DISEÑO E IMPLEMENTACIÓN DE UN PROCESADOR IP CORE EN UN DISPOSITIVO DE LÓGICA PROGRAMABLE FPGA

T.G 0946

ANAMARÍA VIVEROS ROCHA

Informe Final

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA
2010

DISEÑO E IMPLEMENTACIÓN DE UN PROCESADOR IP CORE EN UN DISPOSITIVO DE LÓGICA PROGRAMABLE FPGA

T.G 0946

ANAMARÍA VIVEROS ROCHA

Directora
Ing. LUISA FERNANDA GARCÍA VARGAS, M.Sc.
Profesora

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA
2010

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

RECTOR MAGÍFICO:	RP. JOAQUÍN SÁNCHEZ GARCÍA. S.J.
DECANO ACADEMICO:	ING. FRANCISCO JAVIER REBOLLEDO MUÑOZ.
DECANO MEDIO UNIVERSITARIO:	RP.SERGIO BERNAL RESTREPO. S.J.
DIRECTOR DE CARRERA:	ING. JUAN MANUEL CRUZ M.Sc.
DIRECTOR DE PROYECTO:	ING. LUISA FERNANDA GARCÍA VARGAS. M.Sc.

ARTÍCULO 23 DE LA RESOLUCIÓN No. 13 DE JUNIO DE 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

Dedico este trabajo de grado a:

A DIOS, por darme el conocimiento y la sabiduría en todos los momentos cruciales e importantes de mi vida.
Por darme momentos de felicidad y por ofrecerme este reto.

A mi Papá, porque tú me has ayudado a encontrar siempre el camino buscando la luz cuando estoy en las tinieblas.
Porque eres y seguirás siendo la persona que más admiro.

A mi Mamá, porqué tú eres la que más celebra todos mis éxitos, la que siempre me ha apoyado y me has dado tu hombro para llorar.
Por ser mi verdadera y única confidente.

A mis hermanas, Ange y Lily, por demostrarme que todos los sueños se pueden realizar sólo si eres constante y dedicado. Por mostrarme que para triunfar en esta vida hay que tener metas fijas por las cuales luchar.

AGRADECIMIENTOS

Quiero agradecerle a mi papá por todas las dudas que me ayudó a resolver, por ser el usuario y crítico más activo de este trabajo de grado. Gracias a sus ideas, su conocimiento y su motivación, el proceso de diseño del procesador fue más sencillo.

Quiero agradecer a mi mamá y a mis hermanas, Ange y Lily, por su apoyo constante, por el ánimo y la fe que me brindaron durante este periodo.

Quiero darle un especial agradecimiento a Luisa, mi directora, porque siempre estuvo pendiente de todo el trabajo, porque es una guía excelente y su exigencia logró motivarme para hacer un gran trabajo.

Quiero agradecerles a todas las personas que me ayudaron en este proceso y compartieron buenas ideas para este Trabajo de Grado.

Finalmente, quiero agradecer a la Dirección de Carrera de Ingeniería Electrónica por todo lo que me han enseñado, tanto en la parte académica como en la parte moral, por su disponibilidad y por brindarme siempre su apoyo.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

En los últimos años, los sistemas digitales han tomado un nuevo rumbo en las nuevas creaciones de la electrónica. Muchas cosas que conocemos en la actualidad son controladas por sistemas digitales complejos como los nuevos automóviles, las calculadoras e inclusive el sistema de acceso a los cubículos que existe en el Departamento de Electrónica.

En la Sección de Técnicas Digitales del Departamento de Electrónica, en el curso de Arquitectura de Procesadores, se ha enfatizado en una metodología en la cual el estudiante debe deducir cuales serían las partes básicas de un procesador dependiendo de ciertos parámetros que da el profesor. El estudiante, utilizando la información adquirida en los anteriores cursos, logra llegar a una aproximación de un procesador. Adquiere una visión básica de su funcionamiento que es suficiente para poder entender y utilizar un microprocesador o un micro controlador comercial.

Además, un apoyo a estos cursos ha sido los trabajos de grado que se han realizado en la Sección. Un ejemplo de estos es PCSIM [5], el cual es un software demostrativo del funcionamiento interno de un procesador que incluye conexiones a diversos periféricos y está basado en la arquitectura del RIC [3]. A partir de esta arquitectura, se realizaron dos trabajos de grado, ALTERIC [2] y BINARIC [1], los cuales son la implementación en un dispositivo de lógica programable, del procesador simulado en PCSIM, para uso académico y son el complemento del simulador. El simulador PCSIM se utiliza actualmente en el curso de Arquitectura de Procesadores.

El objetivo principal de este trabajo de grado, que se presenta en este documento, es el diseño, desarrollo y la implementación de un procesador IP CORE (núcleo de propiedad intelectual) [4], en una FPGA de Altera. El procesador que se mostrará es de propósito general con un conjunto de instrucciones seleccionado, con la capacidad de interconectarse a varios periféricos y a núcleos iguales del mismo procesador. Una de las ideas principales es poderlo utilizar en las asignaturas de la sección de Técnicas Digitales, trabajos de grado y en futuras aplicaciones de procesamiento paralelo.

Para su diseño y desarrollo se utilizó la metodología de diseño que se ha desarrollado en las asignaturas de la sección de Técnicas Digitales, que comprende una descripción del sistema hasta llegar a las especificaciones, el diagrama de bloques, una descripción en el lenguaje AHPL, la codificación en VHDL, una simulación inicial en el software de ALTERA llamado QUARTUS II 9.1 y la implementación sobre la FPGA.

Este documento se divide en cinco (5) capítulos:

- CAPÍTULO 1: es el marco teórico, el cual da una visión general y deja que el lector se contextualice con algunos conceptos que se van a ir manejando durante el libro.
- CAPÍTULO 2: son las especificaciones del procesador tales como el diagrama general de bloques, la interfaz entrada/salida, interrupciones y el manejo de la pila, Instrucciones, modos de direccionamiento y el protocolo de comunicación entre dos procesadores del mismo núcleo. También están descritas algunas consideraciones que se deben tener en cuenta cuando se conectan en dos procesadores o más en paralelo.
- CAPÍTULO 3: es la parte de desarrollo donde se encuentran los recursos de la *FPGA Cyclone III* EP3C25F324 y de la tarjeta de desarrollo, el protocolo de pruebas utilizado para el análisis de

resultados del procesador y la metodología utilizada para realizar el AHPL y la codificación en VHDL.

- CAPÍTULO 4: en este capítulo está el análisis de resultados de los programas de prueba realizados para el procesador. Además se encuentra el análisis de recursos, tiempos y potencia. Estas pruebas se realizaron para un solo procesador y para dos procesadores.
- CAPÍTULO 5: aquí se encuentra las conclusiones y los posibles trabajos futuros que se puede realizar con este trabajo de grado.

1. MARCO TEÓRICO

1.1. EL PROCESADOR

Para el lector, es importante conocer todos los conceptos referentes a este trabajo de grado para entender tanto el desarrollo como las herramientas utilizadas en el mismo. En esta parte se escribirá de forma resumida ya que el objetivo de este documento es describir el diseño que se realizó.

Un procesador es un sistema digital complejo, compuesto de unidad de control, ALU (*Arithmethic Logic Unit*), registros y las diferentes interconexiones, las cuales son las que proporcionan la comunicación entre todos los componentes.

Los procesadores desde su concepción, han conservado sus principales características pero su arquitectura ha ido evolucionando para que estos tengan mayor desempeño y adquieran mayor velocidad. Cuando se creó el primer microprocesador en 1971 por Intel **[7]**, el primer *chip* que contuvo todos los componentes mencionados, se multiplicaron las aplicaciones en las cuales se utilizaba un procesador porque los costos bajaron al reducir el tamaño de los *chip*.

En la actualidad, se siguen diseñando procesadores con las mismas metas, más velocidad y más eficiencia. Una de las formas utilizadas para obtener más velocidad es el procesamiento en paralelo, en el cual se conectan de dos o más núcleos del mismo procesador para hacer una o varias tareas al mismo tiempo. En la sección 1.4 se explicará este concepto.

1.2. LENGUAJE DE PROGRAMACIÓN DE *HARDWARE* AHPL

Cuando se está diseñando un sistema digital complejo, es necesario seguir las señales del diseño de forma sencilla. Para esto hay diferentes maneras de hacer la descripción del sistema, algunos son el diagrama de bloques, los diagramas de flujo y las tablas de conectividad. Otra manera de describir un sistema digital complejo es utilizar un lenguaje de descripción de *hardware* que muestre todo el funcionamiento del sistema de una manera fácil de leer y de implementar posteriormente con un software apropiado.

En los años 80's, los Ingenieros Fredrick J.Hill y Gerald R.Peterson **[3]** desarrollaron un lenguaje de hardware, llamado AHPL(*A Hardware Programming Language*). Se basó en el lenguaje desarrollado por K.E Inverson llamado APL (*A programming language*), el cual solo se utilizaba primordialmente en programación interactiva teniendo la descripción de *hardware* como una de sus aplicaciones. Hill y Peterson utilizaron algunas notaciones y simplificaciones del APL y le añadieron características adicionales para describir el comportamiento del *hardware*.

Para el procesador de este trabajo de grado se utilizó el AHPL como una de las herramientas más importantes del diseño. La ventaja del AHPL es que es un lenguaje muy cercano al hardware, que

permite al diseñador visualizar fácilmente el circuito resultante y por tanto facilita su tarea. Previamente es necesario realizar un diagrama en bloques del sistema a diseñar y luego es la guía para la realización del VHDL.

1.3. LENGUAJE DE DESCRIPCIÓN DE HARDWARE VHDL

El VHDL describe el comportamiento y la estructura de sistemas digitales, que se pueden implementar como un circuito electrónico. La sigla VHDL se puede dividir como VHSIC-HDL.

HDL son las iniciales de "Hardware Description Languaje" del cual se pueden resaltar las siguientes tres características tal como lo menciona el Ingeniero Guillermo Jaquenod:

> "Describe Procesos o actividades que ocurren de forma simultánea (Concurrencia).
>
> Posibilita la construcción de una estructura jerárquica, donde es posible combinar estilos de descripción.
>
> Permite describir módulos con acciones que serán evaluadas en forma secuencial, donde todo el módulo será visto como una acción concurrente."[1]

Otra característica o ventaja es que al ser un lenguaje estándar el código puede ser reusado en diferentes aplicaciones.

VHSIC es la abreviación para *Very High Speed Integrated Circuits*, circuitos integrados diseñados por el Departamento de Defensa de los Estados Unidos de América en 1980 y que permitieron la creación del VHDL. Este fue el primer lenguaje de descripción de *hardware* que fue estandarizado por la IEEE.

[1] JAQUENOD, Guillermo. Diseño Digital con FPGA. Apuntes de Clase. Año 2010

2. ESPECIFICACIONES

El procesador diseñado tiene un bus de direcciones de 14 bits lo cual determina un tamaño máximo de una memoria de 16k posiciones y un bus de datos de 32 bits que corresponde al tamaño de la palabra, tiene además ocho modos de direccionamiento, un repertorio de 58 instrucciones para realizar diferentes aplicaciones, los circuitos necesarios para realizar una comunicación con periféricos y las señales de conexión que constituyen el protocolo de comunicación entre dos o más procesadores iguales.

El tamaño del bus de direcciones se escogió por las memorias embebidas de la FPGA que se utilizaron para la implementación del proyecto. El dispositivo cuenta con 64 bloques M9k de 512×16 cada uno, lo cual equivale a 16384 posiciones de memoria, por lo tanto se necesitan 14 bits en el bus de direcciones. El criterio para el tamaño del bus de datos se escogió partiendo que un procesador comercial puede ser de 32 y de 64 bits. Con esto, se decidió que el bus fuera de 32 bits para que se pudieron utilizar formatos de instrucción de 6bit en el código de la instrucción, 3 para los modos de direccionamiento y que el tamaño de los datos fuera apreciable.

A continuación se hará una descripción de los diferentes subsistemas que conforman la arquitectura del procesador OokiCore.

2.1 INTERFAZ ENTRADA/SALIDA

En el diagrama de interfaz que se muestra en la figura 1, se observan 9 entradas y 11 salidas desde el procesador.

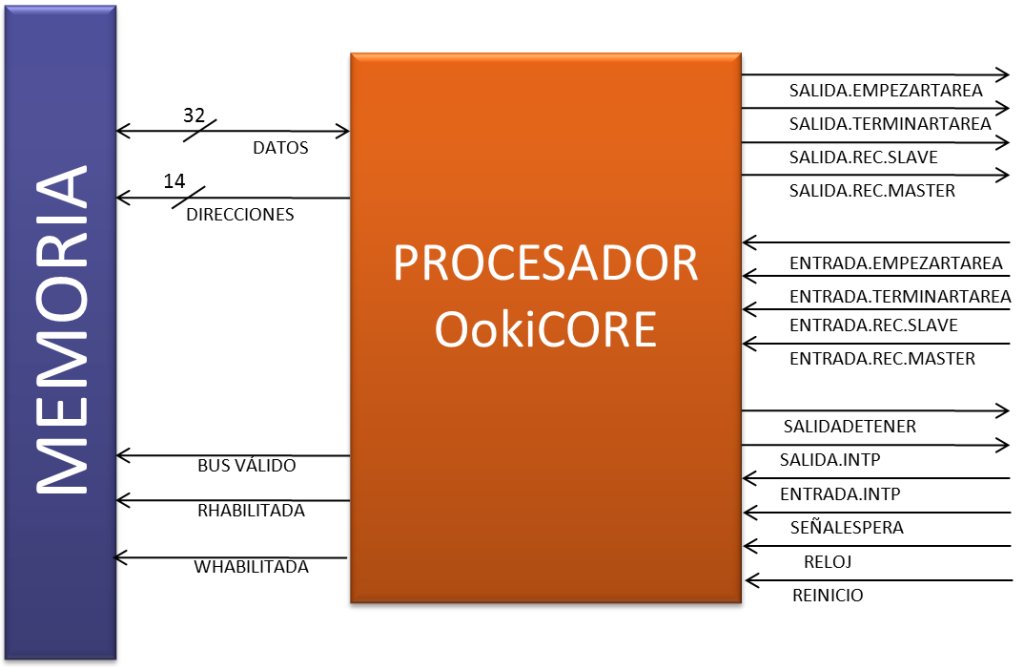

Figura 1. Interfaz Entrada/Salida

Las señales de entrada - salida son:

- La entrada al procesador llamada **DATOS**, es un bus bidireccional de 32 bits, el cual es la interconexión de datos entre la memoria, periféricos y el procesador.
- La señal de entrada **ENTRADA.INTP,** es un bit que indica al procesador cuando hay una interrupción generada por algún periférico.

- La señal de entrada **REINICIO** (Reset), es una señal asíncrona que cuando se activa deja al procesador en su estado inicial.

- La señal de entrada **RELOJ** (Clk)**,** es la señal de reloj de frecuencia 25 MHz (Máxima frecuencia del procesador). Con el borde de subida se maneja la carga de todos los registros. El bloque de control utiliza el borde se bajada de este.

- La señal de entrada **SEÑALESPERA** (SENALWAIT), es un bit, que se utiliza cuando el usuario necesita que el procesador quede en estado de espera debido a que un sistema externo conectado a él, no tiene la velocidad necesaria para comunicarse. Cuando está activo este bit, el procesador detiene la tarea que está haciendo y espera hasta que se desactive la señal para reanudar lo que esté haciendo.

- La señal de salida **DIRECCIONES** es un bus de 14 bits el cual tiene la dirección de memoria o del periférico al cual se desea acceder.

- La señal de salida **RHABILITADA** (RENABLE), es un bit que indica que el procesador va a *leer* los datos que se encuentran en la memoria.

- La señal de salida **WHABILITADA (**WENABLE), es un bit que indica que el procesador va a *escribir* los datos que se encuentran en la memoria.

- La señal de salida **SALIDADETENER,** es un bit que se activa cuando el usuario utiliza la instrucción de Detener el procesador (*HLT*, ver *Manual del Usuario*). La máquina de estados del procesador queda en un lazo infinito de tal manera que la única manera de salir de él es generando una señal de reinicio (*Reset)*.

- La señal de salida **BUS VÁLIDO**, es un bit que indica que los datos que se encuentran en el bus de datos y direcciones son válidos para el procesador.

- La señal de salida **SALIDA.INTP,** es un bit que indica que el procesador ya reconoció la interrupción.

Adicionalmente se encuentran las señales de entrada/salida que sirven para comunicar dos núcleos de este procesador, la configuración de Maestro o Esclavo depende del usuario.

