AEMO Paralelo para Resolver el Problema del Floorplanning Térmico en 3 Dimensiones

Ignacio Arnaldo, José L. Risco-Martín, José L. Ayala,
J. Manuel Colmenar, Alfredo Cuesta-Infante, J. Ignacio Hidalgo

Resumen— **Los algoritmos de floorplanning térmico tratan de ubicar de forma estratégica los distintos módulos hardware que componen una arquitectura con el fin de obtener una temperatura reducida y homogénea. La mayor parte de las propuestas utilizan algoritmos evolutivos para realizar esta tarea. La fase de evaluación térmica de las distintas configuraciones se convierte en el cuello de botella de estos algoritmos, consumiendo el 99.5 % del tiempo de ejecución en el caso del mejor floorplanner propuesto hasta la fecha. La contribución de este trabajo es una paralelización de la fase de evaluación siguiendo un modelo Maestro-Esclavo que permite reducir de forma notable el tiempo de optimización del floorplanner térmico.**

Palabras clave— **Floorplanning, Algoritmo Evolutivo Multi-Objetivo, modelo Maestro-Esclavo**

I. Introducción

En los últimos años, el número de transistores en los circuitos integrados ha aumentado según la tendencia propuesta por Gordon E. Moore, quien pronosticó que el número de transistores se doblaría cada dieciocho meses (su observación se conoce como la Ley de Moore). Esta progresión ha sido posible gracias a la continua reducción del tamaño de los transistores, lo que ha liberado área de integración abriendo nuevas posibilidades en el diseño de procesadores. La tendencia seguida para aprovechar esta mayor disponibilidad de área es incluir recursos adicionales en un sólo chip. En efecto, la industria se ha decantado por la fabricación de máquinas paralelas capaces de proporcionar el aumento de rendimiento esperado por los usuarios. Así, las arquitecturas Multi-Processor Systems-on-Chip (MPSoCs) no sólo han llevado a una mejora del rendimiento global, sino que también han permitido disminuir la frecuencia de funcionamiento. De esta forma, se ha conseguido mantener la potencia disipada en niveles aceptables. Hasta la fecha, el chip en desarrollo Single-chip Cloud Computer (SCC) fabricado por Intel es el que cuenta con más procesadores, alcanzando los 48 cores integrados en un sólo chip [1].

Ignacio Arnaldo está financiado por el proyecto Avanza Competitividad I+D+I: TSI-020100-2010-962. Este trabajo también ha sido financiado con la becas TIN 2008-00508 y MEC CONSOLIDER CSD00C-07-20811.

José L. Risco-Martín, José L. Ayala, y J. Ignacio Hidalgo son miembros del Departamento de Arquitectura de Computadores y Automática (DACYA) de la Universidad Complutense de Madrid (e-mail: jlrisco@dacya.ucm.es; jayala@fdi.ucm.es; hidalgo@dacya.ucm.es)

J. Manuel Colmenar y Alfredo Cuesta-Infante trabajan en el C.E.S. Felipe II (UCM, Campus de Aranjuez) (e-mail: jmcolmenar@ajz.ucm.es; acuestai@pdi.ucm.es)

No obstante, la reducción del tamaño de los transistores acarrea problemas como el aumento de las corrientes de fuga y el aumento de la temperatura del chip, directamente relacionada con la densidad de potencia [2]. Estos problemas afectan a la fiabilidad y vida media de los circuitos integrados [3] y su impacto se ve incrementado cuando aumenta el número de elementos en el chip. Estos efectos se pueden atenuar con políticas de emplazamiento que ubican estratégicamente los componentes con mayor densidad de potencia. Como consecuencia, el problema de la ubicación de bloques o Floorplanning gana relevancia. En efecto, si un bloque caliente se emplaza entre dos bloques fríos, la temperatura del primero desciende [4].

A parte del calor generado, la inclusión de un mayor número de elementos en un chip provoca un aumento de los retardos debidos a la mayor longitud del cableado que conecta los diferentes componentes, afectando gravemente al rendimiento del chip. La integración en 3D se ha perfilado como una solución de futuro en el campo de los MPSoCs [5]. Esta tecnología está en fase de desarrollo [6] y se predice que ofrecerá ventajas como el aumento de prestaciones debido a la reducción del cableado, la reducción del área total del chip o el potencial para integrar arquitecturas heterogéneas en un sólo chip [7]. Los beneficios de la tecnología 3D han sido ampliamente reconocidos. Sin embargo, existe una gran necesidad de herramientas que den soporte de simulación y de diseño para estas nuevas y complejas arquitecturas ya que éstas plantean nuevos retos. En particular, se considera que el aumento de temperatura en estos chips es la principal dificultad. En efecto, en una configuración en 3D la densidad de potencia aumenta con el número de capas ya que se ubica un mayor número de componentes en un área menor.

Con el desarrollo de este trabajo, se proporciona una herramienta capaz de realizar el Floorplanning en tres dimensiones de arquitecturas heterogéneas compuestas por una gran cantidad de procesadores integrados de forma óptima y en un tiempo reducido mediante la implementación paralela de un algoritmo evolutivo multi-objetivo.

El resto del trabajo está organizado de la siguiente manera. El trabajo relacionado se presenta en la siguiente sección. En la sección 3 se describe el Algoritmo Evolutivo Multi-Objetivo. Los resultados experimentales se comentan en la sección 4 y, por último, las conclusiones se exponen en la sección 5.

