DISEÑO DE LA ARQUITECTURA DE UNA CELDA PROGRAMABLE PARA IMPLEMENTACION DE BANCO DE FILTROS EN FPGA

Joaquín Louzao jlouzao@adinet.com.uy
Santiago Paz sanpaz@adinet.com.uy
Daniel Tejera tejera@athenea.ort.edu.uy
Gustavo Bellora bellora@athenea.ort.edu.uy
Guillermo Langwagen langwagen@athenea.ort.edu.uy

Facultad de Ingeniería. Bernard Wand-Polak. Ingeniería en Telecomunicaciones.
Universidad ORT Uruguay

ABSTRACT

In this paper we describe the architecture´s design of a programmable cell belonging to a filter bank to be implemented in a FPGA. This cell is composed by a modulator and a filter stage, where an average moving filter (FIR filter) is employed. The user may program the amount of samples from the carrier signal as well as the average moving filter delays. This work takes into consideration both the chip area ocupied by the design and the highest working frecuency.

The cell implementation took place in a Virtex XSV100 FPGA from Xilinx. The cell is capable of working at a maximun frecuency clock of 53,6 MHz occuping an area of 438 slices.

RESUMEN

En este artículo se describe el diseño de la arquitectura de una celda programable perteneciente a un banco de filtros a ser implementada en un FPGA. Dicha celda está formada por una etapa moduladora y una etapa de filtrado, en la que se utiliza un filtro promedio móvil (filtro FIR). Es programable por el usuario tanto la cantidad de muestras correspondiente a la señal portadora, como la cantidad de retardos pertenecientes al filtro promedio móvil. Este estudio se realizó considerando el área de chip ocupada por el diseño y la máxima frecuencia de trabajo.

La implementación de la celda se realizó en un FPGA Virtex XSV100 (100.000 compuertas) de Xilinx. La celda es capaz de trabajar a una frecuencia máxima de reloj de 53,6 MHz ocupando un área de 438 Slices.

DISEÑO DE LA ARQUITECTURA DE UNA CELDA PROGRAMABLE PARA IMPLEMENTACION DE BANCO DE FILTROS EN FPGA

Joaquín Louzao jlouzao@adinet.com.uy
Santiago Paz sanpaz@adinet.com.uy
Daniel Tejera tejera@athenea.ort.edu.uy
Gustavo Bellora bellora@athenea.ort.edu.uy
Guillermo Langwagen langwagen@athenea.ort.edu.uy

Facultad de Ingeniería. Bernard Wand-Polak. Ingeniería en Telecomunicaciones.
Universidad ORT Uruguay

ABSTRACT

En este artículo se describe el diseño de la arquitectura de una celda programable perteneciente a un banco de filtros a ser implementada en un FPGA. Dicha celda está formada por una etapa moduladora y una etapa de filtrado, en la que se utiliza un filtro promedio móvil (filtro FIR). Es programable por el usuario tanto la cantidad de muestras correspondiente a la señal portadora, como la cantidad de retardos pertenecientes al filtro promedio móvil. Este estudio se realizó considerando el área de chip ocupada por el diseño y la máxima frecuencia de trabajo.

La implementación de la celda se realizó en un FPGA Virtex XSV100 (100.000 compuertas) de Xilinx. La celda es capaz de trabajar a una frecuencia máxima de reloj de 53,6 MHz ocupando un área de 438 Slices.

1. INTRODUCCIÓN

El procesamiento de señales digitales ha jugado un papel clave en el desarrollo de sistemas de telecomunicaciones en las últimas dos décadas. Un ejemplo de esto es la utilización de bancos de filtros digitales. El procesamiento en subbandas de señales es empleado en un número importante de aplicaciones como audio, imagen y encoders/decoders de video; canceladores de eco acústico, sistemas de comunicación de "spread spectrum", transmultiplexaciones perfectas, ecualizadores (líneas DSL), analizadores, modulación discreta de multitonos, aplicaciones médicas, y prácticamente todo tipo de proceso que requiera procesamiento digital de señal. [1,2,3]

En este trabajo, en una primera parte, se presenta el banco de filtros, describiendo la estructura del filtro pasabanda elegido. A continuación, se muestra qué representa la celda, es decir cuales son las etapas que ésta implementa con respecto a la totalidad de las necesarias para la implementación de uno de los filtros del banco.

Luego de mostrar las consideraciones de diseño realizadas, se describe la arquitectura de la celda implementada, describiendo sus grandes bloques, la funcionalidad de los componentes y sus distintos modos de funcionamiento.

Uno de los puntos que se destaca para poder lograr un diseño eficiente, es el estudio e implementación de distintos algoritmos de suma y multiplicación en punto fijo.

Por otro lado, se debe lograr que la configuración del sistema pueda ser programada desde el exterior. Para lograr esto, se utiliza un software corriendo en un PC para dialogar con la celda y poder ajustar o configurar sus parámetros. Para la puesta en marcha del sistema fue necesario realizar otras aplicaciones que se utilizaron como herramientas para probar el buen funcionamiento del filtro. En esta sección se presentan la placa de desarrollo y la herramienta de síntesis utilizadas. Por último, se analizan los resultados obtenidos así como también se llevan a cabo recomendaciones para futuras aplicaciones del diseño.

2. DESCRIPCIÓN: BANCO DE FILTROS

La realización de la Celda surge de una idea más amplia que consta en desarrollar un banco de filtros digital programable, en el que se pueda configurar su transferencia. La arquitectura de dicho banco es la suma de "varios filtros en paralelo", donde cada uno de éstos determina la amplitud de cierto rango de frecuencias. En la siguiente figura se muestra cuales son los elementos que componen cada filtro de los N posibles pertenecientes al banco de filtros.

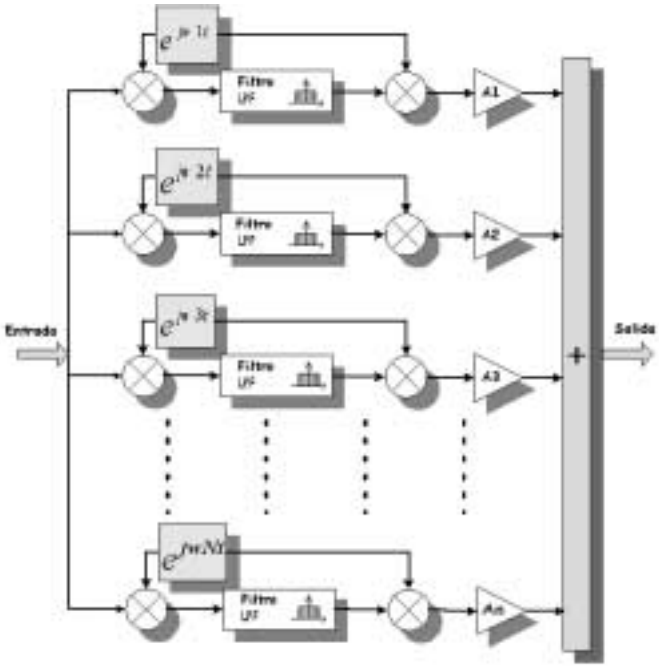

Fig. 1: Banco de Filtros

Estos "filtros en paralelo" son filtros pasabanda cuyas bandas pasantes se encuentran adyacentes de forma de cubrir el espectro completo de la señal de entrada. Cada uno de estos filtros puede dividirse en cuatro etapas. La primera es una etapa moduladora que corre en el espectro a la señal de entrada para llevar el rango de frecuencia deseado a banda base. La segunda etapa consta de un filtro pasabajo (LPF, Low Pass Filter) que determina el ancho de la banda pasante. La tercera etapa es un demodulador sincronizado con el modulador para llevar la señal modulada a su posición original en el espectro. La última etapa de estos filtros pasabanda es un multiplicador que adecúa la amplitud de la banda tratada; con esta etapa se logra la forma de la transferencia deseada.

