Reporte del Proyecto Final

Mauricio Caamaño, Marco Espinoza, Tomás González

Agosto 2017

1. Parte 1

1.1. Diseño de prototipado

Para el planteamiento del prototipado del sistema requerido se definió primero cuáles componentes son los que se pueden implementar en la FPGA, así como cuales se pueden prorotipar mediante componentes externos, ya sea en una misma tarjeta de desarrollo o en otro componente separado. La premisa de utilizar una tarjeta de desarrollo o evaluación se eligió por la facilidad y conveniencia que ofrecen las tarjetas de evaluación para protitpar sistemas, pues muchas contienen componentes útiles como memorias RAM, sensores, e interfaces de comunicación, así como conectores estandarizados para adaptar otros dispositivos, como convertidores, o módulos de comunicación de red, entre otros.

La tabla 1 muestra los componentes del sistema, definidos en los requerimientos de la parte 1 del proyecto. Se clasificó cada componente según el tipo de prototipado que se utilizaría, siendo estos completo, parcial o innecesario/imposible:

- Completo: Se puede prototipar por completo el componente, y es posible de probar su funcionalidad completa, tal como se haría en el ASIC.
- Parcial: Sólo se puede prototipar parcialmente el bloque. Algunas funciones no son posibles de prototipar.
- Innecesario o imposible: No es posible o no es necesario prototipar el bloque, ya sea por limitaciones de hardware, o porque no es relevante.

La misma tabla además especifica si el bloque se implementaría utilizando como fuente un bloque IP de la librería, código HDL, inferencia HDL, o mediante un dispositivo o tarjeta externa. En caso de ser externo, debe ser posible simular el comportamiento del bloque en el testbench. Además, no se requirió utilizar IPs externos a la librería de Altera incluida en la licensia gratuita.

Todos estos componentes se interconectan principalmente mediante la interfaz de comuncación Avalon de Altera, tal como se observa en la figura 1. Se tiene así el el diagrama de la arquitectura planteada para el prototipo del controlador DSP. Se observa en una isla de primer nivel la parte del sistema contenida dentro de la tarjeta de desarrollo, y la lógica a implementar dentro de la FPGA. De este modo, se definió como único componente totalmente externo a la tarjeta a una tarjeta de convertidores con ADC y DAC. La elección de esta tarjeta de convertidores se realizó posteriormente, en la sección 1.2.2.

Los detalles del diseño de los bloques de la tabla 1 así como la arquitectura del diseño de la figura 1 se describieron en el documento de "Arquitectura y diseño" adjunto en el apéndice A.

1.2. Implementación del prototipo

En esta sección se describe el proceso para elegir dos posibles tarjetas de desarrollo aptas para implementar el prototipo del sistema, cumpliendo con los requisitos de recursos y componentes externos requeridos.

Cuadro 1: My caption

Bloque	**Prototipado (C / P / I)**	**Fuente (HDL, IP, Ext)**	**Justificación**
uControlador RISC	Completo	Soft core IP	Se prototipa utilizando el Soft Core de Nios II, el cual también es arquitectura RISC.
Controlador de reloj	Completo	IP	Se prototipa mediante el uso del core de ALTERA PLL con opción de reconfiguarción para permitir la variación de los factores de multiplicación y división. Para esto, se agrega el uso del core de Altera PLL Reconfig.
Contador descremental	Completo	Inferencia HDL	Los dos contadores se implementan meidante código HDL. La lógica de interrupción e inicialización se incluye en un bloque personalizado para ser instanciado mediante Qsys.
Temporizador Watchdog	Completo	IP	Se prototipa mediante el IP de Timer, configurado para operar como Watchdog, con el período requerido de 30 segundos a la máxima frecuencia.
Transformada de Fourier	Completo	IP	Se prototipa mediante el IP de FFT de la librería. Este IP utiliza una interfaz de streaming por Avalon, por lo que es necesario otro IP para convertir paquetes.
ADC y DAC	Completo	Externo	Debido a que pocas tarjetas de desarrollo contienen ambos componentes, se decide utilizar una tarjeta con convertidores externa, con interfaz I2C o SPI. Esto permite que sea posible simular su comportamiento en el testbench.
SPI e I2C	Completo	IP	Se prototipa mediante los IP de la librería.
Memoria en chip	Completo	Externo	Se prototipa utilizando memoria de la tarjeta de desarrollo, tipo RAM. Esto se decide debido a la cantidad de memoria requerida, pues es mucha para ser implementada en los componentes de memoria de la FPGA.
Controlador de energía	Parcial	Código HDL	Se prototipa parcialmente mediante código HDL. Se pueden implementar los modos de operación “activo” y “P1”, mas no es posible probar el “P2”, ya que requiere de un hardware específico que permita levantar el dispositivo cuando el procesador y todos los periféricos se encuentran apagados.
Bancos de I/O	Innecesario		No es necesario prototipar los bancos de I/O. Sin embargo sí se toma en cuenta la cantidad de puertos necesarios a la hora de elegir la tarjeta de desarrollo.