Las señales son:

- La señal de salida **SALIDA.EMPEZARTAREA**, es un bit que le indica a otro procesador que debe comenzar una tarea. Se modifica con las instrucciones de protocolo E*mpezar tarea (SET)* y *Borrar señal de empezar tarea(BET).*

- La señal de salida **SALIDA.TERMINARTAREA,** es un bit que indica cuando el procesador que genera la señal ha terminado la rutina asignada. Se modifica con las instrucciones de protocolo *Terminar tarea (STT)* y *Borrar señal de terminar tarea(BTT).*

- La señal de salida **SALIDA.REC.SLAVE,** es un bit que se modifica con las instrucciones de protocolo *Reconocimiento del slave(SRS)* y *Borrar señal de reconocimiento del slave(BRS).*

- La señal de salida **SALIDA.REC.MASTER,** es un bit que se modifica con las instrucciones de protocolo *Reconocimiento del master(SRM)* y *Borrar señal de reconocimiento del master(BRM).*

- La señal de entrada **ENTRADA.EMPEZARTAREA**, es un bit que indica si se ha generado la señal SALIDA.EMPEZARTAREA. Esta señal modifica la bandera ET del registro de estados.

- La señal de entrada **ENTRADA.TERMINARTAREA,** es un bit que indica si se ha generado la señal SALIDA.TERMINARTAREA. Esta señal modifica la bandera TT del registro de estados.

- La señal de entrada **ENTRADA.REC.SLAVE,** es un bit que indica si se ha generado la señal SALIDA.REC.SLAVE. Esta señal modifica la bandera RS del registro de estados.

- La señal de entrada **ENTRADA.REC.MASTER,** es un bit que indica si se ha generado la señal SALIDA.REC.MASTER. Esta señal modifica la bandera RM del registro de estados.

El protocolo de comunicación que las utiliza se explica en la sección 2.8.

2.2 DIAGRAMA DE BLOQUES DEL PROCESADOR

El procesador está compuesto de 8 bloques registro, un contador de corrimiento, un bloque de control, un selector y por una unidad algorítmica lógica (ALU). En la figura 2 se muestra el diagrama de bloques del procesador.

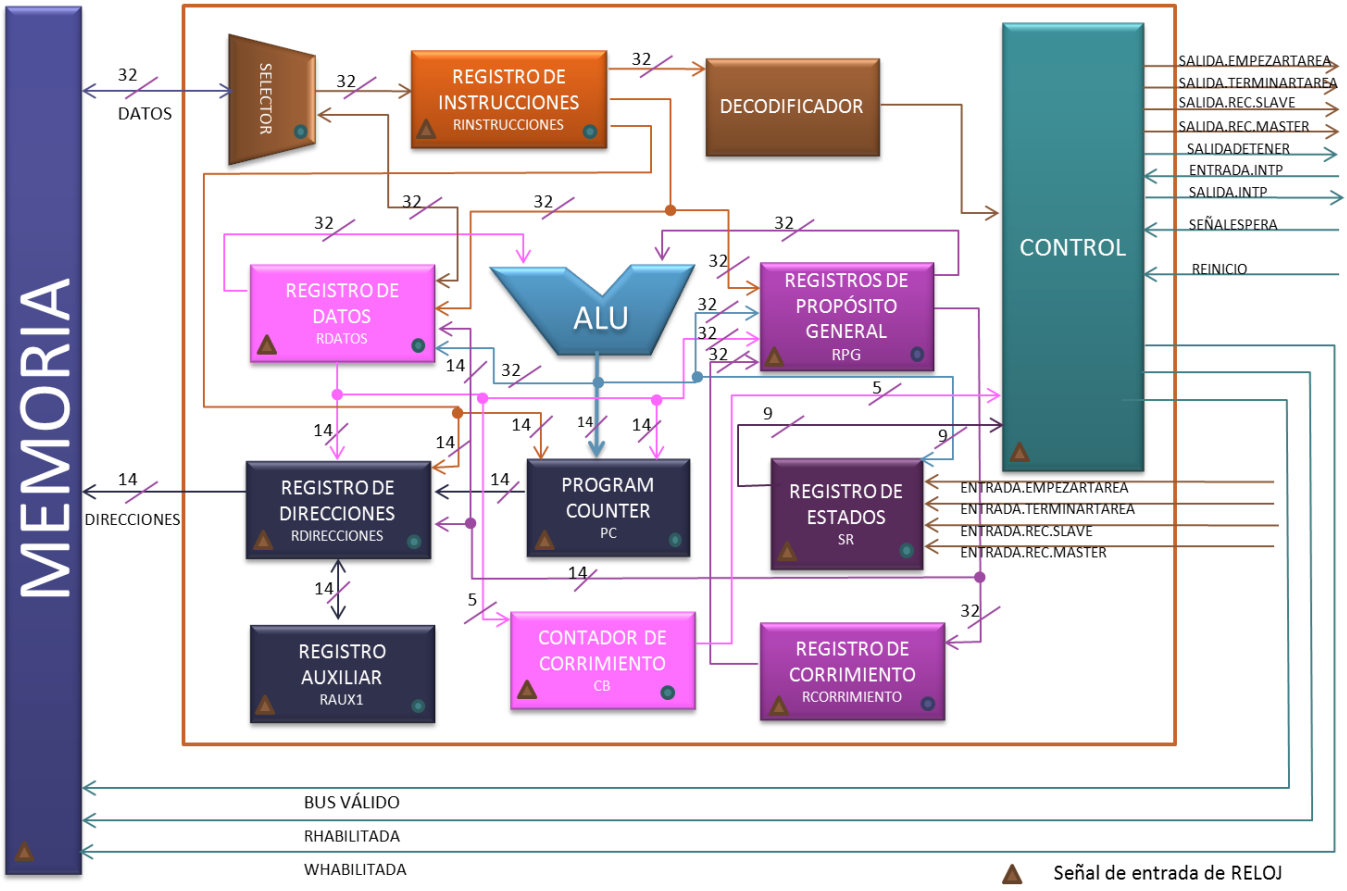

Figura 2. Diagrama de Bloques Principal

Como se implementó el proyecto en la FPGA Cyclone III EP3C25F324, se utilizaron las memorias embebidas con dos configuraciones: *True Dual Port* y *Simple Port*. La configuración *Simple Port* tiene un bus de entrada de escritura a la memoria y un bus de salida de lectura para obtener los datos de la memoria y la configuración *True Dual Port* tienen por cada puerto un bus

de escritura y un bus de lectura. Por esa razón, fue necesario hacer unos cambios al diagrama de bloques como se ve en la figura 3.

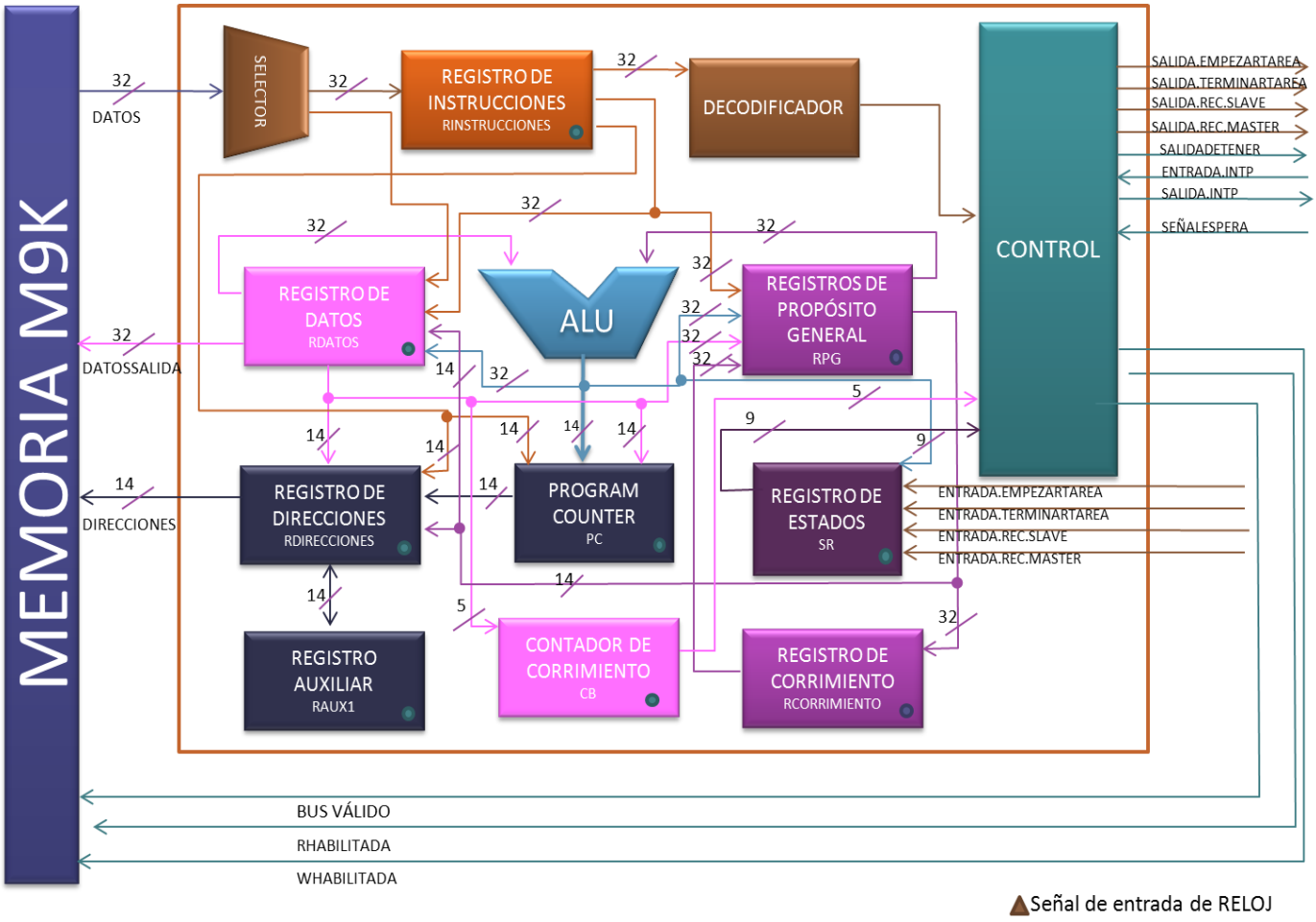

Figura 3 Diagrama de Bloques del procesador con la configuración de las memorias embebidas.

1. Bloque Memoria: el bloque de memoria es externo al procesador. Este bloque hace referencia a una memoria de 512 × 16, que puede almacenar tanto instrucciones como datos. Para manejar las direcciones se cuenta con un bus de 14 bits y para los datos un bus de 32 bits.

2. Bloque selector (SELECTOR): tiene como entrada un bus bidireccional de 32 bits. El bloque se encarga de escoger a que registro, ya sea al Registro de Instrucciones o el Registro de Datos, debe mandar los datos de la entrada.

3. Registro de instrucciones (RINSTRUCCIONES): es un registro que recibe como entrada un bus de 32 bits y almacena la instrucción a ejecutar por el procesador.

4. Bloque Decodificador (DECODIFICADOR): recibe la instrucción de 32 bits y se encarga de decodificar el tipo de instrucción y el modo de direccionamiento. La salida de este bloque está conectada al bloque de control; esta información es necesaria para el funcionamiento del procesador.

5. Registro de Datos (RDATOS): es un registro que almacena los datos a procesar o almacenar, estos pueden venir del bus de datos, del registro de instrucciones, del PC o de los RPG.

6. Registro de direcciones (RDIRECCIONES): es un registro que almacena una dirección especifica de 14bits dada ya sea por el *Program Counter*, o el registro de instrucciones, o algún RPG o calculada según el modo de direccionamiento. Se encarga de indicar a la memoria o periférico en cual dirección se va a escribir o leer.

7. *Program Counter* (PC, contador de programa): es un registro que almacena e incrementa la dirección de 14 bits de la siguiente instrucción que se va a ejecutar. La dirección de arranque, cuando se está utilizando un solo procesador, que está almacenada en PC es la 00000001h. En el caso de utilizar dos procesadores conectados ver la sección 2.7.

8. *Status Register* (SR, Registro de Estados): es un registro de bits. Los primeros 4 bits indican si alguna de las operaciones realizadas por la ALU tienen un resultado igual cero, o si dio un valor negativo, si tienen *overflow* (desborde) o *carry* (acarreo). Los otros 4 bits se refieren al protocolo de comunicación entre dos procesadores iguales como referencias para las instrucciones de salto para realizar una tarea en paralelo. El último bit se utiliza para las rutinas de interrupciones (Ver sección 2.3). El orden de las banderas se muestran en la figura 3 y están organizados de la siguiente manera:
   C: *carry* (bit 0)
   O: *overflow* (bit 1)
   N: negativo (bit 2)
   Z:cero (bit 3)
   ET: empezar Tarea (bit 4)
   RS: reconocimiento *Slave* (bit 5)
   TT: terminar Tarea (bit 6)
   RM: reconocimiento *Master* (bit 7)
   Intp: interrupción (bit 8)

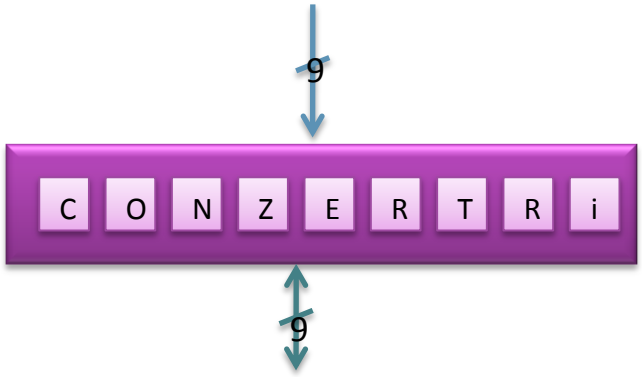

Figura 4. Registro de Estados

Los bits correspondientes al protocolo de comunicación son modificados automáticamente por el procesador al cambiar la entrada de protocolo que le corresponde.

9. Registros de propósito general (RPG): a diferencia de los otros bloques, este se refiere a un conjunto de 16 registros de 32 bits.
   Entre estos registros se deben destacar dos:
   a. Registro AC: es el comúnmente llamado registro acumulador en otras arquitecturas. Es igual a todos los registros pero con la excepción que es el único registro que tiene un bus de salida conectado al Registro de datos, cuando se desea pasar este dato a memoria.

Además en las instrucciones de un operando, es el registro implícito cuando se desea realizar alguna operación.

b. Registro *Stack Pointer* (Apuntador de pila)*:* es un registro que se encarga de almacenar la dirección de memoria donde se encuentra la cabeza de la pila. A partir de esta dirección de memoria se almacenan las direcciones a las que se debe retornar cuando se ejecutan subrutinas e interrupciones o los datos que se quieran almacenar.

   Después de insertar un dato en la pila, se decrementa la dirección para apuntar de esta manera a la siguiente posición en que puede quedar almacenado un nuevo dato. Cuando se desea extraer un dato de la pila, se incrementa la dirección para apuntar al dato que se desea sacar y luego se realiza la extracción. La posición de la pila en la memoria es definida por el usuario, mediante una instrucción de mover al registro SP. En la sección 2.3 se explica el manejo de la pila y el papel del *Stack Pointer*.

   En un programa que no tenga interrupciones o subrutinas, el usuario tiene la libertad de utilizarlo como un registro general de 32 bits.

Los otros 14 registros pueden almacenar tanto datos como direcciones según lo requiera el usuario.

Los RPG están codificados tal como se muestra en la Tabla 1:

REGISTRO	CÓDIGO	REGISTRO	CÓDIGO
AC	0000	**RG**	1000
SP	0001	**RH**	1001
RA	0010	**RI**	1010
RB	0011	**RJ**	1011
RC	0100	**RK**	1100
RD	0101	**RL**	1101
RE	0110	**RM**	1110
RF	0111	**RN**	1111

Tabla 1 Codificación de los registros de propósito general

10. Unidad Aritmético Lógica (ALU): este bloque se encarga realizar las operaciones aritméticas y lógicas según la instrucción a ejecutar. También se utiliza la ALU cuando se desea incrementar o disminuir una dirección de memoria. Para todas las operaciones aritméticas, los datos utilizados deben ir en notación complemento 2.

11. Registro de Corrimiento (RCorrimiento): este registro se utiliza cuando el usuario desea realizar un corrimiento (lógica o aritmética) o una rotación, ya sea hacia la derecha o hacia la izquierda. Se utiliza este registro ya que su lógica de entrada y la lógica de su habilitación es diferente del resto de registros (ver el capítulo 3 Desarrollo).

12. Contador de Corrimiento (CB): es el registro donde se almacena el número de veces que se desea correr o rotar un dato. Esto se puede hacer máximo 32 veces.

