Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

Tipo de artículo: Artículo original
Temática: Inteligencia artificial
Recibido: 14/11/2014 | Aceptado: 19/01/2015

Partición hardware software de un codificador JPEG utilizando escalador de colinas estocástico

JPEG encoder hardware software partitioning using stochastic hill climbing optimization technique

Humberto Díaz Pando[1*], **Roberto Sepúlveda Lima**[1], **Alejandro Rosete Suárez**[1], **Sergio Cuenca Asensi**[2]

[1]Facultad de Ingeniería Informática, Instituto Superior Politécnico José Antonio Echevarría, Calle 114 No 11901 e/ Ciclovía y Rotonda, Marianao, La Habana, Cuba.

[2] Universidad de Alicante, España.

*Autor para la correspondencia: hdiazp@ceis.cujae.edu.cu

Resumen

La partición hardware/software es una etapa clave dentro del proceso de co-diseño de los sistemas embebidos. En esta etapa se decide qué componentes serán implementados como co-procesadores de hardware y qué componentes serán implementados en un procesador de propósito general. La decisión es tomada a partir de la exploración del espacio de diseño, evaluando un conjunto de posibles soluciones para establecer cuál de estas es la que mejor balance logra entre todas las métricas de diseño. Para explorar el espacio de soluciones, la mayoría de las propuestas, utilizan algoritmos metaheurísticos; destacándose los Algoritmos Genéticos, Recocido Simulado. Esta decisión, en muchos casos, no es tomada a partir de análisis comparativos que involucren a varios algoritmos sobre un mismo problema. En este trabajo se presenta la aplicación de los algoritmos: Escalador de Colinas Estocástico y Escalador de Colinas Estocástico con Reinicio, para resolver el problema de la partición hardware/software. Para validar el empleo de estos algoritmos se presenta la aplicación de este algoritmo sobre un caso de estudio, en particular la partición hardware/software de un codificador JPEG. En todos los experimentos es posible apreciar que ambos algoritmos alcanzan soluciones comparables con las obtenidas por los algoritmos utilizados con más frecuencia.

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

Palabras clave: codificador JPEG, escalador de colinas estocástico, escalador de colinas estocástico con reinicio, partición hardware/software.

Abstract

Hardware/software partitioning is a key task for embedded system co-design. The goal of this task is to decide which components of an application will be executed in a general purpose processor (software) and which ones on a specific hardware. To support this decision a design space exploration is executed, by the evaluation of several solutions to establish the best trade-off reached. To accomplish this task, metaheuristics algorithms are used by the most proposals; highlighting Genetic Algorithms and Simulated Annealing. Many times this decision is not taken by a comparative study over several algorithms. In this article the application of Stochastic Hill Climbing and Restart Stochastic Hill Climbing for solving the hardware/software partitioning problem is presented. A case study of JPEG encoder is presented. The results show that comparable solutions are reached by those algorithms.

Keywords: *hardware/software partitioning, JPEG encoder, restart stochastic hill climbing, stochastic hill climbing.*

Introducción

La Partición Hardware/Software (PHS) de sistemas embebidos es considerada la etapa más importante en el co-diseño de sistemas embebidos (Wolf, 2003; De Micheli and Gupta, 1997; Vahid and Givargis, 2002). Esta etapa define la forma en que será implementado el sistema, teniendo en cuenta la arquitectura del dispositivo de hardware. El resultado es una especificación de qué componentes del sistema serán implementados en un co-procesador de hardware y qué componentes serán implementados sobre un procesador de propósito general. La partición encontrada está en correspondencia con los requisitos impuestos por el usuario final.

Durante las últimas décadas, varios modelos López-Vallejo and López (2003); Mann (2004); Jigang and Srikanthan (2006b); Amin *et al.* (2007); Jigang *et al.* (2008b, a); Bhattacharya *et al.* (2008) han sido propuestos. Estos modelos, permiten llegar a una solución, que si bien en algunos casos pudiera ser un óptimo local, es aceptada como suficientemente buena. Los algoritmos metaheurísticos son ampliamente utilizados en estos modelos para realizar la exploración del espacio de búsqueda y obtener la mejor solución al problema PHS. En muchos de estos modelos la selección del algoritmo no está fundamentada en un estudio comparativo, predominando un criterio de conveniencia o elitismo. Por otra parte, la búsqueda de la solución suele estar guiada por la optimización de varias métricas.