1.2.1. Estimación de recursos

Para planear la implementación del prototipo, se diseñó un sistema en Qsys con todos los componentes internos a la FPGA que requirieran ser instanciados desde un IP de la librería. Los únicos componentes internos a la FPGA, excentos de ser instanciados mediante un IP, fueron los contadores decrementales y la lógica para el controlador de energía. Sin embargo, se utiliza como criterio de estimación, el reporte de recursos utilizados a partir de la instanciación de los bloques IP sólamente. La figura 2 muestra el reporte del sistema en Quartus que implementa todos los bloques IP del sistema, incluyendo el soft core de Nios II.

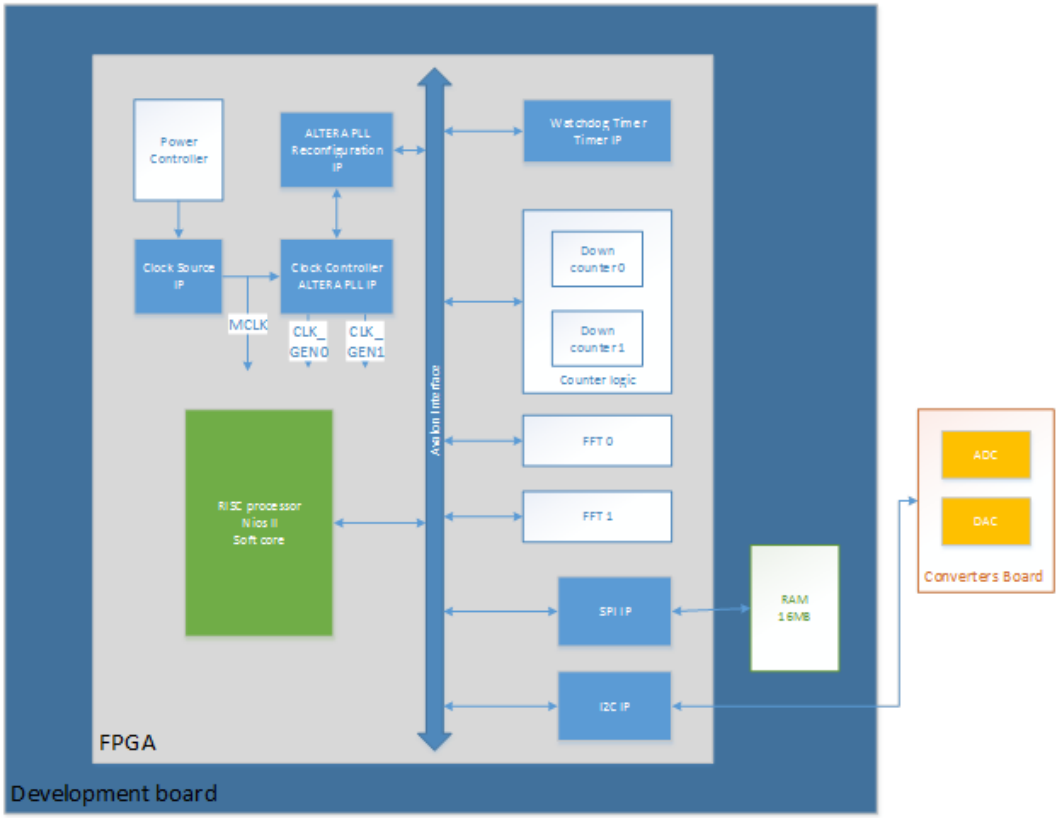

Figura 1: Arquitectura del sistema micro-controlador DSP

1.2.2. Selección de plataforma de prototipo

En base a los resultados mostrados en la figura 2, se buscaron tarjetas de desarrollo adecuadas para prototipar el sistema. Ya que la cantidad de recursos es relativamente baja, se buscaron opciones de bajo costo que tuvieran suficiente memoria para implmenetar los 16 MB requeridos, así como conectores para utilizar las interfaces de comunicación SPI e I2C. Las dos opciones elegidas fueron la DE0-Nano de Terasic (`http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=593&PartNo=2`), y la DECA de Arrow (`https://www.arrow.com/en/products/deca/arrow-development-tools`), mostradas en la figura 3. La table 2 muestra las características y componentes de cada una de las tarjetas de desarrollo.

