Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Diseño de una herramienta de generación automática de código para las plataformas MPSoC KeyStone

RWTH Aachen University

Chair for Software for Systems on Silicon, SSS.

Institute for Communication Technologies and Embedded Systems,ICE

Anteproyecto de Graduación

Ronny Jiménez Araya

200811178

Declaro que el presente Anteproyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo realizado y por el contenido del correspondiente informe final.

Ronny Jiménez Araya

Cartago, 19 de diciembre de 2013

Cédula: 1-1373-0768

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Entorno del proyecto

Los sistemas embebidos o empotrados se utilizan en muchos de los dispositivos y equipos con los cuales el ser humano interacciona o tiene relación alguna en la actualidad. Desde dispositivos móviles como un teléfono celular, pasando por televisores de alta definición, equipos de seguridad en medios de transporte y hasta en la industria aeroespacial, la cual requiere de los elementos más robustos y de soporte crítico, son ejemplos claros de la utilización de los sistemas embebidos.

En general, un sistema empotrado se considera como aquel conjunto de elementos de carácter electrónico y/o computacional-informático que se encuentran relacionados entre sí y que tienen como función una aplicación específica dada.[3] [9]

Hasta hace algunos años la arquitectura de estos dispositivos generalmente fue la utilización de un único procesador, con la tendencia del escalado en frecuencia [5] para aumentar el rendimiento de los mismos. Sin embargo, las tendencias han variado y han surgido los Sistemas en Chip de Múltiples Procesadores (*MPSoC, Multi-Processor Systems on Chip*), los cuales integran una gran variedad de dispositivos entre los cuales se encuentran elementos con múltiples núcleos de procesamiento.

Actualmente estos Sistemas en Chip (*SoC, Systems-on-Chip*) incorporan múltiples núcleos de diferente naturaleza entre los cuales se encuentran los Procesadores de Propósito General (*GPP, General Purpose Processors*) como la gama de procesadores ARM, Procesadores Digitales de Señales (*DSP, Digital Signal Processors*) que se encuentran de diversos tipos como los especializados para imágenes (*DIP, Digital Image Processor*), así también como los dedicados al tratamiento de gráficos llamados GPU (*GPU, Graphics Processing Units*) [10]. Todos los anteriores encapsulados en un mismo circuito integrado o bien incorporados en el SoC que los alberga.

Éste cambio en el paradigma de la computación de pasar de sistemas de un solo núcleo de procesamiento a los MPSoC trae consigo retos que antes no se conocían, tanto para los proveedores como para los desarrolladores de las plataformas. Los modelos de programación cambian de un estándar secuencial a uno paralelo que justamente aproveche el paralelismo inherente a los múltiples procesadores, lo que conlleva nuevos métodos de programación, compilación, y ejecución que deben de cumplir y satisfacer los requerimientos característicos de los sistemas embebidos como operación en tiempo real [4], eficiencia energética y procesamiento limitado.

La Universidad de Aquisgrán (*RWTH Aachen University*) es un centro académico y de investigación en diversas ramas de la ingeniería. En la actualidad es reconocida mundialmente por los trabajos realizados en las áreas de ingeniería mecánica, electrónica y ciencias de la computación [1]. Dentro de su sistema académico, la universidad cuenta con diversos centros de investigación (institutos) que realizan aportes de vanguardia con nuevas tecnologías y avances, concretando estos mediante publicaciones en revistas internacionales de alto renombre. Estos centros de investigación son dirigidos mediante Juntas (*Chairs*) que tienen a cargo diversos proyectos tanto en áreas especificas como en áreas multidisciplinarias.

Uno de los institutos con más publicaciones y aportes a la ciencia es el Instituto de Tecnologías de Comunicaciones y Sistemas Embebidos (*ICE, Institute for Communications Technologies and Embedded Systems*) [2], el cual integra tres de las juntas de mayor importancia en el área de la electrónica y las ciencias de la computación (*SSS, Software for Systems on Silicon; ISS, Integrated Signal Processing; Research Group MPSoC Architectures UMIC Research Cluster*); estos "*Chairs*" colaboran entre si en la generación de conocimiento, y cuentan con tres áreas vitales de las comunicaciones eléctricas y los sistemas embebidos.

- Algoritmos y esquemas de transmisión y recepción en los sistemas de comunicaciones inalámbricas.
- Hardware y arquitecturas de los chips para sistemas embebidos, en particular para los MPSoC.
- Diseño de herramientas a nivel de sistema (*ESL, Electronic System-Level*) para sistemas y circuitos integrados complejos.

