Reconfiguración Parcial

Xilinx

Edwin Jahir Rueda Rojas
Ingeniería de Sistemas
UIS
Bucaramanga-Santander
ejrueda95@hotmail.com

Angelica Mercedes Basto Garcia
Ingeniería de Sistemas
UIS
Bucaramanga-Santander
angelica.basto@hotmail.com

***Resumen*: Xilinx es una empresa tecnológica estadounidense reconocida por inventar los FPGAs, estos se basan en dos tipos de reconfiguración parcial: basados en módulos y basados en diferencias. La reconfiguración parcial basada en modulo utiliza los conceptos de diseño modular para reconfigurar grandes bloques de la lógica. Por otro lado la reconfiguración parcial basada en la diferencia es un método para hacer pequeños cambios en el diseño de un FPGA, como lo es cambiar las normas I/O y el contenido de bloques de RAM [1].**

***Abstract:* Xilinx is a American technology company recognized for invent the FPGAs, these are based on two types of partial reconfiguration: module-based and difference-based. Module-based partial reconfiguration uses modular design to reconfigure large blocks of logic. Module-difference partial reconfiguration is a method to make small changes in the design of an FPGA, as is change the rules I/O and the content of blocks of RAM [1].**

Keywords—FPGAs, PR, PRC, FPGA_Editor, NCD, LUT, UCF, INIT, PRR., JTRS, SEU, SDR.

INTRODUCCIÓN

Xilinx es la empresa tecnológica pionera en la reconfiguración parcial, esta tecnología permite a los diseñadores cambiar la funcionalidad de los FPGAs aun cuando estos estén en marcha, lo que elimina la necesidad de reconfigurar completamente y volver a establecer vínculos, mejorando dramáticamente la flexibilidad que ofrecen los FPGAs.

Los tipos de reconfiguración, basados en reconfiguración parcial propuestos por Xilinx nos permiten hacer cambios en el diseño de cualquier FPGA, cada uno de los dos tipos se ajusta a cierto tipo de cambio, así mismo cada uno se usa para una labor diferente, module-based o basado en modulo se usa para reconfigurar grandes bloques de lógica, no obstante hay que tener en cuenta que los FPGAs deben ser diseñados previamente con módulos reconfigurables para que cumplan con criterios específicos para poder utilizarse en ellos la reconfiguración parcial. Por otro lado difference-based o basado en diferencia es usado para hacer cambios pequeños en los diseños de FPGAs. Cabe resaltar que los FPGAs que soportan la reconfiguración parcial del Sistema, permiten que una parte del diseño sea programable mientras el resto de partes siguen funcionando.

ESTADO DEL ARTE

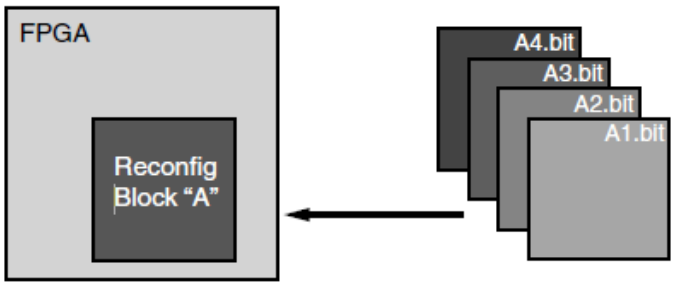
Xilinx en su anuncio del 4 de Junio de 2016 da a conocer el controlador de reconfiguración parcial (PRC), IP proporciona funciones de gestión para los diseños de reconfiguración parcial. Cuando se producen eventos de activación de hardware o software, PRC extrae flujos de bits parciales de la memoria y los entrega a un Puerto de acceso de configuración interna (ICAP). El PRC también asiste con el desacoplamiento lógico y la puesta en marcha de eventos adaptables por reconfiguración parcial.

Algunas características y ventajas claves del nuevo controlador de reconfiguración parcial es que soporta hasta 32 sockets virtuales (particiones reconfigurables y periferia) y 128 módulos reconfigurables por el socket virtual. También tiene la opción de reestablecer opcionalmente los módulos reconfigurables después de cargarlos.

RECONFIGURACIÓN PARCIAL

La tecnología FPGA ofrece flexibilidad a la hora de programar y re-programar sin tener que hacer una re-fabricación con un diseño modificado. La reconfiguración parcial PR, realiza esta etapa a un paso, permitiendo la modificación de un diseño de FPGA operativo cargando un archivo de configuración parcial, por lo general es un archivo BIT parcial. Después que un archivo de BIT configura la FPGA, archivos BIT parciales se pueden descargar para modificar regiones reconfigurables en la FPGA sin comprometer la integridad de las aplicaciones que se ejecuten en las partes donde no se está reconfigurando.