En artículos previos Díaz-Pando *et al.* (2013a,b), los autores del presente artículo establecen una comparación entre varios algoritmos metaheurísticos, utilizando modelos de optimización con dos de las métricas de diseño más utilizadas:

Áreay Tiempo. Además, en Díaz-Pando *et al.* (2013a), se presenta una versión del modelo basado en soft computing. Los resultados de estos artículos establecen que los algoritmos Escalador de Colinas Estocástico y Escalador de Colinas Estocástico con Reinicio alcanzan buenos resultados con respecto a otros algoritmos utilizados más frecuentemente en la literatura.

El codificador JPEG es un sistema apto para ser implementado en un sistema embebido Wallace (1992); Lin *et al.* (2006), ya que es de los estándares más utilizados para la compresión de imágenes digitales. En la actualidad se han propuesto un conjunto de trabajos Lin *et al.* (2006); Lee *et al.* (2007c); Nath and Datta (2014) que están dirigidos a realizar la partición hardware/software de este sistema.

En este artículo se presenta la aplicación de los algoritmos Escalador de Colinas Estocástico y Escalador de Colinas Estocástico con Reinicio sobre un sistema de codificación JPEG. El modelo PHS utilizado es una variante del modelo utilizado por los autores en artículos previos, a fin de que los resultados pudieran ser comparados con los resultados presentes en el estado del arte.

El artículo está organizado de la siguiente forma, en la siguiente sección se presenta un vistazo acerca de los principales algoritmos y técnicas utilizadas para resolver el problema PHS. El sistema de codificación JPEG es descrito en la sección. La sección 4 presenta una breve descripción de los algoritmos Escalador de Colinas Estocástico y Escalador de Colinas Estocástico con Reinicio junto con la función objetivo utilizada. La sección 5 presenta los experimentos realizados y un análisis de los resultados obtenidos, por último las conclusiones del artículo se presentan en la sección 6.

Materiales y métodos

Trabajos relacionados

De acuerdo con la clasificación presentada por Niemann (1998), las propuestas dirigidas a solucionar el problema de Partición Hardware/Software (PHS) pueden ser agrupadas siguiendo varios criterios. Algunos de estos criterios son: complejidad del problema, arquitectura de destino, función objetivo, estrategia de búsqueda, entre otros. De acuerdo con los objetivos trazados en este artículo, las propuestas serán agrupadas según este último criterio.

La estrategia de búsqueda define los algoritmos que permitirán explorar el espacio de soluciones para buscar las soluciones que cumplan con los parámetros establecidos por el diseñador. Las estrategias pueden ser agrupadas según la técnica de optimización utilizada, de esta forma es posible mencionar: las basadas en algoritmos de programación dinámica Madsen *et al.* (1997); Jigang and Srikanthan (2006a, b); Jigang *et al.* (2008b, b); Jigang and Srikanthan (2006a) y en programación entera Bhattacharya *et al.* (2008); Arató *et al.* (2003); Schwiegershausen *et al.* (1996);

Arató *et al.* (2005). Todas estas soluciones exactas, si bien alcanzan el óptimo global, tienen como inconveniente que el tiempo de ejecución crece de forma exponencial con respecto al aumento del tamaño del problema a resolver Arató *et al.* (2003).

Para contrarrestar este inconveniente surgen las soluciones aproximadas, las cuales en la mayoría de los casos no alcanzan el óptimo global pero ofrecen una solución considerada como buena en un tiempo de ejecución razonable. Las estrategias basadas en algoritmos metaheurísticos son las más utilizadas dentro de este grupo, destacándose a su vez un conjunto de algoritmos como los más utilizados. En este caso se encuentran los Algoritmos Genéticos ya sea en variantes mono-objetivos Wiangtong *et al.* (2002); Arató *et al.* (2003); Arató *et al.* (2005); Amin *et al.* (2007); Purnaprajna *et al.* (2007); Bhattacharya *et al.* (2008) y multi-objetivo (Liu and Li, 2009; Lu *et al.*, 2009; Beux *et al.*, 2010; Jagadeeswari and Bhuvaneswari, 2011; Nath and Datta, 2014), Recocido Simulado Ernst *et al.* (1993); Eles *et al.* (1997); Vahid and Le (1997); Henkel and Ernst (2001); Wiangtong *et al.* (2002); López-Vallejo and López (2003) y Enjambre de Partículas variantes monoobjetivo Amin *et al.* (2007); Tong *et al.* (2008); Bhattacharya *et al.* (2008); Abdelhalim and Habib (2011) y multi-objetivo Jagadeeswari and Bhuvaneswari (2011).