13. Registro Auxiliar1(RAux1): este registro almacena al RDIRECCIONES, cuando se desea guardar una copia del mismo (ver Anexo A).

14. Bloque de Control: es el bloque que toma todas las decisiones, indica que bloque activar, desactivar o que señales enviar según la instrucción en el que se encuentre. Además, es el bloque que genera las señales de protocolo para la comunicación entre los procesadores, las señales de RENABLE, WENABLE y BUS VALIDO para poder utilizar la memoria y periféricos cuando se necesite y verifica las interrupciones.

2.3 INTERRUPCIONES Y MANEJO DE PILA

El sistema sólo tendrá una línea de interrupción por la cual se conectan los diferentes periféricos externos, es decir, que todos los periféricos manejan una única entrada de interrupción. El manejo de interrupciones es de tipo auto vectorizado. Si el usuario desea conectar dos procesadores en la sección 2.7 se explica el manejo de interrupciones para este caso.

Cuando hay una interrupción, el procesador accede a la posición de memoria 00000000h (para un solo procesador) y se actualiza para iniciar la ejecución del programa en la nueva dirección. El usuario debe colocar en la posición de memoria anteriormente mencionada la dirección de inicio de la rutina o programa de interrupción

Cuando inicia el sistema, todas las interrupciones están deshabilitadas, motivo por el cual es necesario que el usuario se encargue de habilitar la interrupción, si esta hace parte de alguno de sus programas. Para habilitar la interrupción es necesario poner un '1' en el bit 9 del *Status register*, utilizando la instrucción *(LBI)*. Si el usuario no desea que el procesador acepte la interrupción, esta se puede deshabilitar, para lo cual el usuario debe dejar en '0' el bit de interrupción del *Status Register*, utilizando la instrucción *(BBI)*.

El procesador maneja una pila tipo LIFO (*Last in-First out*) ubicada en memoria, la dirección inicial es establecida por el usuario y se debe almacenar en el *Stack Pointer*. Se utiliza en los saltos a subrutinas e interrupciones almacenando la dirección actual del programa para luego utilizarla en su retorno. Opcionalmente, si el usuario lo desea, se pueden almacenar los datos provenientes de los RPG y del registro de estado.

2.4 FORMATOS DE INSTRUCCIÓN

Para las instrucciones se manejaron 6 formatos de instrucción diferentes, los cuales se muestran a continuación.

El formato 1 que se muestra en la figura 4, se utiliza para instrucciones de un solo operando, donde los primeros 6 bits son el código de operación, los siguientes 3 son el modo de direccionamiento y los últimos 23 bits corresponden al operando.

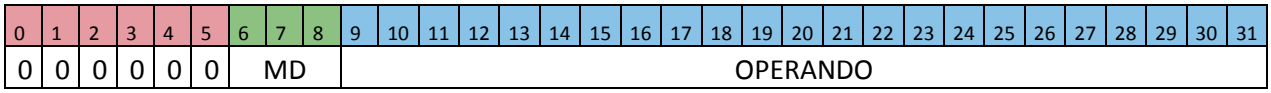

Figura 5. Primer Formato utilizados para instrucciones de un solo operando

El formato 2 que se muestra en la figura 5, se utiliza para un solo operando en el modo de direccionamiento relativo a registros (ver sección 2.5), donde la dirección se encuentra en el campo de operando y el desplazamiento en el registro. También se utiliza para instrucciones de dos operandos, uno colocado en el campo de operando y el otro en un registro. Para este formato, el campo de operando se reduce de 23 a 19 bits y los últimos 4 bits son para el código de un RPG.

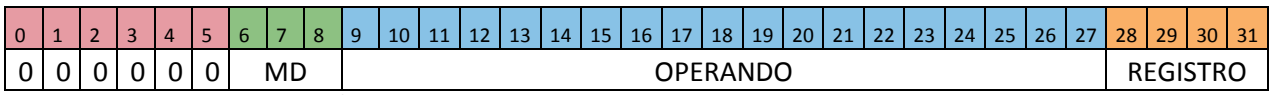

Figura 6 Segundo formato utilizado para instrucciones de un solo operando y de dos operandos

El formato 3 que se muestra en la figura 6, se utiliza para instrucciones de dos operandos cuando el usuario desea realizar operaciones entre registros. En el campo de REGISTRO 1, se coloca el registro donde quedaría el resultado final. Nótese que el campo de operando no se utiliza y por tanto puede quedar en cualquier valor.

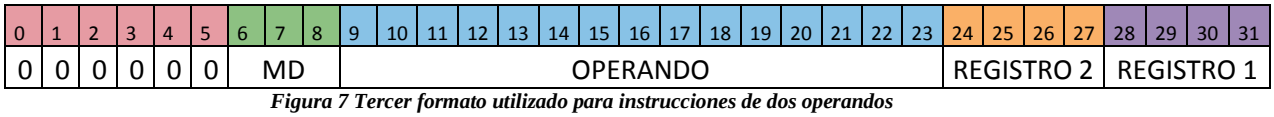

Figura 7 Tercer formato utilizado para instrucciones de dos operandos

El formato 4 que se muestra en la figura 7, se utiliza para las instrucciones de saltos condicionales y para las señales de protocolo (Ver manual de Usuario). Los primeros 6 bits son el código de operación de la instrucción, los bits del 6 al 13 son para la condición y el campo del operando es de 18 bits. En el campo del operando el usuario coloca el desplazamiento desde la dirección donde se encuentra la instrucción hasta la dirección donde debe saltar.

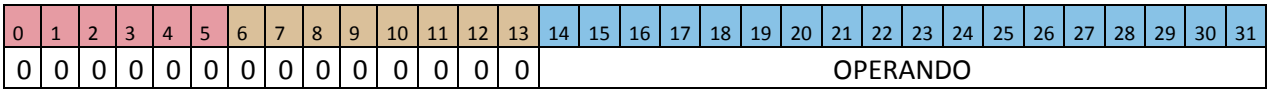

Figura 8.Cuarto formato que se utiliza para las instrucciones de salto condicional y de protocolo

El formato 5 que se muestra en la figura 8, se utiliza para las instrucciones que no tiene ningún modo de direccionamiento. Los primeros 6 bits son los del código de operación y los siguientes están predefinidos.

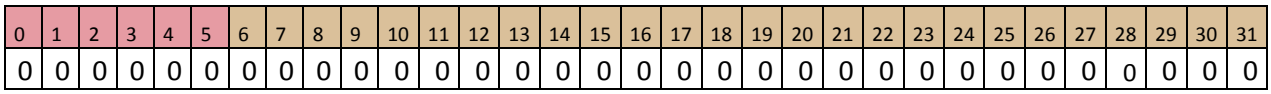

Figura 9 Quinto formato que se utiliza para algunas instrucciones sin modo de direccionamiento

En las instrucciones de negación aritmética, negación lógica y las instrucciones para el manejo de la pila se utiliza el formato 6 que se muestra en la figura 9, donde los primeros 6 bits son el código de operación, los siguientes 22 están predefinidos de acuerdo con la instrucción y los 4 últimos bits son el código del RPG donde se encuentra el dato.

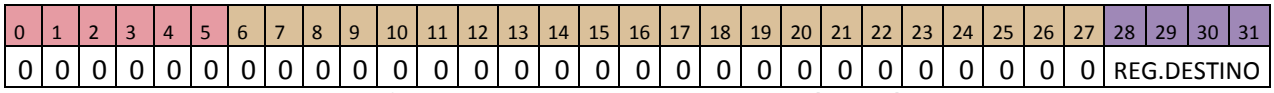

Figura 10 Sexto formato para las instrucciones de NOT y Negación Aritmética

2.5 MODOS DE DIRECCIONAMIENTO

El procesador tiene 7 modos de direccionamiento:

1. Directo (000): en este modo de direccionamiento en el operando se encuentra una dirección efectiva de memoria.

2. Indirecto (001): en este modo de direccionamiento en el operando se encuentra una dirección de memoria en la cual se encuentra almacenada la dirección efectiva.

3. Inmediato (010): en este modo de direccionamiento el operando no almacena ninguna dirección, el dato que contiene se puede utilizar ya sea para fijar valores de inicio a las variables, transferencia de datos o para realizar operaciones.

4. Relativo a registros (011): en este modo de direccionamiento la dirección efectiva se calcula sumando el desplazamiento colocado en el registro con la dirección que se encuentra en el campo de operando (ver Figura 5).

5. Registros (100): en este modo de direccionamiento en el operando de la instrucción se encuentran referenciados dos registros de propósito general los cuales tienen almacenados datos que se desean operar o mover de uno al otro. En formato de instrucción 3(figura 6) los últimos 4 bits

hacen referencia al registro donde está uno de los datos a operar y los 4 bits, del 24 al 27, indican el segundo registro que tiene el segundo dato en el caso de una operación.

6. Post incremento (101): este modo de direccionamiento, tiene como registro implícito al *Program Counter* y el campo del operando hace referencia al valor que se desea restar a la dirección almacenada a este. Cuando se está realizando un proceso, utilizando este modo de direccionamiento, primero la dirección efectiva es la que se encuentra almacenada en el registro. Esta se pasa al *Program Counter* y luego a este se le suma la dirección del registro con el valor que se indica en el campo del operando dado en la instrucción.

7. Pre decremento (110): este modo de direccionamiento, tiene como registro implícito al *Program Counter* y el campo del operando hace referencia al valor que se desea sumar a la dirección almacenada a este. Primero se resta la dirección que se tiene almacenada en el *Program Counter* con el valor que se indica en el campo operando. Luego la dirección efectiva es la dirección almacenada en este registro.

8. Relativo al registro PC(111): en este modo de direccionamiento, el registro implícito que se utiliza es el *Program Counter*. En las instrucciones de un solo operando, que utilizan este modo de direccionamiento, los bits del campo operando en el formato de la instrucción 1(figura 4) hacen referencia a el desplazamiento que se desea sumar con la dirección actual que se encuentra almacenada en el *Program Counter.*

   Cuando las instrucciones son de dos operandos, el valor de la dirección efectiva sería la dirección actual del *Program Counter* más el dato que está almacenado en un Registro RPG, en el formato de instrucción 3, sería el registro que está ubicado en el campo REGISTRO 1.

2.6 REPERTORIO DE INSTRUCCIONES

El procesador cuenta con un repertorio de 58 instrucciones las cuales se encuentran a continuación. Para una descripción más detallada mirar el Manual del Usuario.

INSTRUCCIONES DE TRANSFERENCIA DE DATOS						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
1	Mover un dato a registro	**MVT**	**1**	Directo	13	
				Indirecto	17	
				Inmediato	10	
				Relativo al PC	13	
				Relativo a otros registros	13	
			2	Directo	13	
				Indirecto	17	
				Inmediato	10	
				Relativo al PC	13	
				Registros	10	
2	Mover un dato de un registro a memoria	**MVF**	**1**	Directo	15	
				Indirecto	19	
				Relativo a otros registros	15	
				Relativo al PC	15	

INSTRUCCIONES DE OPERACIONES ARITMÉTICAS						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
3	Suma con carry de entrada	**ADC**	**1**	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo a otros registros	13	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
			2	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
				Registros	10	C,O,N,Z
4	Suma sin carry de entrada	**ADD**	**1**	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo a otros registros	13	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
			2	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
				Registros	10	C,O,N,Z
5	Resta con carry de entrada	**SUBC**	**1**	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo a otros registros	13	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
			2	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
				Registros	10	C,O,N,Z
6	Resta sin carry de entrada	**SUB**	**1**	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo a otros registros	13	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
			2	Directo	13	C,O,N,Z
				Indirecto	17	C,O,N,Z
				Inmediato	10	C,O,N,Z
				Relativo al PC	13	C,O,N,Z
				Registros	10	C,O,N,Z
7	Negación aritmética	**NEG**		ÚNICO	7	

INSTRUCCIONES DE OPERACIONES LÓGICAS						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
8	Operación lógica AND	**AND**	**1**	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z

				Relativo a otros registros	13	N,Z
				Relativo al PC	13	N,Z
			2	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z
				Relativo al PC	13	N,Z
				Registros	10	N,Z
9	Operación lógica OR	**OR**	1	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z
				Relativo a otros registros	13	N,Z
				Relativo al PC	13	N,Z
			2	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z
				Relativo al PC	13	N,Z
				Registros	10	N,Z
10	Operación lógica XOR	**XOR**	1	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z
				Relativo a otros registros	13	N,Z
				Relativo al PC	13	N,Z
			2	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z
				Relativo al PC	13	N,Z
				Registros	10	N,Z
11	Operación lógica NOT	**NOT**		ÚNICO	6	
INSTRUCCIONES DE COMPARACIÓN						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
12	Comparación aritmética	COMP	1	Directo	13	N,Z
				Indirecto	17	N,Z
				Inmediato	10	N,Z
				Relativo a otros registros	13	N,Z
				Relativo al PC	13	N,Z
INSTRUCCIONES DE CONTROL DE FLUJO DE PROGRAMA						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
A.	SALTOS INCONDICIONALES					
13	Salto incondicional con dirección absoluta	JMP	1	Directo	13	
				Indirecto	17	
				Registros	10	
				Postincremento	12	
				Predecremento	12	
14	Salto incondicional con dirección relativa al PC	JRPC	1	Directo	13	
				Indirecto	17	
				Registros	10	
15	Salto incondicional a	JSR	1	Directo	18	
				Indirecto	22	

	subrutina con dirección absoluta			Registros	15	
				Postincremento	17	
				Predecremento	17	
16	Salto incondicional a subrutina con dirección relativa al Program Counter	JRSR	1	Directo	18	
				Indirecto	22	
				Registros	15	
17	Retorno de Subrutina	RSR		ÚNICO	11	
B.	SALTOS CONDICIONALES					
18	Salto condicional si es igual a cero	BEQ	1	UNICO	7	
19	Salto condicional si NO es igual a cero	BNE	1	UNICO	7	
20	Salto condicional si hay carry	BSC	1	UNICO	7	
21	Salto condicional si NO hay carry	BNC	1	UNICO	7	
22	Salto condicional si hay overflow	BSO	1	UNICO	7	
23	Salto condicional si NO hay overflow	BNO	1	UNICO	7	
24	Salto condicional si es mayor	BGT	1	UNICO	7	
25	Salto condicional si es menor	BLT	1	UNICO	7	
26	Salto condicional si es menor o igual	BLE	1	UNICO	7	
27	Salto condicional si es mayor o igual	BGE	1	UNICO	7	
INSTRUCCIONES DE ROTACIÓN Y DESPLAZAMIENTO						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
28	Corrimiento lógico hacia la derecha	CLD	**1**	Directo	14	
				Indirecto	17	
				Inmediato	11	
				Relativo a otros registros	14	
				Relativo al PC	14	
				Registros	11	
29	Corrimiento lógico hacia la izquierda	CLI	**1**	Directo	14	
				Indirecto	17	
				Inmediato	11	
				Relativo a otros registros	14	
				Relativo al PC	14	
				Registros	11	
30	Corrimiento aritmético hacia la derecha	CAD	**1**	Directo	14	
				Indirecto	17	
				Inmediato	11	

				Relativo a otros registros	14	
				Relativo al PC	14	
				Registros	11	
31	Corrimiento aritmético hacia la izquierda	CAI	**1**	Directo	14	
				Indirecto	17	
				Inmediato	11	
				Relativo a otros registros	14	
				Relativo al PC	14	
				Registros	11	
32	Rotación a la izquierda	RIZ	**1**	Directo	14	
				Indirecto	17	
				Inmediato	11	
				Relativo a otros registros	14	
				Relativo al PC	14	
				Registros	11	
33	Rotación a la derecha	RDR	**1**	Directo	14	
				Indirecto	17	
				Inmediato	11	
				Relativo a otros registros	14	
				Relativo al PC	14	
				Registros	11	
INSTRUCCIONES PARA ATENCIÓN DE INTERRUPCIONES						
	INSTRUCCIÓN	**ABREVIACIÓN**	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
34	Borrar bandera de interrupción	BBI		UNICO	7	
35	Levantar bandera de interrupción	LBI		UNICO	7	
36	Retorno de interrupción	RIN		UNICO	18	
INSTRUCCIONES PARA MODIFICAR LA BANDERA DE CARRY						
		INSTRUCCIÓN	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
37	Borrar bandera de Carry	BBC		UNICO	6	
38	Levantar la bandera de Carry	LBC		UNICO	6	
INSTRUCCIONES PARA MANEJO DE PILA						
		INSTRUCCIÓN	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
39	Introducir en la pila	INP		UNICO	11	
40	Extraer de la pila	EDP		UNICO	12	
INSTRUCCIONES DE PROTOCOLO						
		INSTRUCCIÓN	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
A.	Generales					
41	Empezar Tarea	SET		UNICO	7	
42	Reconocimiento del Slave	SRS		UNICO	7	
43	Borrar señal de empezar tarea	BET		UNICO	7	

44	Borrar señal de reconocimiento del Slave	BRS		UNICO	7	
45	Terminar Tarea	STT		UNICO	7	
46	Reconocimiento del Master	SRM		UNICO	7	
47	Borrar señal de terminar tarea	BTT		UNICO	7	
48	Borrar señal de reconocimiento del Master	BRM		UNICO	7	
B.	Saltos de protocolo					
49	Salto cuando si hay bandera de empezar tarea ET	SBET		UNICO	7	
50	Salto si no hay bandera de empezar tarea ET	SNET		UNICO	7	
51	Salto si hay bandera de reconocimiento slave RS	SBRS		UNICO	7	
52	Salto si no hay bandera de reconocimiento slave RS	SNRS		UNICO	7	
53	Salto si hay bandera de terminar tarea TT	SBTT		UNICO	7	
54	Salto si no hay bandera de terminar tarea TT	SNTT		UNICO	7	
55	Salto si hay bandera de reconocimiento master RM	SBRM		UNICO	7	
56	Salto si no hay bandera de reconocimiento master RM	SNRM		UNICO	7	
INSTRUCCIONES DE SISTEMA						
		INSTRUCCIÓN	**CANTIDAD DE OPERANDOS**	**MODOS DE DIRECCIONAMIENTO**	**CICLOS DE RELOJ**	**BANDERAS**
57	No hacer ninguna operación	NOP		UNICO	6	
58	Detener	HLT		UNICO	7	

Tabla 2 Resumen del repertorio de instrucciones

2.7 CARACTERÍSTICAS ESPECIALES CUANDO SE CONECTAN DOS PROCESADORES DEL MISMO NÚCLEO

Para este proyecto se limitó a la conexión de sólo dos procesadores pero con posibilidad de extender a más conexiones. Sin embargo, es necesario tener en cuenta algunas características que deben tener en cuenta tanto el diseñador (para ampliar la cantidad de conexiones), como el usuario (cuando esté realizando algún programa) al utilizar está configuración.

