Flujo de desarrollo de sistemas reconfigurables escalables sobre FPGAs

Julián Caba, Julio D. Dondo, Fernando Rincón y Juan C. López [1]

Resumen— **Una de las características más atractivas de las FPGAs (Field-Programmable Gate Array) durante estos últimos años ha sido la reconfiguración dinámica parcial. Esta característica ofrece un gran abanico de posibilidades, desde minimizar el área utilizada en la FPGA, hasta disponer de la posibilidad de modificar un diseño después de ser desplegado, sin necesidad de una reprogramación completa. Sin embargo, el aprovechamiento de esta característica no se ha extendido ampliamente entre los desarrolladores de sistemas, debido a la complejidad que conlleva realizar proyectos de este tipo. La aportación de este trabajo consiste en establecer un nuevo flujo de desarrollo, que facilite al usuario la construcción de proyectos dinámicos sobre hardware reconfigurable, así como la gestión de bitstreams y áreas dinámicas. Los desarrollos realizados y los resultados obtenidos, utilizando esta propuesta, muestran una mejora muy significativa en los tiempos de diseño de sistemas dinámicamente reconfigurables.**

Palabras clave— **Reconfiguración dinámica parcial, Field-Programmable Gate Array, bitstream.**

I. Introducción

LOS incesantes avances en la electrónica digital reconfigurable ofrecen productos cada vez más completos que disponen de un mayor espacio útil, de un mejor rendimiento y de mayor eficiencia, debido a las exigencias del mercado. Pero no sólo ha habido mejora en cuanto a espacio y rendimiento se refiere, si no que también a la incorporación de nuevas características, como por ejemplo la reconfiguración dinámica parcial, que aumenta la flexibilidad de una FPGA permitiendo la reprogramación de zonas del dispositivo mientras que el resto de aplicaciones que no están implicadas continúan ejecutándose. Pero si esta característica no ha terminado de ser explotada, se debe a la alta dificultad que supone la construcción de proyectos reconfigurables [1].

La reconfiguración dinámica parcial ofrece una serie de ventajas en el diseño de proyectos, puesto que permite al usuario introducir más funcionalidades en un menor espacio, obteniendo una reducción de costes. Un ejemplo de esta estrategia, es el uso de la reconfiguración dentro de proyectos de *software defined radio* (SDR), donde es posible actualizar la forma de onda sin necesidad de tener una instancia desplegada por cada tipo, sino utilizando la forma de onda necesaria en cada instante mediante la reconfiguración parcial del dispositivo. Otra ventaja relacionada con la anterior, es la flexibilidad que ofrecen los diseños reconfigurables, puesto que es posible implementar nuevas funcionalidades posteriormente al despliegue del diseño [2].

[1]Departamento de Tecnología y Sistemas de la Información. Universidad de Castilla-La Mancha, e-mails: [julian.caba, juliodaniel.dondo, fernando.rincon, juancarlos.lopez]@uclm.es

Sin embargo, no todo son ventajas a la hora de aplicar la reconfiguración dinámica a los diseños hardware. El alto coste (tiempo/beneficio) que implica seguir esta técnica provoca un abandono en su utilización por parte de los desarrolladores, sustituyéndola por un mayor esfuerzo en el uso de técnicas de codiseño, con el fin de minimizar el tiempo de desarrollo [3]. A esto se le suma la complejidad del flujo de desarrollo propuesto por `Xilinx` para sus FPGAs, puesto que el usuario debe trabajar sobre niveles de abstracción muy bajos [4]

En este artículo se expondrá un modelo de desarrollo para tratar el problema de la reconfiguración parcial sobre hardware reconfigurable, planteando mejoras sobre el flujo utilizado hoy en día, y resolviendo el problema del manejo de áreas, haciendo uso de las herramientas de bajo nivel que proporciona `Xilinx`, pero sin necesidad de trabajar en dicho nivel. El modelo aportado pretende eliminar la problemática en el proceso de diseño de sistemas reconfigurables, aportando a la vez transparencia al desarrollador de FPGAs.