En la siguiente imagen se puede ver el detrás de la reconfiguración parcial [2]:



Como podemos ver, la figura representa la reconfiguración implementada en el bloque A la cual es modificada por alguno de los archivos de bit parciales, A1.bit, A2.bit, A3.bit, o A4.bit. La lógica del diseño FPGA se divide en dos partes a tener en cuenta, una es la lógica reconfigurable que se representa mediante el bloque de color negro "Reconfig Block A", la otra es la lógica estática que la representa el color gris que es la que permanece en funcionamiento y no es afectada por la carga del archivo BIT parcial. Cabe resaltar que la lógica reconfigurable se sustituye por el contenido del archivo BIT parcial.

RECONFIGURACIÓN PARCIAL BASADA EN DIFERENCIA

Una característica importante en las arquitecturas Virtex ™ es la capacidad de reconfigurar una porción de la FPGA mientras que el resto del diseño es todavía operativo. La reconfiguración parcial es útil para aplicaciones que requieren la flexibilidad para cambiar partes de un diseño sin tener que reconfigurar completamente todo el dispositivo. Con esta capacidad, nuevas áreas de aplicación son posibles:

- En el campo de las actualizaciones de hardware y actualizaciones de los sitios remotos.
- Reconfiguración en tiempo de ejecución

Por otro lado nuevos beneficios potenciales se incluyen:

- Reducción del conteo de dispositivos
- Reducción del consumo de energía
- Un uso más eficiente del espacio disponible en la board

La reconfiguración parcial basada en diferencia es útil para hacer pequeños cambios en la marcha de los parámetros de diseño tales como ecuaciones lógicas, parámetros de filtro, y estándares I/O [3]. Este flujo de diseño no se recomienda para hacer grandes cambios en la funcionalidad o la estructura de un diseño, por ejemplo en el cambio de todo un algoritmo.

El objetivo principal de la reconfiguración parcial basada en la diferencia es permitir los pequeños cambios en el diseño, por ejemplo hacer cambios en la programación LUT los cuales tienen que ser cambiados y cargados. Estos cambios se pueden hacer fácilmente editando directamente el archivo NCD mediando un editor de FPGAs como lo es FPGA_Editor de Xilin. Por otro lado si el contenido del bloque de memoria RAM necesita ser modificado, la utilidad Data2MEM se puede utilizar en reemplazo del FPGA_Editor.

Una vez hechos los cambios, el programa BitGen es usado para producir un flujo de bits que diferencia el diseño original del nuevo diseño, este flujo de bits puede ser mucho más pequeño que el flujo de bits inicial. Estas corrientes de bits se cargan fácilmente mediante el software, lo único que se requiere es como saber hacer los cambios lógicos mediante la aplicación FPGA_Editor, y las opciones pertinentes para seleccionar en BitGen.

A continuación se mostrarán ciertos ejemplos de cambios utilizando la reconfiguración parcial basada en diferencia:

Cambios en las ecuaciones LUT: el elemento lógico más pequeño que se puede seleccionar es la rebanada. Primero, el bloque debe mostrarse. Una rebanada individual se puede encontrar usando el botón "Find" de la parte derecha de la ventana. Después la rebanada seleccionada, (marcada de rojo en la figura 1) se oprime el botón Editblock para abrir el editor de bloques de la barra de herramientas [4].

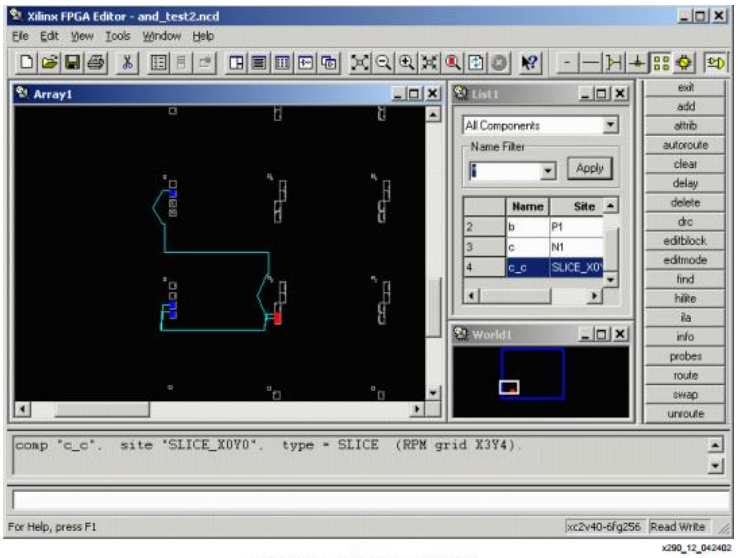


Figure 1: **Viewing a Block**

Para evitar modificaciones accidentales, por defecto, la parte interna de una rebanada no se puede modificar. Cada vez que se abre un bloque, para que sea editable, el botón de edición "Edit" se debe seleccionar. Esto cambia el color de las ventanas a negro.