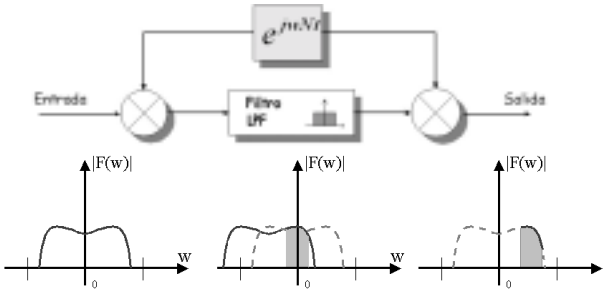

Fig. 2: Filtro Pasabanda en función de un Filtro Pasabajo (LPF)

2.1. Etapa Moduladora

La modulación consta de multiplicar la señal de entrada por un exponencial de exponente complejo. El manejo con complejos se logra operando con partes reales e imaginarias por separado y llevando apropiadamente a lo largo del filtro ambos vectores hasta reconstruir luego de una remodulación y de una suma nuevamente una señal real. Ver figura 3. Dado que una de las entradas del multiplicador es una señal moduladora, tendremos que diseñar tablas de datos, que contengan los valores discretos de dicha señal. Por lo tanto, en una de las etapas se debe multiplicar por una tabla que resulta de muestrear al coseno, mientras que en la otra, lo que se realiza es multiplicar por una tabla que resulta de muestrear al coseno desfasado, correspondiente al seno.

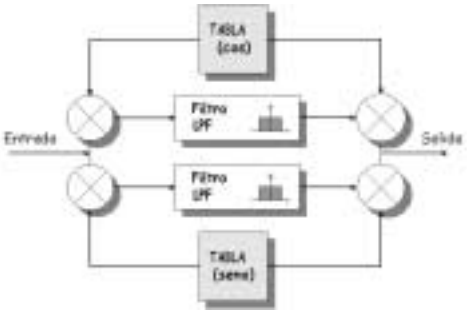

Fig. 3: Implementación de la modulación

2.2. Etapa de Filtrado

El filtro a implementar es un filtro transversal: promedio móvil (Moving Average Filter). Éste es un filtro de respuesta impulsiva finita (FIR, del inglés Finite Impulse Response), consta de una cascada de retardos con tomas de señal a la salida de cada retardo. La salida del filtro es el promedio de las últimas N señales retardadas, siendo N la cantidad de retardos implementados.

Los bancos de filtros de coseno modulado o de DFT utilizan un prototipo pasabajo que luego se reubica en la frecuencia de trabajo mediante la modulación con un coseno o una exponencial compleja, de allí el nombre.

Un sistema de reconstrucción perfecta que descompone la señal en sub-bandas cuyas salidas sumadas son iguales a la señal original puede construirse mediante un filtro pasabajo de promedio móvil y la modulación con las exponenciales armónicas exp (j2π k f_m / p), k = 0,1,....,(p-1) siendo f_m la frecuencia de muestreo del sistema.

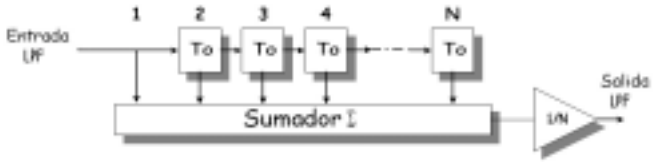

Fig. 4: Filtro LPF elegido: Filtro Promedio Móvil con N Retardos

Aunque existen elecciones mucho más satisfactorias para el prototipo pasabajo, el filtro promedio móvil tiene la enorme ventaja de la simplicidad: todos sus coeficientes son iguales y pueden moverse a un único lugar en la estructura. Ver figura 4. Como el próposito de la celda es utilizarla eventualmente en sistemas masivamente paralelos se optó por la simplicidad. [4,5]

3. ¿QUÉ ES LA CELDA?

El trabajo realizado se centra en llevar a cabo en el lenguaje de descripción de hardware VHDL [6,7] las dos primeras etapas del filtro pasabanda. La primera es una etapa moduladora (multiplicación) y la segunda etapa consta de un filtro pasabajo (LPF, Low Pass Filter), eligiendo un filtro promedio móvil (filtro FIR). Observar que para poder implementar la etapa moduladora, explicada anteriormente, necesitamos 2 celdas.

Es importante mencionar que dicha implementación se enmarca dentro del comienzo de una investigación más amplia, como es el desarrollo del banco de filtros por completo y de otros filtros.

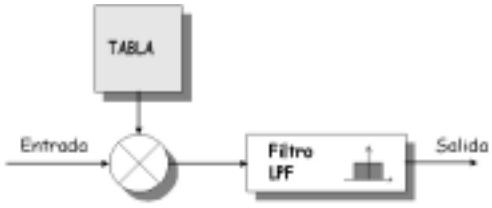

Fig.5: Celda

4. CONSIDERACIONES DE DISEÑO

4.1. Representación Numérica, Formato y Tamaño de Buses

Se optó por definir la señal de entrada, así como los valores de la "Tabla", con un tamaño de 16 bits con signo. Se utiliza representación en punto fijo [8]. Dentro de dicha representación se opta por la representación de los datos en punto fijo binario, que es la representación que más se adapta al sistema ya que maneja signo, parte entera y fracción y no desperdicia ningún bitio en la conversión de decimal a punto fijo binario. Disminuyendo de esta forma el "error de cuantificación" o el error por conversión de formatos. Se utiliza representación en complemento a 2 para los valores negativos.

Para los valores de entrada, el formato elegido de punto fijo binario es: S.15.O, siendo S un bit de signo. Consideramos que la entrada no debe tener parte fraccional, por lo que debe ser puramente parte entera. Esto se debe a que en la cuantificación no se tiene en cuenta la parte fraccional. Con respecto a los valores del muestreo de la señal portadora (valores de la "Tabla"), el formato será el siguiente: S.O.15. Esta elección de considerar que la señal está formada solo por parte fraccional con signo, se explica por el hecho de que la señal que se muestrea es el coseno y sus valores varían entre –1 y 1. De la elección de los formatos para la entrada y para la señal portadora obtenemos el formato resultante de la multiplicación, que es S.16.15.

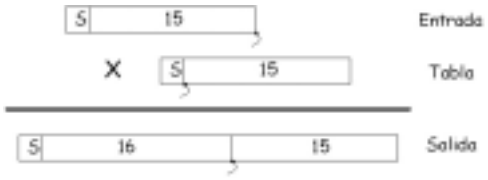

Fig. 6: Representación Numérica Elegida

4.2. Cantidad de Retardos

El largo del cadena de retardos está vinculado con el ancho espectral de la transferencia del filtro o sea con su ancho de banda. Cuando el banco de filtros que se desea implementar es un sistema de reconstrucción perfecta el número de filtros que deben construirse es igual al largo de la respuesta impulsiva. Teniendo esto en cuenta y a fin de crear una celda flexible capaz de adaptarse a diferentes situaciones se eligió un largo máximo para la cadena de retardos de 1024. De esta forma la máxima resolución obtenible en frecuencia es aproximadamente igual a una milésima parte de la frecuencia de muestreo del sistema.

4.3 Acumulación y División de N Retardos

Analizando la sumatoria que representa la ecuación del filtro, buscando alguna simplificación, notamos que no es necesario realizar toda la sumatoria de los N valores por cada valor que ingrese al sistema. Alcanza con mantener una suma total a la cual se resta el valor más antiguo y se suma el nuevo valor ingresado. Una vez calculada la suma del filtro, se debe realizar la división perteneciente a la ecuación del filtro, es decir, falta dividir entre la cantidad de N retardos que utilizamos para terminar de calcular el promedio de los últimos N valores. Impusimos que la cantidad de retardos sea una potencia de 2, para que la división en el sistema se transforme en un "shift" del dividendo (suma total), tantas posiciones a la derecha como indica el número x.

5. IMPLEMENTACIÓN DEL BANCO DE FILTROS

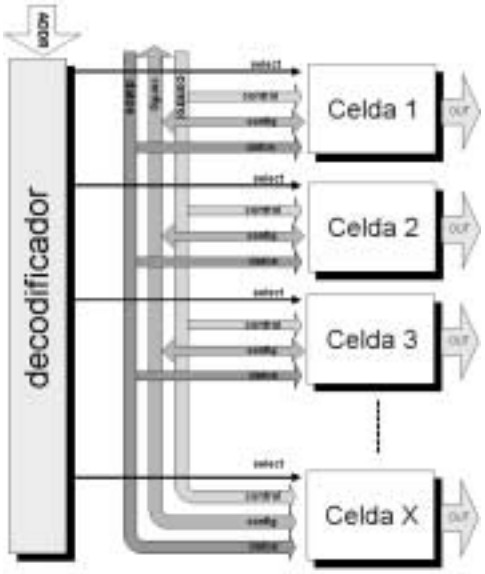

Fig. 7: Implementación del Banco de Filtros

En este trabajo se implementa únicamente una Celda, pero es importante destacar cuál es su interface, cuál es el esquema de configuración y cómo se implementa para llevar a cabo el banco de filtros. En este esquema de configuración se destacan tres buses: uno de datos, uno de configuración y otro de control. También existe un bus de direcciones que, después de ser decodificado, se convierte en una línea de "Select" que se dirige a cada Celda.