En muchos casos es posible apreciar que los algoritmos son seleccionados de forma arbitraria o siguiendo la preferencia de los autores. No obstante existen un grupo de trabajos López-Vallejo and López (2003); Arató *et al.* (2005); Jigang and Srikanthan (2006b); Jigang *et al.* (2008b, a); Jagadeeswari and Bhuvaneswari (2011), donde los autores plantean comparaciones entre varios algoritmos sobre un mismo modelo PHS; con el objetivo de determinar el algoritmo que mejor se ajusta a dicho modelo.

En el caso del codificador JPEG, en los últimos años es posible apreciar su utilización como aplicación real en casos de estudio que abordan el problema PHS. Este caso de estudio fue introducido en Lee *et al.* (2007c) y seguidamente ha sido utilizado como sistema de referencia para probar diferentes estrategias de búsqueda Lee *et al.* (2007a, b, c); Abdelhalim and Habib (2011).

Sistema de codificación JPEG

Según se hace referencia en Nath and Datta (2014), existen una variedad de formatos para la compresión de imágenes, destacándose por su popularidad el formato JPEG Wallace (1992). En la figura 1 se muestra el grafo de flujo de control-datos y flujo perteneciente a este sistema, el cual está tomado de Lee *et al.* (2007c). Estos autores asumen que el convertidor RGB a YUV es implementado en software, por lo que no estará involucrado en el proceso de partición. Además, los procesos agrupados en un mismo nivel pueden ser ejecutados de forma concurrente ya que no existe dependencia entre ellos.

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

Para poder utilizar el caso de estudio, es necesario disponer de detalles de las estimaciones de las variables utilizadas, la función de coste y los resultados de la partición, todos estos detalles se encuentran en Lee *et al.* (2007c), por lo cual es posible establecer comparaciones entre diferentes estrategias de búsqueda.

Para obtener las estimaciones de los costes en hardware, los componentes fueron implementados en una tarjeta ML310 usando la plataforma de diseño Xilinx ISE 7.1i. Mientras que las estimaciones de los costes en software se obtuvieron con Xilinx Embedded Design Kit (EDK 7.1i). La tarjeta ML310 contiene una FPGA Virtex2-Pro XC2vP30FF896, la cual a su vez contiene 13696 slices y 2448 Kbytes de memoria y dos procesadores IBM Power PC embebidos. Las estimaciones se muestran en la tabla 1.

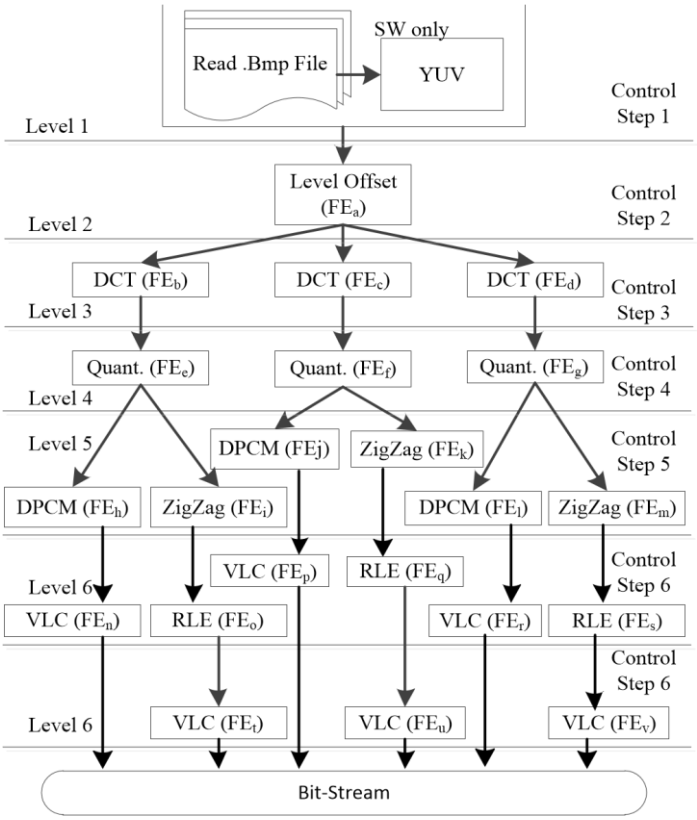

Figura 1. Grafo de flujo de control-datos para el sistema de codificación. Tomado de (Lee *et al.*, 2007c)

