital Signal Processing Circuits. *IEEE Trans. on Circuits and Systems II, Express Briefs*, 53(5):339–343, May 2006.
- [CCC07] J. A. Clarke, G. A. Constantinides, and P. Y. K. Cheung. On the Feasibility of Early Routing Capacitance Estimation for FPGAs. In *Proc. on Field-Programmable Logic and Applications*, pages 234–239, August 2007.
- [CCCS08] J. A. Clarke, G. A. Constantinides, P. Y. K. Cheung, and A. M. Smith. Glitch-aware Output Switching Activity from World-level Statistics. In *Proc. on ISCAS*, pages 1792–1795, May 2008.

- [CCF03] D. Chen, J. Cong, and Y. Fan. Low-Power High-Level Synthesis for FPGA Architectures. In *Proc. of ISLPED'03*, pages 134–139, August 2003.
- [CCLH04] D. Chen, J. Cong, F. Li, and L. He. Low Power Technology Mapping for FPGA Architectures with Dual Supply Voltages. In *Proc. of FPGA'04*, pages 109–117, February 2004.
- [CCX05] D. Chen, J. Cong, and J. Xu. Optimal Module and Voltage Assignment for Low Power. In *Proc. on ASP-DAC*, pages 850–855, 2005.
- [CGC05] J. A. Clarke, A. A. Gaffar, and G. A. Constantinides. Parameterized Logic Power Consumption Models for FPGA-based Arithmetic. In *Proc. of 11th Int. Workshop on Field-Programmable Logic and Applications, LNCS vol. 2147*, pages 626–629, August 2005.
- [CGCC06] J. A. Clarke, A. A. Gaffar, G. A. Constantinides, and P. Y. K. Cheung. Fast word-level power models for synthesis of FPGA-based arithmetic. In *Proc. ISCAS*, pages 1299–1302, August 2006.
- [Che94] C. L. E. Cheng. Accurate and Efficient Placement Routability Modelling. In *Proc. on ICCAD*, pages 690–695, 1994.
- [CJC00] J. Choi, J. Jeon, and K. Choi. Power Minimization of Functional Units by Partially Guarded Computation. In *Proc. of ISLPED*, pages 131–136, 2000.
- [CJMP03] S. Choi, J-W. Jang, S. Mohanty, and V. Prasanna. Domain-Specific Modeling for Rapid Energy Estimation of Reconfigurable Architectures. *The Journal of Supercomputing*, 26(3):259–281, November 2003.
- [CKCW06] N. Chan King Choy and S. J. E. Wilton. Activity-Based Power Estimation and Characterization of DSP and Multiplier Blocks in FPGAs. In *International Conference on FPT*, pages 253–256, December 2006.
- [CLW06] S. Y. L. Chin, C. S. P. Lee, and S. J. E. Wilton. Power Implications of Implementing Logic using FPGA Embedded Memory Arrays. In *Proc. of the Int. Conference on Field-Programmable Logic and Applications*, pages 1–8, August 2006.
- [Cor] Lattice Semiconductor Corp. www.latticesemi.com.
- [CP97] J. M. Chang and M. Pedram. Energy Minimization using Multiple Supply Voltages. *IEEE Trans. on VLSI*, 5(4):436–443, December 1997.
- [CPHC03] K. Cho, J. Park, J. Hong, and G. Choi. 54x54-bit Radix-4 Multiplier based on Modified Booth Algorithm. In *Proc. on GLSVLSI'03*, pages 233–236, April 2003.
- [CRP94] T. Chou, K. Roy, and S. Prasad. Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching. In *Proc. of the 1994 IEEE/ACM Int. Conference on Computer-aided Design*, pages 300–303, June 1994.

- [CS00] P. Christie and D. Stroobandt. The Interpretation and Application of Rent's Rule. *IEEE Trans. on VLSI*, 8(6):639–648, December 2000.
- [CW08] C. Chu and Y. Wong. FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. *IEEE Trans. on Computer-Aided Design*, 27(1):70–83, January 2008.
- [CX08] J. Cong and J. Xu. Simultaneous FU and Register Binding based on Network Flow Method. In *Proc. of DATE'08*, pages 1057–1062, March 2008.
