Scheduling and Resource Planner Based on Multicriteria Method for Partially Dynamically Reconfigurable Systems

A. Nuñez, J. Dondo, M. Berón, C. Sosa, and M. Murdocca

***Abstract*—Dynamically Reconfigurable FPGAs allow to accelerate the reconfiguration process downloading partial bitstreams to those areas created for that. The partial reconfiguration capability allows to share hardware resources between different tasks during the design lifecycle. In this paper we propose an algorithm based on Logic Score of preference (LSP) multicriteria decision method for floorplanning, scheduling and order of deployment of these partial bitstreams during design lifecycle. The proposed method is well suited to be implemented in preemptive systems in which the deadline of a critical task can demand the preemption of a lower priority one.**

***Index Terms*—Multicriteria Methods, LSP, FPGA, Partial and dynamyc reconfiguration.**

I. Introducción

En la actualidad los problemas computacionales se resuelven utilizando distintas tecnologías integradas como lo son los SoC heterogéneos que combinan recursos software (CPU-GPU) con recursos hardware reconfigurables (FPGA). Esto ha llevado a la necesidad de implementar complejos planificadores de ejecución de tareas buscando optimizar el uso de los recursos del sistema en cada una de sus decisiones. Por lo tanto, la complejidad del sistema en su conjunto hace que la toma de decisión que debe asumir el planificador de tareas presente mútiples aristas o criterios de decisión en donde las posibles decisiones a implementar pueden ser, en algunos casos, infinitas. De todas las posibles soluciones en un momento particular de cómputo del sistema, se debe elegir la que mejor satisfaga todos los criterios que componen el sistema, es decir, que todos los criterios que mejoran el desempeño del sistema estén en su máximo posible y que todos los criterios que empeoran el desempeño del sistema estén en su mínimo posible.

Para lograr este objetivo es que se propone la implementación de un método multicriterio para la toma de decisión del planificador de tareas. En este trabajo se propone el método multicriterio LSP como el más simple de implementar y el que permite dar un ranking de puntajes de todas las posibles soluciones que se pueden dar en la planificación de tareas.

Para el desarrollo del decisor se ha aprovechado la posibilidad que presentan hoy en día las FPGAs en donde se pueden implementar sistemas de harware reconfigurable dinámicamente. El uso de la reconfiguración parcial dinámica[1] permite modificar un diseño basado en FPGAs en tiempo de ejecución, donde diferentes funcionalidades hardware, llamadas en este trabajo Objetos Hardware (OH), comparten los recursos lógicos de Áreas Dinámicamente Reconfigurables (ADR)[2].

En este sentido, dado un Sistema Dinámicamente Reconfigurable (SDR) formado por varias ADR y varios OH, se propone el diseño de un algoritmo que permite decidir que OH se instancia en que ADR, aplicando el método multicriterio LSP.

Alejandro Nuñez Manquez, Julio Dondo Gazzano, Carlos Sosa Paez, y Martín Murdocca pertenecen al Departamento de Electrónica de la Facultad de Ciecias Físico Matemáticas y Naturales de la Universidad Nacional de San Luis.

Mario Berón pertenece al Departamento de Informática de la Facultad de Ciecias Físico Matemáticas y Naturales de la Universidad Nacional de San Luis.

II. Sistemas Reconfigurables

Los sistemas dinámicamente reconfigurables están compuestos por diferentes OH que son implementados en la FPGA de acuerdo a las demandas surgidas a lo largo del ciclo de vida del sistema. Estos OH (bitstreams parciales) se alojan en una biblioteca a la que se accederá cuando el sistema requiera la implementación de un objeto hardware específico, los cuales estarán asignados a sus correspondientes áreas. Además, es posible tener diferentes bitstreams del mismo objeto hardware correspondientes a las diferentes áreas en las que podrán ser implementados, teniendo la posibilidad de instanciar un OH en una u otra área de acuerdo a las necesidades de ejecución del sistema.

Esto permite tener diferentes posibilidades o perfiles de ejecución, dependiendo del tipo de recursos que se utilicen, permitiendo al sistema trabajar en un contexto de alta performance, ejecutando tareas en OH con recursos necesarios para computación de alto rendimiento, o en contextos de bajo consumo, ejecutando la misma tarea pero con OH que funcionen a menor frecuencia, por ejemplo.

Uno de los problemas que surge en la implementación de este tipo de sistemas es la planificación de ejecución de las tareas asociadas al sistema, donde además del orden de ejecución de las mismas se agrega el tipo de recursos que hacen falta para ejecutarlas.

El despliegue de un objeto hardware en un área y la posterior reutilización de los recursos del área por otro objeto requiere resolver algunos problemas tales como tamaño y recursos de las áreas disponibles, prioridad de ejecución del objeto nuevo a instanciar con respecto al instanciado, cantidad de recursos lógicos necesitados, si el OH puede ser implementado en diferentes áreas, entre otros.

Estos planificadores deben tener además una cola en la cual se ubicarán los OH que el sistema desea que se ejecuten, como así también los OH que fueron desalojados de alguna de las ADR por otros OHs de mayor prioridad. En caso de que hayan objetos con la misma prioridad que se instancien en una misma área, estos pueden ir alternándose en la ejecución utilizando Round-Robin.

III. Trabajos Relacionados

Hay muchos trabajos que han estudiado el problema de planificar las ejecución de las tareas hardware, en donde se pone de manifiesto que la cantidad de información necesaria para decidir incluye no tan sólo la localización de la tarea hardware en la FPGA, sino además en qué momento la tarea debe ser instanciada. Por lo tanto los planificadores resultan ser mucho más complejos que aquellos que sólo analizan cuando ejecutar una tarea.