Estos buses presentan las siguientes características:

- El bus de datos y el bus de configuración son compartidos por todas las celdas.
- El bus de control con las líneas ENA (Enable), DR (Data Ready), RTR (Ready to Receive), RST (Reset), L/E (Lectura/Escritura) llega a todas las celdas.
- Las líneas de "Select" son individuales, una por celda.

La Celda posee tres modos de uso: "Modo Normal", "Modo Cambiar Configuración" y "Modo Leer Configuración". Podríamos considerar que el banco de filtros tiene básicamente dos modos de operación, por un lado el modo "Normal" que representa el procesamiento de señal, y por otro, el modo de configuración de la Celda. Dicha configuración es guardada en un "Registro de Configuración".

Cuando el banco se encuentra funcionando en modo "Normal", en el bus de direcciones (ADDR) debe haber una dirección que no corresponda a ninguna Celda. Esto se puede lograr utilizando la dirección cero o deshabilitando el bus mediante alguna línea de control. En este caso cada Celda escucha todas las señales de control, pero sólo responde a la línea de Reset y Enable.

Los siguientes son los pasos a seguir si se desea configurar una de las celdas:

1) Seleccionar la acción a realizar: leer o escribir la configuración (L/E).
2) Poner la dirección de la Celda deseada en el bus de direcciones, esto activará la línea "Select" correspondiente.
3) Esta Celda deshabilita la entrada de datos y toma el control de las líneas RTR (Ready to Receive) y DR (Data Ready) (sale de alta impedancia y dependiendo de la línea L/E coloca valores en DR o RTR).
4) Comienza el handshake y la transferencia de datos, ya sea leer o escribir el contenido del registro de configuración y los valores de la Tabla. En las transferencias, siempre el primer dato corresponde al valor del Registro de Configuración.
5) Una vez finalizada la programación, poniendo una dirección genérica en el bus de direcciones o deshabilitando el decodificador, la Celda vuelve a operar. Luego de haber sido seleccionada una Celda, en el flanco de bajada de la señal "Select", la Celda pasa a un estado de reset para limpiar todos los valores espurios de configuraciones anteriores.

El hecho de que el banco de filtros funcione de esta manera permite programar solamente una subbanda sin alterar el funcionamiento de las demás. Un escenario de aplicación es una transmisión del tipo modulación en múltiples portadoras donde una de las subbandas se ve interferida y se desea cambiarla de posición. Se podría hacer esto sin alterar el funcionamiento del resto del banco.

Como se mencionó anteriormente para lograr la modulación por la señal e^{jwt} se deben utilizar dos celdas, es decir dos tablas (coseno y seno). Esto trae consigo una restricción de fase, ya que las dos señales moduladoras deben estar sincronizadas. En la figura 8 se sugiere el esquema de interconexión de celdas vecinas. Se aprecia que al seleccionar una celda del par, la otra recibe un reset, esto hace que cuando termine la programación de una de ellas, comience el reset de la otra.

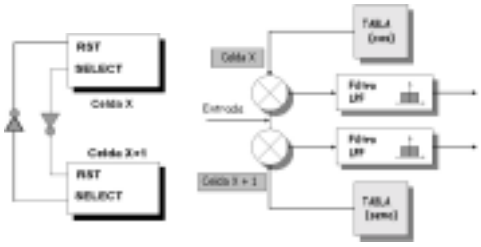

Fig. 8: Interconexión Celdas Vecinas

6. ARQUITECTURA DE LA CELDA

La arquitectura [9,10] dispone de dos grandes unidades. Por un lado, la **Unidad de Proceso** es la que contiene los recursos del sistema, es donde tienen lugar las distintas operaciones, transferencias de información y almacenamiento de datos en registros. Por otro lado, la **Unidad de Control** es la encargada de decidir en cada ciclo de reloj las transferencias de información que deben de tener lugar entre los componentes en la Unidad de Proceso así como el siguiente estado de control a ejecutar.

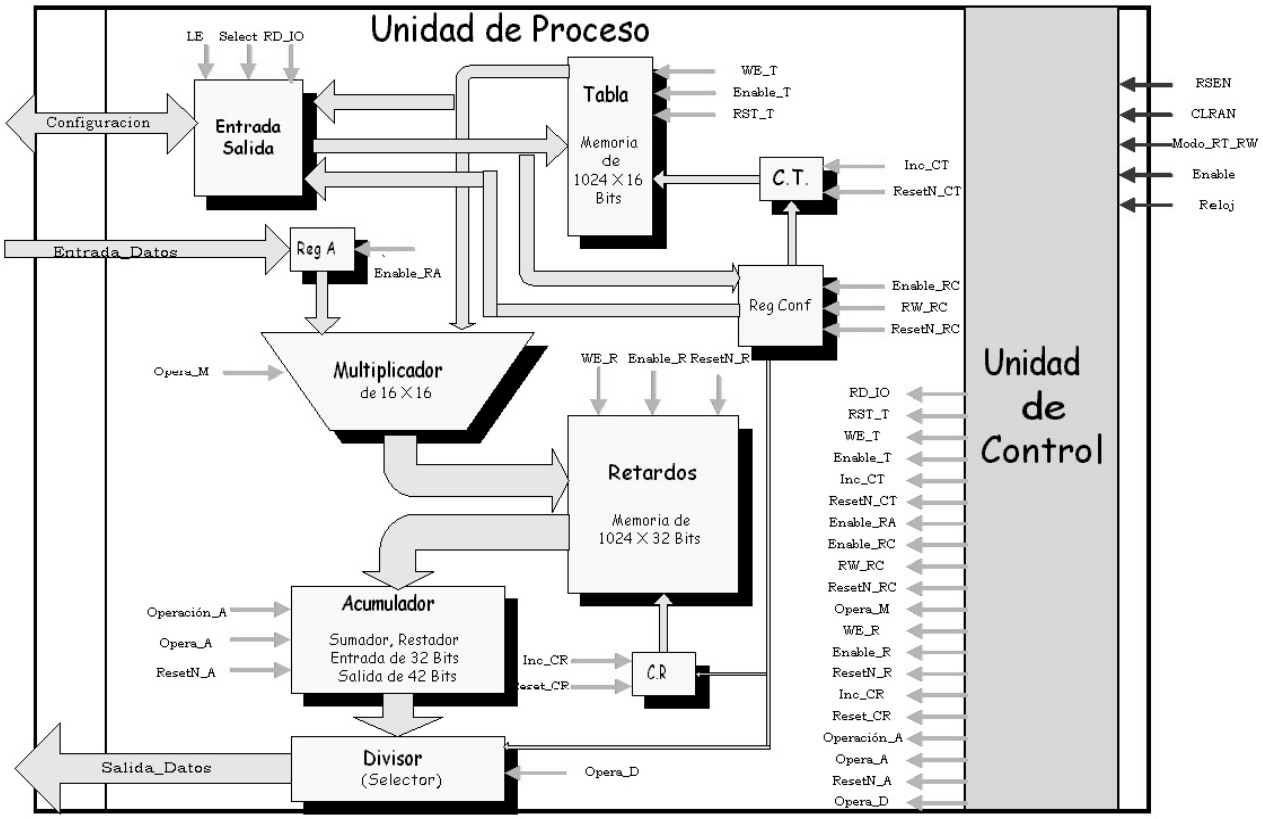

Fig. 9: Arquitectura de la Celda

6.1. Unidad de Proceso

En esta unidad podemos distinguir 2 grandes grupos de componentes: uno que forma el bloque modulador, y otro que implementa el bloque del filtro.

6.1.1. Descripción de los Componentes del Bloque Modulador

- **Memoria "Tabla"**

Para implementar el bloque modulador necesitamos que el sistema contenga los valores de la señal portadora. Un medio de almacenamiento adecuado es una memoria, ésta debe ser de rápido acceso para lograr una buena velocidad de trabajo.

La "Tabla" precisa un tamaño de 1024 entradas de 16 bits. Para esto es necesario un bus de datos de 16 bits y uno de direcciones de 10 bits. Para su implementación se utiliza la memoria RAM embebida en la pastilla.