Para prototipar los convertidores, se eligió la Tarjeta PCF8591 AD DA (`https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf`), con convertidores de 8-bit, cuya frecuencia de muestreo es definida por la interfaz I2C. Esta tarjeta, mostrada en la figura 4 se puede conectar a cualquiera de las dos tarjetas de desarrollo elegidas. Además, tiene un costo muy pequeño de tan sólo $6, lo cual la hace muy conveniente para mantener el sistema de bajo costo.

Analysis & Synthesis Resource Usage Summary		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	1747
2		
3	Combinational ALUT usage for logic	2774
1	-- 7 input functions	36
2	-- 6 input functions	501
3	-- 5 input functions	709
4	-- 4 input functions	601
5	-- <=3 input functions	927
4		
5	Dedicated logic registers	1773
6		
7	I/O pins	6
8	Total MLAB memory bits	0
9	Total block memory bits	18432
10		
11	Total DSP Blocks	0
12		
13	Total PLLs	1
1	-- Fractional PLLs	1
14		
15	Maximum fan-out node	clk_clk~input
16	Maximum fan-out	1683
17	Total fan-out	19496
18	Average fan-out	4.16

Figura 2: Consumo de recursos del sistema con bloques IP

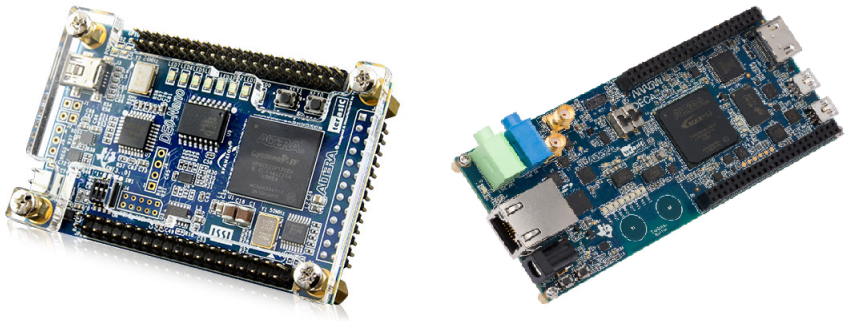

Figura 3: Tarjetas de desarrollo elegidas para la implementación del prototipo de la parte 1.

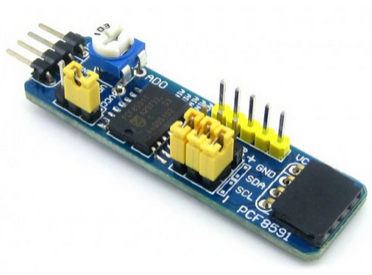

Figura 4: Tarjeta con convertidores de 8-bits PCF8591.

Cuadro 2: Descripción de tarjetas de desarrollo seleccionadas para implementación del prototipo.

	Terasic DE0-Nano	**Arrow DECA**
FPGA	Cyclone® IV EP4CE22F17C6N	Max 10 10M50DAF484C6G
- Elementos lógicos	22320	49760
- Memoria embebida	594 Kbits	1.638 Kb
- Multiplicadores	66 de 18 x 18	144 de 18 x 18
- PLLs	4	4
- FPGA I/O	153	360
Frecuencia op.	50 MHz	50 MHz
Memorias	2Kb I2C EEPROM 32MB SDRAM	64MB Flash 512MB SDRAM
ADC	1	2
JTAG	Si	Si
GPIO	72	92
USB	1	2
Ethernet	No	1
Display	No	LCD y HDMI
Audio	No	Si
Switches	6	4
LEDs	8	8