II. Trabajo Relacionado

Los algoritmos de floorplanning térmico tratan de ubicar las unidades funcionales de forma estratégica para obtener una distribución de la temperatura homogénea en el chip. Este problema ha sido típicamente formulado como un problema de optimización combinatoria, siendo adecuado el uso de algoritmos evolutivos para su resolución. Existen numerosas propuestas como Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) o Algoritmos Genéticos (GA). Estas técnicas implementan estrategias diferentes, no obstante poseen las siguientes similitudes:

1. en cada iteración se evalúa un reducido número de soluciones (denominado población) en comparación con la cardinalidad del espacio de soluciones
2. las soluciones (o individuos) se representan de forma adecuada para su evaluación
3. existe un método iterativo para producir la población de la próxima generación
4. existe una función de fitness que evalúa los individuos

Las propuestas tradicionales están basadas en representaciones eficientes de las soluciones como la notación polaca [8], los árboles ordenados (O-tree) [9] o Combined Bucket Array [10]. Una desventaja común a todos estos métodos reside en el hecho de que fueron concebidos para reducir área o cableado. Esto representa una grave limitación ya que hoy en día es habitual trabajar con área fija y los objetivos del floorplanning son reducir cableado y temperatura [11]. Así, en dos dimensiones Hung *et al* redujeron la temperatura máxima usando algoritmos genéticos [12]. Con el mismo propósito se utilizó la técnica del enfriamiento simulado en [13]. Para plataformas de tres dimensiones, Healy *et al* [14] propusieron una combinación de programación lineal y enfriamiento simulado. Estos trabajos resuelven un problema de optimización mono-objetivo que sólo tiene en cuenta el impacto en la fiabilidad causado por las altas temperaturas. Por lo tanto, no proporcionan soluciones óptimas que minimicen simultáneamente la temperatura y los retardos debidos a la longitud del cableado.

Los Algoritmos Evolutivos Multi-Objetivo se presentan como una buena opción para tratar el problema del 3D Thermal-Aware Floorplanning ya que en éste, se trata de minimizar simultáneamente el cableado y la temperatura del chip. Estos algoritmos permiten una exploración eficiente del espacio de soluciones siendo, por tanto, buenos candidatos para realizar tareas de Architecture Exploration. Así, en [15] se consigue reducir notablemente la temperatura de arquitecturas manycore utilizando un algoritmo evolutivo basado en Non-Dominated Sorting Genetic Algorithm (NSGA-II) [16]. Sin embargo este trabajo presenta un cuello de botella en la evaluación de las soluciones debido a la complejidad computacional de ésta, resultando en largos tiempos de ejecución al tratar de optimizar arquitecturas complejas.

Este trabajo toma como punto de partida el algoritmo presentado en [15] y [17]. Nuestra contribución consiste en la paralelización del thermal-aware floorplanner propuesto, obteniendo una reducción notable en el tiempo de ejecución. De esta forma, proporcionamos una herramienta de diseño capaz de agilizar tareas de exploración de nuevas arquitecturas.

Los algoritmos evolutivos son intrínsecamente paralelos pero es en la función de evaluación donde se puede obtener el mayor speed-up. La Tabla I muestra los tiempos de ejecución de los diferentes métodos involucrados en el algoritmo considerado. En este caso el valor de fitness se obtiene utilizando un modelo térmico que, en conjunción con la verificación de la factibilidad de las soluciones, consume el 99,5 % del tiempo de ejecución. Esta tarea constituye por lo tanto un evidente cuello de botella, justificando así la implementación paralela propuesta en este trabajo.

Método/Operador	Tiempo
evaluación	$3{,}13 \times 10^7 ms$
selección	$2{,}57 \times 10^2 ms$
reducción	$2{,}51 \times 10^2 ms$
mutación	$5{,}76 \times 10^1 ms$
cruce	$1{,}0\ ms$

TABLA I
Tiempo total de ejecución de los diferentes métodos involucrados en el algoritmo propuesto en [17]

Los algoritmos evolutivos pueden ser paralelizados a nivel de población o de individuo. En el primer caso, la población se divide en varias subpoblaciones que evolucionan independientemente pero con una probabilidad de interacción. Los dos modelos típicos son el modelo de islas y el modelo de vecindad. En el modelo de islas, los individuos pueden ser migrados de una isla a otra con una cierta probabilidad. Existe una gran variedad de topologías, siendo las más frecuentes la de anillo, la de estrella y las mallas n-dimensionales. Generalmente, cuando tiene lugar una migración, se reemplazan los k peores individuos de una isla por los recién llegados, que son a su vez los k mejores individuos de otra isla. En los modelos de vecindad, los individuos son ubicados en grids de una o dos dimensiones y cada individuo interacciona solamente con sus vecinos. En el caso de la paralelización a nivel de individuo, es típico paralelizar únicamente la fase de evaluación ya que ésta suele ser la operación de mayor coste computacional. Así, todos los métodos involucrados se ejecutan secuencialmente salvo la evaluación, que se realiza en paralelo. El modelo Maestro-Esclavo es especialmente adecuado para llevar a cabo esta idea, ya que únicamente la evaluación es realizada en paralelo. En este trabajo, implementamos este modelo. La particularidad de nuestra propuesta es que se trata de una ejecución multi-threaded, lo que permite incrementar el núme-

ro de trabajadores aprovechando los tiempos de idle de los distintos threads.

III. Algoritmo Evolutivo Multi-Objetivo Paralelo

En esta sección presentamos un algoritmo evolutivo multi-objetivo paralelo capaz de optimizar plataformas heterogéneas manycore en tiempo reducido.