2.7.1 Dirección de arranque para cada uno de los procesadores

Cuando se tienen dos procesadores conectados, uno de los procesadores, tiene su dirección de arranque para el *Program Counter* en la dirección 00000001h y el otro la tiene en la posición 00000003h.

Por esta razón al usuario se le sugiere utilizar cualquier instrucción de salto (Ver Manual Usuario) como primera instrucción de sus programas, que se ubica en 00000001h, ya que si esto no se hace es posible pasar por encima de las direcciones 00000003h y 00000002h que ya han sido reservadas para el manejo del otro procesador.

2.7.2 Manejo de interrupciones y de la pila para dos procesadores

Anteriormente, en la sección 2.3 se explicó el manejo de interrupciones para un solo procesador y como es el manejo de la pila LIFO en memoria. Cuando se tienen dos procesadores, cada uno debe tener un autovector asociado, uno está en la posición 00000000h y el otro está en el 00000002h. El usuario es el encargado de colocar la dirección de inicio de la rutina de interrupción de cada uno dejando la posibilidad que cualquiera de los dos procesadores pueda ser interrumpido.

De esta forma, cuando hay dos procesadores no hay ningún sistema de prioridad y es cuestión del usuario si desea que alguno de los procesadores sea interrumpido. Al usuario se le sugiere utilizar una instrucción de salto como primera instrucción de los programas para que no poner instrucciones en posiciones de memoria reservadas para el otro procesador.

Teniendo dos procesadores utilizando la misma memoria es posible que cada uno maneje su propia pila tipo LIFO (ver sección 2.3) o utilizar una para cada uno ya que el usuario es el encargado de colocar la dirección inicial que se almacena en el *Stack Pointer*.

2.8 PROTOCOLO DE COMUNICACIÓN ENTRE PROCESADORES DEL MISMO NUCLEO

Una de las características interesantes de este procesador es que contiene un conjunto de señales e instrucciones para realizar tareas en paralelo. El mecanismo de comunicación de esta interfaz básica es llamado comúnmente como *handshake*. El diagrama de bloques general muestra las entradas y salidas de un solo procesador, ver sección 2.1, para el *handshake*, de las 8 señales se utilizan solo 4 para cada procesador, dependiendo de cómo se trabaje ya sea como *Master* (maestro) o como *Slave* (esclavo). A continuación en la figura 10 se muestra un diagrama de bloques donde están conectados dos procesadores

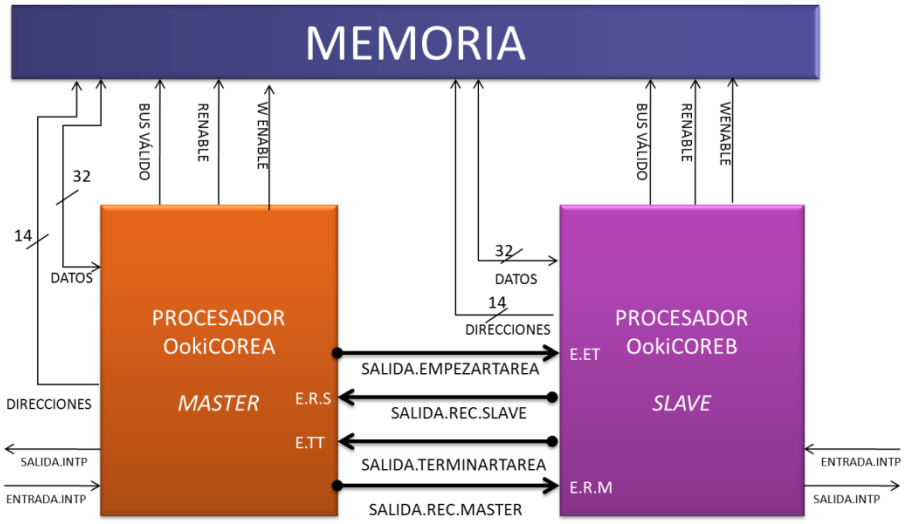

Figura 11. Diagrama de bloques conexión entre dos núcleos iguales

En el esquema de la figura 11, se maneja una memoria tipo *True dual Port* la cual tiene dos puertos síncronos independientes de lectura y escritura. Está se implementó en este trabajo dentro de la FPGA *Cyclone III*, pero es posible utilizar otros dispositivos de lógica programable. Por esta razón, se modeló como una sola pero depende del usuario la conexión de los núcleos y la cantidad de memorias que desea utilizar.

Cuando se realiza un programa que pueda utilizar el procesamiento en paralelo, si está utilizando esta configuración de la memoria embebida, cómo la que se usó en este proyecto, es necesario evitar la escritura por los dos puertos simultáneamente.

A continuación se mostrará la definición de un protocolo en 6 etapas utilizando como referencia el diagrama de tiempos mostrada en la figura 12. Es necesario aclarar que todo depende de las instrucciones utilizadas dentro del programa realizado por el usuario, tanto el orden como la lógica, aquí se presentará una manera de realizar el paralelismo.

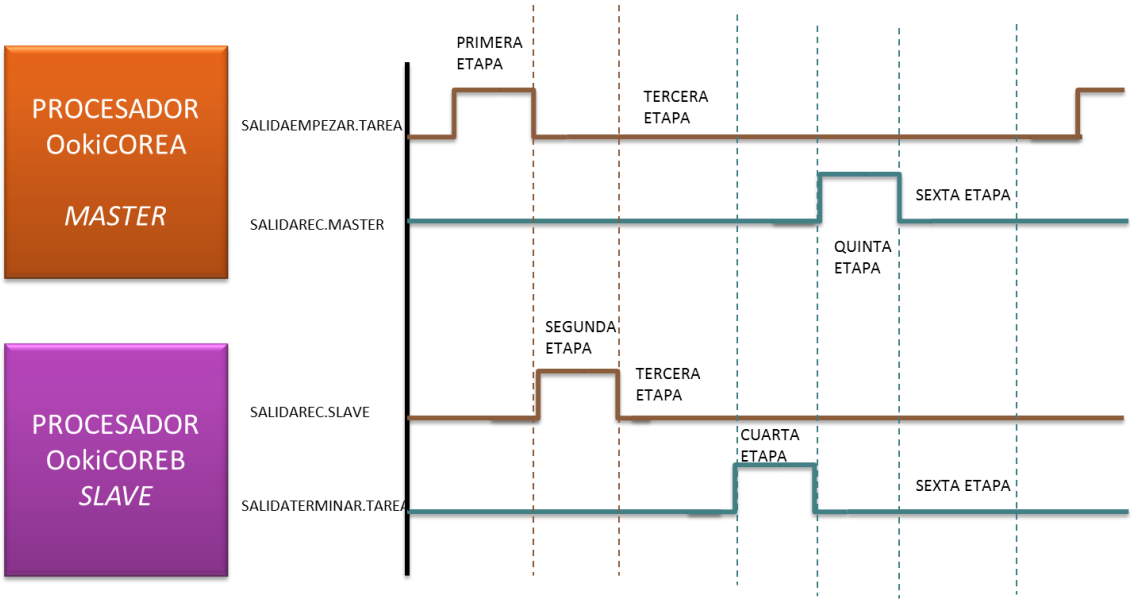

Figura 12.Diagrama de tiempos del protocolo de comunicación

Primera Etapa (Master a Slave)

Se está realizando una rutina, el *master* le da la orden al *Slave* para qué comience su trabajo. Se genera y se asigna el valor de 1 a la señal SALIDA.EMPEZARTAREA del *master* como se muestra en la figura 12.

Segunda Etapa (Slave a Master)

Cuando el *slave* reconoce que hay un 1 en su señal de entrada llamada ENTRADA.EMPEZARTAREA (E.ET) se debe activar la señal de salida llamada SALIDA.REC.SLAVE como se muestra en la figura 12. Esta es una forma de avisar que ya va a empezar a trabajar la rutina.

La señal ENTRADA.EMPEZARTAREA asigna un 1 a la bandera ET del registro de estado del *slave*.

Tercera Etapa (Master a Slave y Slave a Master)

El *master* reconoce un 1 en su señal ENTRADA.REC.SLAVE (E.R.S), luego el *master* asigna un valor de cero a la señal SALIDA.EMPEZARTAREA como se muestra en la figura 12.

La señal ENTRADA.REC.SLAVE asigna el valor de 1 a la bandera RS del registro de estado del *master*.

Cuando el *slave* reconoce '0' en la entrada ENTRADA.EMPEZARTAREA, él también debe asignar el valor de '0' a su señal SALIDA.REC.SLAVE.

La señal ENTRADA.EMPEZARTAREA asigna un 0 a la bandera ET del registro de estado del *slave*.

Cuarta Etapa (Slave a Master)

Cuando el *slave* termine el trabajo asignado, debe informarle esto al *master* para que le sea asignado otro trabajo o simplemente termine su tarea. El *slave* le asigna un 1 a la señal SALIDA.TERMINARTAREA.

Quinta Etapa (Master a Slave)

Cuando el *master* reconoce que hay un 1 en su señal de entrada llamada ENTRADA.TERMINARTAREA (E.TT en la figura 12). Se debe generar una señal de salida llamada SALIDA.REC.MASTER. Esta es una forma de avisar, que ya se percató que el s*lave* ha terminado.

La señal ENTRADA.TERMINARTAREA asigna el valor de 1 a la bandera TT del registro de estados del *master*

Sexta Etapa (Master a Slave y Slave a Master)

El *Slave* reconoce un '1' en su señal ENTRADA.REC.MASTER (E.R.M) como se muestra en la figura 12, este asigna un valor de cero a su señal SALIDA.TERMINARTAREA.

La señal ENTRADA.REC.MASTER asigna un 1 a la bandera RM del registro de estado del *slave*.

Cuando el *Master* reconoce el '0' en la entrada ENTRADA.TERMINARTAREA (E.TT) también le asigna el valor de 0 a su señal SALIDA.REC.MASTER. La señal ENTRADA.TERMINARTAREA asigna un 0 a la bandera TT del registro de estado del *master*.

3. DESARROLLO

Para el diseño de este procesador se utilizó la metodología que se ha desarrollado en las asignaturas de la sección de técnicas digitales, Diseño de Sistemas Digitales y Arquitectura de Procesadores.
La metodología de diseño está compuesta por los siguientes pasos:

- Descripción general del sistema
- Interfaz de Entrada/Salida del sistema
- Descripción de las señales de entrada y de salida del sistema
- Planteamiento del diagrama de bloques
- Descripción de cada bloque
- Descripción del comportamiento del *hardware* utilizando AHPL
- Obtención de los circuitos esquemáticos.
- Realización de las tablas de conectividad
- Descripción del *hardware*, utilizando VHDL
- Escogencia del dispositivo donde se desea implementar el proyecto

En el capítulo anterior se muestran los primeros 5 pasos de la metodología de diseño y ahora se van a describir los siguientes pasos. Aparte de la metodología es importante nombrar en el desarrollo de este trabajo de grado, la escogencia de los programas realizados, las instrucciones y los modos de direccionamiento.

3.1 ESCOGENCIA DE PROGRAMAS

Un objetivo de este proyecto es diseñar e implementar un procesador que tenga la capacidad de interconectarse con varias instancias del mismo y para cumplirlo fue necesario diseñar programas especiales para esto. El primer paso que se realizó fue escoger problemas aritméticos sencillos que pudieran utilizarse en el manejo del protocolo anteriormente mencionando en el capítulo 2. De este paso salieron los tres siguientes programas,

a. ¿Cómo obtener lo números primos del 1 al 100?
b. Promedio de notas
c. Multiplicación de matrices

Debido a la complejidad del último programa, este no se implementó utilizando el OokiCore . De los primeros dos programas se realizaron dos diagramas de flujo (Ver Anexo H), uno para un solo procesador y el otro para dos procesadores en configuración *Master/Slave*. Teniendo los diagramas de flujo y utilizando las instrucciones de BINARIC[1] se realizaron los programas para un solo procesador. Para probar que los programas funcionaran se utilizó PCSIM.

También, con ayuda de estos, se definieron las instrucciones y los modos de direccionamiento para el procesador.

3.2 AHPL

El AHPL de este procesador es la guía principal para definir el funcionamiento del todo el sistema. El punto de partida del AHPL fue el estado inicial en el cual debe comenzar el

procesador, en donde se asignan los valores iniciales de cada una de las señales descritas en la interfaz entrada/salida de todo el sistema (ver Capítulo 2).

Después de esto se pensó en la mejor estrategia, para darle organización a todo el sistema y con esto no repetir pasos que pueden ser común en la mayoría de procesos. En la figura 20 está el diagrama de jerarquía de la construcción del AHPL.

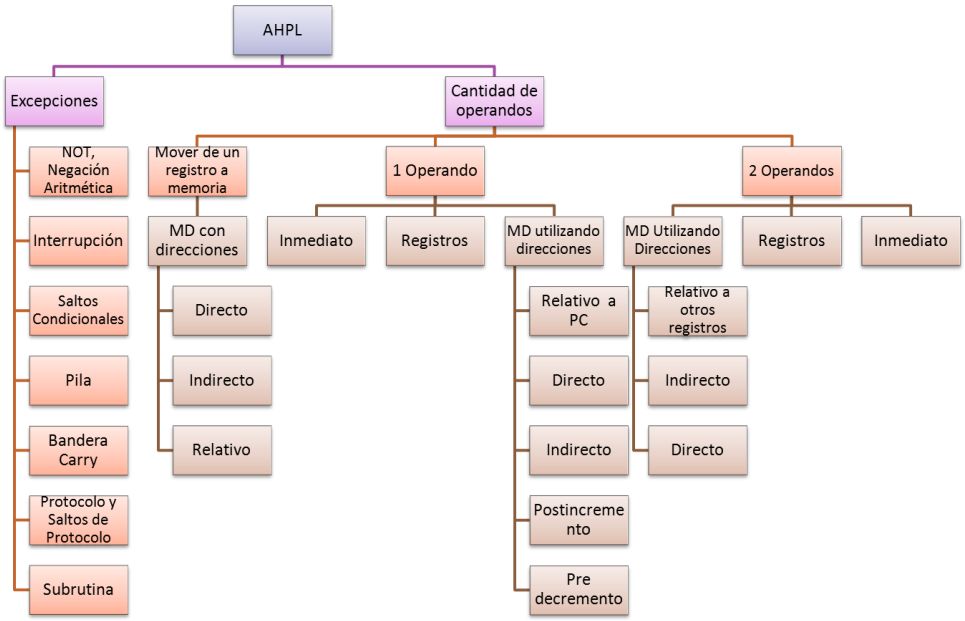

Figura 13. Diagrama de Jerarquía del AHPL

En el desarrollo del AHPL, primero se realizaron las instrucciones que denotaban excepciones, ya que no tiene ningún modo de direccionamiento. Luego se realizaron las instrucciones que tienen dos operandos porque la cantidad de instrucciones es menor a las que tienen un operando. Por último se tomaron las instrucciones de un solo operando ya que estas eran más y algunas manejaban modos de direccionamiento que las de dos operandos no manejaban, como pre decremento y pos incremento.