En la tabla, la primera columna representa el nombre y el identificador de cada componente que aparece en la figura 1. La segunda y tercera columna representa el tiempo de ejecución de cada uno de los componentes en las dos implementaciones. Las columnas cuatro y cinco representan el porcentaje del área total que ocupa cada uno de los componentes. Estos porcentajes están referidos a los recursos disponibles en la tarjeta utilizada. Las dos últimas columnas muestran el consumo de potencia de cada componente según el tipo de implementación.

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

Escalador de colinas estocástico

El Escalador de Colinas Estocástico (ECE), es una técnica de optimización basada en un punto propuesta por Jones (1995). El ECE está inspirado en la forma en que un alpinista escala una montaña, bajo esta situación el alpinista se puede encontrar en un punto en que solo conozca que los puntos más próximos ascienden o descienden. Los algoritmos basados en este principio son muy sencillos, aunque no gozan de gran popularidad en el contexto de investigación. No obstante se tiene información sobre los buenos resultados dados en otros escenarios de aplicación, en Rosete-Suárez (2000) se realiza una revisión de estas aplicaciones y además se muestran los resultados del empleo de este algoritmo en el problema del trazado de grafos.

Tabla 1. Estimaciones obtenidas para el caso de estudio. Tomado de (Lee *et al.*, 2007c).

Función	Tiempo de ejecución		Área ($x10^{-3}$)		Potencia consumida	
	HW (*µseg*)	SW (*µseg*)	HW	SW	HW (mw)	SW (mw)
Level Offset	0,155264	9,38	7,31	0,58	4	0,096
DCT	1,844822	20000	378	2,88	274	45
DCT	1,844822	20000	378	2,88	274	45
DCT	1,844822	20000	378	2,88	274	45
Quant.	3,51232	34,7	11	1,93	3	0 , 26
Quant.	3,51232	33,44	9,64	1,93	3	0 , 27
Quant.	3,51232	33,44	9,64	1,93	3	0 , 27
DPCM	0,005334	0,94	2,191	0,677	15	0,957
ZigZag	0,399104	13,12	35	0,911	61	0,069
DPCM	0,005334	0,94	2,191	0,677	15	0,957
ZigZag	0,399104	13,12	35	0,911	61	0,069
DPCM	0,005334	0,94	2,191	0,677	15	0,957
ZigZag	0,399104	13,12	35	0,911	61	0,069
VLC	2,054748	2,8	7,74	14,4	5	0,321
RLE	1,148538	43,12	2,56	6,034	3	0,021

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

VLC	2,197632	2,8	8,62	14,4	5	0,321
RLE	1,148538	43,12	2,56	6,034	3	0,021
VLC	2,197632	2,8	8,62	14,4	5	0,321
RLE	1,148538	43,12	2,56	6,034	3	0,021
VLC	2,668288	51,26	19,21	16,7	6	0,018
VLC	2,668288	50	1,91	16,7	6	0,018
VLC	2,668288	50	1,91	16,7	6	0,018

En la variante clásica de este algoritmo cuando se alcanza una solución, para pasar a la siguiente, se explora la vecindad de manera exhaustiva. En el caso que el valor de la solución candidata sea menor que el valor de la solución actual, el algoritmo queda estancado en un óptimo local; siendo este el principal inconveniente del algoritmo. Otro inconveniente lo constituye que en la mayoría de los escenarios, la vecindad de una solución puede ser un espacio lo suficientemente grande como para que el tiempo de ejecución de cada iteración sea muy grande.

A partir de esta última dificultad surge el Escalador de Colinas Estocástico con Mejor Ascenso (ECE-MA), el cual no explora la vecindad completa, sino que hace una exploración aleatoria de una parte. En el caso que no se mejore la solución actual, no se asume que sea un óptimo local pues no se exploró la vecindad completa. En este caso el algoritmo cumplirá con un número de iteraciones establecidas. Este algoritmo será utilizado como parte de los experimentos.