A. Especificación del AEMO

En [17], se propone un Algoritmo Evolutivo Multi-Objetivo (MOEA) basado en el NSGA-II para resolver el problema del floorplanning de chips 3D. Las plataformas estudiadas están compuestas por una serie de capas de área fija apiladas en las que se ubican los distintos componentes. El proceso de floorplanning trata de optimizar el emplazamiento de los componentes de estos chips a la vez que minimiza el retardo debido a la interconexión de los distintos elementos. En este trabajo, mejoramos la propuesta presentada en [17] con una implementación paralela del algoritmo que reduce de forma notable el tiempo de ejecución.

A.1 Problema de la ubicación de bloques

Todos los bloques que modelan los distintos componentes de un sistema manycore tienen que ser ubicados en la pila 3D. Ésta impone una longitud máxima L, un ancho máximo W y una altura máxima H. Cada componente es representado por un bloque B_i de longitud l_i, ancho w_i, altura h_i. Se utilizan las coordenadas (x_i, y_i, z_i) de la esquina inferior izquierda de cada bloque para definir su ubicación en la pila, de tal forma que:
$0 \leq x_i \leq L - l_i$, $0 \leq y_i \leq W - w_i$, $0 \leq z_i \leq H - h_i$, Estos bloques no se pueden superponer. La Figura 1 muestra un esquema de la representación usada en este trabajo.

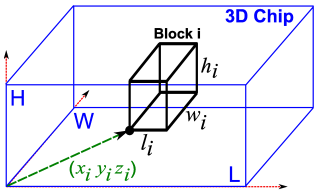

Fig. 1. Representación de un bloque

Para trabajar con un MOEA, se han de diseñar funciones de fitness, operadores genéticos y una representación de las soluciones adecuada para el problema.

Representación de las soluciones. Cada individuo representa un posible floorplan del sistema y se codifica con la secuencia de componentes ordenados según el turno en el que serán ubicados. Por ejemplo, si consideramos una arquitectura con 4 procesadores (C_1, C_2, C_3, C_4) y 4 memorias (L_1, L_2, L_3, L_4), un posible cromosoma sería $[C_1, L_2, C_3, L_4, C_4, L_3, C_2, L_1]$. En este caso el espacio de búsqueda tendría cardinalidad 8! (40320).

Operadores. La selección se lleva a cabo por torneo, seleccionando parejas aleatoriamente y escogiendo el mejor individuo de cada una. Los individuos seleccionados son cruzados para obtener la descendencia. El operador de cruce ha de tener en cuenta que todos los componentes han de aparecer exactamente una vez en el cromosoma. Utilizamos un cruce cíclico ya que éste asegura que los cromosomas obtenidos sean permutaciones de sus padres. La mutación consiste en, o bien intercambiar el contenido de dos posiciones del cromosoma, o bien en rotar un componente elegido aleatoriamente. Tanto el operador de cruce cíclico como la mutación por intercambio se muestran en la Figura 2.

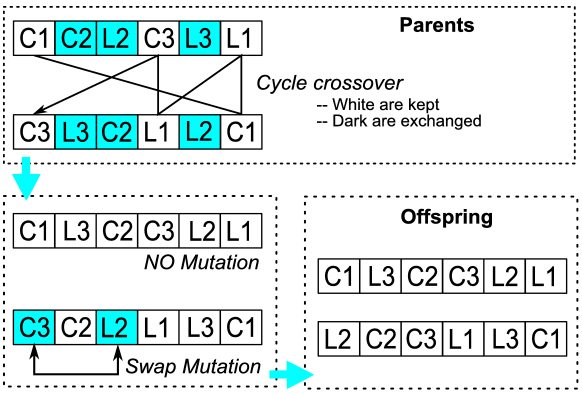

Fig. 2.Cruce cíclico y mutación por intercambio para permutaciones de seis elementos

Decodificación. Los individuos se decodifican de la siguiente manera. Los bloques se ubican secuencialmente en la pila 3D siguiendo el orden impuesto por la codificación de la solución. Para seleccionar las mejores coordenadas $r_i = (x_i, y_i, z_i)$ para cada bloque B_i, se establece una relación de dominancia teniendo en cuenta los bloques ya ubicados $B_j, j < i$.

Fitness. Consideramos tres funciones objetivo:
- El primer objetivo J_1 corresponde al número de restricciones topológicas violadas por una determinada configuración de los elementos.
- El segundo objetivo es el cableado, aproximado como la distancia de Manhattan entre bloques conectados entre sí, $J_2 = \sum_{j<i} |r_i - r_j|$.
- Por último, el impacto térmico se mide a través de la potencia disipada por celdas unitarias del chip. Usamos un modelo térmico que tiene en cuenta la densidad de potencia de una determinada celda y la de sus vecinas para aproximar la temperatura del chip. El uso de un modelo térmico preciso no es viable ya que éstos conllevan una gran carga computacional. Así, evaluamos la respuesta térmica de un determinado individuo de la siguiente manera:

$$J_3 = \sum_{i<j \in 1..n} (dp_i * dp_j)/(d_{ij}) \qquad (1)$$

donde dp_i es la densidad de potencia del bloque i y d_{ij} es la distancia euclídea entre los bloques i y j. Se ha demostrado que este modelo es lo suficientemente preciso para nuestro propósito [18].

B. Paralelización del Algoritmo

En este trabajo proponemos una implementación paralela del Algoritmo Evolutivo Multi-Objetivo presentado en la sección anterior utilizando un modelo Maestro-Esclavo. Como mencionamos previamente, la fase de evaluación consume el 99 % del tiempo de ejecución. Esto se debe a que las soluciones se han de decodificar y evaluar en cada generación del proceso por lo que el modelo Maestro-Esclavo se adecúa a nuestras necesidades. En efecto, resulta interesante explotar la naturaleza intrínsecamente paralela de los algoritmos evolutivos y llevar a cabo la evaluación de la población concurrentemente. La Figura 3 muestra la estrategia de paralelización usada en este trabajo: el maestro se queda con un subconjunto de la población y distribuye el resto entre n esclavos. A partir de ahora, nos referimos con el término trabajador tanto al maestro como a los distintos esclavos ya que todos realizan la tarea de evaluación.