- **Contador de Tabla (C.T.)**

Definimos el componente "Contador de Tabla" con la función de contener la dirección de memoria del siguiente coeficiente almacenado en la "Tabla". El incremento debe ser de uno en uno por cada operación ya que deseamos acceder a posiciones consecutivas. Se debe considerar que la cantidad de coeficientes a utilizar en la "Tabla" sea programable por el usuario, lo cual obliga a agregar una entrada al "Contador de Tabla" para indicar cuál es su valor máximo. Una vez alcanzado dicho valor el próximo incremento llevará la cuenta a cero. Utilizamos 10 bits, tanto para la entrada como la salida de este componente, lo cual nos permite direccionar las 1024 posiciones de la memoria "Tabla".

- **Registro A (Latch)**

Con este registro de almacenamiento temporal se logran guardar los datos presentados a la entrada del sistema, para su posterior proceso. Logrando que el sistema encargado de presentar los valores de entrada quede libre.

- **Multiplicador**

Realiza la multiplicación de una muestra de entrada por un coeficiente perteneciente a la "Tabla". Tiene dos entradas de 16 bits, por lo tanto, una salida de 32 bits.

6.1.2. Descripción de los Componentes del Bloque del Filtro

- **Memoria "Retardos"**

El secreto de los filtros transversales está en retener las últimas entradas al filtro, disponemos de componentes que retrasan la señal de entrada un tiempo To. Entonces, podemos imaginar que en un determinado instante tenemos señales colgadas a la salida de cada uno de estos componentes. Al ingresar un nuevo valor, todas las señales se corren un lugar a la derecha, eliminando el valor más antiguo. Si consideramos que el tiempo de un retardo es To, y N es la cantidad de retardos que posee un filtro, se constata que una señal colocada a la entrada demora un tiempo N.To en salir del filtro (ver figura 4).

Para implementar esta idea en la arquitectura utilizamos una memoria circular de N posiciones, donde cada valor ingresado en la memoria permanece allí un tiempo N.To. Una vez que tenemos las N posiciones almacenadas, con el siguiente valor sobreescribimos la primer posición, eliminando el valor más antiguo. Es como una memoria FIFO ("primero en entrar primero en salir"). El tamaño de

la memoria está determinado por los 32 bits de salida del bloque modulador, y la cantidad máxima de retardos que puede contener el filtro (1024, por consideración de diseño).

Al igual que para la implementación de la memoria "Tabla" se utiliza la memoria RAM embebida en la pastilla. Dicha implementación presenta un tamaño de 512 retardos debido a la cantidad de memoria disponible.

- **Contador de Retardos (C.R.)**

Su finalidad es controlar la próxima posición a acceder dentro de la memoria de "Retardos". En este caso, cambia con respecto al "Contador de Tabla" la forma de indicar la entrada que determina el máximo valor al que puede llegar el contador. El ancho del bus de entrada pasa de 10 a 4 debido a la consideración de diseño realizada de la cantidad de retardos utilizada. Dicha cantidad a utilizar en un determinado instante es igual a 2^x donde x es un número natural comprendido entre 0 y 10, permitiendo un valor máximo de 1024. Por lo tanto, en lugar de indicar directamente el máximo valor, indicamos a través de 4 bits de entrada el número x. Los restantes valores de 4 bits correspondientes a los valores del 11 al 15 serán tomados como el valor "0000".

- **Acumulador**

Por medio de este componente realizamos los cálculos necesarios para obtener la sumatoria del filtro. Este guarda la suma total de los valores contenidos en la memoria "Retardos". Tiene la capacidad de restar o sumar un valor presentado a la entrada, a la suma almacenada, dependiendo si la línea "Operación" vale "1" o "0". Un flanco ascendente en "Opera" provoca realizar la operación indicada por "Operación": 1 resta, 0 suma.

El ancho de 32 bits de entrada está determinado por la salida del bloque "Multiplicador". El ancho de 42 bits de la salida, se debe al considerar el máximo valor que puede llegar a sumarse, es decir 1024 sumandos de 32 bits.

- **Divisor**

Fijamos la cantidad de retardos que puede contener el filtro con el fin de transformar la división entre N por un "shift" de la suma total, tantas posiciones a la derecha como indica el $\log_2 N = X$. El funcionamiento es el siguiente: cuando se produce un flanco positivo en la señal "Opera" selecciona 32 bits consecutivos de los 42 bits de entrada, dependiendo de una entrada de selección de 4 bits, y los copia a la salida. Agregamos la capacidad de almacenamiento para mantener la salida estable, mientras el sistema procesa un nuevo valor.

6.1.3. Descripción de los restantes componentes

- **Registro de Configuración**

Es un registro de 16 bits que guarda la información de toda la configuración del sistema; permite realizar su programación y reconfiguración.

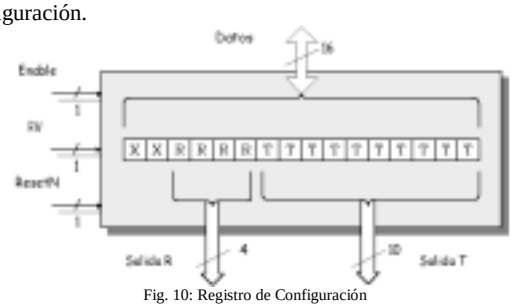

Fig. 10: Registro de Configuración

Está muy relacionado con los contadores (de la "Tabla" y de "Retardos") y el "Divisor", ya que es de éste que se toman los datos de configuración. Los 10 bits menos significativos representan la dirección más alta a la que puede acceder el sistema en la memoria "Tabla". Esto determina la cantidad máxima de muestras de la señal moduladora que la celda puede soportar. Los 4 bits siguientes representan el exponente de 2 de la cantidad de retardos a utilizar. Los 2 bits más significativos quedan sin especificar. El contenido de este registro puede ser modificado o leído por el usuario cuando lo desee.

- **Entrada - Salida**

El bloque "Entrada-Salida" cumple la función de adaptar el bus bidireccional de configuración con los buses direccionales internos. En sentido, desde el "Registro de Configuración" hacia el exterior y desde la "Tabla" hacia el exterior presenta entradas independientes, en cambio, en el sentido desde el exterior a la "Tabla" y al "Registro de Configuración" presenta un mismo bus de salida.

6.2. Unidad de Control (Modos de Funcionamiento)

Disponemos de 3 modos de funcionamiento: "Normal", "Cambiar Configuración" y "Leer Configuración". En el modo "Normal", la Celda procesa los datos del bus de entrada y vuelca los resultados en el bus de datos de salida. El modo "Cambiar Configuración" escribe el Registro de Configuración y los valores de la "Tabla" mientras que el modo "Leer Configuración" permite leer ambos.

Son dos las líneas necesarias para determinar en qué modo debe funcionar el sistema. Una llamada "Modo", asociada a una línea de "Selección" por la cual le indicamos a la Celda que fue seleccionada que debe salir del modo "Normal". La restante señal, llamada "L/E" (Lectura-Escritura) determina en que modo de configuración debe trabajar la Celda, haciendo referencia si deseamos leer o cambiar la configuración.

LE & Modo	***Significado***
B"X0"	Modo Normal
B"11"	Modo Configuración – Cambiar Configuración
B"01"	Modo Configuración – Leer Configuración

Las señales "L/E" y "Modo" son conectadas en paralelo a la Unidad de Control y la Unidad de Proceso, esto permite simplificar la Unidad de Control y determinar el comportamiento del componente "Entrada–Salida" directamente con estas señales, sin la necesidad de que la Unidad de Control intervenga.

Los tres modos de uso comparten la característica de que un valor bajo en "Enable" provoca que el sistema se detenga en el estado donde se detecta el cambio y no pasa al próximo estado si la señal "Enable" no pasa al estado alto. En el reset, los contadores de Retardos y de la Tabla son llevados a la posición inicial. El estado de Reset determina en cual de los modos nos encontramos según las líneas "Modo" y "LE". Luego, del Reset se pasa al conjunto de estados que forman el modo seleccionado por estas líneas.

6.2.1. Modo Normal

Una vez que la Celda se encuentra operando en este modo, espera que un agente externo ubique un valor válido en el bus de entrada y genere un flanco positivo en "Hay Dato" (HD). Cuando esto sucede, el valor del bus es almacenado en el Registro A y queda listo para ser multiplicado con un valor de Tabla. Para continuar con el cálculo, el resultado es guardado en la memoria "Retardos". Luego es sumado en el "Acumulador" para realizar la división final. Una vez que el resultado es calculado y se encuentra en el bus de salida, la Unidad de Control genera un pulso en la línea "Dato Listo" (DL) para

informarle al sistema externo de la finalización del ciclo de operación.

Para llegar a un modelo optimizado se identificaron las operaciones básicas que requieren estados de la Unidad de Control, buscando a su vez el paralelismo entre las mismas, para lograr una reducción de estados o de tiempo necesario para obtener una salida.