1.3. Plan de pruebas

El plan de pruebas a realizar es el siguiente:

1.3.1. Pruebas por IP

Consiste en realizar pruebas directas para cada IP de manera que se pueda comprobar la funcionalidad de cada uno de ellos por separado, además se utilizarán BFMs para simular el comportamiento de otros IP en caso de ser necesario. A continuación se explican las pruebas para cada uno de los IPs.

a. Procesador:
Debido a que el procesador utilizado no es el real, las pruebas para el procesador se realizaran para probar el mismo interactuando con el sistema completo.

b. Controlador de reloj: Para el controlador de reloj no se pueden realizar pruebas directas debido a que no se tienen registros para validar la frecuencia al que el sistema está ejecutando. Por lo que se deben realizar pruebas indirectas, de manera que al cambiar la frecuencia del sistema, se verifique que los resultados obtenidos no alteran el funcionamiento el sistema.

c. Contador decremental:
Para el contador decremental, se utilizaran también pruebas directas, ya que la lógica del mismo es simple, por lo que no se requiere hacer pruebas aleatorias exhaustivas, únicamente se chequea que el mismo se decremente adecuadamente y se genere la interrupción.

d. Temporizador Watchdog:
El temporizador watchdog también se valida con pruebas directas, ya que se verifica que la interrupción suceda en el tiempo esperado.

e. Bloque FFT
   Para este bloque se realizan pruebas directas y pruebas aleatorias, de manera que se verifique que el bloque esta realizando las transformaciones adecuadamente.

f. SPI y I2C: Se hacen pruebas directas para verificar el compotamiento adecuado de estos bloques, además se pueden hacer pruebas aleatorias para corroborar que la comunicación sucede de manera adecuada. Para ello se debe de realizar un testbench que tiene una interfaz para enviar los estímulos a dichos bloques y poder revisar la salida del mismo.

g. Memoria en chip Para el caso de la memoria, se hacen pruebas directas y aleatorias para poder verificar la comunicación correcta con la memoria. Para ello se debe de hacer un BFM que modele el comportamiento de la memoria.

h. Controlador de energía Se hacen pruebas directas para verificar que el IP puede cambiar entre los modos de operación esperados. Pero, no se puede verificar de manera digital los cambios de voltaje esperados, por lo que únicamente se valida el comportamiento del IP.

1.3.2. Pruebas para el sistema completo

Para este caso, se realizan pruebas que verifiquen la funcionalidad del circuito completo, ya utilizando el procesador, y los IPs pero sin verificar la memoria, ya que como la misma es externa no se tendrá a la hora de realizar las pruebas funcionales. Para este caso se harán las siguientes pruebas:

1. Directas:
   Pruebas directas a nivel de ensamblador para verificar el correcto funcionamiento del procesador y la interacción del mismo con los IPs. Se realizan operaciones directas y conocidas para realizar esta verificación.

2. Aleatoria:
   Se realizan pruebas aleatorias para poder probar la interaccion del procesador con los IPs enviando datos de manera exhaustiva, de manera que se generan operaciones y transacciones random.

2. Parte 2

2.1. Diseño

El primer paso del diseño fue hacer un módulo que haga la multiplicación de matrices y el cálculo del determinante. Se diseño un módulo que de manera puramente combinacional hace los cálculos. En la primera etapa del módulo se calculan los 4 elementos de la matriz resultante de la multiplicación y en la segunda etapa se hace el cálculo del determinante.

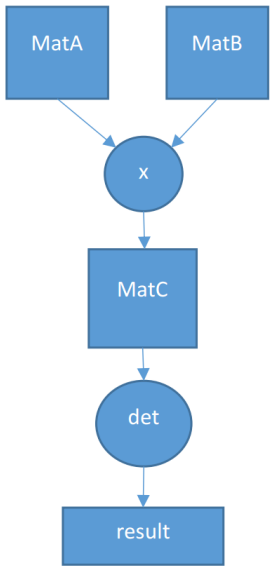

Figura 5: Multiplicador de matriz y cálculo de determinante.

Para poder llamar al módulo de multiplicación y cálculo de determinante desde el procesador NIOS II se necesita una interface compatible. Para esto se creó un módulo que encapsula al módulo de multiplicación y que provee esta interfaz. En la figura 6 se aprecia el diagrama de la interfaz elaborada.