- [DAR07] Y. A. Durrani, A. Abril, and T. Riesgo. Efficient Power Macromodelling Technique for IP-based Digital System. In *Proc. of the IEEE Int. Symposium on Circuits and Systems*, pages 1145–1148, May 2007.
- [DDM98a] J. A. Davis, V. K. De, and J. Meindl. A Stochastic Wire Length Distribution for Gigascale Integration (GSI) - Part I: Derivation and Validation. *IEEE Trans. On Electron Devices*, 45(3):580–589, March 1998.
- [DDM98b] J. A. Davis, V. K. De, and J. Meindl. A Stochastic Wire Length Distribution for Gigascale Integration (GSI) - Part II: Applications to Clock Frequency, Power Dissipation, and Chip Size Estimation. *IEEE Trans. On Electron Devices*, 45(3):590–597, March 1998.
- [dHAJW⁺07] J. A. de Holanda, J. Assumpcao Jr., D. F. Wolf, E. Marques, and J. M. P. Cardoso. On Adapting Power Estimation Models for Embedded Soft-Core Processors. In *IEEE Second International Symposium on Industrial Embedded Systems*, pages 348–354, July 2007.
- [Don79] W. E. Donath. Placement and Average Interconnection Lengths for Computer Logic. *IEEE Trans. on Circuits and Systems*, 26(4):272–277, December 1979.
- [DR06] Y. A. Durrani and T. Riesgo. Power Macromodelling for IP modules. In *Proc. of the IEEE Int. Conf. on Electronics, Circuits and Systems*, pages 1172–1175, December 2006.
- [DT05] V. Degalahal and T. Tuan. Methodology for High Level Estimation of FPGA Power Consumption. In *Proc. of 2005 conference on ASP-DAC*, pages 657–660, January 2005.
- [ED01] R.W. Erickson and D.Maksimovic. *Fundamentals of Power Electronics*. Springer, 2001.
- [EJH⁺04] D. Elléouet, N. Julien, D. Houzet, J. G. Cousin, and E. Martin. Power Consumption Characterization and Modeling of Embedded Memories in Xilinx Virtex 400E FPGA. In *Euromicro Symposium on Digital System Design*, pages 394–401, September 2004.
- [EJH06] D. Elléouet, N. Julien, and D. Houzet. A High Level SoC Power Estimation based on IP Modeling. In *Intern. Parallel and Distributed Processing Symposium*, September 2006.

- [ESJ06] D. Elléouet, Y. Savary, and N. Julien. An FPGA Power Aware Design Flow. In *Proc. on PATMOS 2006, LNCS vol. 4148*, pages 415–424, September 2006.
- [FWAW05] M. French, L. Wang, T. Anderson, and M. Wirthlin. Post Synthesis Level Power Modelling of FPGAs. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, pages 281–282, April 2005.
- [Geo] Geosteiner. www.diku.dk/geosteiner/.
- [GGH97] R. Gonzalez, B. M. Gordon, and M. A. Horowitz. Supply and Threshold Voltage Scaling for Low Power CMOS. *IEEE Journal of Solid-state circuits*, 32(8):1210–1216, August 1997.
- [GJ77] M. Garey and D. S. Johnson. The Rectilinear Steiner Problem is NP-complete. *SIAM J. Appl. Math.*, 32(3):826–834, June 1977.
- [GN00] S. Gupta and F. N. Najm. Power Modeling for High Level Power Estimation. *IEEE Trans. On VLSI Systems*, 8:18–29, March 2000.
- [Gro] Andraka Consulting Group. Multiplication in FPGAs. <http://www.fpga-guru.com/multipli.htm>.
- [GTL] GTL. <http://www.infosun.fim.uni-passau.de/GTL/>.
- [GTV⁺04] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan. Reducing Leakage Energy in FPGAs using Region-constrained Placement. In *Proc. on FPGA*, pages 51–58, 2004.
- [GZJ03] P. Gupta, L. Zhong, and N. K. Jha. A High-level Interconnect Power Model for Design Space Exploration. In *Proc. of ICCAD'03*, pages 551–558, April 2003.
- [HLL⁺06] C. H. Ho, P. H. W. Leong, W. Luk, S. J. E. Wilton, and S. Lopez-Buedo. Virtual Embedded Blocks: A Methodology for Evaluating Embedded Elements in FPGAs. In *Proc. on FCCM*, pages 35–44, April 2006.
