

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE MADRID

TESIS DOCTORAL

COSÍNTESIS DE SISTEMAS HETEROGÉNEOS COMPLEJOS

Autor:

José Manuel Moya Fernández
Ingeniero de Telecomunicación

Director:

Juan Carlos López López
Dr. Ingeniero de Telecomunicación
Universidad de Castilla-La Mancha

Tutor:

María Luisa López Vallejo
Dr. Ingeniero de Telecomunicación
Universidad Politécnica de Madrid

2003

TESIS DOCTORAL: Cosíntesis de Sistemas Heterogéneos Complejos

AUTOR: José Manuel Moya Fernández

DIRECTOR: Juan Carlos López López

TUTOR: María Luisa López Vallejo

El tribunal nombrado para juzgar la Tesis arriba indicada, compuesto por los siguientes doctores:

PRESIDENTE: Dr. Carlos A. López Barrio
VOCALES: Dr. Javier Uceda Anselmi
Dr. Antonio Núñez Ordóñez
Dr. Román Hermida Correa
SECRETARIO: Dr. Teresa Riesgo Alcalde

acuerda otorgarle la calificación de: Sobresaliente cum laude (5 votos)

Madrid, 11 de Abril de 2003

El Secretario del Tribunal

Resumen

Los rápidos cambios tecnológicos y la fuerte demanda de aparatos electrónicos de consumo, cada vez más complejos, están haciendo que el coste de diseño sea dominante para un amplio rango de sistemas empotrados. Esta complejidad inherente de los sistemas se ve agravada por la heterogeneidad de los recursos y de las herramientas de síntesis de bajo nivel. Por tanto, cada vez es más patente la necesidad de nuevas metodologías y herramientas de diseño que permitan aumentar drásticamente la productividad de los diseñadores.

El enfoque más común consiste en especificar el sistema completo, con un nivel de abstracción elevado, y después realizar un particionado y completar el diseño de hardware y software de forma independiente. Esta aproximación no escala bien con el aumento de complejidad, y además añade numerosos problemas de difícil solución.

Esta tesis propone un enfoque radicalmente diferente para el diseño de sistemas heterogéneos de elevada complejidad: homogeneizar los recursos y ofrecer abstracciones de muy bajo nivel, que eviten la separación del diseño de hardware y software hasta el final del proceso de diseño. Nuestro enfoque puede resumirse con la frase

Construyamos sistemas complejos homogéneamente sobre un conjunto heterogéneo de recursos, en lugar de construir sistemas complejos heterogéneamente sobre varios conjuntos homogéneos de recursos.

La metodología propuesta extiende los conceptos utilizados en el desarrollo de software

empotrado, utilizando un compilador hardware-software como herramienta principal, y un pequeño substrato de sistema operativo hardware-software, que ofrece los servicios básicos de comunicación y sincronización entre los diferentes componentes del sistema. A diferencia de otras aproximaciones, la interfaz de todos los componentes del sistema es idéntica.

Se ha implementado un prototipo de entorno de desarrollo para soportar la metodología propuesta, basado en las herramientas de GNU de desarrollo cruzado de software. Al mismo tiempo se han desarrollado una serie de ejemplos que demuestran la validez del enfoque.

Palabras clave: Codiseño hardware-software, síntesis de sistemas heterogéneos, diseño de sistemas complejos, sistemas operativos distribuidos.

Abstract

The fast technology developments, and the growing demand of consumer appliances, more and more complex every time, are making the design cost to be of highest concern for a spread range of embedded systems. This inherent complexity is getting worse because of the heterogeneity of the resources and the low-level synthesis tools. Therefore, it is becoming clear that we need new methodologies and design tools, capable of dramatically increasing the productivity of the designers.

The most common approach consists in specifying the whole system at a high abstraction level, then making a partition, and then completing the design of hardware and software separately. This approach does not scale well as complexity grows, and it also adds many other problems, with no easy solution.

This PhD thesis proposes a new, completely different approach to the design of highly complex heterogeneous systems: making the resources homogeneous and offering low-level abstractions, to avoid the separation of hardware and software design, until the last steps in the design flow. Our approach can be summarized as follows

Let's build complex systems homogeneously on top of an heterogeneous set of resources, instead of building complex systems heterogeneously on top of several sets of homogeneous resources.

The proposed methodology extends the concepts used in the development of embed-

ded software, using a hardware-software compiler as the main tool, and a small hardware-software operating system substrate, to provide the basic communication and synchronization services between the different system components. Unlike previous approaches, all the components (hardware and software) share a common interface.

We have built a prototype of a codesign environment supporting the proposed methodology, based on the GNU tools for cross-development of embedded software. At the same time, we have developed some examples that prove the suitability of our approach.

Keywords: Hardware-software codesign, heterogeneous system synthesis, complex system design, distributed operating systems.