Del AHPL se obtiene la cantidad de estados de la máquina de control *"one hot"*, que en este caso es de 130 estados. El AHPL documentado está en el Anexo A.

3.2. CIRCUITOS ESQUEMÁTICOS

El siguiente paso es realizar todos los esquemáticos, los cuales se pueden obtener a partir del AHPL y del diagrama de bloques. Utilizando los esquemáticos se pudo realizar el VHDL con más facilidad sobre todo la lógica de entrada de algunos registros, la ALU, el Registro de Corrimiento y los RPG. En el Anexo B se encuentran estos esquemáticos.

3.3 TABLAS DE CONECTIVIDAD

Las tablas de conectividad son una herramienta que relaciona los nombres de las señales de la entidad genérica y el nombre dado en cada instancia en VHDL. Estas conectan cada una de las señales de entrada/salida de cada bloque.

Estas tablas, facilitan la codificación de la descripción del sistema en el lenguaje VHDL, debido a que los nombres genéricos asignados a las entradas y salidas de una entidad son diferentes a los que se asignan a cada una de las instancias de esa misma entidad. Si el número de instancias se incrementa lo mismo que el número de entidades, los nombres asignados a sus entradas y salidas se hace tan grande que puede llevar a confusiones difíciles de corregir.

En el Anexo C se encuentran las tablas de conectividad que se realizaron para un solo procesador y cuando se conectan dos procesadores.

3.4 VHDL

El VHDL se realizó utilizando todos los elementos que se explicaron anteriormente, se utilizó el diagrama de bloques, el AHPL, los circuitos esquemáticos y las tablas de conectividad. Para un solo procesador, el VHDL se realizó de forma jerárquica donde la entidad principal está compuesta por tres entidades, el procesador, la memoria (*Single Port)* y el divisor de frecuencia.

Luego se describió la arquitectura del procesador y si se desea dentro de la herramienta Quartus II al hacer click sobre la entidad procesador, se puede observar el diagrama de bloques del sistema tal cual como aparece en la figura 2. Los registros utilizados dentro de esta entidad cargan los datos con el borde de subida. Después se empezaron a desarrollar las diferentes entidades que conforman el diagrama de bloques, empezando por las más sencillas.

Luego se realizaron los bloques de más complejidad como el RPG, el registro de corrimiento, la ALU y por último la máquina de control. Las lógicas combinatorias de entrada fueron bloques adicionales en VHDL, que gracias a los circuitos esquemáticos, su paso al código fue realizado con gran facilidad. Utilizando las tablas de conectividad, se conectó el procesador estando listopara pruebas. En la figura 14 se muestra el diagrama de jerarquía para el VHDL

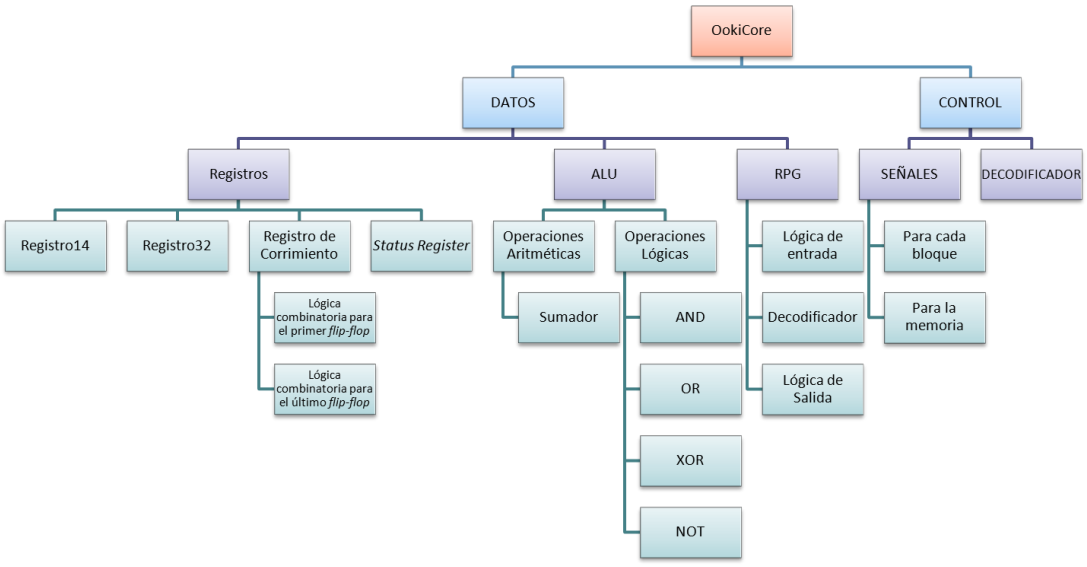

Figura 14 Diagrama de Jerarquia para el VHDL

Para la conexión de los dos procesadores, también hay una entidad principal donde se encuentran los dos procesadores, la memoria (*True Dual Port)* y el divisor de frecuencia. Se realizaron dos lógicas de entrada para el *Stack Pointer,* ya que era necesario modificar la dirección de arranque para alguno de los procesadores. También se utilizaron las tablas de conectividad para conectarlos.

3.5 PROTOCOLO DE PRUEBAS

En este protocolo se mostrarán las diferentes pruebas que se realizaron para la verificación del funcionamiento del procesador, tanto para uno solo como para dos procesadores conectados. Se muestran las pruebas que se realizaron en el orden expuesto en el diagrama de jerarquía del VHDL (figura 19).

Se probaron los bloques generales en VHDL en el siguiente orden:

a. Registros de 32 y 14 bits los cuales cargan el dato con el borde de subida del reloj.
b. ALU:
   - La escogencia del sumador se realizó haciendo una simulación de 3 posibles configuraciones. El primero un sumador '+' que se encuentra en la biblioteca IEEE.std_logic_unsigned.all, el cual utiliza las cadenas de *carry* dentro de la FPGA. El segundo un sumador de rizado realizado con *full adder*, hecho con compuertas XOR. El tercero un sumador *carry-look ahead*. Después de las simulaciones se tomó la decisión por el primero ya que el retardo fue menor y utiliza menos elementos lógicos.
   - Se probaron los bloques de operaciones lógicas: XOR, OR, NOT y AND .
   - Se probó la lógica de salida de la ALU, la bandera de *zero*, la bandera de negativo, la bandera de *carry* y la bandera de *overflow*.

c. Memorias: Se realizaron diferentes programas para verificar el tiempo de respuesta tanto para la lectura como para la escritura de datos. Además se realizaron programas de lectura y de escritura por los dos puertos en la misma posición de memoria de forma simultánea para probar las memorias en configuración *True Dual Port*.

d. Máquina de Control: se realizó la máquina *one hot* de 130 pasos de AHPL. Se probó sin las conexiones externas, solo para verificar que por cada borde de bajada del reloj siguiera al paso que le correspondía. A continuación se realizaron 15 Pruebas (Anexo E) con diferentes instrucciones para verificar que cada instrucción si realizara su respectiva rutina. También se probaron todos los modos de direccionamiento para los dos tipos de formato de instrucción (un operando y dos operandos).

   La métrica de cobertura que se utilizó fue FSM coverage (Finite state machine coverage), donde se probó que todos los estados fueron estimulados utilizando todas las posibles rutas al utilizar el set de instrucciones con todos los modos de direccionamiento.

e. Bloque RPG: se probó la lógica de entrada, el control RPG y la lógica de salida por aparte luego se probó nuevamente cuando se unió en una sola entidad llamada RPG. Además, como los registros AC y SP tienen un uso especial se probó el funcionamiento de estos a través de las pruebas realizadas en el Anexo F.

f. Lógicas de Entrada: para los bloques RDatos, RDirecciones,PC, los dos operandos de la ALU, se hicieron diferentes selectores los cuales son manipulados por señales de control. Se verificó el correcto funcionamiento pero solo al final se conectaron a los registros respectivos y se hicieron los programas para un solo procesador (Anexo F).

g. Status Register: cuando se hicieron las pruebas de la ALU, se probó que las banderas de *zero*, negativo, *carry* y *overflow* estuvieran recibiendo el dato verdadero. El resto de las banderas se probaron con los programas realizados para un solo procesador (Anexo F)

Prueba de todo el procesador conectado (OokiCore con memoria)

Se conectaron todos los bloques del procesador y se realizaron 7 pruebas de pequeños programas (Anexo F), los cuales tienen diferentes instrucciones y se utilizan diferentes modos de direccionamiento. Hubo programas en los cuales se probaron instrucciones que no tienen modo de direccionamiento. Se utilizó una memoria de un solo puerto para todas las pruebas.

Teniendo los programas realizados y luego de verificar el correcto funcionamiento del procesador se simularon los programas propuestos para un solo procesador (Ver sección 3.1). Se probaron los dos programas propuestos verificando el correcto funcionamiento de estos.

Cuando se verificaron las simulaciones se implementó el procesador en la FPGA y por medio de salidas de observabilidad, la salida de RDatos y la salida de RDirecciones, se probó el funcionamiento del procesador utilizando el analizador lógico Tecktronix TLA5201 comparando los resultados con las simulaciones realizadas.

Prueba de los dos procesadores conectados (OokiDualCore con memoria)

Se conectaron dos procesadores con una memoria de doble puerto (*True Dual Port*) embebida dentro de la FPGA. Luego de esto se realizaron 2 pruebas de pequeños programas, los cuales tienen diferentes instrucciones y se utilizan diferentes modos de direccionamiento. Se probó cada uno con un programa diferente sin que hubiera comunicación entre ellos y otro que si involucró las señales de protocolo.

Luego de esto se probaron los dos programas propuestos (Ver sección 3.1) verificando el correcto funcionamiento de estos primero en simulación y luego al Implementar el procesador en la FPGA con las memorias embebidas.

Por medio de salidas de observabilidad, la salida de RDatos y la salida de RDirecciones, para cada uno de los procesadores, se comprobó el funcionamiento de los dos procesadores utilizando el analizador lógico Tecktronix TLA5201 y se comparó con las simulaciones realizadas. Con esto se verificó que las señales de protocolo de verdad estén comunicando a los dos núcleos.

Pruebas con periféricos

Para el manejo de interrupciones del procesador se utilizó como periférico un pulsador de la tarjeta de desarrollo *Cyclone III Starter Board*. Se utilizó un led, donde su estado inicial era de encendido, si este se apagaba esto se significaba que el pulsador sí había interrumpido al procesador. El programa realizado con interrupciones se encuentra en el Anexo F.

3.6. TARJETA DE DESARROLLO Y FPGA

Para este trabajo se escogió una FPGA *Cyclone III* EP3C25F324 utilizando las memorias embebidas que podían manejar bloques de máximo 512*16 en configuración *True Dual Port.*

Los elementos principales de esta FPGA son:

- Cantidad de elementos lógicos: 26,624
- Cantidad de bloques de memoria M9k: 66
- Total RAM bits: 608,256
- Multiplicadores 18 x 18: 66
- PLL's:4
- Redes de reloj global :20
- Pines E/S: 215

De estos elementos generales se utilizaron 64 bloques de memoria de M9k que equivalen a 524288 bits, o visto de otra forma una memoria de 16k palabras de 32 bits. Para un solo procesador se utilizaron 1746 elementos lógicos y 98 pines E/S. Para los dos procesadores conectados se utilizaron 3519 elementos lógicos y 114 pines E/S.

La tarjeta de desarrollo se llama *Cyclone III Starter Board*, el cual se muestra en la figura 20

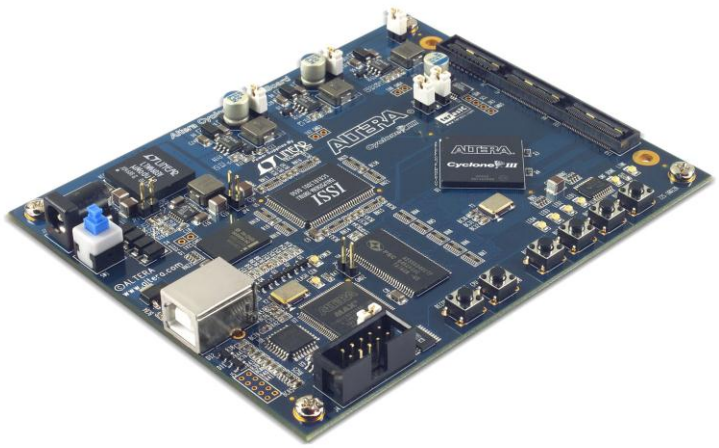

Figura 15 Tarjeta de desarrollo Cyclone III Starter Board

Los componentes principales de la tarjeta de desarrollo son:

- *Cyclone III* EP3C25F324
- Sistema de manejo de reloj
    - Un oscilador de 50MHz
    - La cylone III distribuye los siguientes relojes a través de sus PLL's:
        - DDR *clock*
        - SSRAM *clock*
        - Flash *clock*
- Conector HSMC: Provee 12 V y 3.3 V a las tarjetas de conexión. Tiene 84 pines de E/S de comunicación a las tarjetas de expansión para este puerto.
- Para la interfaz del usuario tiene : 4 LED's , 2 LED's específicos para la tarjeta
  6 botones: uno de Reset del sistema, otro de reset para el usuario y 4 botones de uso general.
- Subsistemas de memoria

    - SRAM Sincrona: 1 Mbyte, 167 MHz y comparte un bus con el dispositivo flash en paralelo.
    - Parallel Flash Device: 16 Mbyte para configuración en paralelo y almacenamiento.
    - DDR SDRAM: 56 pin, 32M byte DDR SRAM, 167 MHZ y conectada a la FPGA por un bus dedicado de 16 bits.
- Interfaz USB-Blaster
    - CPLD EPM3128A
    - Para configuración externa del dispositivo *Cyclone III*

Todas estas especificaciones se pueden encontrar en el manual de la tarjeta de desarrollo y de la FPGA que estará Anexo al este trabajo.

De los componentes anteriormente mencionados, se utilizaron para este proyecto: la Cyclone III, el oscilador de 50 MHz para poder hacer el divisor de frecuencia a la mitad y lograr los 25 MHz del reloj principal del procesador, el conector HSMC, 1 LED de prueba para las interrupciones y para las pruebas en paralelo, un botón para el reset y otro para simular la interrupción y por último el USB Blaster para poder configurar la FPGA.

Esta tarjeta de desarrollo tiene un conector HSMC (*High Speed Mezzazine Connector)* al cual se le puede conectar diferentes periféricos. Sin embargo, se utilizó una tarjeta de expansión, hecha por la empresa TERASIC que se muestra en la figura 21, ya que era imposible la conexión del conector HSMC a los POD del analizador lógico utilizado para las pruebas de *hardware*.

Figura 16 Tarjeta de desarrollo

4. ANÁLISIS DE RESULTADOS

Antes de hacer un análisis se va a a resumir los resultados de las pruebas que están en los Anexos F y G para el procesador OokiCore y cuando se conectan los dos procesadores (OokiDualCore). Luego de esto se van a hacer los análisis de recursos, tiempo y potencia utilizando los reportes dados por QUARTUS II 9.1.

4.1. RESULTADOS PRUEBAS REALIZADAS CON EL PROCESADOR

En todas las pruebas que se realizaron, se puede observar el correcto funcionamiento del procesador. Los programas fueron variados así que se tuvo la oportunidad de ver varias de las instrucciones utilizando diferentes modos de direccionamiento. Para las pruebas con el analizador lógico, sólo se extrajeron la señal de entrada al procesador, llamada DATOS y el bus de DIRECCIONES ya que no alcanzó el número de pines en la tarjeta de expansión (Figura 21).

La figura 22 es una imagen obtenida del analizador, mostrando una de las instrucciones que se probaron con el procesador. En la imagen se puede ver una instrucción de salto (7C000001) en la dirección de memoria 0000000Fh que hace que el procesador salte a la dirección 00000001h. El resto de pruebas se encuentran en los Anexos F y G con los respectivos programas que se realizaron.