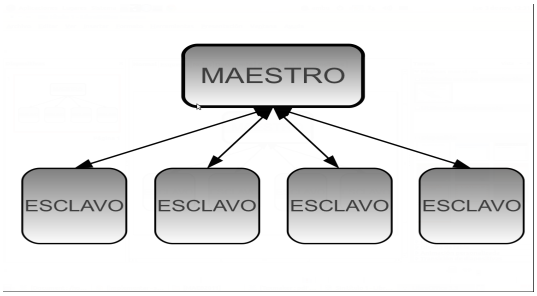

Fig. 3. Configuración Maestro-Esclavo

De esta forma, la carga computacional se reparte en $n + 1$ tareas distintas. Cuando los esclavos acaban su tarea, envían el resultado de la evaluación al maestro. El proceso se detiene hasta que todos los trabajadores hayan acabado su tarea. Esta repartición de la carga computacional es mucho más rápida que la ejecución secuencial ya que, al tratarse de una implementación multi-threaded, los tiempos de comunicación son despreciables.

Proponemos una implementación multi-hilo en la que el maestro ejecuta el hilo principal del algoritmo. Para obtener un rendimiento máximo, tanto el maestro como los esclavos realizan la evaluación de subconjuntos disjuntos de la población. De esta forma, esperamos obtener un speed-up similar al número de trabajadores (maestro + esclavos) involucrados en la ejecución del algoritmo. En un trabajo futuro, se estudiarán otras técnicas como el uso de arquitecturas masivamente paralelas para reducir el tiempo consumido por la fase de evaluación.

IV. Diseño de Experimentos y Análisis de Resultados

El trabajo experimental analiza el speed-up obtenido con la versión paralela del Algoritmo Evolutivo Multi-Objetivo a la vez que demuestra que las soluciones obtenidas son de la misma calidad. También analizamos la optimización térmica alcanzada por nuestro floorplanner.

Para evaluar nuestro algoritmo, consideramos dos escenarios en los que estudiamos dos arquitecturas heterogéneas inspiradas en la plataforma Niagara:

- Escenario A: estudio de la plataforma denominada A48. Esta arquitectura cuenta con 32 procesadores SPARC y 12 Power6, 72 memorias y 6 crossbars usados para la comunicación entre procesadores.
- Escenario B: estudio de la plataforma denominada A128. Esta arquitectura cuenta con 128 procesadores: 96 SPARC y 32 Power6. Además incluye 192 memorias y 16 crossbars, sumando un total de 336 componentes.

El algoritmo ha de ubicar todos estos elementos en 4 y 9 capas respectivamente. La primera arquitectura refleja el estado del arte en integración manycore mientras que la segunda es representativa de arquitecturas que se fabricarán en el corto plazo.

A. Análisis del Speed-up y Validación de los Resultados

Como primer análisis, estudiamos el speed-up obtenido con la versión paralela del floorplanner. El objetivo de este estudio es encontrar el número óptimo de trabajadores (maestro + esclavos) que maximice el speed-up. Por lo tanto, realizamos un barrido paramétrico del número de trabajadores en el escenario A (arquitectura con 48 procesadores). Fijamos el tamaño de la población en 100 individuos y el número de generaciones en 250. De esta forma comparamos los tiempos de ejecución obtenidos al variar el número de trabajadores. Los experimentos se realizan con un Intel Core-i5 dotado de 4 procesadores trabajando a 2.80GHz. La Tabla II muestra el speed-up (su) obtenido y el tiempo (t) de ejecución de la optimización de la arquitectura de 48 procesadores (escenario A) con un número de trabajadores (tr) entre 1 y 7.

$\#tr$	1	2	3	4	5	6	7
t (s)	24171	12398	8904	6918	6380	6421	6665
su	1	1.95	2.72	3.49	3.79	3.76	3.63

TABLA II
Tiempos de ejecución y speed-ups obtenidos en el escenario A

La Figura 4 muestra los distintos speed-ups obtenidos en el escenario A (48 procesadores). En este caso, el speed-up obtenido depende solamente del número de trabajadores ejecutados simultáneamente ya que el coste de las comunicaciones entre el maestro y los esclavos es despreciable. Se puede ver como el speed-up aumenta de forma casi lineal hasta que el número de trabajadores alcanza el número de procesadores (4 en este caso). Sin embargo, el speed-up

máximo se alcanza con 5 trabajadores. Esto se debe a que los trabajadores son ejecutados en una misma CPU mediante una implementación multi-hilo. El rendimiento se incrementa al incluir un trabajador adicional ya que se aprovechan los tiempos de idle de los distintos threads. Así, la configuración con 1 maestro y 4 esclavos (5 trabajadores) es la elección óptima.