En la actualidad el Instituto cuenta con una aplicación llamada MAPS (*MAPS, MPSoC Application Programming Studio*) [6], la cual consiste en un sistema de herramientas

que asisten a los programadores en la paralelización de código secuencial C. La interfaz del usuario esta basada en las herramientas de C/C++ del ambiente de desarrollo Eclipse.

A pesar de la gran cantidad de tiempo que se ha invertido en el desarrollo de la aplicación MAPS, ésta no se encuentra concluida y solo cuenta con soporte para determinadas plataformas de desarrollo, entre ellas un soporte limitado para la KeyStone I [8], la cual es una plataforma de múltiples unidades de procesamiento (*PE, Processing Elements*) homogéneas, en este caso DSP's. Parte del soporte que del todo no se encuentra es para las arquitecturas de KeyStone II [7], cuya estructura consta de unidades de procesamiento pero heterogéneas, es decir, procesadores de propósito general ARM más DSP's.

Capítulo 2

Definición del problema

2.1 Generalidades

Como generalidad del entorno de los sistemas embebidos se encuentra la dificultad para los desarrolladores de trasladares de un paradigma de programación secuencial a uno que implique la utilización de múltiples unidades de procesamiento en forma paralela, de las nuevas generaciones de plataformas multinúcleo, reutilizando para ello código ya existente (*Legacy Code*).

En términos generales existen herramientas para la plataforma KeyStone I, sin embargo, éstas no se encuentran optimizadas para un mejor consumo de los recursos y de la energía, por lo que la optimización de las mismas es un aspecto a considerar.

De forma más específica, para la plataforma KeyStone II no existe ninguna herramienta que logre programar sus múltiples núcleos heterogéneos de manera automática, generando código paralelo a partir de código secuencial en C.

2.2 Síntesis del problema

Inexistencia y falta de optimización de una herramienta de generación automática de código paralelo a partir de código secuencial C, para las plataformas MPSoC KeyStone de Texas Instruments.

Capítulo 3

Enfoque de la solución

El enfoque que se pretende dar a la solución se basa en la utilización de un lenguaje de programación de alto nivel como lo es C, para el diseño e implementación de un algoritmo que permita la programación de diferentes unidades de procesamiento en plataformas MPSoC. El desarrollo va a ser realizado en un ambiente Linux que facilite el uso de los recursos de la plataforma, así como la depuración y evaluación de los diferentes componentes de hardware que se emplearán.

Además del lenguaje a utilizar, se pretende manejar otros elementos a nivel de software como lo son simuladores, depuradores y las bibliotecas para compilación cruzada, los cuales estarán integrados dentro de un ambiente de desarrollo (*IDE, Integrated Development Enviroment*) de Texas Instruments que podrá ser descargado y utilizado sin algún costo. Sin embargo, deberá desarrollarse interfaces específicas (*API's, Application Programming Interface*) que puedan integrarse al IDE de Texas Instruments para la correcta validación de lo realizado.

Agregado a lo anterior, se manipularán plataformas de hardware destinadas a ser los elementos objetivo de la programación, específicamente las KeyStone I y II. La primera tiene como unidades de procesamiento 8 DSP's TMS320C66x, mientras que la segunda implementa en su arquitectura la misma cantidad y tipo de DSP's más 4 procesadores de propósito general ARM Cortex-A15.

También será necesario utilizar diversos instrumentos de medición como analizadores lógicos y osciloscopios que permitan la depuración mediante señales eléctricas de los comportamientos no deseados en el sistema en general.

Con lo anterior establecido se podrá realizar la programación del algoritmo y su posterior depuración, con el fin de determinar la eficiencia y calidad que va a tener el mismo en cuanto a consumo de recursos de los CPU/DSP como registros, memoria, unidades aritméticas, etc.

Capítulo 4

Meta

La meta a alcanzar con el presente Proyecto de Graduación es brindarle al *Institute for Communications Technologies and Embedded Systems* de la universidad alemana *RWTH Aachen University*, una herramienta que puedan incorporar en su aplicación MAPS, con el fin de proporcionar un entorno mejorado para la programación de los actuales y futuros dispositivos multinúcleo, en los cuales su arquitectura esté basada en unidades de procesamiento heterogéneo y con la mentalidad de una posible comercialización de la aplicación total a un mediano o largo plazo.