Para ver las ecuaciones LUT, hay que hacer clic en el botón Show/Hide. El botón "F=" de la barra de herramientas abre el panel en la parte inferior de la ventana con el nombre de la rebanada, y las dos ecuaciones. Los operadores validos son:

* -> Logical AND

+ -> Logical OR

@ -> Logical XOR

~ -> Unary NOT

Figure 2 shows changing the Geqn from A3*A2 to A3*~A2.

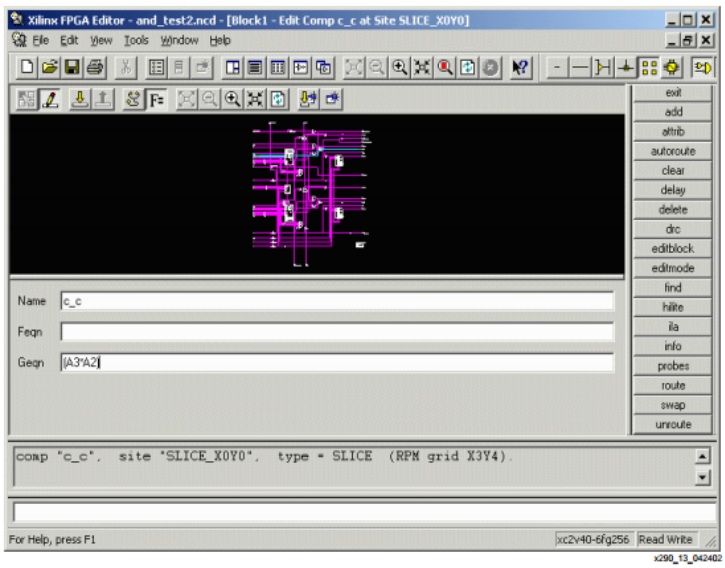


Figure 2: **Changing LUT Equations**

Los valores válidos para las ecuaciones son A1, A2, A3, y A4, representando las cuatro líneas de dirección de entrada a la LUT. Los paréntesis se pueden usar to agrupar secciones de ecuaciones, por ejemplo, (A4*A1)@~A3. Cualquier otro nombre u operadores generaran algún tipo de error, por ejemplo:

ERROR: FPGAEDITOR: 24 - "(A3~A2 + mynet) is not a valid value for the Geqn attribute.*

Después de que los atributos son cambiados, los cambios se guardan y se cierran las ventanas del editor de bloques [5].

Cambios en los contenidos de bloques RAM: El editor de bloques "Block Editor" de los bloques RAMs (figura 3) es similar a las rebanadas o segmentos del "Block Editor". Una vez se despliega la ventana, con el botón Show/Hide "Mostrar/Ocultar" se muestra el contenido de la memoria RAM. El formato que se maneja para los datos es con restricción INIT en un archivo UCF. Una vez hechos los cambios, estos se guardan y se cierra la ventana para volver a la vista de la matriz [6].

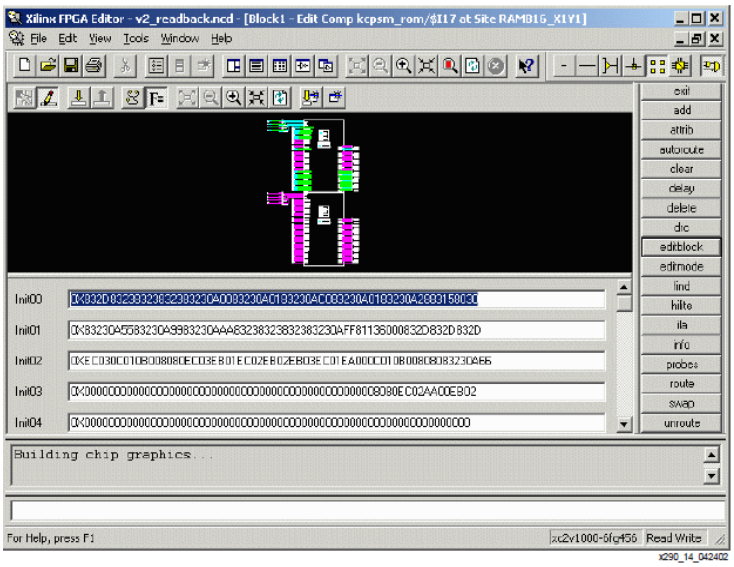
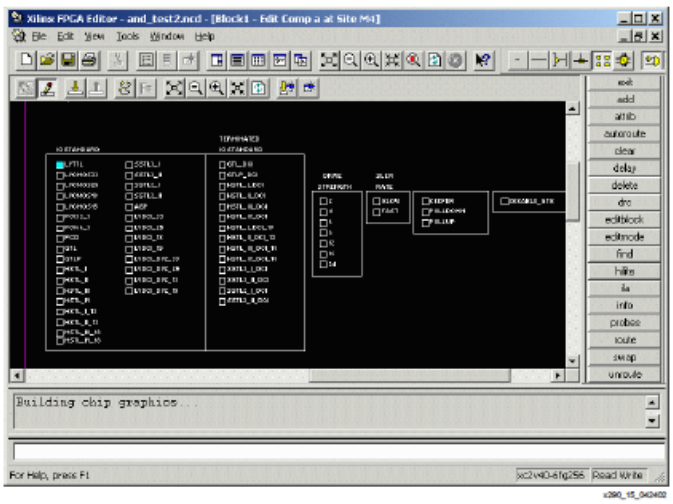