Este documento se ha organizado en diferentes secciones que se resumen a continuación. En la siguiente sección se tratará el *estado del arte*, donde se plasmará la situación actual en la que se encuentra la reconfiguración dinámica y la problemática que conlleva. Seguidamente, en la sección de *objetivos* se indicarán las metas que se desean alcanzar con esta propuesta. Posteriormente, en *modelo de la aplicación* se describirá la metodología utilizada en el modelo que se plantea en este documento, además de indicar los pasos a seguir para la construcción de un proyecto reconfigurable utilizando las herramientas desarrolladas. La sección de *resultados experimentales* aportará al artículo un detalle de los recursos utilizados y la validez del modelo. Por último, en las *conclusiones* se resumirán qué objetivos han sido alcanzados.

II. Estado del arte

La reconfiguración dinámica parcial ha sido uno de los temas de investigación más candentes durante las últimas dos décadas, debido al valor potencial que esta característica aporta. Sin embargo, el uso de la reconfiguración dinámica en la industria como solución a distintos problemas ha sido casi nula, debido a la dificultad existente en el desarrollo de sistemas reconfigurables y al uso de técnicas de codiseño tradicionales, donde se invierte la mayor parte del tiempo de desarrollo al diseño para, de esta forma, reducir los costes económicos [3]. El inconveniente de esta metodología reside en la alta dependencia del trabajo

previo del diseñador, puesto que una de las primeras fases es la partición hardware-software, para posteriormente decidir, en la parte hardware, qué elementos serán dinámicos y cuales estáticos. Todo ello debe realizarse sin el conocimiento de la arquitectura final del sistema, basándose únicamente en la experiencia del desarrollador.

Otros problemas que se encuentra el diseñador a la hora de desarrollar aplicaciones reconfigurables es la pequeña cantidad de dispositivos que disponen esta característica y la inversión económica que supone para la entidad desarrolladora del producto final. Las herramientas utilizadas no facilitan la eliminación de errores, siendo arrastrados hasta las etapas finales del proceso, a lo que se le suma la falta de automatización de las mismas. Además, el flujo de desarrollo utilizado no beneficia la eliminación de estos problemas.

Una solución a los problemas de gestión de la reconfiguración parcial es presentada en [5], la cual se basa en el modelo de programación orientada a objetos en el contexto de la electrónica reconfigurable, aportando a los diseños en FPGAs las ventajas de un sistema distribuido, donde cada componente ofrece un servicio al resto del sistema o incluso al exterior, mientras que los buses son tratados como canales de comunicación. Entre las ventajas de este modelo se encuentra la transparencia de localización, la reutilización sencilla debido a la definición de interfaces y la facilidad para plantear la arquitectura de interconexión entre componentes. Por lo tanto, la solución planteada seguirá esta metodología a la hora de desarrollar componentes hardware.

El proceso de la reconfiguración parcial es gestionado, en la mayoría de los estudios, por software, requiriendo la instanciación de un procesador que se ocupa de tareas propias del proceso, como es la obtención del bitstream desde algún dispositivo de almacenamiento, para su posterior tratamiento y envío al controlador de reconfiguración [6] [7]. Si bien estas alternativas presentan la ventaja de la flexibilidad del software, tienen la desventaja que introducen una latencia elevada en el proceso de reconfiguración parcial, llevando a utilizar técnicas de solapamiento entre ejecución de tareas y reconfiguración, introduciendo complejidad en tareas de planificación, o técnicas de *prefetching*, en la cual el proceso de reconfiguración comienza mucho antes de ser necesitado [8] [9] [10].

El modelo de `Xilinx` para la reconfiguración dinámica impone varias restricciones: geométricas, conectividad y ubicación en una determinada zona de la FPGA [4]. En los últimos años se ha invertido muchos esfuerzos en temas de investigación con el fin de encontrar una solución que permita flexibilizar estas tres restricciones. En el trabajo [11] se expone la posibilidad de ubicar bloques de lógica en diferentes lugares del dispositivo, incluyendo unos componentes especiales que permiten la comunicación entre módulos. En esta técnica no es necesario definir la región o regiones dinámicas.