La frecuencia máxima de "HD" queda determinada por el número de estados que consume un ciclo de operación. Si lo relacionamos con el reloj del sistema, tenemos que en la máxima frecuencia de "HD", un pulso del mismo abarca **7 ciclos de reloj**. A continuación se muestra el flujo de datos, viendo las tareas llevadas a cabo en cada uno de estos ciclos o estados:

1) Inicialmente, luego del reset, mientras se aguarda por un dato (indicación "HD") en la entrada de datos, se lee el valor de la "Tabla" seleccionado por el "Contador de Tabla".
2) Al obtener el dato de entrada, éste es almacenado en el "Registro A". Al mismo tiempo, se lee el valor más antiguo de todos los retardos almacenados de la memoria "Retardos". Cuando están presentes ambos valores, el de entrada y el de la "Tabla" a la entrada del "Multiplicador", la "Unidad de Control" le indica a este último que comience a realizar el cálculo.
3) En forma paralela a la realización de la multiplicación, se resta el valor leído de la memoria "Retardos" en el estado anterior a la suma total almacenada de los últimos valores ingresados en el "Acumulador". Esta última operación debe realizarse imprescindiblemente antes de almacenar el nuevo retardo. De no cumplirse esta condición perdemos el valor más antiguo, perdiendo uno de los operandos para la resta. Se almacena en el "Acumulador" una nueva suma (suma parcial) a la que habrá que sumarle un nuevo valor.
4) Una vez finalizada la multiplicación, el resultado es volcado al bus para sobreescribir en la memoria "Retardos" al valor más antiguo. Pasando a formar el nuevo retardo del sistema. Es importante señalar que el "Contador de Retardos" no ha cambiado su valor.
5) A continuación, se realiza la suma de este nuevo retardo a la suma parcial almacenada en el "Acumulador", para así obtener la sumatoria del filtro. No es necesario leer el valor almacenado anteriormente, debido a que aprovecha la característica de la RAM embebida; dicha memoria pone en el bus de salida el valor que acaba de escribir, siempre y cuando el reset no esté activo. También, se incrementan los contadores de Tabla y de Retardos, para ubicarlos en la posición adecuada para el próximo ciclo de operación.
6) El último paso que queda por realizar es la división, para así obtener el promedio de los últimos retardos. La "Unidad de Control" le indica al divisor que desplace los bitios a la derecha para obtener la salida del sistema. Dicho corrimiento dependerá de la cantidad de retardos sumados.
7) Finalizado este estado tenemos en el bus de salida el primer resultado válido. Se indica al mundo exterior que el proceso ha finalizado, llevando la línea "DL" a "1". Del último estado pasamos al primero para comenzar con un nuevo ciclo de operación. Este cambio se produce siempre y cuando la señal "HD" contenga un estado lógico "0". Sin esta condición puede ocurrir que se procese el mismo valor repetidas veces.

6.2.2. Modo Cambiar Configuración

Dicho modo permite modificar la cantidad de retardos que posee el filtro, la cantidad de valores a utilizar de la "Tabla" y los propios valores de la "Tabla". Cuando trabajamos en este modo, los datos o parámetros de entrada provienen del exterior a través del bus de configuración. Los datos pasan por "Entrada-Salida" (I/O) directamente a la "Tabla" y al "Registro de Configuración" en forma simultánea. El primer valor corresponde al "Registro de Configuración" y los restantes forman los valores de la "Tabla".

Dos líneas que dialogan con el exterior, "RTR-DR" (Ready to Receive – Data Ready), realizan el handshake en las transferencias de datos de configuración. Este tipo de transferencia con control de flujo por hardware brinda la flexibilidad al sistema de no tener requerimientos de tiempo para la configuración. Los **6 estados**, que implementan la configuración del sistema, pueden agruparse en dos grupos, uno que permite modificar el "Registro de Configuración" y otro que gestiona la escritura de la "Tabla". Ambos grupos manejan el handshake para obtener un dato válido. Estos estados son:

1) El estado inicial realiza las siguientes tareas: informar que está listo para recibir el "Registro de Configuración", resetea el "Contador de Tabla", y espera la indicación de que llegó el valor esperado.
2) Almacena dicho dato en el "Registro de Configuración".
3) Informa al usuario que se ha terminado de almacenar el dato y espera la confirmación de tal situación.
4) Luego de almacenar el "Registro de Configuración", comienza la gestión para cargar los valores en la "Tabla". Para su realización se indica al exterior que está lista para recibir y prepara a la "Tabla" para almacenar el dato que proviene del bus de configuración.
5) Almacenamos en la primer posición de la "Tabla", el valor ingresado.
6) Por último, se incrementa el "Contador de Tabla", notifica al exterior que ya se guardó el dato y espera su confirmación. Cuando se reciba dicha confirmación pasamos al primer estado del subgrupo que permite cambiar los valores de la Tabla. De esta forma el sistema permanece encerrado en 3 estados que permiten modificar todas las posiciones de la memoria "Tabla" que fueron indicadas en el "Registro de Configuración". No hay ningún tipo de control de la cantidad de valores a escribir, lo único que se tiene en cuenta es el valor en el cual se dará vuelta el "Contador de Tabla", por lo que si seguimos escribiendo, comenzaremos a sobrescribir los valores ya almacenados.

Para dar por finalizados los cambios, el usuario debe dejar fijos las líneas del handshake y modificar el modo de funcionamiento mediante el manejo de "L/E" y "Modo".

6.2.3 Modo Leer Configuración

Este modo es muy similar al anterior con la diferencia de que ahora nos permite leer los últimos valores cargados en el sistema, tanto del "Registro de Configuración" como todos los valores cargados en la "Tabla". Para implementar esta funcionalidad, la "Unidad de Control" incorpora 5 estados más.

7. ESTUDIO DE ALGORITMOS DE SUMA Y MULTIPLICACIÓN

Este trabajo incluye la investigación de las estructuras apropiadas para aritmética de punto fijo, es decir el estudio de algoritmos de suma y multiplicación [11,12,13,14,15]. Al analizar la estructura y las operaciones realizadas por el filtro promedio móvil surge la necesidad de probar varios algoritmos, para luego elegir el apropiado para implementar tanto en el bloque "Multiplicador" y como en el bloque "Acumulador" pertenecientes a la arquitectura de la Celda. Nuestro filtro se implementa en FPGA, por lo que, el objetivo de este estudio se centra en encontrar algoritmos que logren un buen desempeño en dicha implementación. En un entorno de diseño lógico, lograr una buena performance o tener buena eficiencia es entendido como la capacidad de implementar una función predeterminada sin desperdiciar recursos como AREA, TIEMPO y POTENCIA. Nuestro objetivo se centra en lograr eficiencia con respecto al área ocupada y el tiempo de cálculo es decir la velocidad de trabajo. No existen reglas generales para lograr un diseño eficiente

en FPGA. De hecho los mejores diseños surgen de los análisis de los resultados obtenidos con diversos modelos.

Luego de un estudio teórico, desde los algoritmos básicos hasta los optimizados para ASIC (Application Specific Integrated Circuit) [14, 15], es importante, analizar la estructura interna de la FGPA VIRTEX, para ver cuáles son las funcionalidades que ésta dispone y cómo éstas pueden aprovecharse adaptando los algoritmos estudiados.

Al implementar cada multiplicador, se toman como datos relevantes de la síntesis: cantidad de slices utilizadas y máximo delay de un camino combinacional, en éste último se verá reflejada la frecuencia máxima de trabajo del multiplicador. A continuación se muestran los resultados obtenidos de aplicar los algoritmos más representativos:

- Array de Carry Save Adder más un Carry Ripple Adder en la suma final: realizado a nivel de componentes, bajo nivel de abstracción.
- Booth-Wallace: realizado a nivel de componentes, algoritmo optimizado para ASIC, bajo nivel de abstracción.
- Log Tree: algoritmo que aprovecha la estructura de la FPGA (cadena de propagación de carry, Carry Logic, y la utilización adecuada de las 4-input LUTs). Posee un nivel de abstracción más alto que los algoritmos anteriores.
- A*B: algoritmo con un alto nivel de abstracción.