Teniendo en cuenta esta dificultad, es propuesta una variante denominada Escalador de Colinas Estocástico con Reinicio (ECE-R) Rosete-Suárez (2000). En esta variante el algoritmo se reinicia la búsqueda desde una nueva solución generada aleatoriamente una vez que se haya detectado que el algoritmo está estancado en un óptimo local.

Función objetivo

La función objetivo utilizada es una versión de la empleada en artículos previos Díaz-Pando *et al.* (2013a,b). En aras de hacer la comparación lo más fiel posible se hace necesario modificar la instancia del modelo, propuesta en Díaz Pando (2014). La modificación consta de tres elementos principales: Considerar dos procesadores de propósito general en la arquitectura, considerar la ejecución simultánea de dos funciones que se ejecuten en procesadores independientes, modificar el cálculo del área de hardware ocupada por la solución.

Sea ah_i el área que ocupa la implementación de las funciones como coprocesadores de hardware, el área total consumida por el diseño se calcula como:

$$A = \sum_{i=1}^{n} (x_i * ah_i) \qquad (1)$$

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

Sea L la cantidad de niveles en el sistema (en la figura 1, $L = 6$) y l_i es la cantidad de tareas en el i-´esimo nivel (en la figura 1, $l_6 = 4$). El tiempo de ejecución para la j-´esima tarea en el i-´esimo nivel en hardware y software se define como th_{ij} y ts_{ij} respectivamente. x_{ij} es una variable binaria que indica el tipo de implementación de la función i-´esima en el nivel j-´esimo, un valor de 1 indica implementación en hardware y 0 en software. De acuerdo con esto, el tiempo total del sistema se calcula como:

$$T = \sum_{i=1}^{n} max\{ts_{ij}(1 - x_{ij}), th_{ij}x_{ij}; j = 1, \ldots, l_i\} \quad (2)$$

Sea me_i la memoria consumida por cada una de las funciones implementadas en el procesador, la memoria total consumida por el sistema se calcula como:

$$Me = \sum_{i=1}^{n}(1 - x_i)me_i \quad (3)$$

Sea ps_i la potencia consumida por la función i si es implementada sobre un procesador empotrado y ph_i la potencia cuando se implementa en un coprocesador de hardware. La potencia consumida por el diseño completo P, se define como:

$$P = \sum_{i=1}^{n}[(1 - x_i)ps_i + x_i ph_i] \quad (4)$$

Sea $R = R_P, R_{Me}$ el conjunto que define las restricciones impuestas al diseño. R_P representa el valor máximo de potencia que puede consumir la solución y R_{Me} la cantidad de memoria máxima disponible de la plataforma para la ejecución de las funciones sobre el procesador empotrado. Estas restricciones serán tratadas como penalizaciones, siendo P_P y P_M las penalizaciones aplicadas por incumplir el consumo de potencia y la memoria respectivamente, ambas penalizaciones se definen como:

$$P_P = \begin{cases} \frac{\mu(P) - R_P}{R_P} & \mu(P) > R_P \\ 0 & \mu(P) \leq R_P \end{cases} \quad (5)$$

$$P_M = \begin{cases} \frac{\mu(Me) - R_{Me}}{R_{Me}} & \mu(Me) > R_{Me} \\ 0 & \mu(Me) \leq R_{Me} \end{cases} \quad (6)$$

Sea Ir el Índice de rendimiento propuesto por [40], una métrica compuesta por las métricas Área y Tiempo mediante la multiplicación ($Ir = A * T$). En este trabajo las métricas serán consideradas como conjuntos difusos, es por esto que Ir se calculará a partir de la función de pertenencia Gamma y utilizando la función $x * y$ de la operación T-Norma; es decir Ir se calcula como: $Ir = \mu(a) * \mu(t)$.

Para evaluar las soluciones se define la función objetivo como:

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

$$FO = \min Ir + P_P + P_M \quad (7)$$

Experimentos

El objetivo principal de los experimentos es demostrar la validez de la hipótesis de que el comportamiento de los algoritmos EC y ECR empleando un modelo HSP basado en lógica difusa, en un escenario real, es similar o mejor al alcanzado por otros algoritmos y modelos más utilizados.