Por otro lado, el estudio [12] introduce un nuevo paradigma de reconfiguración, donde varios diseños compiten de forma dinámica por un espacio en la FPGA. La solución a cómo enfrentase a este desafío se basa en obtener el mayor beneficio de la escalabilidad y la adaptabilidad de los diseños, alcanzando un equilibrio entre rendimiento y flexibilidad. La técnica utilizada consiste en considerar la FPGA como una única área reconfigurable, a la cual se le incluyen de forma dinámica los módulos necesarios, estos dispondrán de unos puertos de comunicación, los cuales estarán conectados a otros módulos o al exterior. Esta conexión es realizada mediante la herramienta `Torc`, que es capaz de modificar las netlists y los bitstreams generados [13]. La ventaja de esta técnica es la facilidad que ofrece a la hora de reubicar un componente en cualquier zona de la FPGA, sin necesidad de volver a sintetizar el proyecto.

Las propuestas aportadas por los trabajos [11] y [12] son dos de las soluciones que permiten mejorar la problemática de la reconfiguración dinámica, como queda demostrado en sus respectivos resultados experimentales. El inconveniente de ambas soluciones reside en el alejamiento del flujo de desarrollo propuesto por `Xilinx`, el primero de ellos incluye unos componentes especiales de comunicación, mientras que el segundo utiliza una herramienta que realiza modificaciones sobre el sistema diseñado. Además, en ambos casos el flujo de desarrollo es complejo y costoso, razón por la que la reconfiguración dinámica parcial no ha sido explotada industrialmente hoy en día. En este punto es donde se plantea la posibilidad de aportar una nueva solución al problema de la reconfiguración sin necesidad de distanciarse del flujo original de `Xilinx`, teniendo presente las limitaciones que éste tiene e intentando minimizarlas lo máximo posible.

III. Objetivos

En este trabajo se pretende facilitar el desarrollo de sistemas reconfigurables, mediante el uso de un conjunto de herramientas que han sido desarrolladas para guiar el flujo de diseño de una forma sencilla y transparente, evitando la ardua tarea que supone generar un sistema reconfigurable desde el inicio o convertir uno de los proyectos, creados previamente por el usuario, en un proyecto reconfigurable.

Una área dinámica por el hecho ser reconfigurable proporciona flexibilidad al usuario, pero esta flexibilidad esta limitada al tamaño de la misma. Esto supone una limitación importante para el diseñador, que le impide tener componentes dinámicos mayores. La solución planteada debe permitir el uso de varias áreas con el fin de albergar aquellos componentes que no pueden ser instanciadas en una sola área, eliminando la restricción que actualmente existe.

IV. Modelo de la aplicación

La construcción de un sistema reconfigurable es un proceso largo y complejo, con el objetivo de alcanzar los resultados planteados inicialmente. Para minimi-

zar este problema, la solución pasa por seguir una serie de pasos de forma secuencial, sin desviar la tarea que el usuario venía haciendo hasta ahora. En la figura 1 se representa el nuevo flujo de desarrollo para la construcción de sistemas reconfigurables sobre hardware reconfigurable.

Fig. 1. Flujo de desarrollo.

Este flujo de desarrollo se basa en la utilización de tres nuevas herramientas, que han sido desarrolladas para facilitar de manera importante la construcción de sistemas reconfigurables. Este modelo aporta la transparencia suficiente para que un usuario sea capaz de construir sistemas reconfigurables de forma sencilla y eficaz.

A. *Diseño del proyecto estático*

El primer paso que debe dar el usuario, consiste en construir un proyecto estático, como hasta ahora ha venido trabajando con la herramienta `Xilinx XPS`. Pero debe tener presente no incluir aquellos componentes que van a formar parte de la zona dinámica. Por norma general, estos componentes deberán estar comunicados con la parte estática del proyecto, cuya comunicación se realizará a través de señales que permitan dicha interacción. El nombre de las señales de comunicación tendrán obligatoriamente un prefijo que identifique unívocamente qué conjunto de señales corresponde a una determinada área reconfigurable. Por lo tanto, cada área reconfigurable tendrá asociado una serie de señales por las cuales podrá interaccionar de forma directa con la parte estática del proyecto. Si se quiere mayor movilidad de componentes entre áreas reconfigurables se pueden conectar estas a un bus y en este caso las interfaces serían todas iguales, la del bus.