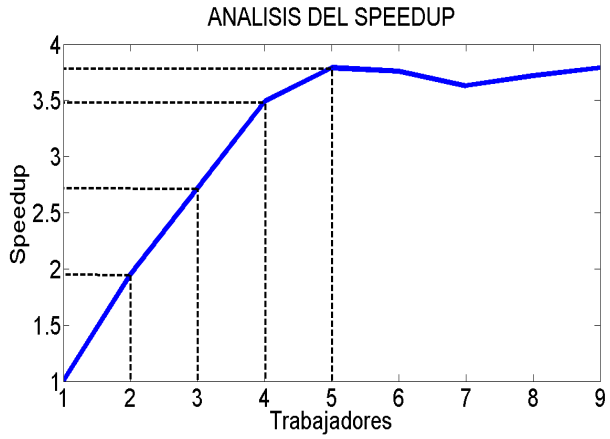

Fig. 4. Speed-up obtenido en el escenario A

Una vez obtenido el número óptimo de trabajadores, optimizamos la plataforma de 128 cores (escenario B). Esta vez realizamos un barrido paramétrico reducido en el número de trabajadores, eligiendo siempre valores cercanos al encontrado previamente (5 trabajadores). Esta estrategia es necesaria ya que el coste de la fase de evaluación crece con el tamaño de la arquitectura. El número de generaciones ha de ser escalado cuando aumenta el tamaño del problema, es decir, cuando se incrementa la cantidad de componentes a ubicar. Por ello, fijamos el número de generaciones igual al número total de componentes. En este caso tenemos 336 elementos a ubicar: 128 procesadores, 192 memorias y 16 crossbars. El tamaño de la población sigue siendo de 100 individuos. Ejecutamos el algoritmo paralelo con 3, 4, 5 y 6 trabajadores. La tabla III muestra los tiempos de ejecución obtenidos en este caso.

#nodes	3	4	5	6
time	204165s	175855s	162184s	163093s

TABLA III
Tiempos de ejecución obtenidos en el escenario B

En este caso, podemos ver que la ejecución más rápida también se obtiene para la configuración de 5 trabajadores (un maestro y 4 esclavos). La Figura 5 muestra los speed-ups obtenidos en el escenario B (128 cores). Como en el caso anterior, se puede ver que el speed-up máximo se obtiene con 5 trabajadores, es decir, un maestro y 4 esclavos. De nuevo, esto se debe a la implementación multi-hilo del modelo Maestro-Esclavo que, al aprovechar los tiempos de idle de los distintos threads, permite mejorar el rendimiento incluyendo un trabajador adicional.

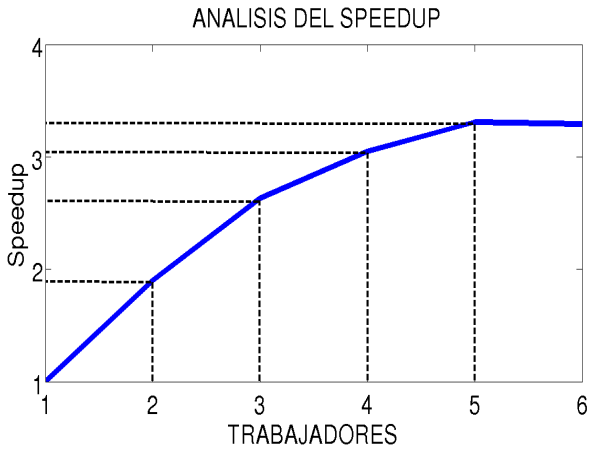

Fig. 5. Speed-up obtenido en el escenario B

Una vez presentado el speed-up obtenido, confirmamos que las soluciones propuestas por la versión paralela son de la misma calidad que las propuestas por la versión secuencial. Para ello, realizamos un análisis térmico de las soluciones optimizadas propuestas en los escenarios A y B (48 y 128 cores respectivamente).

B. Análisis Térmico

Por último, presentamos la optimización térmica realizada por nuestro algoritmo. Nuestro floorplanner trabaja con un área fija y trata de minimizar tanto el cableado como la temperatura máxima del chip. Para realizar los experimentos, utilizamos valores de consumo de potencia y de área reales. En [19] encontramos que el consumo de potencia del procesador SPARC es de 4W cuando trabaja a 1,4GHz. En el caso del POWER6, el valor estimado de consumo de potencia es de 2,6W (ver [20]). Consideramos las siguientes áreas: $3{,}24mm^2$ and $1{,}5mm^2$ para los procesadores SPARC y POWER6 respectivamente (ver [19] y [20]). El consumo de potencia y el área de las memorias se obtienen con el software CACTI [21].

B.1 Escenario A: Plataforma de 48 cores

Comparamos una configuración optimizada de la arquitectura heterogénea de 48 cores A48 con la plataforma homogénea denominada NIAGARA48 presentada en la Figura 6 (compuesta también por 48 procesadores). En esta última se aplica una estrategia de replicación ubicando una arquitectura original de 12 cores en cada una de las capas. Como consecuencia, los procesadores SPARC (SPC) se alinean verticalmente dando lugar a puntos calientes. Por otra parte, la Figura 7 muestra los mapas térmicos de las diferentes capas de una solución no dominada propuesta por el floorplanner. Esta figura muestra una ubicación optimizada de los procesadores SPARC (SPC) y Power6 (P6), de las memorias (L2) y de los crossbars (Cross) obtenida con nuestro algoritmo. En esta configuración, los principales focos de calor (procesadores SPARC) tienden a ser

colocados en los bordes del chip y en las capas exteriores de éste, tratando de separarlos lo máximo posible. Además, nuestro floorplanner tiene en cuenta la difusión vertical del calor, por lo que evita la superposición vertical de los cores. Los crossbars son ubicados en las capas intermedias para minimizar el cableado.