Figure 3: **Changing Block RAM Contents**

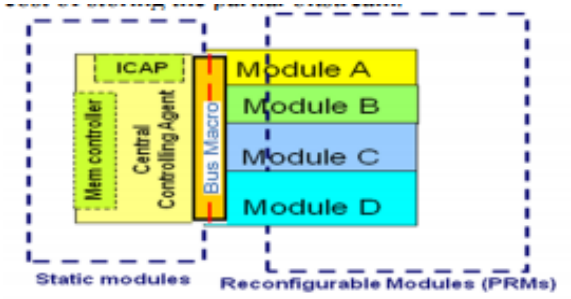
Cambios en los estándares I/O: Los cambios en los estándares I/O se efectúan parecido a los anteriores con el Block Editor. Los estándares I/O aparecen en la parte superior de la ventana del FPGA Editor como lo muestra la figura 4. [5] Para poder cambiar los estándares I/O, estos se deben elegir de las cajas de la parte derecha de la ventana del FPGA Editor, estos estándares especifican por ejemplo la velocidad de entrada y salida de los puertos. Hay que tener en cuenta que a la hora de modificar los voltajes de los estándares I/O en las FPGAs se debe tener en cuenta que concuerden con los estándares I/O de los otros FPGAs con los cuales se va a interconectar o si no podrían surgir problemas en la conexión [7].



Creando flujos de reconfiguración parcial basado en diferencia: El comando *–g ActiveReconfig:Yes* es requerido para activar la opción de reconfiguración parcial, lo que significa que el dispositivo sigue en funcionamiento mientras el nuevo flujo de bits parcial se está descargando. Si este comando no se especifica, la reconfiguración parcial no queda activada por lo tanto los cambios hechos no se aplicarán a los FPGAs. Por otro lado si se quiere trabajar en el modo SelectMAP es indispensable activar su modo con el comando: *-g persist:Yes.* Por ultimo también se activa la seguridad con el comando *–g security:none* para así poder empezar el flujo de bits.

RECONFIGURACIÓN PARCIAL BASADA EN MÓDULOS

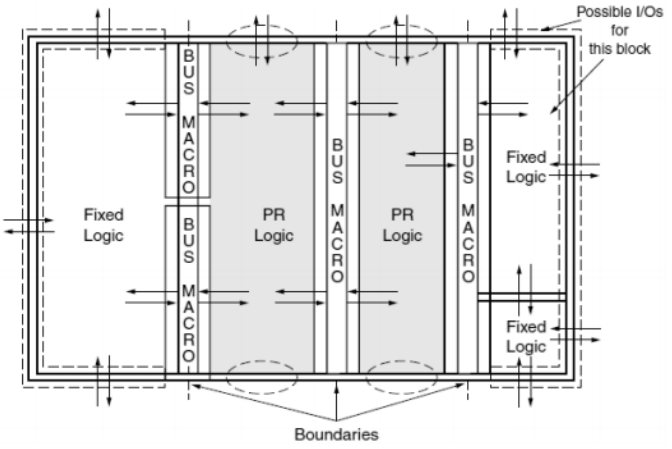
La reconfiguración parcial basada en módulos se utiliza para hacer cambios en las distintas partes del diseño de los módulos. El problema de este diseño de flujo es que el flujo de bits de reconfiguración parcial solo puede ser colocado en una región determinada, durante la reconfiguración dinámica si la región de configuración parcial está ocupada por otro módulo de reconfiguración parcial, un nuevo archivo parcial de flujos de bits debe ser generado. Esto incrementa el costo de almacenar el flujo de bits parciales. Figura 4 [8]



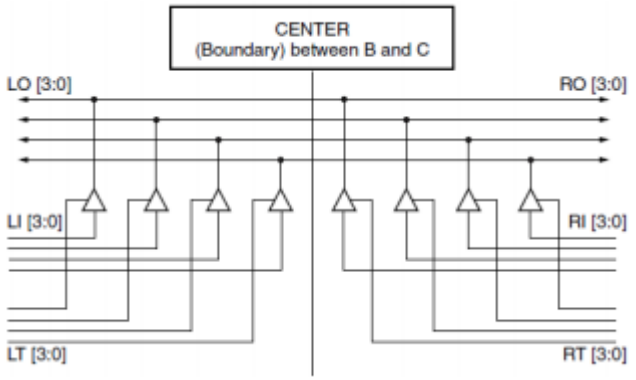
Estos módulos FPGA se dividen en dos partes: módulos estáticos y módulos reconfigurables. Los módulos reconfigurables son los que se muestran en la figura 4 como (A, B, C, D, etc.) son independientes del diseño de entrada que no tienen que estar activos durante toda la ejecución de la aplicación.