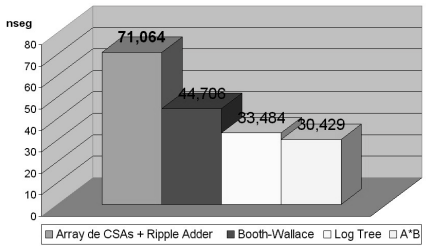

Fig. 11: Máximo delay de un camino combinacional para distintos multiplicadores de 16*16 bits

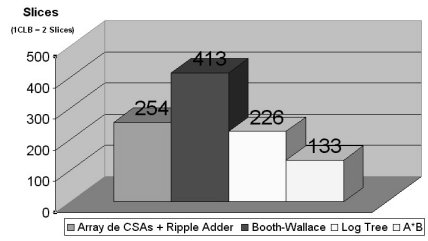

Fig. 12: Cantidad de Sices utilizadas por distintos multiplicadores de 16*16 bits

Al realizar varias pruebas de distintos algoritmos concluimos que los algoritmos optimizados para ASIC no logran un buen mapeo, es decir una buena implementación en FPGA. Es muy importante que el diseñador considere las características de la FPGA al momento de realizar un diseño.

Los resultados más satisfactorios surgieron de implementar tanto el "Multiplicador" como el "Acumulador" sintetizado, en los que se trabaja en un nivel de abstracción más alto, aplicando directamente las operaciones matemáticas suma, resta y multiplicación.

8. PLACA, PASTILLA Y HERRAMIENTA DE SÍNTESIS

La placa utilizada en este trabajo es la XSV100 de Xess [16]. Esta placa está equipada con una PFGA Virtex de 100.000 compuertas. La placa dispone de dos bancos de memoria SRAM independientes de 512K x 16 utilizados por el FPGA para almacenamiento de propósito general. Dispone de un oscilador programable de 100 Mhz.

La herramienta de diseño está ligada al fabricante y modelo de la pastilla que seleccionamos para bajar y probar el diseño [17]. La herramienta de Xilinx utilizada fue ISE 4 (Integrated Syntesis Environment), es una serie de herramientas integradas que permite producir, comprobar e implementar diseños para las FPGAs o CPLDs de Xilinx. Incorpora el programa de síntesis "FPGA Express".

9. SISTEMA DE PRUEBA

El diseño de nuevos componentes (que en algunos casos presentan una complejidad semejante a la Celda), la conexión de cada uno de éstos, la creación de un software adecuado para la interacción de la "Celda" con el usuario y la interconexión entre el PC y el sistema fueron las tareas necesarias para probar el funcionamiento de la Celda. Fue así que surgió el llamado "Sistema de Prueba".

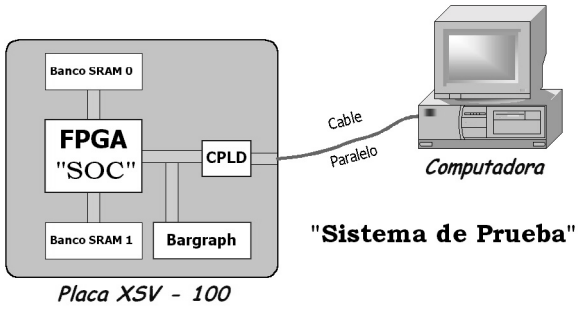

Fig. 13: Implementación del Sistema de Prueba

Por un lado, disponemos de un software corriendo en el PC para controlar la Celda, y por otro la placa con la FPGA. Esta comunicación se resolvió utilizando el puerto paralelo. Por esta vía de comunicación se manejan las líneas de control y se intercambian los datos de configuración.

Los datos de entrada se escriben en uno de los bancos de memoria (banco 0) de la placa XSV-100, mientras que los resultados obtenidos luego del procesamiento de la Celda lo hacen en el banco restante (banco 1). Para monitorear la actividad de la Celda se utiliza el bargraph de la placa.

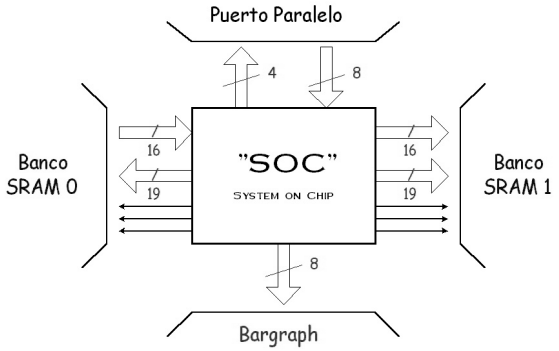

Fig. 14: Interface del SOC

Como se aprecia en la figura 14 quedan definidas tanto la interface para leer datos desde un banco de memoria como para escribir en otro banco de memoria. El bus de salida de 8 bits permite fácilmente encender o apagar 8 de los 10 leds que contiene la barra de leds (Bargraph). El resto de líneas que se aprecian en la figura, forman dos grupos importantes. Uno contiene todas las señales de control y el otro representa la transferencia de datos entre el SOC y el exterior.

Disponemos de 12 líneas de las 16 que brinda el puerto paralelo de la computadora, para formar la comunicación con el SOC. Desde el punto de vista del sistema embebido en la pastilla, las 8 líneas del puerto de Datos (Puerto A o Puerto base) son líneas de entrada y las 4 líneas que utilizamos del puerto Status (Puerto B o Puerto base +1) son líneas de salida.

En la siguiente figura apreciamos claramente que hay 5 bloques que complementan a la Celda.

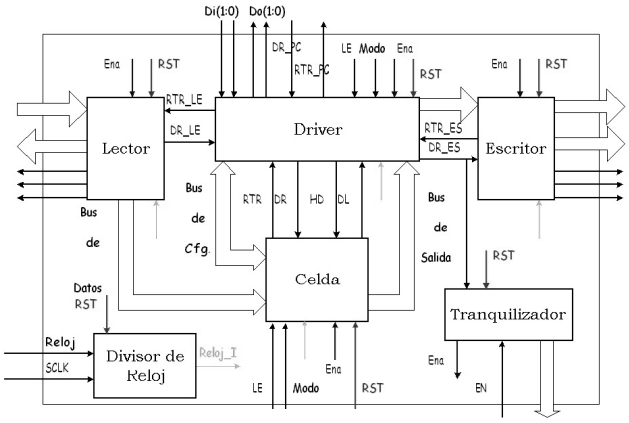

Fig. 15: Componentes del SOC

A continuación se realiza una descripción de cada uno de los bloques:

- **Driver**

El bloque central, llamado "Driver", establece una comunicación con cada uno de sus colegas y maneja la secuencia de eventos adecuados para que todo el conjunto de elementos funcione correctamente. En el modo de funcionamiento "Normal", éste es el encargado de solicitar un dato de memoria, indicarle a la Celda que hay un dato válido para ser procesado, y una vez que la salida de la Celda es actualizada, toma el resultado y realiza las solicitudes correspondientes para el almacenamiento en memoria.

El Driver también incorpora soporte para testear los modos de configuración de la Celda. Permite modificar y leer los parámetros de configuración. Lo interesante es que utiliza mecanismos distintos para el tratamiento de datos a procesar y de configuración. Estos últimos tienen un tamaño de 16 bits, lo que supera ampliamente el ancho de datos que puede viajar en forma paralela por el puerto paralelo. El ancho en bits de los datos enviados en forma paralela entre el PC y el SOC o el SOC y el PC, quedó determinado por la cantidad de líneas libres, luego de asignar líneas a las señales de control. Para solucionar esta dificultad el Driver realiza una traducción de comunicación. Para la transferencia de datos se tomaron 4 líneas, 2 en un sentido y 2 en sentido contrario. Esto determina que para llegar a transmitir 16 bits de datos, hay que realizar 8 envíos de 2 bits, tanto en un sentido como en el otro. También asignamos 2 líneas con sentidos opuestos para manipular el envío de información. Estas líneas de control, DR y RTR, manejan un handshake que establece el control de flujo de la transferencia.

- **Lector y Escritor**

El "Lector" es un dispositivo capaz de leer en forma secuencial uno de los bancos de memoria que tiene la placa. Es el controlador de lectura del Banco 0 de SRAM. Permite tomar una palabra de 16 bits desde el exterior de la FPGA, y capturarla para un posterior proceso. Los datos leídos provienen de posiciones consecutivas de memoria, salvo que el reset actúe provocando que el Lector pase a leer la primera posición. De forma similar, el "Escritor" es un dispositivo que permite escribir datos de 16 bits, en el otro banco SRAM de memoria.