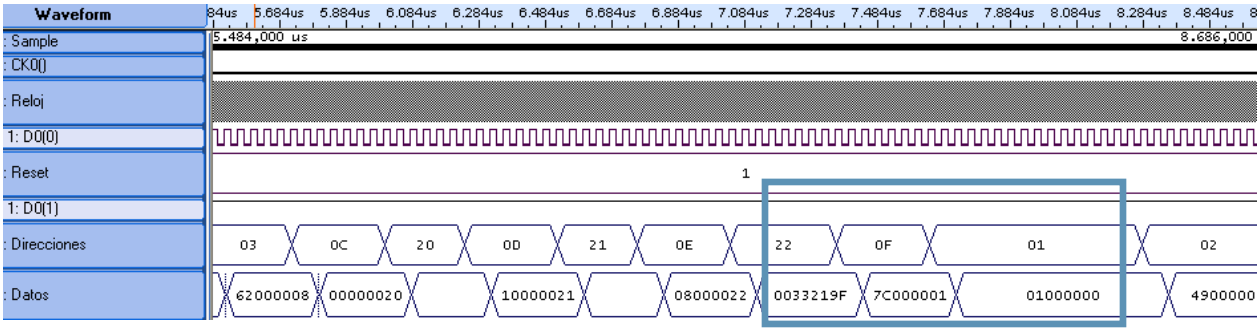

Figura 17 Resultado de una prueba donde se muestra un salto absolut

La figura 23 muestra parte del proceso que sigue el programa propuesto de los números primos cuando están dos procesadores conectados en paralelo. Esta parte es la primera y segunda etapa del protocolo de comunicación que se explicó en la sección 2.8

Para el *Master*, se puede observar como en la salida SALIDAEMPEZARTAREA se activa utilizando la respectiva instrucción y para deshabilitarla espera hasta que se genere la señal de respuesta en el *Slave*.

o

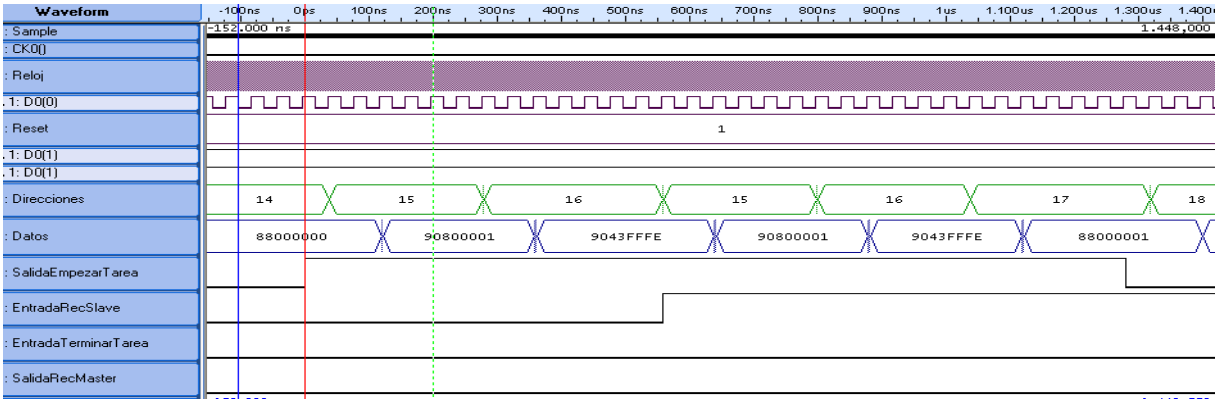

Figura 18 Habilitación de la señal de salida SALIDAEMPEZARTAREA

4.2. ANÁLISIS PARA UN PROCESADOR

Para el análisis de recursos, de tiempos y de potencia se utilizó el programa QUARTUS II 9.1, el cual suministra los datos necesarios en sus reportes de compilación. Además, se utilizó el programa propuesto de la búsqueda de los primos (Ver Anexo H) aunque hubiera podido escoger cualquiera porque el programa es independiente del *hardware* del procesador.

Los ajustes para la compilación, que se utilizaron para realizar los diferentes análisis son los siguientes:

- Características de potencia del dispositivo: Normal
- Early Timing Estimate: Realistic
- Physical Synthesis Optimizations: Effort Level: Normal
- Técnica de Optimización: Balanceada
- PowerUp power optimization: Normal
- VHDL Version : VHDL 1993
- Fitter Effort: AutoFit
- Timing Analysis Settings: Se utilizó el *Classic Time Analizer*

4.2.1. Análisis de Recursos

Este análisis indica en la cantidad de recursos de la FPGA que se utilizaron en el procesador y en la figura 24 se indica el primer reporte que entrega QUARTUS II al terminar de compilar.

Flow Summary

Flow Status	Successful - Wed Jul 21 05:13:42 2010
Quartus II Version	9.1 Build 304 01/25/2010 SP 1 SJ Web Edition
Revision Name	OokiCoreMemoria2
Top-level Entity Name	OokiCoreMemoria2
Family	Cyclone III
Device	EP3C25F324C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1,746 / 24,624 (7 %)
Total combinational functions	1,505 / 24,624 (6 %)
Dedicated logic registers	803 / 24,624 (3 %)
Total registers	803
Total pins	98 / 216 (45 %)
Total virtual pins	0
Total memory bits	524,288 / 608,256 (86 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 19 Reporte de la compilación en QUARTUS II 9.1

Este primer reporte muestra la cantidad de elementos lógicos que se utilizaron, el cual es 1746 de 24624, el 7% de la FPGA. Los recursos que sobran son una ventaja ya que el usuario puede utilizarlos para lo que necesite, ya sea para una aplicación que utilice el procesador o para conectar más de dos procesadores. Además, se debe recordar que una de las razones para escoger la FPGA, fue porque tiene los bloques de memorias embebidas M9K.

Los pines que se utilizan son 98 de 216 los cuales se distribuyen entre el bus de DIRECCIONES (14 bits), las 8 señales de protocolo, el reloj, el reset, la SENALWAIT, el bus bidireccional de DATOS que se toman como dos buses cada uno de 32 bits, la salida SALIDAHLT, BUS VALIDO, RENABLE, WENABLE y por último las señales de observabilidad del reset, del reloj y de la interrupción que se utilizaron como *Trigger* en el analizador lógico, dependiendo del caso.

El segundo reporte de la figura 25 muestra la cantidad de elementos lógicos que se utilizan por dos de los tres modos de operación de los elementos lógicos que maneja la FPGA, normal, contador y aritmético. En modo normal hay 1467 elementos lógicos, los cuales están repartidos en registros y las lógicas combinatorias que están en el procesador. Del modo aritmético hay 37 elementos lógicos, los cuáles se refieren al sumador que se implementó en la ALU ya que utiliza las cadenas de *carry* de la FPGA

Analysis & Synthesis Resource Usage Summary

	Resource	Usage
1	Estimated Total logic elements	2,041
2		
3	Total combinational functions	1504
4	⊟ Logic element usage by number of LUT inputs	
5	-- 4 input functions	1057
6	-- 3 input functions	336
7	-- <=2 input functions	111
8		
9	⊟ Logic elements by mode	
10	-- normal mode	1467
11	-- arithmetic mode	37
12		
13	⊟ Total registers	803
14	-- Dedicated logic registers	803
15	-- I/O registers	0
16		
17	I/O pins	98
18	Total memory bits	524288
19	Maximum fan-out node	RELOJ:RelojOokiCore|Q
20	Maximum fan-out	868
21	Total fan-out	9199
22	Average fan-out	3.58

***Figura 20**. Reporte general de los recursos utilizados*

El número total de registros utilizados es de 803 registros lógicos dedicados. Este número debe coincidir con el número de registros que se calculó en el diseño, los cuales se muestran en la tabla 3:

NOMBRE DEL REGISTRO	NÚMERO DE REGISTROS
MÁQUINA DE CONTROL	130
REGISTRO DE DATOS	32
REGISTRO DE INSTRUCCIONES	32
REGISTRO DE DIRECCIONES	14
PROGRAM COUNTER	14
RPG	512
STATUS REGISTER	9
REGISTRO DE CORRIMIENTO	32

REGISTRO AUXILIAR	14
CONTADOR DE CORRIMIENTO	5
PROTOCOLO DE SALIDA	4
DIVISOR DE FRECUENCIA	1
TOTAL	799

***Tabla 3**. Número de registros utilizados para un solo procesador*

Los 4 registros que hacen falta son utilizados por la memoria que se está utilizando, en este caso una *Single Dual Port*. Esto se pudo ver en el siguiente reporte de la figura 26

Analysis & Synthesis Resource Utilization by Entity

	Compilation Hierarchy Node	LC Combinationals	LC Registers
1	⊟ |OokiCoreMemoria2	1504 (0)	803 (0)
2	⊞ |Memoria2:MemoriaOokiCore|	37 (0)	4 (0)
8	⊞ |OokiCore:OokiCoreM|	1466 (0)	798 (0)
69	|RELOJ:Reloj|OokiCore|	1 (1)	1 (1)

Figura 21. Reporte de la cantidad de recursos utilizados por cada entidad

Otra cosa que se analizó fue cuántos de estos registros tiene carga síncrona y cuantos tienen carga asíncrona, por lo tanto si se ve la figura 22, se puede observar

General Register Statistics

	Statistic	Value
1	Total registers	803
2	Number of registers using Synchronous Clear	0
3	Number of registers using Synchronous Load	5
4	Number of registers using Asynchronous Clear	130
5	Number of registers using Asynchronous Load	0
6	Number of registers using Clock Enable	661
7	Number of registers using Preset	0

***Figura 22**. Reporte de las cargas asíncrona y síncrona de los registros*

Los que tienen carga asíncrona utilizando *clear* son los 130 registros utilizados por la máquina de control. Los que utilizan el *Clock Enable* son 661 y se puede decir que son la mayoría de registros del procesador. En el reporte aparecen solo 5 de carga síncrona.

En la figura 29, también aparece el nodo de máximo *fan out* en este caso es el reloj. Esto tiene sentido ya que el reloj al ser global es el nodo que se propaga para todos los registros que tiene el procesador y la memoria. El máximo *fan out* es de 868, los cuáles están repartidos en los 803 registros mencionados y adicionalmente en los 63 bloques de memoria utilizados.

Multiplexer Restructuring Statistics (Restructuring Performed)

	Multiplexer Inputs	Bus Width	Bas... Area	Area if Restru...	Saving if Restructured	Example Multiplexer Output
1	3:1	4 bits	8 LEs	4 LEs	4 LEs	|OokiCoreMemoria2|OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:RM|Output
2	3:1	2 bits	4 LEs	2 LEs	2 LEs	|OokiCoreMemoria2|OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:Zero|Output
3	3:1	5 bits	10 LEs	5 LEs	5 LEs	|OokiCoreMemoria2|OokiCore:OokiCoreM|Counter5:CBOokiCore|cnt[4]

Figura 23, Multiplexores dentro del sistema

En la Figura 28, se muestran los multiplexores que el compilador de QUATUS infirió para la realización de algunos bloques como: el contador, la salida en el status register del *flip-flop* zero y la salida del *flip-flop* de Reconocimiento Master debido a la lógica combinatoria implementada.

4.2.2. Análisis de Tiempo

4.2.2.1. Ciclos de las instrucciones para un solo procesador

Para el análisis de los programas fue necesario contabilizar la cantidad de ciclos de reloj que se utilizan en cada instrucción. Los ciclos que se tienen en cuenta desde que se solicita la instrucción (*fetch*), cuando de esta decodificando (*decode*) y el proceso de ejecución(*execute*). La tabla 2 donde se encuentra el resumen del repertorio de instrucciones se encuentra este cálculo de los ciclos.

Para el análisis de tiempos es importante saber cuáles instrucciones son las que más se demora y las que menos se demoran. Esto se puede obtener utilizando los ciclos de reloj y la frecuencia de reloj utilizada en la implementación. Por tanto, si se trabajó a 25MHz, el periodo del reloj fue de 40ns.

Las instrucciones que se demoran más ciclos de reloj y también las instrucciones que se demoran menos se encuentran en la tabla 4

INSTRUCCIÓN	MODO DE DIRECCIONAMIENTO	CICLOS DE RELOJ	TIEMPO DE DURACIÓN
Operación lógica NOT	ÚNICO	6	0.240us
Borrar bandera de Carry	ÚNICO	6	0.240us
Levantar la bandera de Carry	ÚNICO	6	0.240us
No hacer ninguna operación	ÚNICO	6	0.240us
Salto incondicional a subrutina con dirección absoluta	Indirecto	22	0.880us
Salto incondicional a subrutina con dirección relativa al Program Counter	Indirecto	22	0.880us

Tabla 4 Ciclos de reloj de las instrucciones que más se demoran.

4.2.2.2. Análisis de tiempos utilizando QUARTUS II

En el resumen de compilación dado por QUARTUS II 9.1 se puedo sacar el siguiente reporte de tiempos para su análisis que se encuentra en la figura 24

Timing Analyzer Summary

	Type	Actual Time	From	To
1	Worst-case tsu	3.504 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[54]
2	Worst-case tco	16.347 ns	Memoria2:MemoriaOokiCore|altsyncram:altsyncram_compone...	DATOS[25]
3	Worst-case tpd	5.417 ns	EntradaIntP	OutInterrupcion
4	Worst-case th	1.555 ns	EntradaIntP	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[52]
5	Clock Setup: 'Clk1'	29.86 MHz (period = 33.488 ns)	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[6]	OokiCore:OokiCoreM|RPG:RPGOokiCore|RegistrosRPG:Registr...
6	Total number of failed paths			

Figura 24. Reporte del analizador de tiempos de QUARTUS II 9.1

Este reporte muestra los peores casos donde los tiempos de *setup* (t_{su}), *hold* (t_h), *clock to output* (t_{co}) y *pin- to –pin* (t_{pd}) al límite donde estos dejarían de ser válidos. En el reporte se puede ver que la frecuencia máxima es de 29.86MHz de reloj cuando se pasa desde el *flip-flop* 6 de la máquina de control al bloque RPG. Esta frecuencia corresponde a la máxima de todo el procesador, si se llegará a utilizar un valor mayor a este es probable que deje de funcionar algunos caminos entre señales y el procesador no funcione bien.

Los resultados para los tiempos fueron:

Tiempo de *setup*

El tiempo de *setup* es de 3.504ns, se da cuando la señal SENALWAIT pase al bus de DATA del *flip-flop* 54 de la máquina de control antes que haya señal de reloj en el pin *Clock* del mismo. A continuación en la figura 25 se mostrara un reporte que tiene los peores y los mejores casos para el t_{su}

tsu

	Slack	Required tsu	Actual tsu	From	To
1	N/A	None	3.504 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[54]
2	N/A	None	1.505 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[30]
3	N/A	None	0.927 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[125]
4	N/A	None	0.923 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[114]
5	N/A	None	0.901 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[53]
6	N/A	None	0.895 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[56]
7	N/A	None	0.754 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[31]
8	N/A	None	0.753 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[126]
9	N/A	None	0.750 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[115]
10	N/A	None	0.731 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[57]
11	N/A	None	0.380 ns	EntradaEmpezarTarea	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:ET|Output
12	N/A	None	0.201 ns	EntradaTerminarTarea	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:TT|Output
13	N/A	None	-0.056 ns	EntradaRecSlave	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:RS|Output
14	N/A	None	-0.094 ns	EntradaRecMaster	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:RM|Output
15	N/A	None	-1.363 ns	EntradaIntP	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[2]
16	N/A	None	-1.364 ns	EntradaIntP	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[52]

Figura 25. Reporte de tiempo de setup

El mejor tiempo es de -1.364ns que se da cuando la señal de entrada ENTRADAINTP pasa al bus de DATA del *flip-flop* 52 de la máquina de control antes que la señal de reloj llegue al pin de *Clock* al registro.

Tiempo de *clock to output*

En la figura 24 se puede observar que el mayor retardo *clock to output* es de 16.347ns, cuando un registro utilizado en la memoria saca un dato válido al bit 25 del bus de DATOS de entrada al procesador. En la figura 26 se muestra los diez mejores casos para este tiempo. El mejor tiempo es d 10.840ns cuando es válido el bit de la salida del *flip-flop* 30 del registro de datos y pasa al bus de DATOSSALIDA.

tco

	Actual tco	From	To
190	11.100 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[29]	DATOSSALIDA[29]
191	11.044 ns	OokiCore:OokiCoreM|ProtocoloSalida:SalidaProtocoloOokiCore|Fli...	SalidaRecSlave
192	11.041 ns	OokiCore:OokiCoreM|Registro14:RDireccionesOokiCore|Out14[5]	SalidaRDir[5]
193	10.990 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[21]	DATOSSALIDA[21]
194	10.988 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[11]	DATOSSALIDA[11]
195	10.971 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[28]	DATOSSALIDA[28]
196	10.960 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[20]	DATOSSALIDA[20]
197	10.917 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[13]	DATOSSALIDA[13]
198	10.916 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[17]	DATOSSALIDA[17]
199	10.910 ns	OokiCore:OokiCoreM|Registro14:RDireccionesOokiCore|Out14[0]	SalidaRDir[0]
200	10.840 ns	OokiCore:OokiCoreM|Registro32:RDatosOokiCore|Out32[30]	DATOSSALIDA[30]

Figura 26. Reporte de tiempo de clock to output

Tiempo *pin-to-pin*

En la figura 27 se muestra que el mayor retardo ***pin-to-pin*** es de 5.417ns cuando la señal de entrada ENTRADAINTP, es un pulsador de la FPGA Cyclone III, llegue al pin de la señal de observabilidad OUT INTERRUPCIÓN.

tpd

	Actual P2P Time	From	To
1	5.417 ns	EntradaIntP	OutInterrupcion
2	4.181 ns	Reset	OutReset

Figura 27. Reporte de tiempo de pin to pin

Las señales OutInterrupcion y OutReset se utilizaron para poder manejar el *Trigger* del analizador lógico TLA 5202B. Si estas señales no existen, el t_{pd} tampoco, ya que este es el tiempo requerido para que una señal de un pin de entrada se propague a través de una lógica combinatoria y aparezca en un pin de salida externo.