Principalmente existen dos grandes deficiencias en el método de reconfiguración parcial basado en módulos. El primero es que cada PRR (Regiones Parcialmente Reconfigurables) debe abarcar todo la talla del dispositivo, esto puede forzar a la lógica de modulo a ser puesta y enrutada en un pequeño sector del dispositivo, lo que puede generar o dar lugar a vías de enrutamiento largas dentro del módulo. Por otro lado si se le asigna una región más grande que la requerida para poner el monto de lógica en el módulo, se estaría infrautilizando el área disponible. Un estudio realizado por Kalte et al. [9] mostró que el camino crítico se puede incrementar tanto como el doble de lo que una aplicación óptima podría tardar si un módulo estuviera lleno en su región más estrecha.

La segunda deficiencia de la reconfiguración parcial basada en módulos es que los módulos solo pueden comunicarse entre sus adyacentes. Como se muestra en el ejemplo, la disposición del diseño proporcionada en la nota de aplicación, la siguiente figura. [10]



Todas las comunicaciones entre módulos pasan por los llamados "Bus Macro" colocados entre PRRs y PRRs adyacentes o regiones estáticas de módulos. Un ejemplo se muestra en la siguiente figura la cual se produce desde las notas de aplicación. Los Bus Macro son implementados usando tres estados "tree-state" (es una marca registrada de semiconductor nacional) los cuales abarcan el ancho horizontal del dispositivo [11].



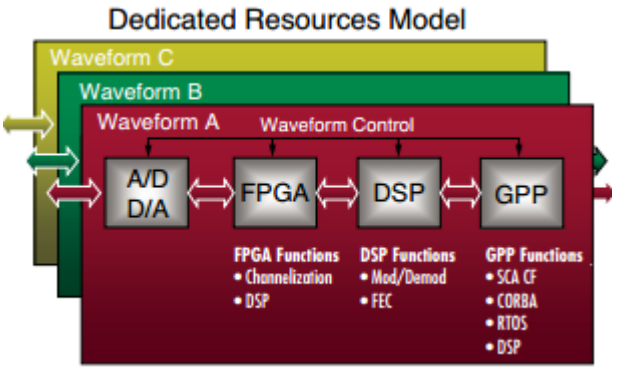
Los Bus Macros son esenciales ya que aseguran que los puertos I/O de cada módulo que se van a colocar en cada lugar puedan conectarse con la misma ubicación.

Estos módulos reconfigurables tienen ciertas propiedades, a continuación se mencionaran algunas de ellas:

- El ancho del módulo oscila entre un mínimo de cuatro cortes a un ancho máximo de todo el dispositivo, este se incrementa de a cuatro divisiones.
- Para ayudar a minimizar los problemas relacionados con la complejidad del diseño, el número de módulos reconfigurables debe ser minimizado (idealmente, un solo módulo reconfigurable, si es posible). Dicho esto, el número de cortes dividido por cuatro columnas es el único límite real para el número de módulos reconfigurables definidos por regiones.
- Un módulo reconfigurable de límite no puede ser cambiado. La posición y la región ocupada por un único módulo reconfigurable siempre es fija.
- Los módulos reconfigurables pueden comunicarse con otros módulos, utilizando un bus especial.
- La implementación debe estar diseñada de manera que las partes estáticas del diseño no dependan del estado del módulo en la reconfiguración mientras la reconfiguración está teniendo lugar. La implementación debe garantizar un funcionamiento correcto del diseño durante el proceso de reconfiguración.
- El estado de los elementos de almacenamiento dentro del módulo reconfigurable son conservados durante y después del proceso de reconfiguración. Los diseños pueden aprovechar este hecho para utilizar el "estado previo" de la información después de una nueva configuración de carga.

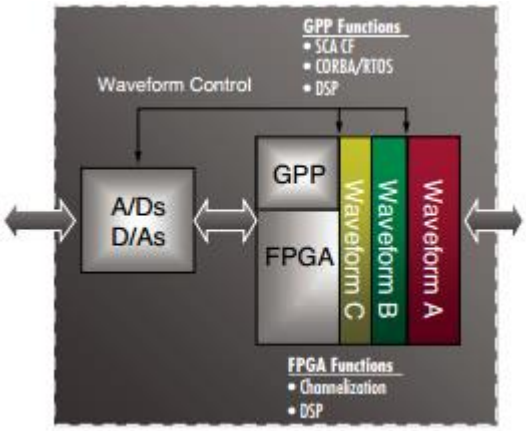
Otras aplicaciones de la reconfiguración parcial a nivel global es el camino hacia el poder-eficiente. A través del programa JTRS, los SDRs se están convirtiendo en una realidad para las industrias de defensa como un instrumento eficaz y necesario para la comunicación. SDRs satisface el estándar JTRS por tener tanto un entorno operativo software- reprogramable como la capacidad de soportar múltiples canales y redes simultáneamente.