Sabemos que el Lector o el Escritor manejan datos de 16 bits. Estos 16 bits representan el ancho de una posición de memoria SRAM. También sabemos que el bus de datos de la Celda es de 16 bits. Por estos motivos el bus de salida del Lector es el mismo bus de datos de la Celda y no requiere un manejo previo del Driver. Pero, la salida de la Celda es de 32 bits, es decir dos posiciones de memoria. Esto impide conectar directamente la salida de la Celda a la entrada del Escritor y requiere un ajuste previo por el Driver. Dicho ajuste consiste en hacer dos solicitudes de escritura por cada salida de la Celda.

- **Tranquilizador**

El "Tranquilizador" es un indicador que informa al usuario que el sistema está operando, informando también qué porcentaje de datos ha sido procesado. La función de este dispositivo es supervisar que la "Celda" esté operando y mediante la barra de leds (bargraph), indicar el estado de la misma. Además incorpora la tarea de detener al sistema una vez que ha operado. La información mencionada es muy importante no sólo para el usuario final, sino que en momentos de depuración del sistema, toma una trascendencia mayor.

El encender un led significa haber almacenado 64K de posiciones en la memoria de resultados. Un led prendido equivale a haber procesado 32K de valores de entrada. Por lo tanto, cada led representa un 12,5% del total a procesar.

- **Divisor de Reloj**

El "Divisor de Reloj" es un componente complementario que reduce la velocidad del reloj, permitiendo probar la Celda en tiempos razonables. Sin la presencia de este elemento es prácticamente imposible inyectar señales desde el exterior ya que antes de indicarle una orden, la Celda ha terminado de operar. El reloj general del sistema entra al DIV_CLK, dando este último la posibilidad de bajar la frecuencia de dicho reloj, dependiendo de una línea de selección. Este dispositivo distribuye el nuevo reloj del sistema a todos los componentes del SOC, tal que funcionen en forma sincronizada. El único dispositivo que no requiere dicho reloj es el Tranquilizador. Este se alimenta de un "reloj interno" al sistema que equivale a la frecuencia de escritura en el banco de memoria de resultados.

10. HERRAMIENTAS DE SOFTWARE

Parte de este trabajo consta en la elaboración de un software que se ejecuta en un PC, para controlar y programar el banco de filtros. Además de este software, llamado **Filter**, se realizaron otras aplicaciones que se utilizaron como herramientas para probar el buen funcionamiento del filtro. Se diseñaron las siguientes aplicaciones: **ParaPort.exe** y **A2H.exe**, llevadas a cabo en Visual Basic de Microsoft.

- **Filter**

Fig. 16: Filter

Este programa es el encargado de controlar y programar el banco de filtros desde un PC. Permite no solo manipular las líneas de "Enable" y "Reset" sino también permite cargar y leer la configuración de cada uno de los filtros. Esta aplicación cuenta con:

- Consola: para manipular las líneas de status del filtro, éstas son reset, enable, sclock, modo y Lectura/Escritura.
- Browsers: dispone de dos browsers para "navegar" hasta los archivos de configuración tanto de entrada como de salida. Su función es la de cargar y levantar los datos de configuración del filtro.
- Launcher de programas programable: para poder levantar otras aplicaciones, es totalmente configurable, tanto los programas a lanzar como los nombres que representarán a los mismos.

❑ **Paraport**

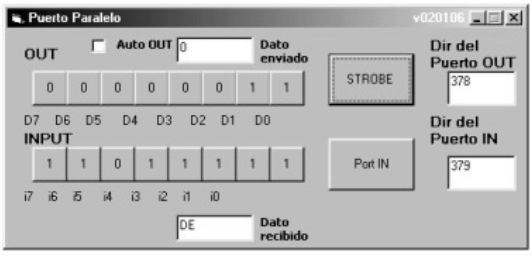

Fig. 17: Paraport

ParaPort es un programa para leer y escribir en los puertos paralelos de un PC. Este cuenta con una interface amigable que permite cambiar las direcciones de los puertos, cambiar las etiquetas que aparecen en pantalla y trabajar con bitios invertidos de forma transparente. También tiene la posibilidad de trabajar de modo "hot-touch" de tal forma que aparentaría estar tocando directamente los bitios del puerto. Todos los datos numéricos que el programa maneja son hexadecimales, excepto por supuesto los botones binarios de los puertos.

❑ **A2H**

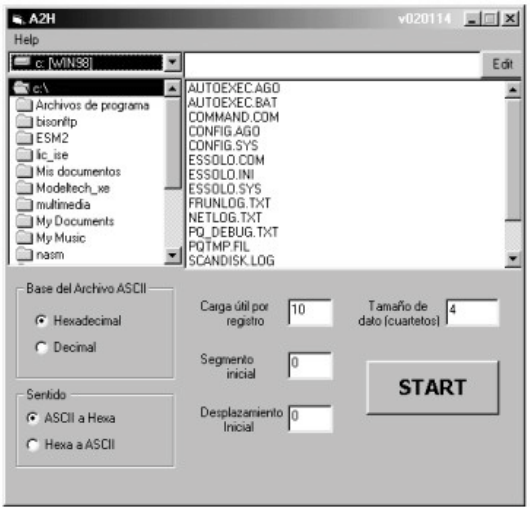

Fig. 18: A2H

A2H es una aplicación diseñada para codificar y decodificar archivos en el formato Hexa de Intel. Un archivo Hexa representa el contenido de una memoria por medio de registros que almacenan la carga útil y mediante encabezados indican las direcciones de memoria donde deberá almacenarse la información. Este software involucra dos archivos, uno Hexa, vinculado con la memoria y otro que se vincula con el usuario escrito en Ascii en el que se escriben línea a línea los datos que se desean cargar en el formato Hexa. Estos datos son considerados con signo y su largo es configurable por el usuario así como también las posiciones de inicio de los segmentos y los desplazamientos. También se puede elegir si los datos de los archivos representarán números hexadecimales o decimales.

11. IMPLEMENTACION EN FPGA

11.1. Resultados de la Síntesis de la Celda

- **Number of Slices**: (1 CLB = 2 Slices)	**438 out of 1,200 36%**
- Number of Slices containing unrelated logic:	0 out of 438 0%
- Total Number Slice Registers:	187 out of 2,400 7%
- Number used as Flip Flops:	139
- Number used as Latches:	48
- Number of 4 input LUTs:	818 out of 2,400 34%
- Number of bonded IOBs:	72 out of 166 43%
- Number of Block RAMs:	8 out of 10 80%
- Number of GCLKs:	4 out of 4 100%
- Number of GCLKIOBs:	1 out of 4 25%
- Total equivalent gate count for design:	139,472
- Additional JTAG gate count for IOBs:	3,504
- The Average Connection Delay is:	2.458 ns
- TheMaximum Pin Delay is:	7.307 ns
- Minimum period:	18.644ns
- **Maximum frequency:**	**53.637 MHz**
- Maximum path delay from/to any node:	20.033ns
- Minimum input arrival time before clock:	11.570ns
- Minimum output required time after clock:	15.361ns

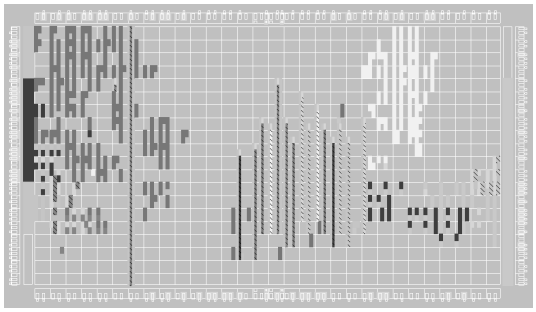

Fig. 19: Place & Route de la Celda

11.2. Análisis de los Resultados Obtenidos

Los resultados se analizan en base a la velocidad de trabajo y al área que ocupa el diseño.

Como muestran los reportes de tiempo, se aprecia que la Celda es capaz de trabajar a una frecuencia máxima de reloj de 53,6 MHz. Entonces si se desea calcular el bit rate máximo que es capaz de procesar, se debe calcular la cantidad de muestras por segundo y multiplicarlo por el tamaño en bits de la muestra. Como se mencionó anteriormente, la Celda demora 7 ciclos de reloj en procesar una muestra, por lo tanto procesa 7,6 muestras por segundo, y considerando que cada muestra es de 16 bits, llegamos a un bit rate de aproximadamente 122,5 Mbps.

Con la velocidad de 122,5 Mbps es posible procesar en tiempo real señales de un gran ancho de banda. Se podría entonces procesar audio y video de alta calidad, e incluso múltiples canales multiplexados, también se podría utilizar para aplicaciones de acondicionamiento de canal para DSL de alta velocidad como ser HDSL.