En la parte estática, se deberá incluir un dispositivo de almacenamiento donde estarán contenidos los bitstreams parciales del sistema. Además, se debe incluir el componente `factory`. Este componente fue diseñado íntegramente en hardware y es el encargado de realizar la reconfiguración parcial de la FPGA. Para ello interpreta los comandos que albergan los bitstreams y los instancia físicamente, haciendo uso del componente `ICAP` que disponen las FPGAs. También es el encargado de obtener de memoria los bitstreams necesarios mediante un bus dedicado (*NPI*), adoptando el rol de una *DMA*. Todo ello, se realiza de forma asíncrona, es decir, sin bloquear el bus de comunicaciones entre componentes, ni el microprocesador, ni ningún otro componente. Una vez terminado el proceso de reconfiguración la factoría notifica que ha finalizado la petición al componente que la invocó (figura 2).

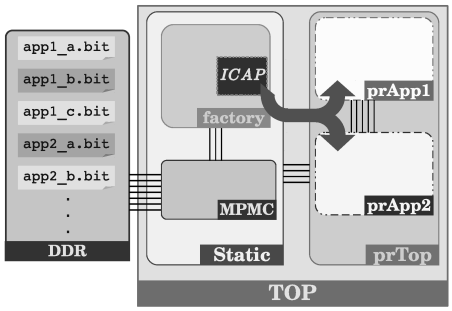

Fig. 2. Top de proyecto reconfigurable.

Seguidamente, se obtendrá la `netlist` del sistema estático, teniendo en cuenta que el sistema no corresponderá al top global del proyecto, sino al top de la parte estática, y por tanto, se deberán configurar las opciones de síntesis para que tenga en cuenta la jerarquía establecida (figura 2).

B. *Definición de áreas reconfigurables*

Una vez creada la parte estática, según las indicaciones que se han mencionado anteriormente, el siguiente paso consiste en definir las áreas que van a componer la parte dinámica. Para dicho fin, es necesario un fichero denominado *archivo de definición de áreas* (archivo mrs). El formato de este archivo consta de: nombre del área, ubicación de la misma en la FPGA y prefijo del conjunto de señales al que está asociada el área.

Existen dos posibles estructuras de este fichero: con jerarquía o sin jerarquía. Ambas formas no pueden ser utilizadas en un mismo fichero a la vez.

- *Sin jerarquía:* Se definen áreas reconfigurables independientes. En la figura 3 se puede observar un ejemplo formado por dos áreas reconfigurables *prApp1*, y *prApp2*.

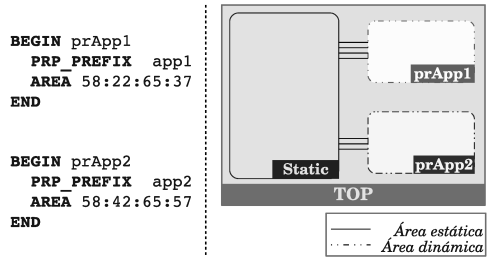

Fig. 3. Archivo mrs y resultado (sin jerarquía).

- *Con jerarquía:* En este modelo es posible definir áreas reconfigurables que estén contenidas en otra área de nivel superior. Para este caso los prefijos se indican en cada una de las áreas inferiores. En la figura 4 se observa un ejemplo formado por dos áreas *topApps* y *prApp3*. A diferencia del anterior, el área *topApps* contiene otras dos áreas reconfigurables definidas dentro

de la primera. Esto permite desplegar bitstreams en cada una de las áreas, *prApp1* o *prApp2*, o bien desplegar un bitstream de mayor tamaño en el área de nivel superior definida por *topApps*.

```
BEGIN_TOP topPR
AREA_TOP 56:20:67:59
  BEGIN topApps
    AREA 58:22:65:37
    BEGIN prApp1
      PRP_PREFIX app1
      AREA 60:24:65:35
    END
    BEGIN prApp2
      PRP_PREFIX app2
      AREA 64:24:65:37
    END
  END

  BEGIN prApp3
    PRP_PREFIX app3
    AREA 58:42:65:57
  END
END_TOP
```

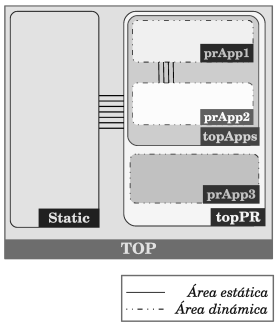

Fig. 4. Archivo mrs y resultado (con jerarquía).