La siguiente figura muestra un modem SDR de tres canales que soporta un software de arquitectura de comunicaciones core framework (SCA CF), según lo dispuesto por JTRS [12].



Las implementaciones actuales de módems SCAenable SDR con múltiples canales requieren múltiples conjuntos de procesamiento de recursos y un conjunto de hardware dedicado para cada canal. Todos los canales SDR deben dar soporte, los recursos dedicados necesitan más. Esto afecta negativamente el espacio, el peso, el consumo de energía y el coste.

Con reconfiguración parcial, la capacidad de implementar un modem SDR usando recursos compartidos se realiza, como se muestra en la siguiente figura: [13]

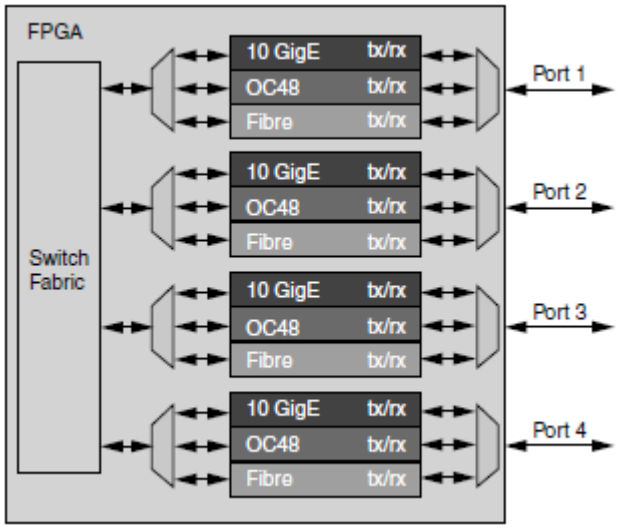


Un modelo de recursos compartidos habilitado por la reconfiguración parcial de un FPGA para soportar múltiples formas de onda puede ser apoyado por el SCA según lo ordene JTRS. FPGA implementa cosas de SDR, con la reconfiguración parcial, da como resultado un uso eficaz de los recursos, menor consumo de energía y ahorro de costes extensos.

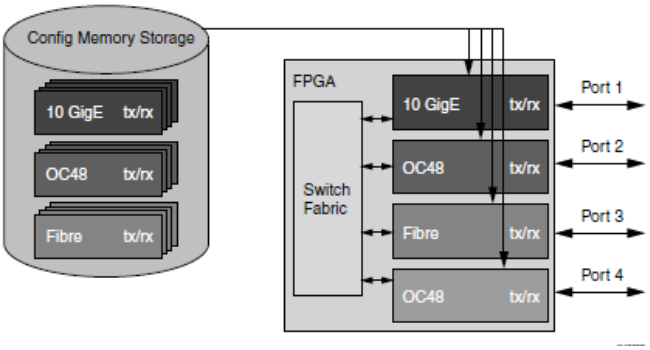
La reconfiguración parcial también se puede utilizar en muchas otras aplicaciones. Otro ejemplo se encuentra en la mitigación y la recuperación de los trastornos de un solo evento (SEU). Dentro de la órbita, el espacio base, las aplicaciones tienen una alta probabilidad de experimentar SEUs. Mediante la realización de reconfiguración parcial, conjuntamente con lectura de retorno, un sistema puede detectar y reparar SEUs en la memoria de configuración sin interrumpir sus operaciones o reconfigurar completamente la FPGA.

Una aplicación común de la reconfiguración parcial es la del multi-puerto de interfaz de red. La reconfiguración parcial optimiza las aplicaciones FPGA tradicionales por el tamaño de la reducción, peso, potencia y coste. Funciones independientes del tiempo pueden ser identificadas, aisladas e implementadas como módulos reconfigurables y enchufar y desenchufar de un único dispositivo, según sea necesario.

Un ejemplo típico es un conmutador de red. Los puertos del switch pueden soportar múltiples protocolos de interfaz; sin embargo, no es posible que el sistema pueda predecir que protocolo se utilizará antes de que se configure el dispositivo FPGA. Para asegurar que el dispositivo FPGA no tenga que volver a configurar y de este modo volver a desactivar todos los puertos, cada posible protocolo de interfaz se implementa para cada puerto, como se ilustra en la siguiente figura [14].



Este es un diseño ineficiente debido a que sólo uno de los estándares para cada puerto está en uso. La reconfiguración parcial permite un diseño más eficiente al hacer cada una de las interfaces de puerto de un módulo reconfigurable como se muestra en la siguiente figura [15]. Esto también elimina los elementos MUX necesarios para conectar varios motores de protocolo a un puerto.