Capítulo 5

Objetivos

5.1 Objetivo General

Diseñar una herramienta de generación automática de código paralelo a partir de código secuencial, para ser utilizada como plataforma de programación en dispositivos de múltiples unidades de procesamiento.
Indicador: Creación de la herramienta

5.2 Objetivos Específicos

1. Optimizar la herramienta existente de generación automática de código C paralelo para la plataforma KeyStone I.
   Indicador: Las modificaciones a la herramienta de generación de la KeyStone I permiten obtener una mejoría en las características generales de rendimiento.

2. Diseñar e implementar la herramienta de generación automática de código C paralelo para los MPSoC KeyStone II.
   Indicador: La herramienta desarrollada permite programar las diferentes unidades de procesamiento del MPSoC KeyStone II.

3. Validar la herramienta creada mediante la programación, compilación y ejecución de programas de prueba (*Benchmarks*).
   Indicador: Gráficos y resultados obtenidos a partir de la aplicación de los Benchmarks.

Capítulo 6

Procedimientos para la ejecución del proyecto

6.1 Metodología

El proyecto primeramente consistirá en el estudio de las arquitecturas KeyStone I y II, con el fin de establecer características propias de cada plataforma que las diferencien tanto en su funcionamiento como en su posible programación, incluyendo aspectos relevantes de los lenguajes que se utilizan para programar DSP's, así como los desafíos que involucra la programación de los mismos.

Una vez concluido esto, se estudiará las herramientas que existen actualmente para programación en paralelo, con el fin de definir pruebas que puedan ser utilizadas para detectar posibles debilidades a optimizar dentro de las aplicaciones, en específico la existente para KeyStone I.

Se procederá en la optimización de los puntos críticos determinados anteriormente y en la validación de la herramienta mediante pruebas de rendimiento, con lo cual se concluiría el primer objetivo específico que se plantea en este documento.

Se debe estudiar lo relevante a los modelos de programación en paralelo como los algoritmos de conversión secuencial a paralelo. En específico debe dársele importante atención las Redes de Procesamiento de Kahn (*KPNs, Kahn Process Networks*).

Se empezará la implementación de los algoritmos y entornos que se requieren para la generación automática de código paralelo, así como su programación en un entorno basado

en Linux y con la herramienta de Texas Instruments Code Composer Studio (*CCS*).

Se dispondrá a diseñar pruebas específicas para la herramienta, con el fin de depurar el sistema en las debilidades más notables y posteriormente comprobar la optimización de la misma, concluyendo así el segundo objetivo específico.

Se pretende exponer la herramienta recién creada a un conjunto de pruebas ya existentes y establecidas específicamente para la arquitectura KeyStone II. Esto con la intención de realizar un perfilado de la aplicación y determinar gráficas de rendimiento, realizando comparaciones entre las diferentes plataformas KeyStone y las herramientas de generación de código.

Finalmente como parte integral del proyecto, se brindarán conclusiones acerca del mismo, y los futuros proyectos que puedan involucrarse, ya sea para optimizar la herramienta o bien en su utilización para nuevos paradigmas.

6.2 Actividades a realizar durante el proyecto

Para la finalización del primer objetivo específico se plantean las siguientes actividades en orden cronológico:

1. Estudiar las arquitecturas presentes en las tarjetas de desarrollo KeyStone I y II.
2. Establecer diferencias importantes entre ambas tarjetas.
3. Familiarizarse con el IDE de TI, Code Composer Studio, realizando para ello ejemplos de pruebas y simulaciones para tarjetas específicas.
4. Estudiar acerca de las herramientas de Texas Instruments como el Multicore Navigator, y las bibliotecas *IPC, Inter-Processor Communication* y *openMP* .
5. Investigar modelos de programación con estructuras en paralelo.
6. Investigar características propias de los sistemas operativos de TI para DSP's, específicamente SYS/BIOS y DSP/BIOS.
7. Estudiar a profundidad la herramienta implementada para KeyStone I para la programación de dispositivos de múltiples unidades de procesamiento homogéneas del tipo DSP.

8. Realizar pruebas mediante *Benchmarks* que logren poner al descubierto las debilidades de la herramienta.
9. Optimizar la herramienta en los puntos más críticos que se establecieron en la actividad pasada.
10. Comprobar la mejora o no en el rendimiento de la herramienta, utilizando como punto de partida la herramienta sin optimizar.