En [3] los autores proponen un planificador que se compone de tres partes. Una de ellas realiza la ubicación espacial de Tareas Hardware (TH), es decir en alguna parte del área de la FPGA, la segunda se encarga de la ubicación temporal de la TH y una tercera que es la encargada de replantear el orden de reconfiguración. El sistema se compone de áreas dinámicamente reconfigurables definidas en una FPGA. Los autores definen algoritmos para ejecutar las tres partes que componen el planificador pero no aplican ningún método multicriterio.

En [4] los autores explican el desarrollo de un sistema compuesto de procesadores y dos FPGA en donde se aplica la reconfiguración dinámica. En el caso de la programación de la ejecución de las TH no explica el modo en que se realiza, aunque se puede observar que las decisiones no son tomadas utilizando algún método multicriterio de decisión.

En [5] los autores plantean un sistema integrado por agentes, los cuales pueden ser implementados en software o en hardware. Los agentes implementados en hardware se instancian en áreas dinámicamente reconfigurables. Para la decisión de su implementación, al ser decisiones multicriterios, no se realizan utilizando ese tipo de métodos de decisión. Desde nuestro trabajo se propone que estas decisiones se pueden realizar utilizando LSP como método decisor.

En [6] los autores proponen un sistema que utiliza lógica programable y procesadores implementados en un mismo chip como lo es la familia Zynq-7000 de Xilinx. En este sistema se computan tanto Tareas Software (TS) como TH. Las TH son ejecutadas en la parte de lógica programable que posee el chip. Estas tareas son instanciadas en áreas dinámicamente reconfigurables y su instanciación es administrada mediante el uso de un microkernel implementado en la zona de lógica programable. Los autores proponen disminuir la latencia usando TS que implementan un proceso que posee su equivalente hardware. Una vez instanciada la TH equivalente se cambia el contexto y el proceso pasa a ejecutarse en la TH equivalente.

El planificador de TH implementa un Round-Robin basado en prioridades. Para esto no se implementa algún método multicriterio para la toma de decisión. En el caso de las TH o TS, pueden ser detenidas en donde se guarda su contexto para lanzarlas nuevamente más tarde. Las TH están contenidas en unas librerías que contiene sus respectivos bitstream.

En [7] los autores proponen un sistema que implementa TS que tienen su homólogo a través de TH. El sistema cuenta con dos áreas dinámicamente reconfigurables en donde se pueden instanciar las TH. Las TS se ejecutan en un procesador que es parte del sistema. Cuando una tarea debe ser ejecutada su ejecución se realiza primero en el microprocesador para, cuando ya esté instanciada ejecutarse en alguna de la áreas dinámicamente reconfigurables (ADR). En el caso de que la ADR esté ocupada puede ser liberada mudando la TH que la ocupa a otra ADR o dejándola en una cola de espera. La decisión que se debe tomar se implementa mediante un algoritmo que no pertenece a la familia de los métodos multicriterios que se desarrolla en nuestro trabajo.

Como se ha podido observar, en los trabajos analizados se describe la necesidad de implementar planificadores de TH o, en el caso de sistemas híbridos, TS / TH. Estos planificadores solamente toman en cuenta el área donde instanciar las TH (recursos hardware necesarios), y en qué momento hacerlo, pero no tienen en cuenta un escalamiento a un sistema n-complejo en donde, además de las áreas y del momento de instanciación se suman otros criterios. Estos criterios a considerar pueden ser diferentes versiones del mismo componente a instanciar con diferentes perfiles de funcionamiento, por ejemplo, mayor o menor consumo, mayor o menor velocidad, dependiendo del contexto de la aplicación.

En este trabajo se propone un método multicriterio que toma en cuenta todas estas posibilidades y que además puede ser fácilmente escalable a sistemas con mayor complejidad tales como grid de FPGAs, en donde se agregan nuevos criterios a la toma de decisiones como tipo de FPGAs del grid, tiempo de transmisión de datos, localización de vecinos, etc [8].

IV. Método Multicriterio LSP

Una vez definido el sistema es necesario identificar los criterios que se utilizarán en la elaboración del método LSP.

LSP es un método multicriterio cuantitativo para la creación de funciones de criterio y para sus aplicaciones en evaluación, optimización, comparación y selección de sistemas complejos en general[9].

LSP plantea los siguientes pasos generales:

1) Construir un Árbol de Criterios correspondiente a los elementos que serán evaluados.
2) Definir una función denominada Criterio Elemental, que normaliza las contribuciones por cada uno de los atributos del árbol de criterios.
3) Construir una Estructura de Agregación, que permitirá computar las contribuciones de los criterios en un único valor. Dicho valor indicará el grado de satisfacción general del elemento evaluado.
4) Llevar a cabo la evaluación y posterior selección de los elementos evaluados.

Las posibilidades de ubicación de los OH en las ADR son muchas de las cuales es necesario elegir la óptima. En este sentido mediante LSP se obtiene una tabla de puntajes en función de todas las posibles soluciones que pueden resolver la ubicación de los OH.

A. Características de las ADR y de los OH

El sistema en donde se desea aplicar LSP está formado por ADRs y una librería de OH que se pueden instanciar en ellas. A cada uno de estos elementos se les han definido los siguientes atributos o características.