Este modelo permite al usuario combinar de forma distinta las áreas reconfigurables, permitiendo la comunicación entre ellas a través de señales. A esto se le debe incluir la posibilidad de aumentar la escalabilidad de los componentes dinámicos, permitiendo que un componente pueda estar localizado en dos áreas. Para ello, el diseño de dicho componente debe dividirse en dos partes que interaccionan entre sí.

Independientemente de la opción elegida, la ubicación de las áreas no deben coincidir en la misma región de reloj local, impidiendo poder ubicar varias áreas dinámicas en una misma región. Esta restricción física coacciona la libertad del desarrollador a la hora de diseñar este tipo de proyectos. Por ello, la opción de definir las áreas con jerarquía, permite al desarrollador incluir componentes de diferentes tamaños, que pueden ir desde un tamaño igual a una región de reloj, hasta un conjunto de varias áreas albergadas en un mismo componente, permitiendo mayor escalabilidad, ya que se puede tener una fila de zonas reconfigurables conectadas entre sí.

C. Generación del sistema (pr_buildSystem)

Una vez desarrollado la parte estática del proyecto y habiéndose definido el modelo de área reconfigurable a utilizar, el siguiente paso consiste en construir el sistema de archivos del proyecto reconfigurable (figura 5), utilizando la primera de las herramientas desarrolladas: `pr_buildSystem`. Será necesario partir de un fichero de definición de áreas y un proyecto estático, creado previamente, con las particularidades que se han explicado anteriormente. También se generan las plantillas necesarias para que el desarrollador implemente los componentes dinámicos, así como las posibles combinaciones de los módulos reconfigurables.

El directorio *regions/static* contiene la parte estática del proyecto reconfigurable, que corresponde al proyecto generado en el primer paso del flujo de desarrollo.

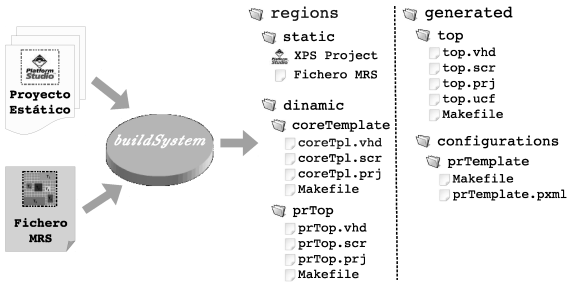

Fig. 5. Herramienta buildSystem.

En el directorio *regions/dinamic* se almacena toda la parte dinámica del proyecto, donde el desarrollador dispone de plantillas para las distintas áreas, a partir de las cuales se pueden iniciar la implementación de los componentes dinámicos necesarios. También, si se ha indicado en el fichero de definición de áreas que existe jerarquía en las zonas dinámicas, aparecerán los directorios correspondientes donde se implementa el componente de alto nivel que alberga los componentes dinámicos. La comunicación entre dichos componentes dinámicos debe realizarla el propio desarrollador y, por tanto, añadir las señales de comunicación a las plantillas que se generaron para las distintas áreas.

El directorio *generated/top* alberga el top del proyecto reconfigurable, cuya estructura dependerá de cómo se describa el archivo de definición de áreas (figuras 3 y 4).