Tiempo de *Hold*

En la figura 28, se muestra que el peor tiempo de *hold* es de 1.555ns cuando el dato se retiene en el *flip-flop* 6 de la máquina de control y va para el *flip-flop* 20 del bloque RPG después de un evento de reloj. En la siguiente imagen se muestran los peores y los mejores casos para el t_h

th

	Minimum Slack	Required th	Actual th	From	To
1	N/A	None	1.555 ns	EntradaIntP	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[52]
2	N/A	None	1.554 ns	EntradaIntP	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[2]
3	N/A	None	0.285 ns	EntradaRecMaster	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:RM|Output
4	N/A	None	0.247 ns	EntradaRecSlave	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:RS|Output
5	N/A	None	-0.010 ns	EntradaTerminarTarea	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:TT|Output
6	N/A	None	-0.189 ns	EntradaEmpezarTarea	OokiCore:OokiCoreM|StatusRegister:StatusRegisterOokiCore|FlipFlop:ET|Output
7	N/A	None	-0.540 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[57]
8	N/A	None	-0.559 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[115]
9	N/A	None	-0.562 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[126]
10	N/A	None	-0.563 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[31]
11	N/A	None	-0.704 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[56]
12	N/A	None	-0.710 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[53]
13	N/A	None	-0.732 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[114]
14	N/A	None	-0.736 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[125]
15	N/A	None	-1.314 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[30]
16	N/A	None	-3.313 ns	SENALWAIT	OokiCore:OokiCoreM|ControlProcesador:ControlOokiCore|Q[54]

Figura 28. Reporte de tiempo de clock to output

El mejor tiempo de *hold* es de -3.313ns cuando el pin de entrada de la señal SENALWAIT va a pasar al *flip-flop* 54 de la máquina de control.

4.2.3. Análisis de Potencia

Para el análisis de potencia se utilizó una herramienta dada por la empresa Altera en la cual se calcula la potencia consumida teniendo en cuenta los recursos utilizados de la FPGA. En la figura 29 se puede ver el reporte dado por esta herramienta. La potencia estática se define dependiendo de la temperatura de la juntura y las características físicas de la FPGA utilizada. La potencia dinámica hace referencia todos los registros y combinatorios utilizados en el proyecto.

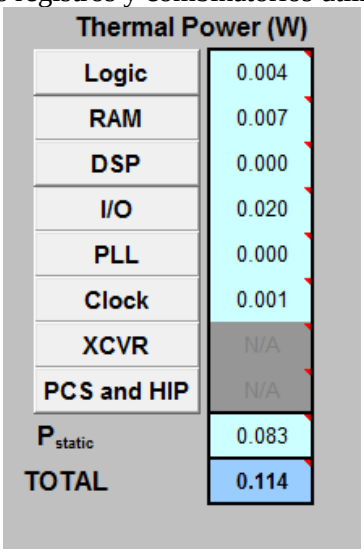

Figura 29 Reporte de la potencia consumida

El total de potencia disipada por la FPGA es de 0.114 W cuando se tiene implementado un solo procesador. Los elementos que más consumen potencia son los pines de Entrada/Salida del dispositivo que más a fondo se muestran en la figura 30.

												Thermal Power (W)		
Module	I/O Standard	Current Strength / Output Termination	Slew Rate	# Input Pins	# Output Pins	# Bidir Pins	Data Rate	Clock Freq (MHz)	Toggle %	OE %	Load (pF)	Routing	Block	Total
0	2.5 V	4mA	2	7	0	0	SDR	25.0	39.3%	0.0%	0	0.000	0.001	0.001
1	2.5 V	Series 50 Ohm	2	0	88	0	SDR	25.0	12.0%	100.0%	0	0.000	0.009	0.009
2	2.5 V	Series 50 Ohm	2	0	2	0	SDR	0.0	0.0%	100.0%	0	0.000	0.000	0.000
3	2.5 V	4mA	2	1	0	0	SDR	0.0	0.0%	0.0%	0	0.000	0.000	0.000

Figura 30 Reporte del consumo de potencia para los pines de entrada y de salida

En la columna de *Toggle* se muestra el porcentaje por ciclos de reloj de los cambios que hay en el valor de los pines, 39.3% para los 7 de entrada y 12 % para los 88 de salida. La disipación de potencia esta en la columna de Block, el cual está determinado por el *Toggle*.

El consumo en las lógicas combinatorias y de los *flip-flop* utilizados es de 0.004W , en la figura 31 se muestra la distribución del consumo de potencia entre las 1500 LUT utilizadas y los 803 *flip-flop*

						Thermal Power (W)		
Module	# LUTs	# FFs	Clock Freq (MHz)	Toggle %	Average Fanout	Routing	Block	Total
0	1505	0	25	15.3%	3	0.001	0.001	0.002
1	0	803	25	12.8%	4	0.001	0.000	0.002

Figura 31 Reporte del consumo de potencia para la lógica

En la figura 32, se muestra el consumo de potencia para los 64 bloques de memoria que se utilizan de la FPGA que es de 0.007W. Como es un solo procesador, se utiliza un memoria *Single Port* , asi que solo habria consumo en el puerto A

						Port A			Port B				Thermal Power (W)		
Module	RAM Type	# RAM Blocks	Data Width	RAM Depth	RAM Mode	Clock Freq (MHz)	Enable %	Write %	Clock Freq (MHz)	Enable %	R/W %	Toggle %	Routing	Block	Total
0	M9K	64	1	8192	Single Port	25.0	50%	50%	0.0	0%	0%	25.0%	0.000	0.007	0.007

Figura 32 Reporte del consumo de potencia para el bloque de memoria M9k

Por último se muestra en la figura 33, el consumo de potencia del reloj que es un pin global como se había comentado anteriormente. Su consumo es de 0.001W.

Domain	Clock Freq (MHz)	Total Fanout	Global Enable %	Local Enable %	Total Power (W)
0	25.0	866	50%	41%	0.001

Figura 33 Reporte del consumo de potencia para el reloj

4.3. ANÁLISIS PARA LOS DOS PROCESADORES

Para los dos procesadores, se una memoria True Dual Port, es decir, una memoria de dos puertos por los cuales se conecta cada uno de los procesadores. Esto es importante para el resto del análisis.

Los ajustes para la compilación, que se utilizaron para realizar los diferentes análisis fueron los siguientes:

- Características de potencia del dispositivo: Normal
- Early Timing Estimate: Realistic
- Physical Synthesis Optimizations: Effort Level: Normal
- Técnica de Optimización: Balanceada
- PowerUp power optimization: Normal
- VHDL Version : VHDL 1993
- Fitter Effort: AutoFit
- Timing Analysis Settings: Se utilizó el *Classic Time Analizer*

4.3.1. Análisis de Recursos

Cuando se conectan los dos procesadores algunos de los recursos se duplican como era de suponerse. En la figura 34, reporte general de la compilación puede ver que si se cumple esa suposición en algunos casos.

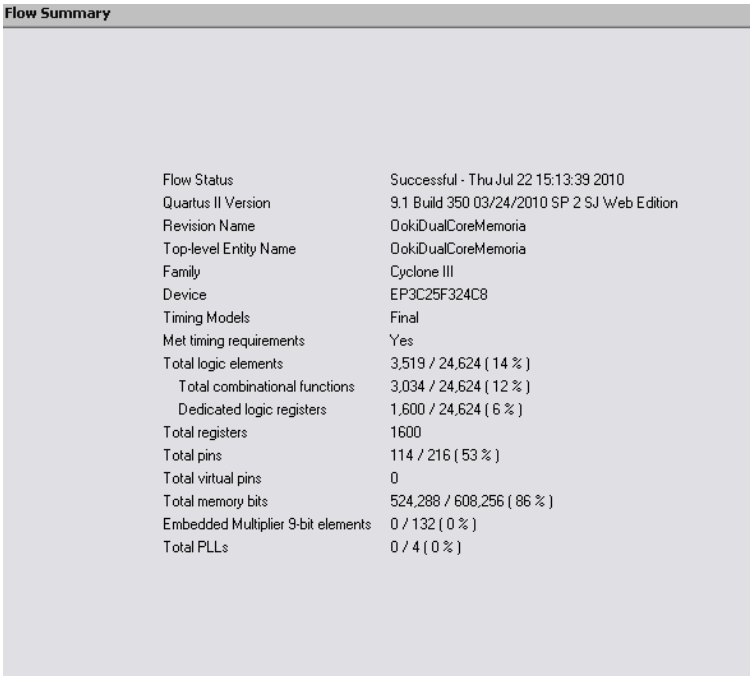

Figura 34. Reporte de compilación para el OokiDualCore

Los pines que se utilizan son 114 de 216 los cuales se distribuyen entre los dos buses de DIRECCIONES (14 bits) de cada procesador , las 8 señales de protocolo, el reloj, el reset, la SENALWAIT, los buses bidireccionales de DATOS, las dos salidas SALIDAHLT, las dos salidas BUSVALIDO, RENABLEA, WENABLEA, RENABLEB,WENABLEB y por último las señales de observabilidad del reset y del reloj que se utilizaron como *Trigger* en el analizador lógico, dependiendo del caso. En este caso no se duplicaron ya que la cantidad de pines que se desean utilizar solo depende del diseñador.

La cantidad de elementos lógicos utilizados es de 3519, se utiliza el 14% de la FPGA, un poco más del doble que se utilizaron para un solo procesador (1746 elementos lógicos). Lo mismo ocurre con los registros ahora son 1600, con un solo procesador eran 803 registros. En el reporte de la figura 40, se puede ver que en modo normal se están utilizando 2959 elementos y en modo aritmético se están utilizando 74 elementos. Esto es porque al tener cada procesador su ALU se tiene dos sumadores.

El *fanout* máximo sigue siendo del reloj pero este aumentó a 1665, como es una señal global maneja todas los registros de los dos procesadores. La cantidad de bits de memoria siguen siendo los mismos, ya que la memoria es la misma sino que se cambió la configuración de esta a una *True Dual Port.*

Analysis & Synthesis Resource Usage Summary		
	Resource	Usage
1	Estimated Total logic elements	4,106
2		
3	Total combinational functions	3033
4	⊟ Logic element usage by number of LUT inputs	
5	-- 4 input functions	2135
6	-- 3 input functions	662
7	-- <=2 input functions	236
8		
9	⊟ Logic elements by mode	
10	-- normal mode	2959
11	-- arithmetic mode	74
12		
13	⊟ Total registers	1600
14	-- Dedicated logic registers	1600
15	-- I/O registers	0
16		
17	I/O pins	114
18	Total memory bits	524288
19	Maximum fan-out node	RELOJ:RelojOokiDualCore|Q
20	Maximum fan-out	1665
21	Total fan-out	18125
22	Average fan-out	3.68

Figura 35. Reporte de compilación detallado para el OokiDualCore

El número total de registros utilizados es de 1600 registros Este número debe coincidir con el número de registros que se calculó en el diseño, los cuales se muestran en la tabla 5.

NOMBRE DEL REGISTRO	NÚMERO DE REGISTROS
MÁQUINA DE CONTROL	260
REGISTRO DE DATOS	64
REGISTRO DE INSTRUCCIONES	64
REGISTRO DE DIRECCIONES	28
PROGRAM COUNTER	28
RPG	1024
STATUS REGISTER	18
REGISTRO DE CORRIMIENTO	64
REGISTRO AUXILIAR	28
CONTADOR DE CORRIMIENTO	10
PROTOCOLO DE SALIDA	8
DIVISOR DE FRECUENCIA	1
TOTAL	1597

Tabla 5. Número de registros utilizados para los dos procesadores conectados

Sin embargo, si la memoria utiliza 4 registros, sobraría uno y serian 1601. Al mirar el reporte de recursos por entidad, se puede observar que el QUARTUS II, como parte de su implementación tomo un *flip-flop* menos para el OokiCoreA (*Master)* en la máquina de estados como se muestra en la figura 41.

Analysis & Synthesis Resource Utilization by Entity

	Compilation Hierarchy Node	LC Combinationals	LC Registers
1	⊟ |OokiDualCoreMemoria	3033 (0)	1600 (0)
2	⊞ |MemoriaTD:MemoriaOokiDualCore|	68 (0)	4 (0)
9	⊞ |OokiCore2:OokiCoreB|	1481 (0)	798 (0)
69	⊞ |OokiCore:OokiCoreA|	1483 (0)	797 (0)
129	|RELOJ:Reloj|OokiDualCore|	1 (1)	1 (1)

Figura 36. Reporte de los recursos utilizados por entidad

Otra cosa que se analizó, igual que para un solo procesador, fue cuántos de estos registros utilizaron carga asíncrona, y carga síncrona como se muestra en la figura 37. Utilizando el *Clock Enable,* se utilizaron 1322, estos son la mayoría de los registros menos la máquina de control, los 10 de carga síncrona y 1 del divisor de frecuencia.

General Register Statistics

	Statistic	Value
1	Total registers	1600
2	Number of registers using Synchronous Clear	0
3	Number of registers using Synchronous Load	10
4	Number of registers using Asynchronous Clear	259
5	Number of registers using Asynchronous Load	0
6	Number of registers using Clock Enable	1322
7	Number of registers using Preset	0

Figura 37. Reporte de cargas asíncronas y síncronas de los registros

4.3.2. Análisis de Tiempos

En el resumen de compilación dado por QUARTUS II 9.1 se puedo sacar en la figura 38 el reporte de tiempos para su análisis,

Timing Analyzer Summary

	Type	Actual Time	From	To
1	Worst-case tsu	0.509 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[31]
2	Worst-case tco	19.157 ns	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[4]	BusValidoA
3	Worst-case tpd	4.181 ns	Reset	OutReset
4	Worst-case th	0.481 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[125]
5	Clock Setup: 'Clk1'	27.68 MHz (period = 36.128 ns)	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[28]	OokiCore2:OokiCoreB|RPG:RPGOokiCore|RegistrosRPG:Registros|Registro32:RJ|Out32[10]
6	Total number of failed paths			

Figura 38. Resumen del analizador de tiempos de QUARTUS II 9.1

Este reporte muestra los peores casos donde los tiempos de *setup* (t_{su}), *hold* (t_h), *clock to output* (t_{co}) y *pin- to –pin* (t_{pd}) al límite donde estos dejarían de ser validos. En el reporte se puede ver que la frecuencia máxima es de 27.68 MHz de reloj cuando se pasa desde el *flip-flop* 28 de la máquina de control del OokiCoreB (*Slave)* al bloque RPG del mismo.

Esta frecuencia es menor a la de un solo procesador ya que hay más caminos críticos entre las señales. Esta fue otra razón por las cuales se decidió utilizar una frecuencia máxima de 25 MHz, para poder hacer todas las pruebas y hallar las diferencias entre un solo procesador y dos conectados.