1) Características de las ADR:

- OH que puede alojar: se define en tiempo de diseño. Cada ADR posee una lista de los OH que puede alojar.
- Recursos: es la cantidad de recursos que posee la ADR y se mide en cantidad de BRAMs, Lógica y DSP.
- Libre: indica si está libre o si está ocupada por un OH.
- Prioridad: indica la prioridad que posee el OH que la ocupa.
- Mudabilidad: es la capacidad que posee el OH que se está ejecutando en la ADR de mudarse a otra ADR.

2) Características de los OH:

- ADRs en donde puede alojarse: es un parámetro que se define en tiempo de diseño
- Prioridad de ejecución: es la prioridad del objeto al momento de querer instanciarlo en una de las ADR.
- Recursos: es la cantidad de recursos que posee el OH y se mide en cantidad de BRAMs, Lógica y DSP.
- Velocidad: es la frecuencia de ejecución del OH. Esto puede ocurrir en el caso de que existan varias versiones de un mismo OH en donde cada versión posee una velocidad de ejecución distinta.

B. Árbol de Criterios

El Árbol de Criterios contiene las distintas características deseables del sistema, las cuales se van desagregando o descomponiendo hasta llegar a atributos que son directamente mensurables. En el caso de este trabajo el objetivo es ubicar un OH en una de las ADR que componen el sistema y que esta decisión sea tomada en función de las características propias del sistema. De acuerdo a esto surge el siguiente árbol de criterios

- Instanciabilidad
- Recursos
  - BRAM
  - Lógica
  - DSP
- Desempeño
  - Velocidad
- Disponibilidad de ADR
  - ADR libre
  - Diferencia de prioridad de OH que la ocupa y OH que la quiere ocupar
  - Mudabilidad del OH que la ocupa

1) Instanciablididad: Este parámetro indica qué OH puede ser instanciados en una determinada ADR.

2) Recursos: Tanto las ADRs como los OH poseen una cantidad de recursos fijos. Este parámetro permite saber si un OH aprovecha en forma óptima los recursos de la ADR que lo puede alojar. Los recursos se miden en cantidad de BRAM, Lógica y DSP.

3) Desempeño: El desempeño mide la eficiencia con la cual se ejecuta un OH. En el caso de este trabajo el desempeño se mide en función de la velocidad de ejecución. Cada OH posee varias versiones, unas más rápidas y otras más lentas. Como cada OH puede ser alojado en más de una de las ADRs que componen el sistema sus velocidades de ejecución pueden variar teniendo mejor desempeño en función de la ADR que lo aloja.

4) Disponibilidad de ADR: Este parámetro permite saber si una ADR está libre o tiene alojado un OH. En el caso de tener un OH alojado permite saber la prioridad de ese OH, y si ese objeto se puede mudar a otra ADR.

C. Criterios Elementales

En función del árbol de criterios antes detallado se pueden definir los criterios elementales. Estos son los valores de entrada de la función LSP.

- **INSTANCIABILIDAD**: Determina si un OH puede o no alojarse en una ADR. Toma valor 100 cuando puede alojarse y 0 en caso contrario. En el caso de que un OH no pueda alojarse en la ADR, la salida del algoritmo LSP debe ser 0. Este es un parámetro esencial.
- **BRAM**: es el porcentaje de BRAMs que utiliza un OH en cada ADR en donde puede alojarse. Este valor se calcula de la siguiente manera:

$$BRAM = \frac{BRAM_{OH}}{BRAM_{ADRx}} * 100 \tag{1}$$

  - $BRAM$: es el porcentaje de BRAM que varía entre 0 y 100
  - $BRAM_{OH}$: es la cantidad de BRAM que usa el OH.
  - $BRAM_{ADRx}$: es la cantidad BRAM que posee la ADRx.

  Este es un parámetro deseable.
- **FF**: es el porcentaje de Flip Flops que utiliza un OH en cada ADR en dónde puede alojarse. Este valor se calcula de la siguiente manera:

$$FF = \frac{FF_{OH}}{FF_{ADRx}} * 100 \tag{2}$$

  - FF: es el porcentaje de FF que varía entre 0 y 100
  - FF_{OH}: es la cantidad de FF que usa el OH.
  - FF_{ADRx}: es la cantidad FF que posee la ADRx.

  Este es un parámetro deseable.
- **DSP**: es el porcentaje de DSPs que utiliza un OH en cada ADR en dónde puede alojarse. Este valor se calcula de la siguiente manera:

$$DSP = \frac{DSP_{OH}}{DSP_{ADRx}} * 100 \tag{3}$$

  - DSP: es el porcentaje de DSP que varía entre 0 y 100
  - DSP_{OH}: es la cantidad de DSP que usa el OH.
  - DSP_{ADRx}: es la cantidad DSP que posee la ADRx.

  Este es un parámetro deseable.
- **VELOCIDAD**: Cada OH tendrá las velocidades de ejecución en ciclos de reloj en cada una de las ADR

en donde puede ser alojado. La velocidad máxima de ejecución está dada por el valor más pequeño mientras que la velocidad menor está dado por el valor mayor. Entonces, si un OH tarda en ejecutarse 30 ciclos en la ADR1, 20 ciclos en la ADR2 y 10 ciclos en la ADR3 la velocidad máxima es 10 ciclos y el cálculo de la velocidad se da de la siguiente manera:

$$VELOCIDAD = \frac{vel_{max}}{vel_{ADRx}} * 100 \quad (4)$$

donde:

- $velocidad$: es un valor de velocidad relativo que varía entre 0 y 100
- vel_{max}: es el valor mínimo en ciclos que un OH puede ejecutarse.
- vel_{ADRx}: es la cantidad de ciclos que tarda en ejecutarse un OH en la ADR en donde se está aplicando el algoritmo LSP.