En el directorio *generated/configurations* se genera una plantilla, que servirá al usuario para implementar el resto de combinaciones posibles (figura 6). Esta plantilla depende del archivo de definición de áreas, donde se creará una partición por cada área definida, importando aquellas partes que son idénticas al resto de combinaciones e implementando aquellas que no lo son, es decir, las partes dinámicas.

```
<Project Name="prTemplate" FileVersion="1.2"
    ProjectVersion="2.0">
 <Partition Name="/top" State="import"
     ImportLocation="../prReference">
  <Partition Name="/top/static" State="import"
      ImportLocation="../prReference">
  </Partition>
  <Partition Name="/top/topPR" State="import"
      ImportLocation="../prReference">
   <Partition Name="/top/topPR/topApps" State="
       import" ImportLocation="../prReference">
    <Partition Name="/top/topPR/topApps/prApp1"
        State="implement" ImportLocation="NONE"
        Reconfigurable="true" ReconfigModuleName="
        prApp1Template">
    </Partition>
    <Partition Name="/top/topPR/topApps/prApp2"
        State="implement" ImportLocation="NONE"
        Reconfigurable="true" ReconfigModuleName="
        prApp2Template">
    </Partition>
   </Partition>
   <Partition Name="/top/topPR/prApp3" State="
       implement" ImportLocation="NONE"
       Reconfigurable="true" ReconfigModuleName="
       prApp3Template">
   </Partition>
  </Partition>
 </Partition>
</Project>
```

Fig. 6. Plantilla pxml (figura 4).

Llegados a este punto, el próximo paso que se debe dar, si las regiones dinámicas descritas en el archivo de descripción de áreas corresponden a un modelo con jerarquía, consiste en definir la comunicación entre las áreas, reflejándolas en el top que las alberga. Posteriormente, se añadirán las señales de comunicación a las plantillas de los cores dinámicos, para de esta forma poder pasar a implementar los componentes dinámicos que formarán parte del proyecto.

D. Configuración de referencia (pr_buildReference)

Una vez que se ha creado al menos un componente por cada área dinámica, se deberá construir una configuración de referencia mediante la herramienta `pr_buildReference`, siendo el punto de partida del resto de configuraciones. El nombre de los directorios que albergan los componentes dinámicos deben contener el prefijo de las plantillas para que la herramienta sea capaz de generar la configuración de referencia. En este caso, todas las partes serán implementadas y ninguna de ellas importada de otra configuración (figura 7).

```
<Project Name="prReference" FileVersion="1.2"
     ProjectVersion="2.0">
 <Partition Name="/top" State="implement"
      ImportLocation="NONE">
  <Partition Name="/top/static" State="implement"
       ImportLocation="NONE">
  </Partition>
  <Partition Name="/top/prApp1" State="implement"
       ImportLocation="NONE" Reconfigurable="true"
       ReconfigModuleName="prApp1_a">
  </Partition>
  <Partition Name="/top/prApp2" State="implement"
       ImportLocation="NONE" Reconfigurable="true"
       ReconfigModuleName="prApp2_a">
  </Partition>
 </Partition>
</Project>
```

Fig. 7. Archivo de referencia pxml (figura 3).

E. Síntesis del sistema reconfigurable

El siguiente paso consiste en sintetizar el proyecto reconfigurable para obtener los bitstreams del sistema. Para ello, las dos herramientas utilizadas hasta el momento han ido generando los *scripts* necesarios para automatizar este proceso, lo que implica ejecutar el script situado en el directorio raíz del proyecto reconfigurable.

El tiempo de síntesis de estas configuraciones es muy bajo, debido a la eliminación de partes idénticas, a esto se le suma la posibilidad de poder ejecutar el resto de configuraciones en paralelo, siempre que la síntesis del proyecto de referencia se haya realizado con anterioridad.

F. Preparación de bitstreams (pr_prepareBits)

Finalizado el proceso de síntesis, de donde se han obtenido los bitstreams, el siguiente, y último, paso consiste en adecuar los bitstream obtenidos para que sean utilizados por el componente `factory` para la reconfiguración parcial de la FPGA.

Para ello se ha diseñado la herramienta `pr_prepareBits` que elimina las cabeceras originales y los datos situados después del comando de finalización. El nuevo bitstream se divide en tres partes: una cabecera reducida, donde se incluye los comandos de configuración y sincronización, los datos del propio bitstream y el comando de finalización. Todos estos datos se encuentran alineados en palabras de 32 bits.

La propia herramienta es capaz de concatenar distintos bitstreams, dejando una única cabecera al inicio del fichero, seguida de los datos de cada uno de los ficheros de configuración y, en última instancia, el comando de finalización (figura 8). El componente encargado para instanciar el objeto físicamente (`ICAP`) será capaz de interpretar los datos que contiene el bitstream generado, permitiendo la configuración de varias áreas reconfigurables de una sola vez, o lo que es lo mismo, un componente cuyo tamaño requiera de varias zonas dinámicas. Estas zonas deben estar albergadas en un mismo top, debido a la comunicación entre ellas.