Para lograr este objetivo se propone la realización de un conjunto de experimentos, los cuales tienen en común la aplicación de la instancia difusa del modelo al juego de datos referido anteriormente. En cada uno de los experimentos se modificarán los parámetros del modelo (los umbrales de las funciones de pertenencia) o los parámetros del algoritmo ECE-R, para evaluar el impacto de estos en la solución obtenida.

Los algoritmos fueron ejecutados 30 veces y en cada ejecución fueron realizadas 50000 evaluaciones de la función objetivo. El resultado de estas ejecuciones es la mejor solución por cada una de las ejecuciones en cuanto a calidad de la solución. Para evaluar los resultados y el comportamiento de los algoritmos y del modelo, se seleccionaron las mejores soluciones en cuanto a calidad de la solución. Estas se compararon con las mejores soluciones obtenidas del estado del arte, utilizando las siguientes métricas: porcentaje del Área de hardware ocupada, Tiempo de ejecución, Potencia y Memoria consumida.

Resultados y discusión

Una vez ejecutados los experimentos, en esta sección se presentarán los resultados y el análisis de estos. En la tabla 2, se presentan los resultados obtenidos por los algoritmos: ECE-MA y ECE-R; junto con los resultados de las propuestas revisadas en el estado del arte: FBP Lee *et al.* (2007c), GHO Lee *et al.* (2007b), GA Lin *et al.* (2006), HOP Lee *et al.* (2007a), PSO-Norm Abdelhalim and Habib (2011).

Con vistas a realizar una comparación lo más justa posible y ganar en claridad a la hora de presentar los datos, fueron retiradas, de esta tabla, variantes de solución propuestas por Abdelhalim and Habib (2011), las cuales están dirigidas a minimizar solo un objetivo (Área o Tiempo o Memoria o Potencia). Además se retiró la variante donde no se asume la restricción de la cantidad de componentes de software por nivel.

En la tabla (2) no se encuentran todos los resultados obtenidos con ambos algoritmos, sino que se muestran todas aquellas soluciones que son de interés en la comparación, es decir aquellas que mejoran al menos una de las métricas con respecto a las propuestas anteriores. A fin de ilustrar el comportamiento de estos dos algoritmos con respecto a los reportados en la bibliografía, se realizará un análisis de las soluciones teniendo en cuenta cada una de las métricas.

Con respecto a la métrica tiempo, es posible apreciar que ninguna de las soluciones obtenidas por el EC y el

Revista Cubana de Ciencias Informáticas
Vol. 9, No.1, Enero-Marzo, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 94-108

Tabla 2. Comparación de los resultados de los algoritmos ECE-MA y ECE-R con el estado del arte en el caso de estudio.

Propuesta	Solución	Tiempo de ejecución (μs)	Memoria (KB)	Área de hardware (%)	Potencia (mw)
FBP	1/001/111/101111/111101/111	20022,26	51,58	53,9	581,39
GHO	1/010/111/111110/111111/111	20021,66	16,507	54,7	586,069
GA	0/010/010/101110/110111/010	20111,26	146,509	47,1	499,121
HOP	0/100/010/101110/110111/010	20066,64	129,68	56,6	599,67
PSO-Norm	0/010/111/101110/111111/111	20030,9	19,98	50,6	521,234
ECR-3	0/100/100/101011/011011/010	20150,32	161,215	45,4843	496,152
ECR-11	0/100/111/110011/110111/111	**20022,1**	54,658	52,9732	563,443
ECR-5	1/010/110/111110/101111/111	20092,50	35,826	53,5653	580,36
EC-3	0/001/010/101110/011101/001	20101,88	181,695	**44,7423**	**494,442**
EC-2	1/001/110/101011/011111/111	20052,18355	58,536	49,5473	517,729
EC-1	1/010/001/101011/111111/111	20052,84118	28,010	49,2213	519,668

ECR logra alcanzar el menor tiempo. No obstante, el incremento es despreciable, mientras que la mejora en el resto de métricas es bastante significativa. Por ejemplo, en la variante ECR-3, que arroja el peor tiempo, solo excede en un 0,6426% a la variante GHO, la cual es la que mejor tiempo alcanza. No obstante, esa misma variante (ECR-3) mejora en un 20,26% y en un 18,12% el Área de hardware y la Potencia consumida por GHO. Por otro lado la variante ECR-11, que presenta el mejor de los tiempos, solo excede a GHO en 0,002197% y al mismo tiempo mejora el Área y la Potencia en 3% y 4% respectivamente.