- **ADR_LIBRE**: 50 si ya hay un objeto instanciado en ella y 100 si está libre. Este es un parámetro deseable.
- **PRIORIDAD**: Diferencia de prioridad entre el objeto a alojar y el objeto ya alojado en la ADRx. Cada OH posee un valor de prioridad: baja(10), regular (25), media (50), alta(80), muy alta (100).

$$PRIORIDAD = pOH - pOH_{ADR} \quad (5)$$

donde:

- $PRIORIDAD$: Es la diferencia de prioridades. Si el resultado de la diferencia de prioridades es negativo el valor de este parámetro es 0.
- pOH: es la prioridad del OH que se desea alojar.
- pOH_{ADR}: es la prioridad del OH ya alojado en la ADR en donde se está aplicando el algoritmo LSP.

Este es un parámetro esencial.
Este parámetro es fundamental a la hora de instanciar OH que requieran instanciación inmediata como es el caso de sistemas preemptivos, en los cuales situaciones de reconfigurabilidad debidas a eventos asíncronos, que no pueden ser programados o que requieran atención inmediata, implican un sistema más reactivo donde la inmediata ejecución de una tarea de mayor prioridad es necesaria. Este reemplazo de una tarea por otra de mayor prioridad debe realizarse preservando la integridad del sistema para hacerlo tolerable a fallos.

- **MUDABILIDAD**: Este parámetro permite saber si un OH que está siendo ejecutado en una ADR puede ser mudado a otra ADR que puede o no estar ocupada. El valor que puede tomar este parámetro es 100 si el OH es mudable y 0 si no lo es.
La mudabilidad del OH depende de la diferencia de prioridad que posee el OH que está alojado en la ADR en donde se aplica el algoritmo LSP y la prioridad del OH que está alojado en la ADR en donde se podría mudar. Si esta diferencia es mayor que cero la $MUDABILIDAD = 100$; si es menor que cero la $MUDABILIDAD = 0$. Este es un parámetro deseable.

D. Estructura de Agregación

Una vez que se ha creado el árbol de criterios y extraído de este las preferencias elementales, el siguiente paso es construir la estructura de agregación. Para ello es necesario construir una estructura que agrupe las preferencias elementales en una única preferencia global.

La estructura de agregación se construye agrupando preferencias como entradas a funciones de Conjunción-Disyunción Generalizadas (CDG). Estas funciones aceptan como datos de entrada un conjunto de preferencias elementales (e_i) con sus respectivos pesos (w_i) y devuelven una preferencia agregada como salida E.

1) Funciones de Conjuncióón-Disyunción Generalizadas(CDG): La función de agregación de preferencias que propone LSP está basada en la media geométrica de potencias pesadas, de acuerdo a la ecuación 6:

$$E = (w_1 * e_1^r + w_2 * e_2^r + ... + w_k * e_n^r)^{\frac{1}{r}} \quad (6)$$

donde:

$$-\infty \leq r \leq \infty$$

$$0 \leq w_i \leq 1$$

$$w_1 + ... + w_n = 1$$

De esta manera se puede notar que en realidad E es un esquema general representativo de funciones de agregación que dependen del valor elegido para **r**, en función de la cantidad de criterios **n** definidos como las entradas de la fución. En la Tabla I se pueden observar las distintas funciones que se pueden generar en función del valor de **r**. También se puede observar que una función CDG puede ser mandatoria o no en función del valor de **r**, como se observa en la columna **Requerimiento Mandatorio (Req Mand)** de la Tabla I. El término **mandatoria** se define más adelante.

TABLA I
FUNCIONES DE DISYUNCIÓN-CONJUNCIÓN GENERALIZADAS

Req Mand	Nombre de la Operación	Símb	Grado de Conj	Valor de r n=2	n=3	n=4	n=5
NO	Disyunción	D	0.000	$+\infty$	$+\infty$	$+\infty$	$+\infty$
NO	Disy Fuerte	D+	0.125	9.52	11.09	12.28	13.16
NO	Disy Media	DA	0.250	3.83	4.45	4.82	5.09
NO	Disy Débil	D-	0.375	2.02	2.19	2.30	2.38
NO	Media Arit	A	0.5	1.00	1.00	1.00	1.00
NO	Conj Débil	C-	0.625	0.26	0.20	0.17	0.16
SI	Conj Media	CA	0.750	-0.72	-0.73	-0.71	-0.67
SI	Conj Fuerte	C+	0.875	-3.51	-3.51	-2.18	-2.61
SI	Conjunción	C	1.000	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Para definir las funciones que se deben utilizar en la estructura de agregación es necesario clasificar los criterios elementales extraídos del sistema en esenciales y no esenciales.

2) Criterios Esenciales: Los criterios esenciales son aquellos que deben cumplir con un valor de satisfacción distinto de cero para que la salida de la función CDG entregue un valor distinto de cero. En el caso que el valor de un criterio esencial sea igual a cero, debe ser cero la salida de la función CDG.

En el sistema los criterios esenciales son:

- INSTANCIABILIDAD: el OH debe poder alojarse en la ADR en donde se está evaluando, sino la salida de la función debe ser cero.
- PRIORIDAD: la prioridad del OH que se desea alojar en alguna de las ADRs del sistema debe ser mayor que la prioridad del OH que ya está alojado. En caso contrario la salida local de la función debe cero. En el caso de que sea cero se calcula la mudabilidad del OH que ocupa la ADR, si también es cero debe ser cero la salida de la función LSP.