- [HSS⁺02] D. Helms, E. Schmidt, A. Schulz, A. Stammermann, and W. Nebel. An Improved Power Macro-Model for Arithmetic Datapath Components. In *Proceedings of the 12th PATMOS '02*, pages 16–24, January 2002.
- [HTF95] H. Al-Twaijry and M. Flynn. Performance/Area Tradeoffs in Booth Multipliers. Technical Report: CSL-TR-95-684, November 1995.
- [Hwa76] F. K. Hwang. On Steiner Minimal Trees with Rectilinear Distance. *SIAM Journal on Appl. Math.*, 30:104–114, January 1976.
- [JC08] R. Jevtic and C. Carreras. Analytical High-level Power Model for LUT-based Components. In *LNCS (Springer)*, volume 5349, pages 369–378, September 2008.
- [JC09] R. Jevtic and C. Carreras. Power Estimation of Embedded Multipliers in FPGAs. *IEEE Trans. on VLSI*, 2009.

- [JCC07a] R. Jevtic, C. Carreras, and G. Caffarena. High-level Switching Activity Models for FPGA Multipliers. In *Proc. of FPGA'07*, pages 224–225, February 2007.
- [JCC07b] R. Jevtic, C. Carreras, and G. Caffarena. Switching Activity Models for Power Estimation in FPGA Multipliers. In *LNCS (Springer)*, volume 4419, pages 201–213, March 2007.
- [JCC08] R. Jevtic, C. Carreras, and G. Caffarena. Fast and Accurate Power Estimation of FPGA DSP Components Based on High-level Switching Activity Models. *International Journal of Electronics*, 95(7):653–668, July 2008.
- [JCH08] R. Jevtic, C. Carreras, and D. Helms. A Comparison of Approaches for High-level Power Estimation for LUT-based DSP Components. In *Proc. of ReCon-Fig'08*, pages 361–366, December 2008.
- [JCP09] R. Jevtic, C. Carreras, and V. Pejovic. Floorplan-based FPGA Interconnect Power Estimation in DSP Circuits. In *Proc. of SLIP'09*, July 2009.
- [JKSN99] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel. A New Parameterizable Power Macro-Model for Datapath Components. In *Proc. on DATE '99*, pages 29–36, January 1999.
- [JKSN00] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel. Power Macromodelling for Firm Macros. In *Proceedings of the 10th PATMOS '00*, pages 24–35, September 2000.
- [JTB04] T. Jiang, X. Tang, and P. Banerjee. Macro-models for High Level Area and Power Estimation on FPGAs. In *Proc. of GLSVLSI'04*, pages 26–28, April 2004.
- [KAA⁺07] F. Klein, G. Araujo, R. Azevedo, R. Leao, and L. C. V. dos Santos. On the Limitations of Power Macromodeling Techniques. In *Proc. of ISVLSI '07*, pages 395–400, March 2007.
- [KBB01] P. Kannan, S. Balachandran, and D. Bhatia. fGREP - Fast Generic Routing Demand Estimation for Placed FPGA Circuits. In *Proc. of 11th Int. Workshop on Field-Programmable Logic and Applications, LNCS vol. 2147*, pages 37–47, August 2001.
- [KBB04] P. Kannan, S. Balachandran, and D. Bhatia. On Metrics for Comparing Interconnect Estimation Methods for FPGAs. *IEEE Trans. on VLSI*, 12(4):381–385, April 2004.
- [KBB06] P. Kannan, S. Balachandran, and D. Bhatia. Interconnect Estimation for FPGAs. *IEEE Trans. on CAD*, 25(8):1523–1534, August 2006.
- [KBN02] E. D. Kyriakis-Bitzaros and S. Nicolaidis. Estimation of Bit-level Transition Activity in Data-paths based on Word-level Statistics and Conditional Entropy. *IEE Proc. on Circuits, Devices and Systems*, 149(4):234–240, August 2002.
- [KC97] D. Kim and K. Choi. Power-conscious High-level Synthesis using Loop Folding. In *Proc. on DAC*, pages 441–445, Jun 1997.