Una amplia variedad de diseños se pueden beneficiar de esta premisa básica. Software Defined Radio (SDR), por ejemplo, es una de las muchas aplicaciones que tiene una funcionalidad mutuamente excluyente y que ve una mejora dramática en la flexibilidad y el uso de recursos cuando se multiplexa era funcionalidad.

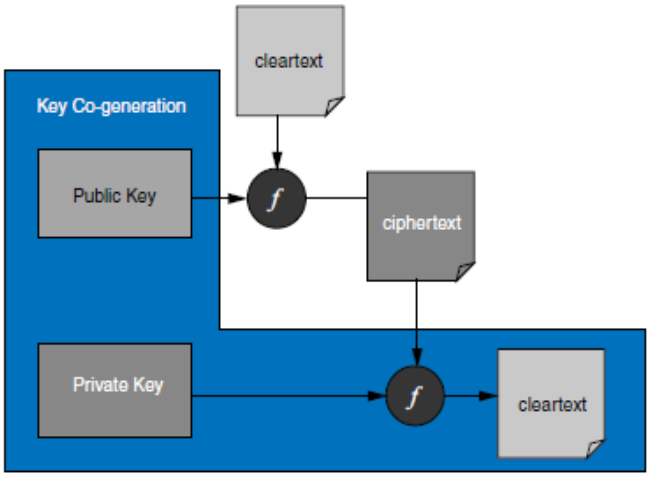
Hay ventajas adicionales al implementar un diseño parcialmente reconfigurable el cual no es la eficiencia. En el ejemplo anterior, un nuevo protocolo se puede apoyar en cualquier momento sin afectar a la lógica estática, la estructura de conmutación en el ejemplo. Cuando se carga un nuevo estándar para cualquier puerto, los otros puertos existentes no se verán afectados de ninguna manera. Normas adicionales pueden ser creadas y se añaden a la biblioteca de memoria de configuración sin requerir un rediseño completo. Esto permite una mayor flexibilidad y fiabilidad con menos tiempo de inactividad para la matriz de conmutación y los puertos. Un

módulo de depuración se podría crear de modo que si un puerto estaba experimentando errores, un puerto no utilizado se podría cargar con el análisis / lógica de corrección para solucionar el problema en tiempo real.

En la figura anterior del ejemplo, un único archivo BIT parcial debe ser generado para cada ubicación física única que podría ser el blanco de cada protocolo. Archivos parciales BIT están asociados con una región explicita en el dispositivo.

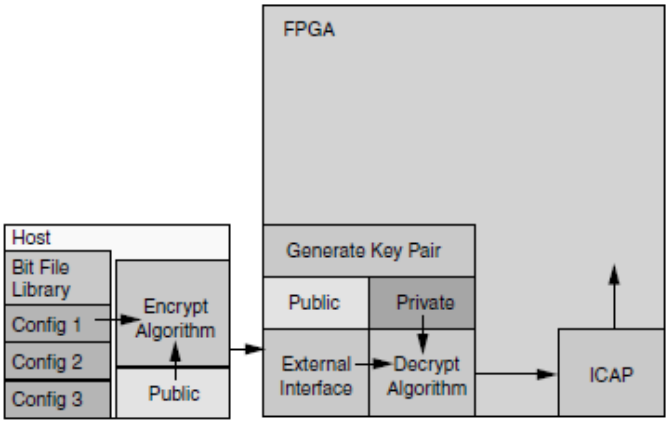
Otro ejemplo de aplicación de la reconfiguración parcial es la clave de cifrado asimétrico [16], un método muy seguro para proteger el archivo de configuración de la FPGA, esto sucede cuando se combina la arquitectura de la reconfiguración parcial y la criptografía asimétrica.

En la siguiente figura [16], todas las funciones del recuadro azul se pueden implementar dentro del paquete físico de la FPGA. La información sin cifrar y la clave privada nunca salen de un contenedor bien protegido.



En una implementación real de este diseño, el BIT de archivo inicial es un diseño sin cifrar que no contiene ninguna información de propiedad. El diseño inicial sólo contiene le algoritmo para generar el par de claves pública-privada y las conexiones de la interfaz entre el host, la FPGA y el ICAP.

Una vez cargado el archivo de bits inicial, el dispositivo FPGA genera el par de claves pública y privada. La clave pública se envía al host que se utiliza para cifrar un archivo bit parcial. El archivo bit parcial cifrado se descarga en el dispositivo FPGA donde se descifra y se envía al ICAP para volver a configurar parcialmente el dispositivo FPGA como se muestra en la siguiente figura [17].



El archivo de bit parcial podría ser la mayoría gran mayoría del diseño FPGA con la lógica en el diseño estático consumiendo un pequeño porcentaje de los recursos globales de la FPGA.