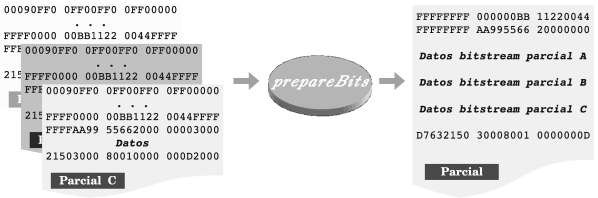

Fig. 8. Herramienta prepareBits.

V. Aplicación

La potencia de este modelo, además de la ofrecida por la facilidad de crear proyectos reconfigurables, se basa en la posibilidad de poder instanciar cualquier objeto hardware, aún cuando necesite un área reconfigurable mayor. Esta característica otorga al modelo de mayor escalabilidad frente al modelo tradicional, donde al usuario se le limita el uso de la FPGA a lo diseñado desde el inicio. Por ejemplo, en la figura 9 se muestra un proyecto reconfigurable, que consta de dos áreas dinámicas, las cuales albergarán algoritmos de encriptación para el posterior envío de datos, simulando un sistema de comunicación seguro. Dependiendo del algoritmo utilizado, se podrán instanciar diferentes algoritmos a la vez, como es el caso del algoritmo `DES` cuyo tamaño ocupa una sola área, y por tanto, es posible utilizar otro algoritmo que ocupe la área que queda libre. En el caso del algoritmo `IDEA` no es posible instanciar dos algoritmos al mismo tiempo, puesto que éste ocupa las dos áreas reconfigurables disponibles.

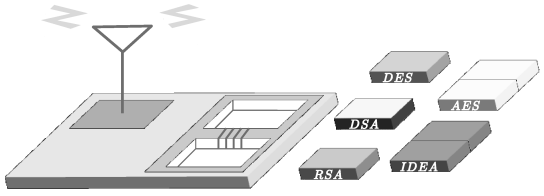

Fig. 9. Modificación de la funcionalidad.

VI. Resultados experimentales

La solución planteada para el problema de la reconfiguración dinámica se ha llevado a cabo bajo un entorno GNU/Linux con dos modelos de FPGAs `Virtex 5`, en concreto `FX70T` y `LX110T`. En ambos casos se ha utilizado el componente `factory`, cuyas características, en cuanto a recursos empleados, se recogen en la tabla I. Este componente reduce de forma considerable la latencia a la hora de configurar parte de una FPGA en tiempo de ejecución [5].

TABLA I
Recursos de factory.

Slices	Slices FFs	LUTs	IOs
1140	456	1116	644

En la parte práctica, se ha conseguido realizar la reconfiguración de varias áreas dinámicas, cada una de tamaño igual a una región de reloj local, ubicadas en la parte derecha de la FPGA con el fin de dejar el resto del espacio para la parte estática. Tanto la ubicación como el tamaño dependen directamente del modelo de FPGA utilizado, además de las restricciones propias de los componentes incluidos.

En la concatenación de áreas reconfigurables para la obtención de una área mayor, necesaria para poder implementar componentes más grandes, no ha existido problema alguno, ya que al inicio del diseño, se ha definido claramente la comunicación entre las áreas o incluso la conexión a un bus.

El flujo de desarrollo, que se ha propuesto como solución a la generación de proyectos reconfigurables, así como las herramientas de reconfiguración descritas, han sido sometidas a estudio por varios desarrolladores de este tipo de sistemas, donde se ha propuesto realizar una serie de proyectos reconfigurables, midiendo los tiempos que se ha tardado en cada uno de ellos. Los resultados que se han obtenido son satisfactorios como refleja la gráfica de la figura 10, donde la curva de aprendizaje, en relación al tiempo empleado para hacer un proyecto reconfigurable y la cantidad de proyectos realizados, corresponde a una exponencial decreciente. En dicha gráfica no se han incluido los tiempos de síntesis del sistema, ya que no forman parte del diseño, y dependen de la potencia de cómputo donde se ejecutará el proceso de síntesis.