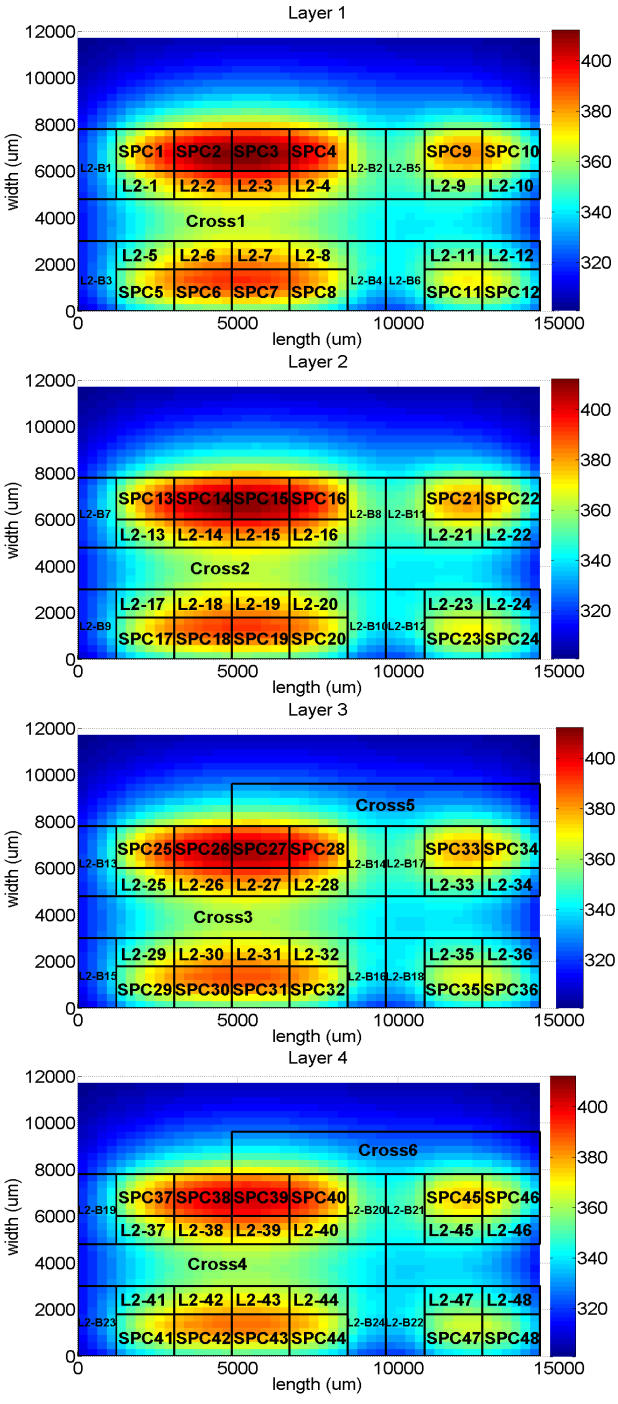

Fig. 6.Mapas térmicos de las 4 capas de la plataforma NIAGARA48

Las métricas utilizadas para el análisis térmico de las dos plataformas propuestas son la temperatura máxima, la temperatura media y el máximo gradiente del chip. En la Tabla IV presentamos los valores obtenidos para estas dos configuraciones. Estos resultados demuestran que nuestro floorplanner

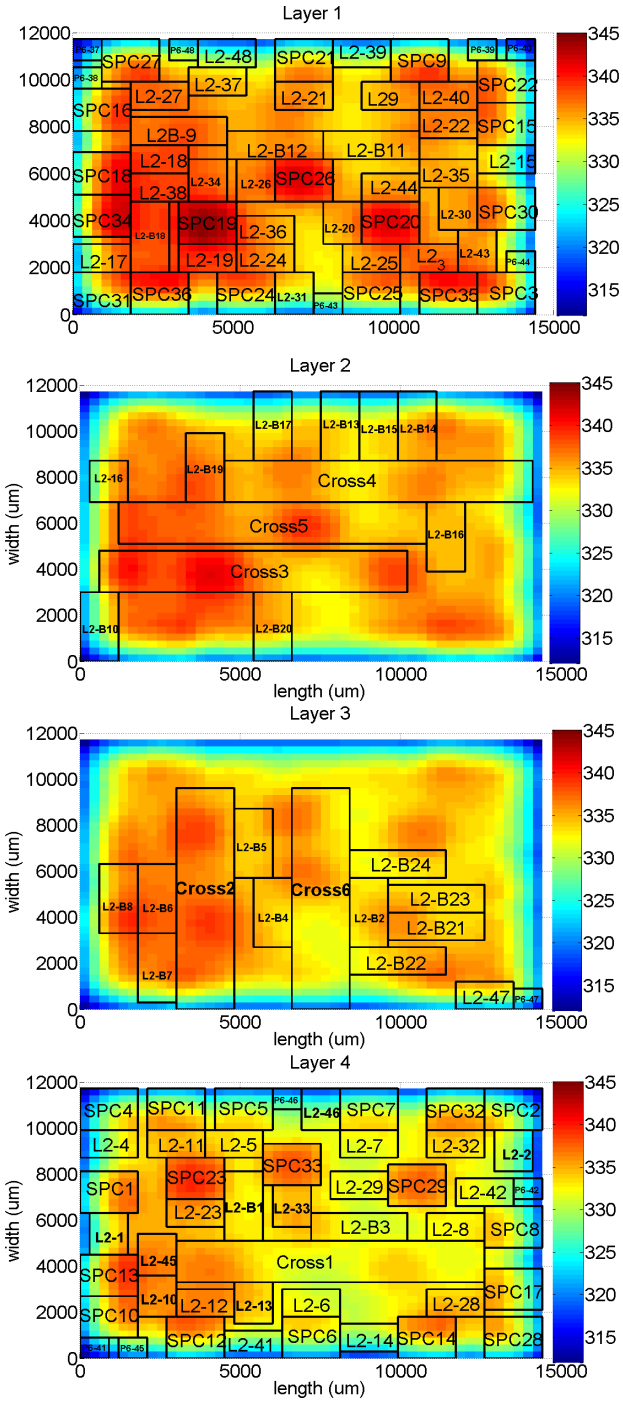

Fig. 7.Mapas térmicos de las 4 capas de una configuración optimizada por el floorplanner de la plataforma A48