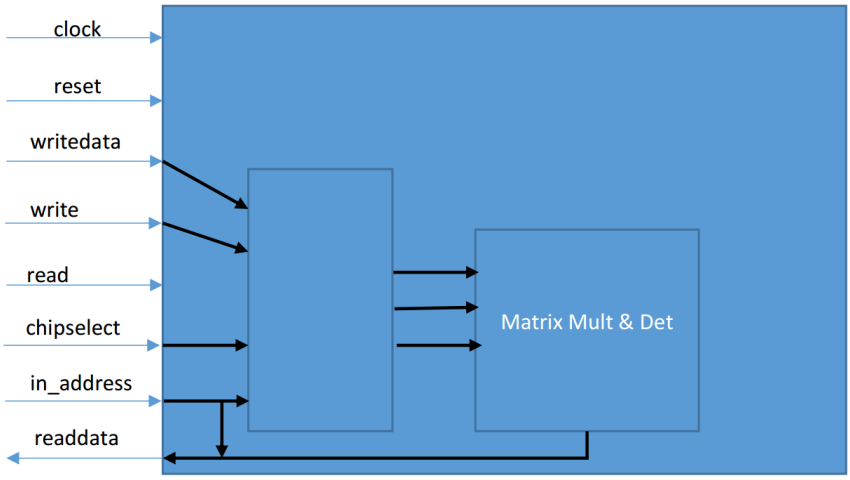

Figura 6: Módulo de interfaz.

Ademas, para este módulo se crearon 4 registros, los cuales todos son de 32 bits:

Cuadro 3: My caption

Dirección	Registro	R/W	Bits	Descripción
0x0	DATA1	W	[31=NU][30:24=MAT_A(0)(0)] [32=NU][22:16=MAT_A(0)(1)] [15=NU][14:8=MAT_A(1)(0)] [7=NU][6:0=MAT_A(1)(1)]	First 2x2 input matrix. Composed as 4 concatenated values of 1 byte each. Order is 00,01,10,11. MSB of each byte is ignored.
0x1	DATA2	W	[31=NU][30:24=MAT_B(0)(0)] [32=NU][22:16=MAT_B(0)(1)] [15=NU][14:8=MAT_B(1)(0)] [7=NU][6:0=MAT_B(1)(1)]	Second 2x2 input matrix. Composed as 4 concatenated values of 1 byte each. Order is 00,01,10,11. MSB of each byte is ignored.
0x2	CONTROL	W	[31:1=NU][0=ENABLE]	Enable=1, Disable=0
0x3	OUT	R	[31:0=RESULT]	RESULT = [MAT_A*MAT_B] Determinant of the multiplication of the two 2x2 input matrices.

- Registro donde se guarda la primera matriz.
- Registro donde se guarda la segunda matriz.
- Registro de 'enable'.
- Registro de lectura donde se guarda el resultado.

El módulo de multiplicación espera datos de 28 bits (7 bits por cada elemento de la matriz), entonces hay que mapear los 32 bits a los 28 requeridos, descartando el más significativo de cada byte. Al ser el módulo de multiplicación puramente combinacional el CPU no tiene que esperar ningún ciclo para recibir la respuesta, por lo que se hace más rápida la comunicación. Ahora, si se piensa utilizar un reloj súmamente rápido, puede que el módulo de multiplicación haya que modificarlo para incluir un pipeline que parta la operaciones en secciones más pequeñas, así el setup timing de registro a registro no se verá comprometido.

Adicionalmente, al diseño se agregó un timer para poder medir el tiempo de ejecución de la aplicación y poder comparar el rendimiento del hardware adicionado contra el diseño sin este hardware.

En la figura 7 se tiene un diagrama de bloques del diseño completo:

Para la parte del código C se tomó el código de ejemplo y se modificó duplicando la parte de las iteraciones para que corra una vez con código C puro y otra vez con el módulo de hardware. Alrededor de las iteraciones se colocó código que configura el timer para poder medir el tiempo que dura corriendo.

De manera que para la parte del código que configura el modulo de hardware, se hace lo siguiente:

- Se escribe el registro que contendrá la primera matriz.
- Se escribe el registro que contendrá la segunda matriz.
- Se habilita la multiplicación de matrices escribiendo al registro de enable.
- Seguidamente se lee del registro de output y se escribe un cero al registro de enable.