Este esquema tiene incorporadas varias ventajas:

- El par de claves pública-privada puede ser regenerado en cualquier momento. Si una nueva configuración se descarga desde el host puede ser encriptada como una clave pública diferente. Si el dispositivo FPGA está configurado con el ismo archivo bit parcial, como después del encendido, un par de clave pública diferente se usa a pesar de que es el mismo archivo bit.
- La clave privada se almacena en la SRAM. si el dispositivo pierde poder la clave privada deja de existir.
- Incluso si el sistema es robado y el dispositivo FPGA permanece accionado, es extremadamente difícil de encontrar la clave privada porque se almacena en el tejido de propósito general de la FPGA. No se almacena en un registro especial. El diseñador ha podido localizar manualmente cada bit de registro que almacena la clave privada en regiones remotas y físicamente no relacionadas.

CONCLUSIONES

La reconfiguración parcial nos permite modificar un diseño FPGA mediante la carga de un archivo de configuración parcial. En el mercado de la reconfiguración parcial la empresa estadounidense Xilinx es pionera en el desarrollo de dos métodos de reconfiguración parcial los cuales cumplen ciertas funciones, y cada uno de ellos fue diseñado para que se desempeñara en cierto tipo de situación. Por un lado está la reconfiguración parcial basada en la diferencia la cual es utilizada para hacer pequeños cambios en el diseño de la lógica, como lo es cambiar la programación LUT o algún estándar de I/O, este estilo de reconfiguración se utiliza en ciertas aplicaciones las cuales solo necesitan algunos cambios pequeños, como en el ejemplo que se vio anteriormente sobre la interfaz multi-puerto de red. Y por el otro lado está la reconfiguración parcial basada en módulos la cual permite efectuar cambios en ciertas partes de diseño lógico, pero por causas de ineficiencia de los resultados la reconfiguración parcial basada en módulos ha ido desapareciendo dando un amplio lugar a la reconfiguración basada en diferencia.

REFERENCIAS

[1] Benefits of Partial Reconfiguration, Xilinx 2005, page 66.

[2] Xilinx, Partial Reconfiguration User Guide, UG702 (v14.1) April 24, 2012, page 8.

[3] Benefits of Partial Reconfiguration, Xilinx 2005, page 66.

[4] Xilinx, XAPP290 (v2.0) December 3, 2007, Difference-Based Partial Reconfiguration, page 3.

[5] Xilinx, XAPP290 (v2.0) December 3, 2007, Difference-Based Partial Reconfiguration, page 4.

[6] Xilinx, XAPP290 (v2.0) December 3, 2007, Difference-Based Partial Reconfiguration, page 5.

[7] Xilinx, XAPP290 (v2.0) December 3, 2007, Difference-Based Partial Reconfiguration, page 6.

[8] Mosule Based And Difference Based Implementation of Partial Reconfiguration On FPGA, Vol.1, Trailokya Nath Sasamal*, Rajendra Prasad**, page 1900.

[9] Kalte, H., Lee, G., Porrmann, M. and Rückert, U. Study on column wise design compaction for reconfigurable systems. In IEEE international Conference o Field-Programmable Technology, pages 413-416, Brisbane, Australia, 2004.

[10] Generating the Communications Infrastrucutre for Module-Based Partial Reconfiguration of FPGAs, Shannon Koh, Bachelor of Computer Science (Honours), UNSW Sydney, 2003, page 33.

[11] Generating the Communications Infrastrucutre for Module-Based Partial Reconfiguration of FPGAs, Shannon Koh, Bachelor of Computer Science (Honours), UNSW Sydney, 2003, page 34.

[12] Benefits of Partial Reconfiguration, Xilinx 2005, page 66.

[13] Benefits of Partial Reconfiguration, Xilinx 2005, page 66.

[14] Xilinx, Partial Reconfiguration User Guide, UG702 (v14.1) April 24, 2012, page 13.

[15] Xilinx, Partial Reconfiguration User Guide, UG702 (v14.1) April 24, 2012, page 14.

[16] Xilinx, Partial Reconfiguration User Guide, UG702 (v14.1) April 24, 2012, page 17.

[17] Xilinx, Partial Reconfiguration User Guide, UG702 (v14.1) April 24, 2012, page 18.

BIBLIOGRAFIA

[1] Xilinx, Partial Reconfiguration User Guide, UG702 (v14.1) April 24, 2012.

[2] International Journal of Computer Applications (0975 – 8887), Volume 66-N°.10, March 2013, Module based Partial Reconfiguration on Bitstream Relocation Filter, M. Angelin Ponrani, G Manoj and R. Rajesvari.

[3] Trailokya Nath Sasamal, Rajendra Prasad/ International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622, Vol. 1, Issue 4, pp.1898-1903.

[4] Generating the Communications Infrastructure for Module-Based Partial Reconfiguration of FPGAs, Shannon Koh, Bachelor of Computer Science (Honours). UNSW Sydney, 2003.

[5] Xilinx, Benefits of Partial Recongiguration, Take advantage of even more capabilities in your FPGA.