propone configuraciones optimizadas. La configuración original NIAGARA48 alcanza una temperatura máxima de 411.82K mientras que para la plataforma A48 optimizada se obtiene un valor de 345.30K. Del mismo modo, se reduce la temperatura media en 12,54K. También se puede ver como el máximo gradiente térmico pasa de 109.75K a 31.81K para la configuración optimizada. Por lo tanto, no sólo se reduce la temperatura del chip, sino que ésta se distribuye de forma más homogénea. En cambio, el cableado de la plataforma optimizada es superior al de la original en un 2.11 %, lo que se traduce en una ligera penalización en el rendimiento.

	Cableado	T_{MAX}	T_{MEAN}	$Grad_{MAX}$
NIAG48	6733	411.82 K	344.29 K	109.75
A48	6875	345.30 K	331.75 K	31.81

TABLA IV
RESPUESTA TÉRMICA DE LAS PLATAFORMAS NIAGARA48 Y A48

B.2 Escenario B: Plataforma de 128 cores

En esta sección presentamos una disposición de los componentes optimizada producida por nuestro floorplanner para la plataforma heterogénea de 128 procesadores A128 (ver Figura 8). Igual que en el escenario anterior, podemos ver que el algoritmo tiende a ubicar los procesadores SPARC en las capas exteriores y en los bordes del chip. Por el contrario, las memorias y los crossbars se colocan en las capas intermedias. De esta forma, se minimiza tanto la temperatura del chip como el cableado. Sin embargo, en esta configuración aparecen puntos calientes. La Tabla V muestra la respuesta térmica de una configuración optimizada de la plataforma de 128 cores. El punto caliente visible en la primera capa corresponde a la temperatura máxima del chip, que alcanza los 396.84K. La temperatura media es de 362.50K mientras que el máximo gradiente térmico es de 75.80K. Dadas las altas temperaturas obtenidas, se requiere un trabajo futuro que incluya simulaciones con técnicas de enfriamiento para estudiar la factibilidad de estas arquitecturas.

	Cabl.	T_{MAX}	T_{MEAN}	G_{MAX}
A128	31587	396.84 K	362.50 K	75.80

TABLA V
Repuesta térmica de la configuración de 128 cores A128

V. CONCLUSIONES

En este trabajo se ha propuesto una herramienta eficiente para realizar el floorplanning térmico de arquitecturas heterogéneas manycore. Se han optimizado plataformas representativas del actual y futuro estado del arte de la integración manycore en tres dimensiones. La paralelización del Algoritmo Evolutivo Multi-Objetivo basada en un modelo Maestro-Esclavo nos ha permitido obtener configuraciones optimizadas de plataformas compuestas por 48 y 128 cores. La implementación multi-hilo del modelo Maestro-Esclavo permite aprovechar los tiempos de idle de los distintos threads, obteniendo un speed-up de 3.79 respecto de la versión secuencial. La optimización de arquitecturas heterogéneas optimizadas térmicamente lleva a una reducción de $66{,}52K$ en la temperatura máxima del chip con un impacto mínimo en el rendimiento.

REFERENCIAS

[1] Intel, ," http://techresearch.intel.com/index.aspx.
[2] S. Borkar, "Design challenges of technology scaling," *Micro, IEEE*, vol. 19, no. 4, pp. 23 –29, jul-aug 1999.
[3] J. Srinivasan, S.V. Adve, P. Bose, and J.A. Rivers, "The impact of technology scaling on lifetime reliability," in *Dependable Systems and Networks, 2004 International Conference on*, june-1 july 2004, pp. 177 – 186.
[4] Karthik Sankaranarayanan, Sivakumar Velusamy, Mircea Stan, Charles L, and Kevin Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *JILP*, vol. 7, no. 1, pp. 8–16, 2005.
[5] Gabriel H. Loh and Yuan Xie, "3d stacked microprocessor: Are we there yet?," *IEEE Micro*, vol. 30, pp. 60–64, May 2010.
[6] M. Motoyoshi, "Through-silicon via (tsv)," *Proceedings of the IEEE*, vol. 97, no. 1, pp. 43 –48, jan. 2009.
[7] W.R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon, "Demystifying 3d ics: the pros and cons of going vertical," *Design Test of Computers, IEEE*, vol. 22, no. 6, pp. 498 – 510, nov.-dec. 2005.
[8] Johan Berntsson and Maolin Tang, "A slicing structure representation for the multi-layer floorplan layout problem," in *EvoWorkshops*, 2004, pp. 188–197.
[9] Maolin Tang and Xin Yao, "A memetic algorithm for vlsi floorplanning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 62 –69, feb. 2007.
[10] J. Cong, Jie Wei, and Yan Zhang, "A thermal-driven floorplanning algorithm for 3d ics," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, Washington, DC, USA, 2004, ICCAD '04, pp. 306–313, IEEE Computer Society.
[11] S.N. Adya and I.L. Markov, "Fixed-outline floorplanning: enabling hierarchical design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, no. 6, pp. 1120 –1135, dec. 2003.
[12] W.-L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M.J. Irwin, "Thermal-aware floorplanning using genetic algorithms," in *Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on*, march 2005, pp. 634 – 639.
[13] Israel Han, Yongkuiand Koren, "Simulated annealing based temperature aware floorplanning," *J. Low Power Electronics*, vol. 3, no. 2, pp. 141–155, 2007.
[14] M. Healy et al., "Multiobjective microarchitectural floorplanning for 2D and 3D ICs," *CADICS, IEEE Transactions on*, vol. 26, no. 1, pp. 38–52, 2007.
[15] I. Arnaldo, J L. Risco-Martin, J L. Ayala, and J I. Hidalgo, "Power Profiling-Guided Floorplanner for Thermal Optimization in 3D Multiprocessor Architectures," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, vol. 6951 of *Lecture Notes in Computer Science*, pp. 11–21. Springer Berlin / Heidelberg, 2011.
[16] K. Deb et al., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
[17] D. Cuesta, J.L. Risco-Martin, J.L. Ayala, and J I. Hidalgo, "A combination of evolutionary algorithm and mathematical programming for the 3d thermal-aware floorplanning problem," in *13th annual conference on Genetic and evolutionary computation (GECCO 2011)*. ACM Press, 07/2011 2011, pp. 1731–1738, ACM Press.
[18] G. Paci et al., "Exploring temperature-aware design in low-power mpsocs," *International journal of embedded systems*, vol. 3, no. 1, pp. 43–51, 2007.
[19] OpenSPARC, ," http://www.opensparc.net/pubs, 2007.
[20] IBM, ," http://www.ibm.com/systems/support/tools/.
[21] HPlabs, ," www.hpl.hp.com/research/cacti/.