Los resultados para los tiempos fueron:

Tiempo de *setup*

El tiempo de *setup* es de 0.509ns se da cuando la señal SENALWAITB, del procesador OokiCoreB pase al bus de DATA del *flip-flop* 31 de la máquina de control del mismo procesador antes que haya señal de reloj en el pin *Clock* del mismo. A continuación se muestra en la figura 44, un reporte que tiene los peores y los mejores casos para el t_{su}

tsu

	Slack	Required tsu	Actual tsu	From	To
1	N/A	None	0.509 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[31]
2	N/A	None	0.507 ns	EntradaIntPB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[2]
3	N/A	None	0.455 ns	EntradaIntPB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[52]
4	N/A	None	0.402 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[115]
5	N/A	None	0.399 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[126]
6	N/A	None	0.394 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[54]
7	N/A	None	0.374 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[114]
8	N/A	None	0.368 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[53]
9	N/A	None	0.342 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[57]
10	N/A	None	0.257 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[125]
11	N/A	None	0.207 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[30]
12	N/A	None	0.042 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[30]
13	N/A	None	0.036 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[56]
14	N/A	None	0.022 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[57]
15	N/A	None	0.020 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[31]
16	N/A	None	0.018 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[126]
17	N/A	None	0.009 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[115]
18	N/A	None	-0.018 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[53]
19	N/A	None	-0.038 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[54]
20	N/A	None	-0.077 ns	EntradaIntPA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[2]
21	N/A	None	-0.106 ns	EntradaIntPA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[52]
22	N/A	None	-0.283 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[114]
23	N/A	None	-0.289 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[56]
24	N/A	None	-0.290 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[125]

Figura 39. Reporte de tiempo de setup

El mejor tiempo es de -0.290 ns que se da cuando la señal de entrada SENALWAITA, del OokiCoreA(*Master)* pasa al bus de DATA del *flip-flop* 125 de la máquina de control de este procesador antes que la señal de reloj llegue al pin de *Clock* al *flip-flop*.

Tiempo de *clock to output*

En la figura 43, se puede observar que el mayor retardo *clock to output* es de 19.157ns, cuando el *flip-flop* 4 de la máquina de control del procesador OokiCoreA saca un dato válido al bus de salida llamado BUSVALIDOA. Con esto también se puede ver que el retardo en este caso es mucho mayor al que se tenía con un solo procesador.

En la figura 45 se mostraran los diez mejores casos para este tiempo. El mejor tiempo es de 13.286ns cuando es válido un bit que sale de la memoria pasa al bus de DATOS[7].

tco

	Actual tco	From	To
190	13.384 ns	MemoriaTD:MemoriaOokiDualCore|altsyncram:altsyncram_compone...	DATOSA[16]
191	13.373 ns	MemoriaTD:MemoriaOokiDualCore|altsyncram:altsyncram_compone...	DATOSB[17]
192	13.348 ns	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[86]	BusValidoA
193	13.335 ns	MemoriaTD:MemoriaOokiDualCore|altsyncram:altsyncram_compone...	DATOSA[4]
194	13.328 ns	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[89]	BusValidoA
195	13.328 ns	MemoriaTD:MemoriaOokiDualCore|altsyncram:altsyncram_compone...	DATOSA[7]
196	13.322 ns	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[71]	BusValidoB
197	13.321 ns	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[70]	BusValidoB
198	13.316 ns	MemoriaTD:MemoriaOokiDualCore|altsyncram:altsyncram_compone...	DATOSB[8]
199	13.292 ns	OokiCore2:OokiCoreB|Registro14:RDireccionesOokiCore|Out14[3]	DIRECCIONESB[3]
200	13.286 ns	MemoriaTD:MemoriaOokiDualCore|altsyncram:altsyncram_compone...	DATOSA[7]

Figura 40. Reporte de tiempo de clock to output

Tiempo *pin-to-pin*

En la figura 43, se muestra que el mayor retardo ***pin-to-pin*** es de 4.181 cuando la señal de entrada RESET, el cual es un pulsador de la FPGA Cyclone III, llegue al pin de la señal de observabilidad OutReset.

tpd

	Actual P2P Time	From	To
1	4.181 ns	Reset	OutReset

Figura 41. Reporte de tiempo de pin to pin

Si esta señal no existe, el t_{pd} tampoco, ya que este es el tiempo requerido para que una señal de un pin de entrada se propague a través de una lógica combinatoria y aparezca en un pin de salida externo.

Tiempo de *Hold*

En la figura 43, se muestra que el peor tiempo de *hold* es de 0.481ns cuando el dato se retiene en el bus de entrada SENALWAITA, del OokiCore A y va para el *flip-flop* 125 del bloque de control del mismo procesador después de un evento de reloj. En la siguiente imagen se muestran los peores y los mejores casos para el t_h

th

	Actual th	From	To
1	0.481 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[125]
2	0.480 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[56]
3	0.474 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[114]
4	0.297 ns	EntradaIntPA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[52]
5	0.268 ns	EntradaIntPA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[2]
6	0.229 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[54]
7	0.209 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[53]
8	0.182 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[115]
9	0.173 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[126]
10	0.171 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[31]
11	0.169 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[57]
12	0.155 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[56]
13	0.149 ns	SenalWaitA	OokiCore:OokiCoreA|ControlProcesador:ControlOokiCore|Q[30]
14	-0.016 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[30]
15	-0.066 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[125]
16	-0.151 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[57]
17	-0.177 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[53]
18	-0.183 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[114]
19	-0.203 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[54]
20	-0.208 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[126]
21	-0.211 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[115]
22	-0.264 ns	EntradaIntPB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[52]
23	-0.316 ns	EntradaIntPB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[2]
24	-0.318 ns	SenalWaitB	OokiCore2:OokiCoreB|ControlProcesador:ControlOokiCore|Q[31]

Figura 42. Reporte de tiempo de hold

El mejor tiempo de *hold* es de -0.318ns cuando el pin de entrada de la señal SENALWAITB, del procesador OokiCoreB(*Slave*), va a pasar al *flip-flop* 31 de la máquina de control.

4.3.3. Análisis de Potencia

Para el análisis de potencia se utilizó también la herramienta dada por la empresa Altera en la cual se calcula la potencia consumida teniendo en cuenta los recursos utilizados de la FPGA. En la figura 48 se puede ver el reporte dado por esta herramienta

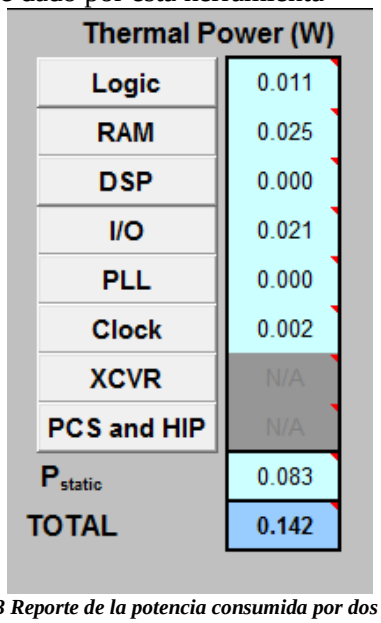

Figura 43 Reporte de la potencia consumida por dos procesadores

El total de potencia disipada por la FPGA es de 0.142 W cuando se tiene implementado dos procesadores mayor a lo que se obtuvo con un solo procesador. La razón es porque se utilizan más recursos de la FPGA. Los elementos que más consumen potencia es la memoria RAM.

En la figura 49 se muestra el consumo de potencia para los pines de Entrada / Salida el cual es de 0.0021W. Aquí se muestran 0.0011W entre los pines de entrada y los pines de salida.

												Thermal Power (W)		
Module	I/O Standard	Current Strength / Output Termination	Slew Rate	# Input Pins	# Output Pins	# Bidir Pins	Data Rate	Clock Freq (MHz)	Toggle %	OE %	Load (pF)	Routing	Block	Total
0	2.5 V	4mA	2	5	0	0	SDR	25.0	50.0%	0.0%	0	0.000	0.001	0.001
1	2.5 V	Series 50 Ohm	2	0	107	0	SDR	25.0	11.2%	100.0%	0	0.000	0.011	0.011
2	2.5 V	Series 50 Ohm	2	0	1	0	SDR	0.0	0.0%	100.0%	0	0.000	0.000	0.000
3	2.5 V	4mA	2	1	0	0	SDR	0.0	0.0%	0.0%	0	0.000	0.000	0.000

Figura 44 Reporte del consumo de potencia para los pines de entrada y de salida

En la columna de *Toggle* se muestra el porcentaje por ciclos de reloj de los cambios que hay en el valor de los pines, 50% para los 5 de entrada y 11.2 % para los 107 de salida. La disipación de potencia está en la columna de Block, el cual está determinado por el *Toggle*.

El consumo en las lógicas combinatorias y de los *flip-flop* utilizados es de 0.0011W , en la figura 50 se muestra la distribución del consumo de potencia entre las 1500 LUT utilizadas y los 803 *flip-flop*

						Thermal Power (W)		
Module	# LUTs	# FFs	Clock Freq (MHz)	Toggle %	Average Fanout	Routing	Block	Total
0	3034	0	25	26.3%	3	0.004	0.004	0.008
1	0	1600	25	12.7%	4	0.002	0.001	0.003

Figura 45 Reporte del consumo de potencia para la lógica para dos procesadores

En la figura 51, se muestra el consumo de potencia para los 64 bloques de memoria que se utilizan de la FPGA que es de 0.0025mW. Para los dos procesadores , se utiliza un memoria *True Dual* entonces hay consumo de potencia por los dos puertos de la memoria.

						Port A			Port B				Thermal Power (W)		
Module	RAM Type	# RAM Blocks	Data Width	RAM Depth	RAM Mode	Clock Freq (MHz)	Enable %	Write %	Clock Freq (MHz)	Enable %	R/W %	Toggle %	Routing	Block	Total
0	M9K	64	1	8192	True Dual Port	25.0	100%	50%	25.0	100%	50%	12.5%	0.000	0.025	0.025

Figura 46 Reporte del consumo de potencia para el bloque de memoria M9k para dos procesadores

Por último se muestra el consumo de potencia del reloj de 0.002W, a una frecuencia máxima de 25MHz y un *fanout* de 1663.

Domain	Clock Freq (MHz)	Total Fanout	Global Enable %	Local Enable %	Total Power (W)
0	25.0	1663	50%	41%	0.002
1	0.0	259	0%	100%	0.000

Figura 47 Reporte del consumo de potencia para el reloj

5. CONCLUSIONES

Se diseñó e implementó un procesador el cual tiene la capacidad de manejar diversos periféricos, incluyendo instancias del mismo, que le permiten ser utilizado como un sistema embebido. Además, con el conjunto completo de instrucciones y los diferentes modos de direccionamiento, el usuario puede utilizar el procesador para alguna aplicación que esté desarrollando. Cuenta además con una entrada de Interrupción autovectorizada.

La metodología de diseño que se viene utilizando en la Sección de Técnicas Digitales para la realización de un sistema digital complejo, favoreció mucho el desarrollo del presente trabajo. La conceptualización del sistema, el diagrama de bloques, la descripción en AHPL, los circuitos esquemáticos y las tablas de conectividad, facilitaron la descripción del sistema en VHDL, la posterior implementación y la puesta a punto del trabajo propuesto.

El uso de una FPGA para la implementación del procesador fue una ventaja ya que gracias a su arquitectura y a la cantidad de elementos lógicos se logró desarrollar un sistema digital complejo como lo es un procesador. En este caso se utilizó un dispositivo Cyclone III, que permitió tener una amplia cantidad de memoria interna (16k x 32 bit) utilizando los bloques M9k.

El procesador ocupa el 7% del dispositivo y en el caso de dos la ocupación es del 14%. Lo anterior va a permitir al usuario conectar otros sistemas digitales para lograr un diseño de mayor complejidad, así como también puede conectar más procesadores con su lógica adicional para generar un sistema multicore. El límite de la cantidad de procesadores que se pueden conectar lo da la FPGA y la memoria que se utilice para implementar el procesador. Un trabajo futuro podría ser hacer procesamiento en paralelo con más de dos procesadores para concretar este límite.

Se pudo comprobar, que la utilización de un sistema con dos procesadores para realizar una tarea específica, logra una reducción en el tiempo de ejecución, que va ser un atractivo para el usuario. Por otro lado la implementación de dos procesadores resulta en un incremento en la ocupación del dispositivo y por tanto en un incremento en el consumo de energía.

Es importante que el usuario programador de un sistema con dos procesadores, tenga en cuenta varios factores en el momento de hacer su programa:

- Debe tener en cuenta que no todos los programas se pueden organizar para que trabajen dos procesadores en forma concurrente. Es necesario determinar si la aplicación que se desea implementar lo permite
- Es necesario diseñar las tareas que debe realizar tanto el Master como el Slave, para producir el efecto deseado de realizar una misma aplicación utilizando los dos procesadores.
- Debe tener en cuenta los señales de protocolo implementadas con el fin de que la comunicación entre los dos procesadores se realice en el momento deseado
- Debe tener en cuenta que cómo los dos procesadores utilizan la misma memoria, debe evitar que haya escritura a la misma posición de memoria simultáneamente.
- Debe cuidar la posición donde desea colocar los programas para cada uno de los procesadores, para evitar interferencia entre las dos tareas.

Se debe pensar que al ser un IP CORE se debería Estandarizar los puertos de entrada salida del procesador según las especificaciones de interfaz Avalon de Altera para poder conectarlo con periféricos embebidos ya existentes o con nuevos periféricos que se pueden crear para uso exclusivo de OokiCore.

Para el futuro de este proyecto se puede pensar en diferentes trabajos en los cuales se logre mejorar las limitaciones del procesador. Uno de esos proyectos puede ser la generación del hardware que permita el acceso de forma simultánea a la memoria por parte de los dos procesadores utilizando la señal SENALWAIT si se desea utilizar las memorias embebidas de la FPGA. Cuando el procesador emite la

SENALWAIT querría decir que se encuentra escribiendo en la memoria (se encuentra ocupado) y que el periférico debe esperar su turno para la escritura.

Teniendo un sistema como estos otro proyecto futuro podría ser generación de un sistema de prioridades de interrupción cuando se manejan uno o dos procesadores. Una de las limitaciones del proyecto cuando se tienen los dos procesadores conectados es la falta de prioridad cuando ocurre una interrupción porque cualquiera de los dos puede ser interrumpido.

Algunas limitaciones del procesador, como la velocidad, podrían ser mejoradas con otros proyectos como modificar la ALU del procesador para realizar operaciones de división, de multiplicación y con punto flotante dando más opciones al usuario para realizar programas donde requiera utilizar números decimales. También se puede proponer la segmentación de algunas de redes combinatorias para aumentar la velocidad sobre todo las de los RPG.

Una limitación importante al realizar el proyecto, fue la tarjeta de desarrollo que se utilizó ya que al tener tantos dispositivos conectados a los pines de salida de la FPGA, no habían los suficientes para observar el correcto funcionamiento del sistema por lo tanto se puede proponer el diseño de una tarjeta de desarrollo para facilitar el trabajo con el procesador.

Por último, ya que una de las ideas principales de este procesador es que sirva como parte del proceso de aprendizaje de un estudiante cuando este aprendiendo procesadores se puede proponer la generación un simulador de esta arquitectura para ser utilizada con fines didácticos y también la generación de software de apoyo como son ensambladores y compiladores.

6. BIBLIOGRAFÍA

1. ALVAREZ RIVERA, Bibiana Andrea, PINILLA PICO, Lady Nathalie. “Implementación de un procesador digital BINARIC”. Trabajo de Grado, Carrera de Ingeniera Electrónica, Pontificia Universidad Javeriana, 2007.
2. BELTRÁN VELEZ, Diego Hernán, HERRERA BUITRAGO, Moisés Fernando, MAYOLO OBREGÓN, Marco Antonio. “Implementación del procesador ALTERIC”. Trabajo de Grado, Carrera de Ingeniera Electrónica, Pontificia Universidad Javeriana, 2004.
3. FOSTER, Harry, PERRY L., Douglas. Applied formal verification. Mc. Graw Hill
4. FOSTER, Harry, BENING, Lionel. Principles of verifiable RTL design. Hewlett Packard Company. 2da Edición.
5. HILL, Frederick J, PETERSON, Gerald R. "Digital Systems, Hardware organization and design". Tercera edición. Singapore. John Wiley & Sons. 1987.
6. http://www.altera.com/education/univ/materials/ip-cores/unv-ip-cores.html [online] Altera University Program- IP Cores for Education. Copyright © 1995-2009 Altera Corporation. All Rights Reserved.
7. MORENO DIAZ, Henry Leonardo. ¨PCSIM RELOADED¨. Trabajo de Grado, Carrera de Ingeniera Electrónica, Pontificia Universidad Javeriana, 2004
8. PEDRONI, Volnei A. “Circuit Design with VHDL”. Massachussets Institute of Technology. United States .2004
9. STALLINGS, William. “Organización y Arquitectura de Computadores”. Séptima Edición. Pearson Education, 2006.

TABLA DE ANEXOS

En el CD adjunto se encuentran los anexos que se han comentado en el este proyecto

ANEXO A AHPL
ANEXO B CIRCUITOS ESQUEMÁTICOS
ANEXO C TABLAS DE CONECTIVIDAD
ANEXO D VHDL
ANEXO E PRUEBAS PARA LA MÁQUINA DE CONTROL
ANEXO F PRUEBAS PARA UN SOLO PROCESADOR
ANEXO G PRUEBAS PARA DOS PROCESADORES
ANEXO H PROGRAMAS PROPUESTOS