Para cumplir con el segundo objetivo específico se proponen estas actividades:

11. Estudiar acerca de concurrencia y paralelismo en núcleos heterogéneos
12. Investigar sobre el funcionamiento de Colas (*FIFO, First In First Out*).
13. Estudiar a profundidad las *KPNs* en donde se utilizan las *FIFO*.
14. Comprender el algoritmo que se utiliza en las *KPNs* con el fin de realizar un diagrama de flujo del método en si.
15. Programar el algoritmo de las *KPNs* utilizando las *FIFO*.
16. Integrar el algoritmo a una herramienta que pueda ser ejecutada mediante un IDE o la línea de comandos de Linux.
17. Realizar pruebas que comprueben el funcionamiento deseado de la herramienta.
18. Determinar con el punto anterior fallos, *"bugs"* y otros comportamientos no deseados de la herramienta.
19. Depurar la herramienta atacando los puntos críticos de la misma.
20. Realizar nuevas pruebas que muestren que se ha corregido los principales fallos de la herramienta.

Para finalizar, el tercer y último objetivo llevará acabo las siguientes actividades:

21. Escoger algoritmos de diferente naturaleza (llámese procesamiento general, procesamiento de señales, procesamiento de video e imágenes, encriptado, transmisión y comunicaciones, etc.) para ser utilizados como los *benchmarks* de prueba de la herramienta implementada.
22. Determinar aspectos relevantes en la ejecución de los *benchmarks*, que puedan ser comparativos entre plataformas y herramientas.

23. Ejecución y perfilado de las pruebas escogidas.

24. Realización de las gráficas de rendimiento basándose en el perfilado resultante del punto anterior.

25. Comparar las plataformas y las herramientas mediante los datos de las gráficas y del perfilado.

26. Brindar conclusiones relevantes acerca de las pruebas realizadas

27. Establecer recomendaciones y futuras investigaciones sobre el proyecto.

6.3 Cronograma de actividades

WBS	Name	Start	Finish	Work	Duration
1	**Objetivo Especifico 1**	**Jan 24**	**Mar 3**	**27d**	**27d**
1.1	Estudiar las arquitecturas KS I y II	Jan 24	Jan 27	2d	2d
1.2	Diferencias entre KS I y II	Jan 28	Jan 29	2d	2d
1.3	Familiarizacion del CCS	Jan 30	Jan 31	2d	2d
1.4	MC Navigator IPC y openMP	Feb 3	Feb 14	10d	10d
1.5	Estudiar programacion paralelo	Feb 17	Feb 18	2d	2d
1.6	SYS/BIOS DSP/BIOS	Feb 19	Feb 25	4d	4d
1.7	Herramienta existente para KS I	Feb 21	Feb 21	1d	1d
1.8	Pruebas de benchmarks	Feb 25	Feb 26	1d	2d
1.9	Optimizacion	Feb 27	Feb 28	2d	2d
1.10	Comprobacion de la optimizacion	Mar 3	Mar 3	1d	1d
2	**Objetivo Especifico 2**	**Mar 4**	**May 15**	**53d**	**53d**
2.1	Concurrencia y paralelismo	Mar 4	Mar 6	3d	3d
2.2	Colas FIFO	Mar 7	Mar 13	5d	5d
2.3	KPNs	Mar 14	Mar 26	9d	9d
2.4	Algoritmo de las KPNs	Mar 27	Apr 10	11d	11d
2.5	Programar el algoritmo KPNs	Apr 11	Apr 23	9d	9d
2.6	Integracion del algoritmo	Apr 24	Apr 30	5d	5d
2.7	Pruebas de funcionamiento	May 1	May 6	4d	4d
2.8	Bugs y fallos	May 7	May 8	2d	2d
2.9	Depuracion	May 9	May 14	4d	4d
2.10	Nuevas pruebas	May 15	May 15	1d	1d
3	**Objetivo Especifico 3**	**May 16**	**Jun 5**	**15d**	**15d**
3.1	Escogencia de algoritmos	May 16	May 19	2d	2d
3.2	Aspectos relevantes de los benchmarks	May 20	May 20	1d	1d
3.3	Ejecucion y perfilado	May 21	May 27	5d	5d
3.4	Graficas de rendimiento	May 28	May 30	3d	3d
3.5	Comparacion de plataformas	Jun 2	Jun 3	2d	2d
3.6	Conclusiones	Jun 4	Jun 4	1d	1d
3.7	Recomendaciones	Jun 5	Jun 5	1d	1d