Índice general

Índice general	11
Índice de figuras	17
1. Introducción	21
1.1. El diseño de sistemas heterogéneos	22
1.2. El diseño de sistemas complejos	24
1.3. Justificación de la investigación	25
1.4. Objetivos de la tesis	26
1.5. Metodología de trabajo	28
1.5.1. Modelo de sistema	28
1.5.2. Modelo del proceso de diseño	28
1.5.3. Compilador Hardware-Software	29
1.5.4. Sistema Operativo Hardware-Software	30

1.5.5. Modelos de Computación y Métodos Formales	31
1.6. Límites y suposiciones principales	32
1.7. Esquema del documento	32
2. Antecedentes	35
2.1. Introducción	36
2.2. Especificación de sistemas	38
2.2.1. Lenguajes de especificación	39
2.2.2. Gestión de memoria dinámica y punteros	45
2.2.3. Modelos de computación	46
2.3. Arquitectura destino	47
2.4. Síntesis de Interfaces	49
2.5. Sistemas operativos para sistemas empotrados	50
2.6. Aspectos metodológicos globales	52
2.7. Deficiencias fundamentales	54
2.7.1. Problemas de escalabilidad	55
2.7.2. Dificultades para reutilizar	55
2.7.3. Dependencias en el grupo de trabajo	56
2.8. Conclusiones	56
3. Un Enfoque Unificado	59
3.1. Introducción	60
3.2. Modelo del sistema	60
3.3. Abstracciones hardware-software	62
3.3.1. Recursos físicos	63
3.3.2. Procesadores virtuales	64
3.3.3. Interfaz entre recursos (ABI virtual)	66

<i>ÍNDICE GENERAL</i>	13
3.3.4. Sistema operativo hardware-software	76
3.3.5. Aplicaciones heterogéneas y reutilización	77
3.3.6. Aspectos dinámicos	78
3.4. Metodología de desarrollo	83
3.4.1. Especificación de sistemas	84
3.4.2. Análisis de resultados	89
3.4.3. Refinamiento de la especificación	91
3.5. Beneficios obtenidos	92
3.6. Conclusiones	94
4. El Entorno de Codiseño FLECOS	97
4.1. Arquitectura extensible	99
4.1.1. Recursos físicos	99
4.1.2. Componentes hardware-software	101
4.2. Estructura del entorno de desarrollo	106
4.2.1. Compilador hardware-software	108
4.2.2. Ensamblador hardware-software	114
4.2.3. Sistema operativo hardware-software	116
4.2.4. Bibliotecas hardware-software	119
4.3. Implementación de la metodología de desarrollo	119
4.3.1. Especificación de sistemas	119
4.3.2. Modelos de computación	122
4.3.3. Análisis de resultados	123
4.3.4. Refinamiento de la especificación	125
4.4. Conclusión	126

5. Resultados y Validación	129
5.1. Estrategia de validación y pruebas	130
5.1.1. Criterios de selección	131
5.1.2. Aplicaciones de ejemplo	131
5.1.3. Condiciones de prueba	132
5.2. Decodificador de MPEG 1 layer 3	132
5.2.1. Especificación	133
5.2.2. Refinamiento de la arquitectura	137
5.2.3. Conclusiones sobre el ejemplo	139
5.3. Filtros de imagen	140
5.3.1. Especificación	141
5.3.2. Refinamiento de la especificación	144
5.3.3. Conclusiones sobre el ejemplo	149
5.4. Sistemas reactivos de control	150
5.4.1. Especificación	151
5.4.2. Simulación	154
5.4.3. Verificación	155
5.4.4. Síntesis	155
5.4.5. Conclusiones sobre el ejemplo	156
5.5. Planificación y sincronización de tareas	156
5.5.1. Tareas hardware-software	157
5.5.2. Planificación de tareas hardware-software	158
5.5.3. Implementación	159
5.5.4. Un ejemplo práctico	161
5.5.5. Conclusiones sobre la planificación de tareas	166
5.6. Limitaciones del prototipo	167

<i>ÍNDICE GENERAL</i>	15
5.7. Conclusiones	168
6. Conclusiones e Implicaciones	171
6.1. Resumen de contribuciones	173
6.1.1. Unificación del diseño de sistemas hardware y software	173
6.1.2. Ortogonalización de aspectos por dominios de conocimiento	174
6.1.3. Escalabilidad en el diseño de sistemas complejos	174
6.2. Conclusiones sobre el problema	175
6.3. Limitaciones	176
6.3.1. Modelo de la arquitectura destino	176
6.3.2. Acoplamiento entre el optimizador y la arquitectura	177
6.3.3. Observabilidad del proceso de síntesis	178
6.4. Líneas futuras de investigación	178
6.4.1. Compilador hardware-software avanzado	178
6.4.2. Exploración automática del espacio de diseño	179
6.4.3. Sistema operativo adaptable	179
Bibliografía	181