Otro de los puntos a tener en cuenta es el tamaño que ocupa el diseño. Como se muestra en los reportes, el área que ocupa en cantidad de slices, no es muy grande. Pero en el caso del área

ocupada es muy importante prestar atención al tipo de área, es decir cuales son los recursos que más consume de la pastilla. En el reporte se ve que uno de los recursos que más ocupa es el de RAM embebida. Esto se debe a la gran cantidad de datos guardados en las tablas. En el caso del banco de retardos, para disminuir la cantidad de recursos que ocupa, se puede pensar en disminuir la cantidad de retardos. Con 128 retardos puede llegar a ser suficiente. También se puede reducir la cantidad de bits por retardo, ya que ésta es del doble de tamaño que la señal de entrada.

Otra tabla que se puede reducir tanto en largo como en ancho es la de modulación que permite alojar hasta 1024 muestras de 16 bits. Todos estos cambios reducen la precisión del filtro, por lo que para hacerlos se deben realizar varios estudios estadísticos para determinar que error produce este redondeo, pero es un compromiso de ingeniería que al momento de generar el banco de filtros deberá ser tenido en cuenta.

Con respecto al área ocupada por el diseño también vale destacar que el bloque multiplicador ocupa aproximadamente 1/3 del área total ocupada por los bloques de cálculo y control de la Celda (140 de 438 slices). Por lo que el pensar en tecnologías con bloques dedicados a la multiplicación como la VirtexII podrían optimizar en área este tipo de diseños.

11.3. Sugerencias para el futuro

Uno de los puntos a destacar de este diseño es la velocidad de proceso. Este diseño es realmente rápido para procesar señales, como también para programarse. De todos modos pensamos que se puede trabajar en un pipeline para la arquitectura que podría reducir en algunos ciclos de reloj el ciclo de operación.

Como la velocidad de programación utiliza menos ciclos de reloj que la de proceso, la cantidad de muestras que se pierde durante una programación es muy poca. Incluso si consideramos un banco de filtros de 32 subbandas (64 celdas) como el de MPEG, reprogramarlo todo llevaría solo alrededor de 5ms. Es muy importante destacar que durante estos 5ms sólo una banda por vez estuvo fuera de operación.
Otra maniobra que se puede realizar es la de “reprogramar y reusar”. Esta maniobra tiene como único fin la economización de área de pastilla, lo que busca es utilizar la misma Celda para procesar más de una subbanda. Para conseguir esto tenemos que poder trabajar sustancialmente más rápido que la señal de entrada. Un ejemplo con audio puede ser el siguiente: una señal de entrada de audio a 128kbps, el equivalente a 8k muestras por segundo; el sistema trabaja a 7 Mega-muestras por segundo y tiene una velocidad de programación de 14Mega-muestras por segundo. Como dijimos anteriormente para programar la misma celda 64 veces demoramos 5ms y con la velocidad de la señal de audio tenemos un tiempo entre muestra y muestra de 125ms. Por lo tanto con una única celda podríamos procesar alrededor de 10 señales de audio de 128kbps. Aclaramos que para esto se deberían realizar modificaciones en los bancos de “Retardos” ya que estos tienen que ser independientes uno de los otros, en realidad sólo el bloque de cálculo y las “Tablas” de modulación son las que se reusan.

Maniobras como estas, las sugerencias de redondeos numéricos, reducción de cantidad de retardos, pipelines, etc. son nuestras recomendaciones a los futuros diseñadores que, a la hora de customisar la Celda, buscarán la optimización de la misma, tanto si conservan a Xilinx como plataforma o si migran a otra.

12. CONCLUSIÓN

Este artículo presenta el desarrollo de una arquitectura, en una plataforma hardware reconfigurable, que permite ejecutar dos etapas pertenecientes a un filtro pasabanda que forma parte de un banco de filtros. Teniendo por un lado una etapa moduladora y una etapa de filtrado, utilizando en este último caso un filtro promedio móvil. La implementación de estas dos etapas es llamada como “Celda”. Es programable por el usuario tanto la cantidad de muestras correspondiente a la señal portadora, como la cantidad de retardos pertenecientes al filtro promedio móvil.

En conjunto a este desarrollo fueron necesarias tres tareas fundamentales. En primer lugar, la elaboración de herramientas de software para poder controlar y programar el banco de filtros, como también herramientas para probar el buen funcionamiento del filtro y para poder utilizar correctamente los bancos de memoria de la placa. Además, adquiere gran importancia la necesidad de crear un sistema de prueba, útil para la puesta en marcha de la “Celda” y poder comprobar su correcto funcionamiento. En este caso, el diseño de nuevos componentes presenta una complejidad semejante a la elaboración de la arquitectura de la “Celda”. Por último, realizado como un estudio en paralelo, este trabajo incluye la investigación de las estructuras apropiadas para aritmética en punto fijo. Se realiza un estudio de algoritmos de suma y multiplicación para luego analizar su implementación en FPGA. Aspecto de suma importancia por la presencia de un bloque multiplicador en la arquitectura de la “Celda” y de la necesidad de implementación de suma y resta. Concluyendo en este estudio, que los algoritmos optimizados para ASIC no logran un buen mapeo, es decir una buena implementación en FPGA, siendo de gran importancia, la necesidad de considerar las características de la FPGA al momento de realizar un diseño.

Fue así que se obtuvo un modelo VHDL sintetizable de la arquitectura de la “Celda” en una placa de la empresa Xess, XSV100, que cuenta con una FPGA Virtex de 100.000 compuertas. Los resultados se analizaron en base a la velocidad de trabajo y al área que ocupa el diseño. Obteniendo que la “Celda” es capaz de trabajar a una frecuencia máxima de reloj de 53,6 MHz ocupando un área de 438 Slices (1 CLB = 2 Slices).

13. REFERENCIAS

[1] DAVIS, Pan 1995. *A Tutorial on MPEG/Audio Compression.* Motorola Inc.

[2] VAIDYANATHAN, P.P. 1993. *Multirate Systems and Filter Banks.* Estados Unidos. Prentice-Hall Inc.

[3] VAIDYANATHAN, P.P. 1998. *Filter Banks in Digital Communications.* Dep. of Electrical Engineering, California Institute of Technology, Pasadena, CA.

[4] CHOU Chi-Jui, MOHANAKRISHNAN, Satish, EVANS, Joseph B., 1993. *FPGA Implementation of Digital Filters*, Proc. Int. Conf. Signal Proc. Appl. & Tech. (ICSPAT’93).

[5] EVANS Joseph B. 1994. *Efficient FIR Filter Architectures Suitable for FPGA Implementation*, IEEE Trans. Circuits & Systems.

[6] TERES, Luis; TORROJA, Yago; OLCOZ, Serafín; VILLAR, Eugenio. 1998. *VHDL Lenguaje Estándar de Diseño Electrónico.* Madrid, McGraw-Hill Interamericana de España, S.A.U.

[7] SMITH, Douglas J. 1996. *HDL Chip Design. A practical guide for designing, synthesizing and simulating ASICs and FPGAs using VHDL or Verilog.* Estados Unidos. Doone Publications.

[8] LIMKEMANN, Steve 1996. Fixed Point Math Tutorial : *Fun With Fixed Point Values.*

[9] MANO, M. Moris. 1982. *Arquitectura de Computadores.* México. Prentice-Hall Inc.

[10] HENESSY, J.L., PATTERSON D.A. Computer Architecture: A Quantitative Approach. Morgan Kaufmann. San Francisco, 1995.

[11] BOEMO, E.I.; JUÁREZ, E.;MENESES J. 1993. *Taxonomía de Multiplicadores.* Dto. de Ingeniería Electrónica. ETSI Telecomunicación. Madrid. España. Universidad Politécnica de Madrid.

[12] MURRA, Fabio. *Optimal Arithmetic Circuits on FPGAs.* [TESIS]. Cork. Irlanda. Department of Electrical Engineering and Microelectronics. University College Cork. 2000. pp. 197.

[13] CHANG K.C. *Digital Systems Design with VHDL and Synthesis.* IEEE Computer Society.

[14] BEWICK, Gary W. *Fast Multiplication: Algorithms and Implementation.* Stanford. Estados Unidos. Stanford University. 1994. pp 170.

[15] BOOTH, A.D. *A Signed Binary Mulltiplication Technique.* Quarterly J. Mechanical Applications in Math, vol. 4, part 2, pp. 236-240. 1951.

[16] XESS Corp. 2001. *XSV Board V1.1 Manual.* Version 1.1.

[17] XILINX Inc. *Virtex 2.5 V FPGAs Datasheet.* April 2, 2001.