Entonces las funciones que tienen como entrada uno o varios criterios esenciales deben ser mandatorias. El concepto de mandatoria indica que si un criterio es esencial y es cero, debe ser cero la salida de la función CDG que lo contiene.

3) Criterios NO Esenciales: Los criterios NO esenciales pueden no cumplir con un valor de satisfacción sin que esto haga cero la salida de la función.

En el sistema los criterios NO esenciales son:

- BRAM: si el OH es instanciable en una ADR se cumple con los requerimientos de este elemento para su funcionamiento.
- FF: si el OH es instanciable en una ADR se cumple con los requerimientos de este elemento para su funcionamiento.
- DSP: si el OH es instanciable en una ADR se cumple con los requerimientos de este elemento para su funcionamiento.
- ADR_LIBRE: la ADR puede estar libre u ocupada.
- MUDABILIDAD: este criterio indica si el OH que ocupa la ADR puede o no ser mudado a otra ADR.

Entonces las funciones que tienen como entrada solamente criterios NO esenciales deben ser NO mandatorias.

Una vez implementadas las funciones CDG necesarias para cumplir con los requerimientos del sistema, la estructura de agregación definida para obtener la preferencia global queda como se muestra en la Fig. 1.

donde

- e1: INSTANCIABILIDAD
- e2: BRAM
- e3: FF
- e4: DSP
- e5: VELOCIDAD
- e6: ADR_LIBRE
- e7: PRIORIDAD
- e8: MUDABILIDAD
- CDGx: son funciones intermedias.
- E: es la salida de la función.
- wx: pesos de preferencias o de funciones intermedias.

E. Función LSP

A partir de la estructura de agregación se puede definir la función LSP del decisor.

$$CDG_1 = (w_{e_2} * e_2^{r_A} + w_{e_3} * e_3^{r_A} + w_{e_4} * e_4^{r_A})^{\frac{1}{r_A}} \quad (7)$$

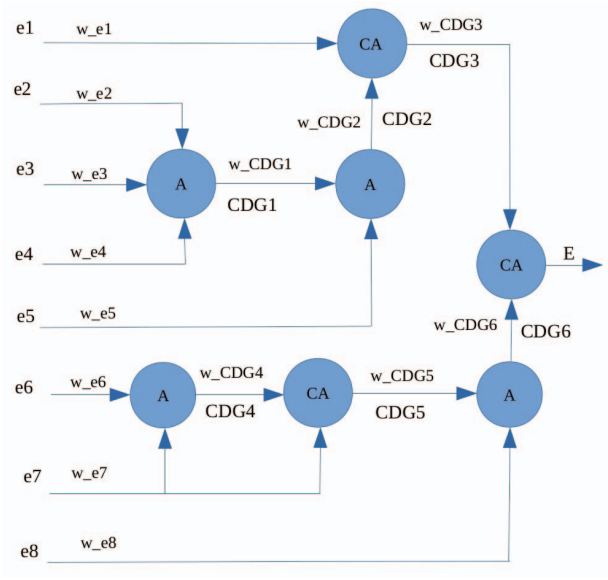

Fig. 1. Estructura de agregación.

$$CDG_2 = (w_{CDG_1} * CDG_1^{r_A} + w_{e_5} * e_5^{r_A})^{\frac{1}{r_A}} \quad (8)$$

$$CDG_3 = (w_{e_1} * e_1^{r_{CA}} + w_{CDG_2} * CDG_2^{r_{CA}})^{\frac{1}{r_{CA}}} \quad (9)$$

$$CDG_4 = (w_{e_6} * e_6^{r_A} + w_{e_7} * e_7^{r_A})^{\frac{1}{r_A}} \quad (10)$$

$$CDG_5 = (w_{CDG_4} * CDG_4^{r_{CA}} + w_{e_7} * e_7^{r_{CA}})^{\frac{1}{r_{CA}}} \quad (11)$$

$$CDG_6 = (w_{CDG_5} * CDG_5^{r_A} + w_{e_8} * e_8^{r_A})^{\frac{1}{r_A}} \quad (12)$$

$$E = (w_{CDG_3} * CDG_3^{r_{CA}} + w_{CDG_5} * CDG_5^{r_{CA}})^{\frac{1}{r_{CA}}} \quad (13)$$

F. Selección de los Pesos W

Cada criterio elemental e_i y funciones CDG que son entradas a funciones **A** y **CA** poseen pesos, como se pueden ver en las funciones generadas. Estos pesos se han elegido de la siguiente manera:

- $w_{e_1} = w_{CDG_2} = 0.5$: Se elijen estos valores para darle el mismo peso a la instanciabilidad y el resultado de los recursos utilizados y velocidad del OH. En este caso la función que agrupa estos dos pesos será cero cuando la instanciabilidad sea cero.
- $w_{e_2} = 0.3; w_{e_3} = 0.4; w_{e_4} = 0.3$: Se le da más peso a los recursos DSP debido a que permiten acelerar la velocidad de los objetos que los utilizan.

- $w_{CDG_1} = w_{e_5} = 0.5$: La función que es alimentada por estos pesos es No Mandatoria. En este caso los pesos valen lo mismo.
- $w_{e_6} = w_{e_7} = 0.5$: Se elijen estos pesos debido a que es tan importante si la ADR está libre como así también la prioridad presente.
- $w_{CDG_4} = w_{e_7} = 0.5$: Se elijen estos pesos para dar igual preferencia.
- $w_{CDG_5} = 0.9; w_{e_8} = 0.1$: Se eligen estos pesos dando prioridad al desalojo por sobre la mudanza de un OH debido a que una mudanza de OH tiene una demora mayor de tiempo que un desalojo.
- $w_{CDG_3} = w_{CDG_6} = 0.5$

La selección de los pesos w_x se hace en función de la incidencia de un criterio **x** con respecto a otro siendo ambos criterios entradas de una misma función (por ejemplo la función CA). En este caso la condición es que la suma de todos los pesos de los criterios que llegan a una función sea igual a 1.