- [KR06] I. Kuon and J. Rose. Measuring the gap between ASICs and FPGAs. In *Proc. of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 21–30, February 2006.
- [Kri04] A. Krishnamoorthy. Minimizing IC Power without Sacrificing Performance. *EETimes*, May 2004.
- [KS01] K.I. Kum and W. Sung. Combined Word-length Optimization and High-level Synthesis of Digital Signal Processing Systems. *IEEE Trans. Circuits Syst.*, 20(8):921–930, August 2001.
- [LCHC03] F. Li, D. Chen, L. He, and J. Cong. Architecture Evaluation for Power-efficient FPGAs. In *Proc. of FPGA'03*, pages 175–184, February 2003.
- [LEG07] M. Lin and A. El Gamal. Routing Fabric for Monolithically Stacked 3D-FPGA. In *Proc. of FPGA'07*, pages 3–12, February 2007.
- [LHW97] Y. Lin, C. Hwang, and A. C. Wu. Scheduling Techniques for Variable Voltage Low Power Designs. *ACM Trans. on Design Automation of Electronic Systems*, 2(2):81–97, April 1997.
- [lib] GNU MP library. www.swox.com/gmp/.
- [LKS01] J. Lou, Krishnamoorthy, and H. S. Sheng. Estimating Routing Congestion using Probability Analysis. In *Proc. on ISPD*, pages 112–117, 2001.
- [LLCC05] H. G. Lee, K. Lee, Y. Choi, and N. Chang. Cycle-Accurate Energy Measurement and Characterization of FPGAs. *Analog Integrated Circuits and Signal Processing, Springer Netherlands*, 42(3):239–251, March 2005.
- [LLHC04] F. Li, Y. Lin, L. He, and J. Cong. Low Power FPGA using Predefined Dual-Vdd/Dual-Vt Fabrics. In *Proc. on FPGA*, pages 42–50, 2004.
- [LLW07] J. Lamoureux, G. G. Lemieux, and S. J. E. Wilton. GlitchLess: An Active Glitch Minimization Technique for FPGAs. In *Proc. of Intern. Sympos. on FPGAs*, pages 156–165, February 2007.
- [LLW08] J. Lamoureux, G. G. Lemieux, and S. J. E. Wilton. GlitchLess: Dynamic Power Minimization in FPGAs Through Edge Alignment and Glitch Filtering. *IEEE Trans. On VLSI Systems*, 16(11):1521–1534, November 2008.
- [LMR01] M. Lundberg, K. Muhammad, and K. Roy. A Novel Approach to High-Level Switching Activity Modeling With Applications to Low-Power DSP System Synthesis. *IEEE Trans. On Signal Processing*, 49(12):3157–3167, December 2001.
- [LR95] P. Landman and J. Rabaey. Architectural Power Analysis: The Dual Bit Type Method. *IEEE Trans. On VLSI Systems*, 3(2):173–187, March 1995.
- [LW03] J. Lamoureux and S. J. E. Wilton. On the Interaction of Power-aware FPGA CAD Algorithms. In *Proc. of ICCAD'03*, pages 701–708, November 2003.

- [MCSB06] V. Manohararajah, G. R. Chiu, D. P. Singh, and S. D. Brown. Difficulty of Predicting Interconnect Delay in a Timing Driven FPGA CAD Flow. In *Proc. on SLIP*, pages 3–8, 2006.
- [MDG⁺97] J. Monteiro, S. Devadas, A. Ghosh, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational Logic Circuits Using Symbolic Simulation. *IEEE Trans. on CAD*, 16(1):121–127, January 1997.
- [MRC03] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi. Transient power Minimization through Datapath Scheduling in Multiple Supply Voltage Environment. In *Proc. on ICECS*, pages 300–303, 2003.
- [MRT08] F. Machado, T. Riesgo, and Y. Torroja. Disjoint Region Partitioning for Probabilistic Switching Activity Estimation at Register Transfer Level. In *Proc. of the PATMOS'08*, pages 1145–1148, September 2008.
- [MS08] F. Machado Sánchez. *Estudio de la actividad de conmutación de circuitos electrónicos digitales descritos en el nivel de transferencia de registros mediante técnicas probabilísticas. Propuesta de un método de estimación*. PhD thesis, Universidad Politécnica de Madrid, 2008.