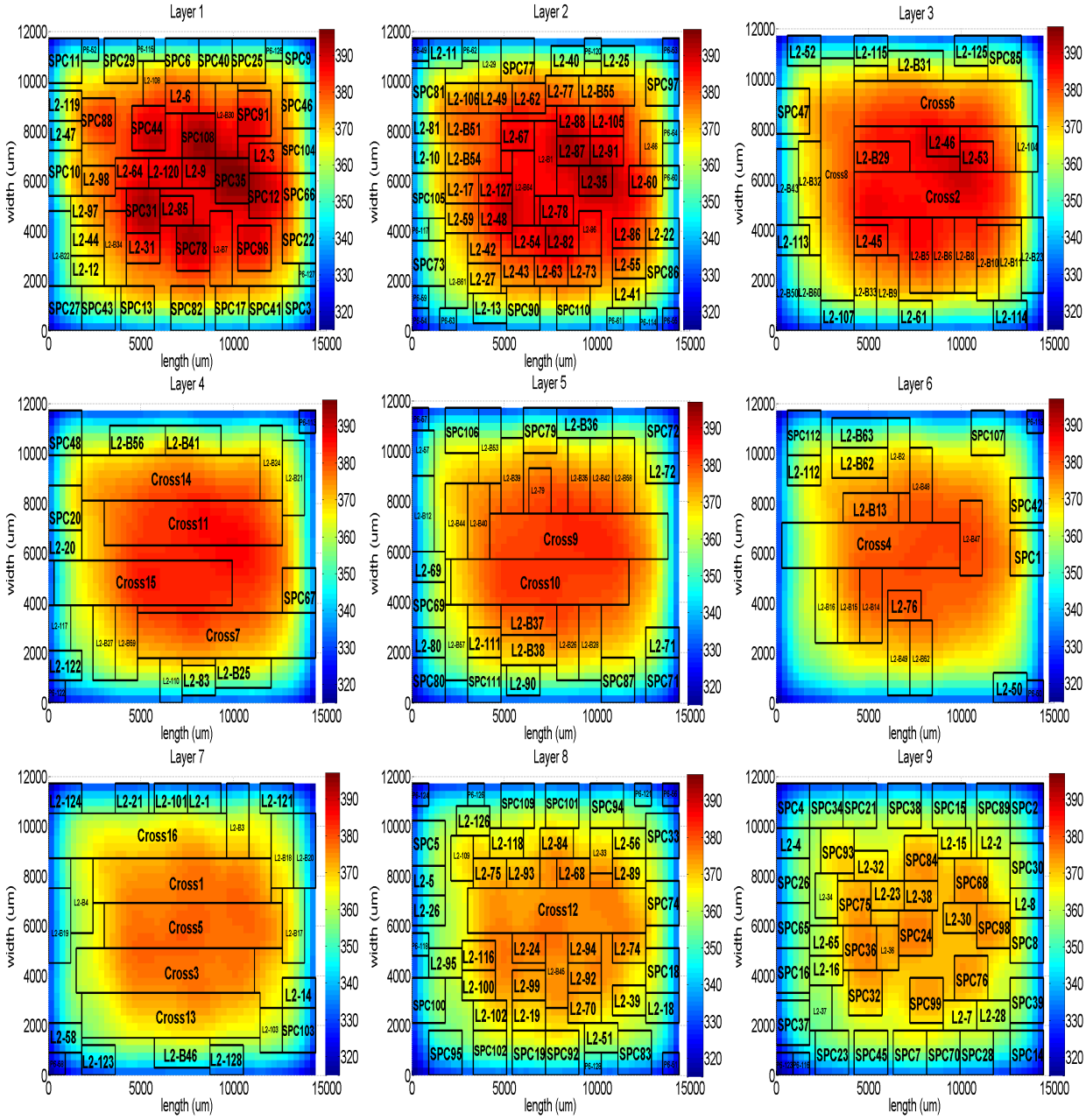

Fig. 8. Mapas térmicos de las 9 capas de una configuración optimizada de la plataforma de 128 cores A128