Los valores r_A y r_CA se seleccionan de acuerdo a la Tabla I.

V. Pruebas y Resultados de la Función LSP

Una vez definido el algoritmo LSP, se realizaron dos implementaciones para su verificación. La pimera en software basado en C++ para el análisis y verificación de su funcionalidad y luego se realizó una implementación real en Sistema en Chip (SoC) hibrido formado por un procesador Cortex A9 dual core y FPGA, para evaluar los tiempos transcurridos durante el proceso de decisión más los tiempos de instanciación de cada OH en la ADR correspondiente. Esta implementación se llevó a cabo en una placa de desarrollo Zedboard que posee una FPGA Zynq ZC702 de Xilinx.

Para ambas pruebas se definió un sistema formado por 4 ADRs y 6 OHs para ser instanciados en las ADRs.

La Tabla II indica las caraterísticas estáticas y dinámicas de las ADR. Los criterios estáticos son: **Aloja el OH** indica cuales OH pueden ser instanciados en qué ADR; **BRAM**, **FF** y **DSP** son los recursos hardware que posee cada una de las ADR. Mientras que los criterios dinámicos dependen del contexto y del momento de reconfiguración, los cuales son: LIBRE, PRIORIDAD y MUDABILIDAD cuyo valor de los dos últimos dependen de si la ADR está o no ocupada en ese momento y, si lo está, de la prioridad del OH que la ocupa.

TABLA II
Características de las ADRs

CRITERIOS	*ADR 1*	*ADR 2*	*ADR 3*	*ADR 4*
OH que aloja	1,2,3	1,4,5	2,4,6	3,5,6
BRAM	300	200	500	200
FF	1000	2000	1000	1500
DSP	50	40	50	40
LIBRE	x	x	x	x
PRIORIDAD	x	x	x	x
MUDABILIDAD	x	x	x	x

La Tabla III indica los criterios definidos para los OH definidos en este sistema.

TABLA III
Características de los OHs

CRITERIOS	*OH1*	*OH2*	*OH3*	*OH4*	*OH5*	*OH6*
BRAM	150	250	150	180	150	200
FF	900	900	900	950	1450	950
DSP	30	40	35	35	35	35
Vel ADR1	100	100	80	0	0	0
Vel ADR2	80	0	0	80	80	0
Vel ADR3	0	80	0	100	0	80
Vel ADR4	0	0	100	0	100	100
PRIORIDAD	X	X	X	X	X	X

En el caso de la prioridad, es un valor que lo asigna el planificador de ejecución de tareas cada vez que se selecciona un OH para ser ejecutado.

A. Implementación de la Función en C++

Para realizar las pruebas del algoritmo generado, se lo ha codificado en C++ y se lo ha implementado como una función. Esta función tiene como argumentos de entrada los criterios antes señalados y como salida retorna el valor calculado E como se puede ver en (13). Este algoritmo se aplica a cada una de las ADR que son parte del sistema.

Como ejemplo supongamos que el OH a instanciar sea del tipo 4 con una prioridad de 80, y que las ADR, en el momento de aplicar la función de decisión de instanciación, se encuentran en el contexto indicado en la Tabla IV.

TABLA IV
Estado de las ADR en el momento de instanciación de un OH tipo 4

CRITERIOS	*ADR 1*	*ADR 2*	*ADR 3*	*ADR 4*
OH que aloja	1,2,3	1,4,5	2,4,6	3,5,6
BRAM	300	200	500	200
FF	1000	2000	1000	1500
DSP	50	40	50	40
LIBRE	SI	NO	SI	NO
PRIORIDAD	0	60	0	10
MUDABILIDAD	0	100	0	100

Se aplica el algoritmo LSP a cada una de las ADR y el resultado se muestra en la Tabla V.

TABLA V
Resultados de las pruebas de la función LSP

CRITERIOS	*Área 1*	*área 2*	*Área 3*	*Área 4*
e1	0	100	100	0
e2	60	90	36	90
e3	95	47.5	95	63.3
e4	70	85.5	70	85.5
e5	0	80	100	0
e6	100	50	100	50
e7	80	20	80	70
e8	0	100	0	100
RESULTADOS	0	49.37	77.93	0

Como se puede observar, aplicando los criterios definidos, se obtiene que la mejor opción a instanciar el OH tipo 4 con una prioridad de 80 es en la ADR3. Se observa además que en el caso de la ADR2 se obtiene un valor distinto de

cero pero menor debido a que esta opción equivale a dos instanciaciones, ya que en este caso implica mudar el OH que está originariamente en la ADR2 y luego reconfigurarla.

Esta prueba se ejecutó usando el compilador gcc (Debian 6.3.0-18+deb9u1), en una notebook Lenovo IdeaPad 320 con un procesador Core i7 cuyo sistema operativo es Debian 10.2, con una velocidad de clock de 3.5 GHz. Al ejecutar el algoritmo el tiempo que llevó su ejecución es de 70 ciclos de reloj promedio.