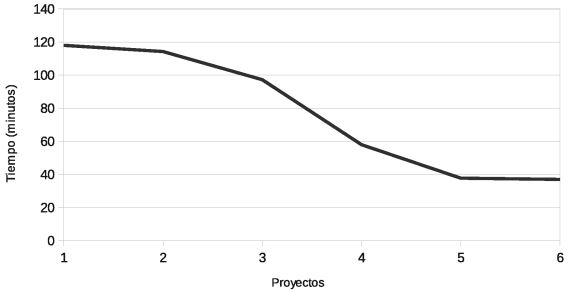

Fig. 10. Curva de aprendizaje.

VII. Conclusiones

El flujo de desarrollo planteado facilita enormemente la construcción de sistemas reconfigurables, minimizando la problemática en este tipo de diseños, como demuestran los resultados experimentales.

Otra característica que se obtiene al aplicar este modelo, es la posibilidad de unificar áreas reconfigurables, con el fin de obtener una zona mayor de reconfiguración, eliminando la restricción física impuesta por el diseño inicial, permitiendo la escalabilidad de los componentes dinámicos y, por tanto, del proyecto reconfigurable.

Además, al hacer uso de las herramientas desarrolladas, se evita arrastrar errores desde las primeras etapas del desarrollo, de esta forma el ahorro en tiempo de diseño es considerable. A esto se le suma la transparencia dada al usuario, evitándole la necesidad de conocer el cómo se construye un sistema dinámico.

Agradecimientos

Esta investigación ha sido financiada por el *Ministerio Español de Ciencia e Innovación* bajo el proyecto `DREAMS (TEC2011-28666-C04-03)`, y por el *Ministerio Español de Industria y el Centro para el Desarrollo Tecnológico Industrial* bajo el proyecto `ENERGOS (CEN-20091048)`.

Referencias

[1] Christophe Bobda, *"Introduction to Reconfigurable Computing: Architectures, algoritnms and applications"*, University of Kaiserslautern, 2007.
[2] David Dye, *"Partial Reconfiguration of Xilinx FPGAs using ISE Design Suite" (UG702)*, Xilinx, 2011.
[3] Wayne Wolf, *"High-Performance Embedded Computing. Architectures, applications, and methodologies"*, Princeton Universit, 2007.
[4] Xilinx, *"Partial Reconfiguration User Guide"*, Xilinx, 2011.
[5] F. Rincón, J. Dondo, J. Barba, F. Moya and J.C. López *"Supporting operating systems for reconfigurable computing: A distributed service oriented opproach"*, ERSA, 2009.
[6] A. Flynn, A. Gordon-Ross and A. D. George *"Bitstream relocation with local clock domains for partially reconfigurable fpgas"*, DATE, 2009.
[7] H. Kalte and M. Porrmann *"Replica2pro: task relocation by bitstream manipulation in virtex-ii/pro fpgas"*, Proceedings of the 3rd conference on Computing frontiers, CF '06, 2006.
[8] J. Noguera and R.M. Badia *"Multitasking on reconfigurable architectures: microarchitecture support and dynamic scheduling"*, ACM, 2004
[9] F. Redaelli, M.D. Santambrogio and D. Sciuto *"Task Scheduling with Configuration Prefetching and Anti-Fragmentation techniques on Dynamically Reconfigurable Systems"*, DATE, 2008.
[10] G. Chen, M. Kandemir and U. Sezer *"Configuration-sensitive process scheduling for FPGA-based computing platforms"*, DATE, 2004.
[11] C. Schuck, M. Kuhnle, M. Hubner, and J. Becker *"A Framework for Dynamic 2D Placement on FPGAs"*, IEEE International Symposium on Parallel and Distributed Processing, 2008
[12] T. Cervero, S. López, R. Sarmiento, T. Frangieh and P. Athanas *"Scalable Models for Autonomous Self-Assembled Reconfigurable Systems"*, International Conference on ReConFigurable Computing and FPGA, 2011.
[13] N. Steiner, A. Wood, H. Shojaei, J. Couch, P. Athanas, and M. French *"Torc: Towards an Open-Source Tool Flow"*, Nineteenth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2011