Figura 6.1: Actividades de Gantt del Proyecto de Graduación

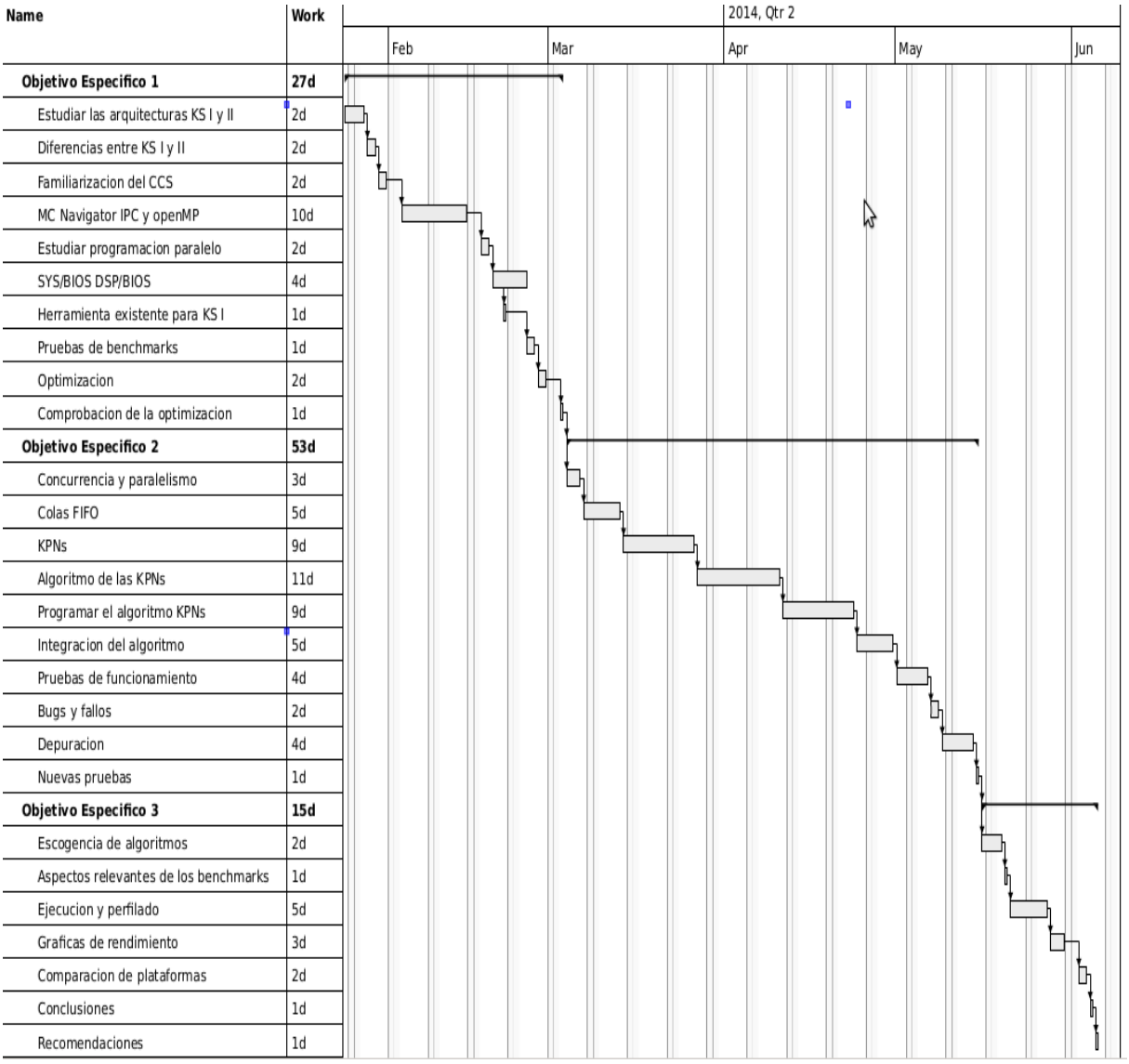

Figura 6.2: Diagrama de Gantt del Proyecto de Graduación

Capítulo 7

Uso de recursos

Los recursos a utilizar durante el proyecto incluyen desde tarjetas de evaluación así como herramientas informáticas, depuradores, simuladores, osciloscopios y diferentes dispositivos periféricos. La lista completa se muestra a continuación:

- 1 Tarjeta de evaluación de Texas Instruments que contenga la arquitectura KeyStone I (C66xx), como la EVM TMS3206657 Lite Evaluation Module.
- 1 Depurador de hardware capaz de realizar emulaciones del sistema y de arquitecturas de DSP C6678 de TI.
- 1 Depurador de software para las arquitecturas de DSP C6678 de TI.
- 1 Tarjeta de evaluación de Texas Instruments que contenga la arquitectura KeyStone II (66AK2H12xx) como la EVM 66AK2H12AAAW2.
- 1 Depurador de hardware capaz de realizar emulaciones del sistema y de arquitecturas ARM Cortex-A15 y de DSP C6678, ambas de TI.
- 1 Depurador de software para las arquitecturas ARM Cortex-A15 y DSP C6678, de Texas Instruments.
- 1 Computadora portátil con un procesador suficientemente robusto que soporte simulaciones y depuraciones de sistemas complejos.
- 1 Instalación nativa de Linux Ubuntu12.04 LTS en la computadora portátil.
- 1 Instalación del entorno de desarrollo integrado (IDE) de Texas Instruments, conocido como Code Composer Studio.
- 1 Instalación de un analizador de sistemas Multicore.

Capítulo 8

Presupuesto

Los gastos incurridos en lo que es equipo, materiales, software, gastos de operación y otros, sin tomar en cuenta lo que son gastos de viaje, transporte, hospedaje y alimentación, son cubiertos en su totalidad por *RWTH Aachen University*, dichos datos se observan en la Tabla 8.2

Con respecto a los costos estimados de los gastos de viaje, transporte y alimentación durante la estadía en la ciudad alemana de Aachen, los cuales van a ser cubiertos en parte por la beca Movilidad Internacional Estudiantil de la rectoría del *Instituto Tecnológico de Costa Rica* y por mi persona, se muestran en la Tabla 8.1

Tabla 8.1: Presupuesto del costo de vida durante 6 meses en Alemania

Gasto / Mes	1	2	3	4	5	6
Transporte($)	50	50	50	50	50	50
Alimentación($)	300	300	300	300	300	300
Hospedaje($)	700	700	700	700	700	700
Telefonía($)	100	100	100	100	100	100
Tiquetes aéreos($)	1500	0	0	0	0	0
Total Semestral($)	2650	1150	1150	1150	1150	1150
Total de los 6 meses completos($)	8400					

Tabla 8.2: Presupuesto del equipo, materiales, software y recurso humano en *RWTH Aachen University*

Gasto / Mes	1	2	3	4	5	6
Equipo						
1 EVM TMS3206678($)	400					
1 Depurador de DSP($)	300					
1 Depurador Software($)	350					
1 EVM 66AK2H12AAAW2($)	100					
1 Depurador ARM Cortex-A15($)	500					
1 Analizador Lógico Agilent 1854A($)	25 000					
1 Osciloscopio digital HF Agilent($)	100 000					
1 Computadora($)	1500					
Recurso Humano						
Pago horas ingeniero($)	500	500	500	500	500	500
Total por mes ($)	129 550	500	500	500	500	500
Total de los 6 meses ($)	132 050					

Bibliografía

[1] Excellence initiative, Noviembre 2013. URL http://www.rwth-aachen.de/cms/root/Die_RWTH/~emq/Exzellenzinitiative/lidx/1/.

[2] Institute for communications technologies and embedded systems, Noviembre 2013. URL http://www.ice.rwth-aachen.de/institute/about-us/.

[3] Doug Abbot. *Linux for Embedded and Real-time Applications*. Newnes, 2003.

[4] Doug Abbot. *Linux for Embedded and Real-time Applications*, chapter 1, pages 2,3. Newnes, 2003.

[5] Jack Ganssle. *The Art of Designing Embedded Systems*. Newnes, 2008.

[6] W. Sheng H. Scharwächter R. Leupers G. Ascheid H. Meyr T. Isshiki H. Kunieda J. Cheng, J. Castrillon. Maps: An integrated framework for mpsoc application parallelization. In *45th Design Automation Conference (DAC '08)*, pages 754–759, June 2008.

[7] Texas Instruments. *KeyStone II User's Guide*, December 2013.

[8] Texas Instruments. *TMS320C6678 Multicore Fixed and Floating-Point Digital Signal Processor*, Abril 2013.

[9] Marilyn Wolf. *Computers as components: Principles of embedded computing design*. Morgan Kaufmann, 2007.

[10] Marilyn Wolf. *High-Performance Embedded Computing*. Morgan Kaufmann, 2007.