- [MSCL07] T. S. T. Mak, P. Sedcole, P. Y. K. Cheung, and W. Luk. Average Interconnect Delay Estimation for On-FPGA Communication Links. *Electronics letters*, 43(17):918–920, August 2007.
- [Naj91] F. N. Najm. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proc. of ACM/IEEE Design Automation Conference*, pages 644–649, June 1991.
- [Neb04] W. Nebel. System-level Power Optimization. In *Euromicro Symposium on Digital System Design*, pages 27–34, September 2004.
- [NHCB02] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee. Accurate Area and Delay Estimators for FPGAs. In *Proc. on DATE*, pages 862–869, 2002.
- [NM97] W. Nebel and J. Mermet. *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, Massachusetts, USA, 1997.
- [NS00] K. Nose and T. Sakurai. Analysis and Future Trend of Short-circuit Power. *IEEE Trans. on CAD*, 19(9):1023–1030, September 2000.
- [Ope] Opencores. www.opencores.org.
- [Pap91] A. Papoulis. *Probability, random variables and stochastic processes*. McGraw Hill, New York, 1991.
- [PHB06] K. Paulsson, M. Huebner, and J. Becker. On-line Optimization of FPGA Power Dissipation by Exploiting Run-time Adaption of Communication Primitives. In *Proc. of the 16th Symp. on Integrated Circuits and Systems Design*, pages 283–288, September 2006.

- [Qui] QuickLogic. www.quicklogic.com.
- [RDJ99] A. Raghunathan, S. Dey, and N. K. Jha. Register Transfer Level Power Optimization with Emphasis on Glitch Analysis. *IEEE Trans. on CAD*, 18(8):1114–1131, August 1999.
- [RJD98] A. Raghunathan, N. K. Jha, and S. Dey. *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Massachusetts, USA, 1998.
- [RMMM03] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-submicrometer CMOS Circuits. *Proc. of the IEEE*, 91(2):305–327, February 2003.
- [RP00] K. Roy and S. Prasad. *Low Power CMOS VLSI Circuit Design*. Editorial John Wiley & Sons, 2000.
- [RPM⁺03] J. Rius, A. Peidro, S. Manich, R. Rodriguez, and E. Boemo. Measuring Power and Energy of CMOS Circuit: a Comparative Analysis. In *Proc. of XVIII Conference on Design of Circuits and Integrated Systems*, pages 89–94, February 2003.
- [RRRL01] V. Raghunathan, S. Ravi, A. Raghunathan, and Lakshminarayana. Transient Power Management through High-level Synthesis. In *Proc. on ICCAD*, pages 545–552, 2001.
- [RSH97] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj. Analytical Estimation of Signal Transition Activity from Word-Level Statistics. *IEEE Trans. on CAD*, 16(7):718–733, March 1997.
- [RSN06] A. Reimer, A. Schulz, and W. Nebel. Modelling Macromodules for High-level Dynamic Power Estimation of FPGA-based Digital Designs. In *Proc. of ISLPED '06*, pages 151–154, January 2006.
- [RW05] N. Rollins and M. J. Wirthlin. Reducing Energy in FPGA Multipliers Through Glitch Reduction. In *Proc. of MAPLD*, September 2005.
- [SGLVLB06] M. A. Sanchez, M. Garrido, M. Lopez Vallejo, and C. Lopez-Barrio. Automated Design Space Exploration of FPGA-based FFT Architectures based on Area and Power Estimation. In *International Conference on Field Programmable Technology 2006*, pages 127–134, September 2006.
- [She99] N. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, Massachusetts, USA, 1999.
- [SHS⁺03a] A. Stammermann, D. Helms, M. Schulte, A. Schulz, and W. Nebel. Binding, Allocation and Floorplanning in Low Power High-Level Synthesis. In *Proc. ACM/IEEE Int. Conference on Computer Aided Design (ICCAD'03)*, pages 544–550, November 2003.

- [SHS⁺03b] A. Stammermann, D. Helms, M. Schulte, A. Schulz, and W. Nebel. Interconnect-driven Low Power High-level Synthesis. In *Proc. of PATMOS'03, LNCS*, volume 2799, pages 131–140, September 2003.