Esto se realiza para realizar cada una de las multiplicaciones, y finalmente se imprimen los resultados para cada uno de los modos (C y hardware) para verificar que son iguales.

2.2. Resultados

Al correr la simulación en ModelSim se obtuvo que el código C puro duró 3514860 ns, mientras que con el módulo de hardware se duró 116900 ns tal y como se aprecia en la figura 7.

Esto demuestra que el módulo de hardware es 30 veces más rápido que en el caso donde se hace la multiplicación únicamente con el procesador, por lo que se tiene un gran incremento en eficiencia, verificando que

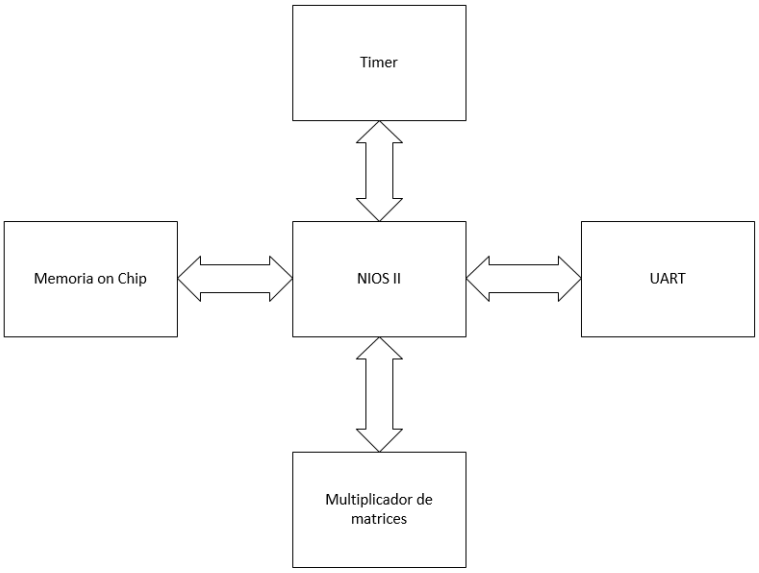

Figura 7: Diagrama de bloques del diseño completo

es muy útil tener módulos de hardware que hagan cálculos pesados para mejorar la eficiencia y velocidad de un SoC.

```
# Time elapsed during software test = 3514860 ns
# Time elapsed during hardware test = 116900 ns
# Iter #0: SW: 14323788 HW: 14323788
# Iter #1: SW: 5250135 HW: 5250135
# Iter #2: SW: 4279296 HW: 4279296
# Iter #3: SW: 5506536 HW: 5506536
# Iter #4: SW: -2171520 HW: -2171520
# Iter #5: SW: -542360 HW: -542360
# Iter #6: SW: 2135824 HW: 2135824
# Iter #7: SW: 2577960 HW: 2577960
# Iter #8: SW: 3543288 HW: 3543288
# Iter #9: SW: -4368960 HW: -4368960
# Iter #10: SW: -363000 HW: -363000
# Iter #11: SW: -6399459 HW: -6399459
# Iter #12: SW: 8243120 HW: 8243120
# Iter #13: SW: 605700 HW: 605700
# Iter #14: SW: 1385624 HW: 1385624
# Iter #15: SW: -855475 HW: -855475
```

Figura 8: Resultados

3. Conclusiones

El proyecto realizado permitió elaborar un prototipado a partir de una especificación de diseño, de manera que se debió decidir a partir de la información y de las condiciones existentes, el tipo de prototipado para cada IP, así como la implementación de los mismos. Con esta información se estimaron los recursos del sistema y además se hizo una selección de plataforma de prototipo. Para finalmente definir el plan de pruebas requerido para la verificación del diseño.

Además, en la parte 2 del proyecto se realizó un hardware para calcular el determinante resultante de una multiplicación de matrices, de manera que el mismo se pudiera comunicar a través de una interfaz a

un procesador NIOS II. De manera que se realizaron cálculos haciendo las operaciones directamente con el procesador y también utilizando como medio el hardware elaborado. Al comparar los tiempos de ejecución se demostró que el hardware elaborado permite disminuir el tiempo de ejecución del código en alrededor de 30 veces. Por lo que se demostró que la implementación de hardware adicional para operaciones complejas permite tener un rendimiento considerable en los sistemas on chip.