B. Implementación del Sistema en un SoC

Luego de esta prueba en PC de escritorio, este escenario fue implementado en un SoC Zynq, en una placa de desarrollo Zedboard. El SoC Zynq-7000 AP SoC consta de un procesador ARM de doble núcleo y una FPGA del tipo Zc702 de Xilinx.

El sistema hardware esta formado por una parte estática en donde se implementó un sistema base y 4 zonas reconfigurables definidas. En esta prueba de nuestro algoritmo se diseñó el sistema utilizando el software dynDeT [11], desarrollado por la Universidad de Castilla-La Mancha, España. Utilizando como punto de entrada para este software el diagrama en bloque de nuestro sistema desarrollado utilizando Vivado, dynDeT permite la creación de los bitstreams parciales a partir de una descripción VHDL de los OH, y de la selección de las diferentes áreas a utilizar. Los diferentes bitstreams generados fueron almacenados en memoria DDR en direcciones ya fijadas para la ubicación de cada uno. Por defecto la plataforma trabaja a una frecuencia de 100 MHz, pero las áreas dinámicas y otros componentes pueden modificar esta frecuencia de reloj durante la ejecución, de esta manera se ajustaron las diferentes frecuencias de operación que sirvieron como uno de los criterios de selección de configuración del método propuesto.

1) Sistema base: El sistema base definido para este caso de estudio está formado por una parte de diseño en zona estática, no reconfigurable dinámicamente, y cuatro zonas reconfigurables donde se irán intercambiando los OH diseñados. La parte estática, Fig. 2 está formada por el Procesador, un componente denominado rController, diseñado para gestionar el proceso de reconfiguración parcial, y otro componente denominado Factoría. El rController es el mecanismo central del proceso de reconfiguración, recibe indicaciones del método LSP y ejecuta el proceso de reconfiguración. Este proceso comprende el arranque o la detención del OH a ser desalojado, salvar su estado si es necesario para su posterior reinserción, y enviar el pedido de reconfiguración con la correspondiente dirección del bitstream parcial que corresponde a la Factoría.

La Factoría se encarga de la manipulación y transferencia de los bitstreams. Es el componente que habilita la instanciación de nuevos componentes transfiriendo el bitstream correspondiente desde la memoria de almacenamiento a la memoria de configuración de la FPGA usando un bus dedicado para tal fin. Se comporta como un DMA especializado que transfiere los datos de memoria al ICAP (Interna Configuration Access Port) de la FPGA

Una vez definido nuestro sistema y generados los correspondientes bitstreams, se ejecutó el método LSP generado

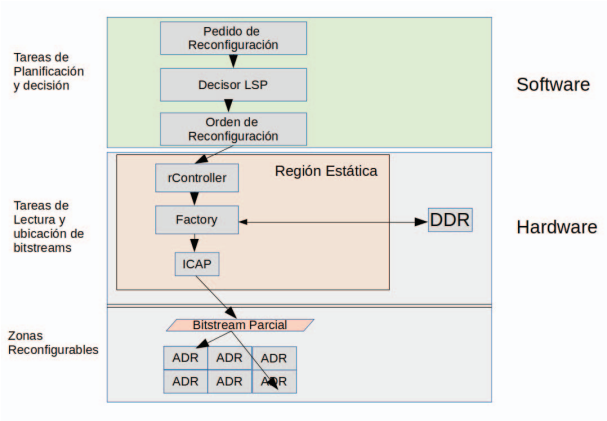

Fig. 2. Sistema Base.

en el procesador ARM y según su resultado, el sistema fue reconfigurado utilizando los bitstreams almacenados en memoria. Los tiempos de reconfiguración dependen del tamaño del bitstream, el que a su vez está determinado por el tamaño del área parcial de reconfiguración.

Los tiempos de reconfiguración obtenidos se muestran en la tabla VI.

TABLA VI
TIEMPOS DE RECONFIGURACIÓN

Área Dinámica	Tamaño bitstream	Tiempo de reconfiguración
Área 1	130Kb	0,65 ms
Área 2	280 Kb	1,45 ms
Área 3	500Kb	2,51 ms
Área 4	360Kb	1,85 ms

VI. CONCLUSIONES

En este trabajo se ha presentado un novedoso sistema de decisión de reconfiguraciones parciales multicriterio. Mediante la aplicación del método multicriterio LSP a sistemas reconfigurables dinámicamente es posible planificar la ubicación de objetos hardware en la zona de reconfiguración más conveniente para el diseño, en función de los criterios que se han definido para la toma de decisiones, y el orden de ejecución de tales reconfiguraciones parciales.

Partiendo de la identificación de criterios de selección tales como la prioridad de ejecución de los objetos a instanciarse, tamaño y recursos lógicos de las áreas a usar, se ha desarrollado un método de decisión que permite obtener las configuraciones óptimas para nuestro sistema en ejecución.

Se observa además, que el método propuesto permite la planificación de recursos y de reconfiguraciones en sistemas preemptivos de tiempo real, y es escalable a nuevos criterios tales como por ejemplo consumo de potencia, cantidad de accesos a memoria, etc, que pueden definirse en sistemas más complejos.

REFERENCIAS

[1] Xilinx.com. (2019). Xilinx - Adaptable. Intelligent.. [online] Available at: https://www.xilinx.com/ [Accessed 21 May 2019].