- [SJ01] L. Shang and N. K. Jha. High-level Power Modeling of CPLDs and FPGAs. In *Proc. of the Int. Conference on Computer Design*, pages 46–53, March 2001.
- [SK08] H. Sankaran and S. Katkoori. Bus Binding, Re-ordering, and Encoding for Crosstalk-Producing Switching Activity Minimization during High Level Synthesis. In *Proc. on International Symposium on Electronic Design, Test and Applications*, pages 454–457, 2008.
- [SKB02] L. Shang, A. S. Kaviani, and K. Bathala. Dynamic Power Consumption in Virtex-II FPGA Family. In *Proceedings of the 2002 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, pages 157–164, February 2002.
- [Smi06] A. M. Smith. *Heterogeneous Reconfigurable Architecture Design: An Optimization Approach*. PhD thesis, Imperial College of London, 2006.
- [SMS02a] A. Singh and M. Marek-Sadowska. Efficient Circuit Clustering for Area and Power Reduction in FPGAs. *ACM Trans. on Design Automation of Electronic Systems*, 7(4):643–663, October 2002.
- [SMS02b] A. Singh and M. Marek-Sadowska. FPGA Interconnect Planning. In *Proc. of SLIP'02*, pages 23–30, April 2002.
- [SP00] J. Satyanarayana and K. K. Parhi. Theoretical Analysis of Word-Level Switching Activity in the Presence of Glitching and Correlation. *IEEE Trans. On VLSI Systems*, 8(2):148–159, April 2000.
- [SWD09] A. M. Smith, S. J. E. Wilton, and J. Das. Wirelength Modeling for Homogeneous and Heterogeneous FPGA Architectural Development. In *Proc. of Intern. Sympos. on FPGAs*, pages 181–190, February 2009.
- [TB05] E. Todorovich and E. Boemo. Statistical Power Estimation for FPGAs. In *LNCS*, vol. 2438, pages 340–349, June 2005.
- [TGS⁺02] E. Todorovich, M. Gilabert, G. Sutter, S. Lopez-Buedo, and E. Boemo. A Tool for Activity Estimation in FPGAs. In *LNCS*, vol. 2438, pages 340–349, June 2002.
- [TL03] T. Tuan and B. Lai. Leakage Power Analysis of a 90-nm FPGA. In *Proc. on ICC*, pages 57–60, 2003.
- [vMSvC96] H. van Marck, D. Stroobandt, and J. van Campenhout. An Accurate Interconnection Length Estimation for Computer Logic. In *Proc. on Great Lake Symp. on VLSI*, pages 50–55, 1996.

- [VO93] D. Villeger and V. G. Oklobdzija. Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partial Products. In *Conf. Record of the 27th ACSSC*, pages 781–784, November 1993.
- [WAL04] S. J. E. Wilton, S. Ang, and W. Luk. The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays. In *Proc. of the Int. Conf. on Field-Programmable Logic and Applications, LNCS vol. 3203*, pages 719–728, August 2004.
- [WBG⁺06] R. Wain, I. Bush, M. Guest, M. Deegan, I. Kozin, and C. Kitchen. An overview of FPGAs and FPGA programming: Initial experiences at Daresbury. Computational Science and Engineering Dept., CCLRC Daresbury Laboratory, Daresbury, Warrington, Cheshire, November 2006.
- [WFDA06] L. Wang, M. French, A. Davoodi, and D. Agarwal. FPGA Dynamic Power Minimization through Placement and Routing Constraints. *EURASIP Journal on Embedded Systems*, 2006.
- [Xil] Xilinx. www.xilinx.com.
- [Xil03] Xilinx. Using Embedded Multipliers in Spartan-3 FPGA. Xilinx Application Note XAPP467, v.1.1, May 2003.
- [Xil05] Xilinx. Xilinx University Program Virtex-II Pro Development System. Hardware Reference Manual, v1.0, March 2005.
- [ZA95] M. Zheng and A. Albicki. Low Power and High Speed Multiplication Design Through Mixed Number Representations. In *Proc. on ICCD'95*, pages 566–570, October 1995.
- [ZJ05] L. Zhong and N. K. Jha. Interconnect-aware Low-power High-level Synthesis. *IEEE Trans. on CAD*, 24(3):336–351, March 2005.