[2] J. Dondo, J. Barba, F. Rincón, F. Moya, J. C. López. Dynamic objects: Supporting fast and easy run-time reconfiguration in FPGAs. Journal of Systems Architecture. Volume 59. Issue 1. 2013. ISSN 1383-7621.
[3] Song Chen, Jinglei Huang, Xiaodong Xu, Bo Ding, Qi Xu. Integrated Optimization of Partitioning, Scheduling, and Floorplanning for Partially Dynamically Reconfigurable Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. arXiv:1803.03748v2. 2018
[4] K. Schleupen, S. Lekuch, R. Mannion, Z. Guo, W. Najjar and F. Vahid, Dynamic Partial FPGA Reconfiguration in a Prototype Microprocessor System. International Conference on Field Programmable Logic and Applications. Amsterdam. 2007. pp. 533-536.
[5] Edward Chen, Victor Lesau, Dorian Sabaz, Lesley Shannon, William Gruver. FPGA Framework for Agent Systems Using Dynamic Partial Reconfiguration. 2011. Pp 94-102. 10.1007/978-3-642-23181-0_9.
[6] Tian Xia, Jean-Christophe Prévotet, Fabienne Nouvel. Microkernel dedicated for dynamic partial reconfiguration on ARM-FPGA platform. (January 2015), 31-36. DOI=http://dx.doi.org/10.1145/2724942.2724947
[7] Charitopoulos G., Koidis I., Papadimitriou K., Pnevmatikatos D. Hardware Task Scheduling for Partially Reconfigurable FPGAs. In: Sano K., Soudris D., Hübner M., Diniz P. (eds) Applied Reconfigurable Computing. ARC 2015. Lecture Notes in Computer Science, vol 9040. Springer, Cham.
[8] Julio Dondo Gazzano, Francisco Sanchez Molina, Fernando Rincón, and Juan Carlos López. Integrating Reconfigurable Hardware-Based Grid for High Performance Computing. The Scientific World Journal, vol. 2015, Article ID 272536, 19 pages, 2015. https://doi.org/10.1155/2015/272536.
[9] A. Dasso, A. Funes, M. Peralta, C. Salgado. Una herramienta para la evaluación de sistemas. III Workshop de Investigadores en Ciencias de la Computación. 2001
[10] J.J. Dujmovic. A Method for Evaluation and Selection of Complex Hardware and Software Systems. The 22nd Int'l Conference for the Resource Management and Performance Evaluation of Enterprise CS. CMG 96 Proceedings, 1:368–378, 1996.
[11] GitHub. (2019). juliancaba/dynDeT. [online] Available at: https://github.com/juliancaba/dynDeT [Accessed 13 Nov. 2019].

Alejandro Nuñez Manquez Completó sus estudios de Ingeniería Electricista Electrónica en la Universidad Nacional de San Luis en 2005. Es docente en las carreras Ingeniería Electrónica, Técnico Universitario en Electrónica y Profesorado en Tecnología Electrónica de la Facultad de Ciencias Físico - Matemáticas y Naturales de la misma Universidad, cumpliendo funciones como jefe de trabajos prácticos. Es director de la carrera Profesorado en Tecnología Electrónica. Es docente investigador, categoría V, integrante del proyecto de investigación "Sistemas embebidos para el procesamiento digital con FPGA".

Sus intereses en investigación son: Sistemas Embebidos, Reconfiguración Dinámica, Métodos Multicriterios Implementados en Sistemas con Reconfiguración Dinámica.

Julio Dondo Gazzano Completó su estudios de Ingeniería Electricista-Electrónica en la Universidad Nacional de San Luis, en 1996. Recibió el Título de Magister en Ingeniería de Software por la misma Universidad en 2007 y el Título de Doctor en Ingeniería de Computadores por la Universidad de Castilla-La Mancha en 2010. Sus principales líneas de investigación son: Diseño de SoC basados en FPGAs, Hardware Dinámicamente Reconfigurable, Sistemas Distribuidos Heterogéneos. Actualmente trabaja como Profesor Asociado y es Director de la Carrera de Máster en Sistemas Embebidos, y de la Especialización en Sistemas Embebidos, en el Departamento de Electrónica de la Facultad de Ciencias Físico-Matemáticas y Naturales, de la Universidad Nacional de San Luis, San Luis, Argentina.

Mario Marcelo Beron es Doctor en Ciencias de la Computación de la Universidad Nacional de San Luis de Argentina con reconocimiento del mismo grado por parte de la Universidade do Minho de Portugal. Se desempeña como docente-investigador del Departamento de Informática (UNSL). Es docente en la Maestría en Ingeniería de Software (UNSL). Los principales tópicos de investigación son Comprensión de Programas, Ingeniería Reversa, Lenguajes de Programación, Seguridad Informática, Sistemas Embebidos, entre otros

Carlos Sosa Paez Ingeniero Electricista Electrónico egresado de la Universidad Nacional de Córdoba en 1986. Se unió a la Universidad Nacional de San Luis en 1987. Profesor de la Cátedra de Diseño de Sistemas Digitales, miembro del Laboratorio de Electrónica, Investigación y Servicios de la Universidad Nacional de San Luis. Área de interés: diseño e implementación de arquitecturas RISC en dispositivos lógicos programables de última generación. Es autor y co-autor de varias publicaciones científicas.

Roberto Martín Murdocca nacido en Buenos Aires, Argentina el 24 de Septiembre de 1974. Profesor en Tecnología Electrónica egresado de la Universidad Nacional de San Luis (UNSL). Se unió a la Universidad Nacional de San Luis en 1998. Instructor de las Cátedras Interfaces y Procesadores II, miembro del Laboratorio de Electrónica, Investigación y Servicios de la Universidad Nacional de San Luis. Áreas de interés: sistemas de adquisición de datos, Sistemas Embebidos.