Con respecto al consumo de memoria, la soluciones generadas por estos dos algoritmos no logran buenos resultados, llegando a ser la peor solución (EC-3) 11 veces superior a la mejor solución del estado del arte (GHO). No obstante, es posible apreciar también que estas soluciones que presentan un consumo de memoria elevado, tienden a mejorar en varios órdenes el área de hardware ocupada, el consumo de potencia y el tiempo de ejecución. Por ejemplo, la solución alcanzada por EC-3 alcanza el valor mínimo de Área y Potencia entre todas las soluciones.

En cualquier caso, es posible decir como caso general que en los sistemas embebidos modernos la métrica menos restrictiva suele ser el consumo de Memoria, debido al bajo precio de las nuevas tecnologías de memoria (DDR, DDR2, etc.).

De acuerdo a los resultados mostrados, es posible constatar que las soluciones aportadas por EC y ECR son las que mejores valores de área de hardware ocupada y consumo de potencia. Muchas de estas soluciones alcanzan el menor valor de estas métricas entre todas las propuestas, por ejemplo la solución EC-3 logra un 5,27% y 0,95% de mejora con respecto a la mejor solución del estado del arte.

A la luz de los resultados y de acuerdo con el propósito de estos experimentos, es posible concluir que no existe un algoritmo que sea mejor que el resto, en este caso de estudio. Para el caso de la métrica Área y Potencia los algoritmos EC y ECR alcanzan los mejores valores, mientras que para la métrica Tiempo alcanzan valores similares al mejor valor; no obstante las soluciones de estos dos algoritmos alcanzan un consumo de Memoria superior en comparación con el estado del arte. De acuerdo con este planteamiento se considera errado el ofrecer como resultado del proceso HSP una única solución, cuando se trata de optimizar varias métricas de forma simultánea.

Con relación a las soluciones de los algoritmos ECE-MA y ECE-R, es posible plantear que están dominadas por la métrica Área, es decir alcanzan soluciones con muy bajo porcentaje de utilización de recursos de hardware.

De acuerdo con los resultados obtenidos en los experimentos, variando los parámetros difusos del modelo, es posible plantear que la lógica difusa además de brindar flexibilidad al modelo permite penalizar de cierta forma las métricas involucradas en el modelo. Esta forma de penalizar las soluciones es más intuitiva para un diseñador, en lugar de establecer pesos por cada una de las métricas. Además, permiten dirigir la búsqueda hacia determinados espacios según las necesidades del diseñador en cuanto al tipo de solución que se desee lograr. Esto se debe a que moviendo los límites de la función de pertenencia es posible lograr valores en las métricas más altos o más bajos.

Conclusiones

En este trabajo se presentó la aplicación de los algoritmos ECE-MA y ECE-R en el problema de partición hardware/software. Para tales propósitos, fue utilizado un sistema codificador JPEG del cual se dispone de todos los datos vinculados con las estimaciones de cada una de las variables. En el modelo utilizado se empleó el índice de rendimiento como métrica para guiar la búsqueda. Fueron empleados también elementos de la lógica difusa, en este caso para modelar cada una de las métricas como variables lingüísticas.

Los resultados obtenidos muestran que los algoritmos ECE-R y ECE-MA, lograron soluciones comparables en relación con otros algoritmos más populares. El algoritmo EC alcanzó la mejor solución en cuanto a Área de Hw y Potencia consumida logrando una mejora del 5,27% y 0,95% respectivamente con relación a la mejor solución del estado del arte. Por su parte, le algoritmo ECE-R alcanzó una solución con un Tiempo de ejecución que solo excede en 0,6426%

a la mejor solución del estado del arte (GHO), presentando una mejora significativa en un 20,26% y en un 18,12% el Área de hardware y la Potencia consumida por GHO, aunque la solución del ECR consume más memoria que GHO.
En problemas donde se optimicen varios objetivos de forma simultánea, es erróneo plantear una única solución como la correcta, ya que existen soluciones buenas en un objetivo y malas en otros. El modelo utilizado con un enfoque difuso es mejor con respecto a las propuestas del estado del arte que utilizan pesos en la función objetivo, ya que permite tratar la importancia de las métricas de forma más intuitiva moviendo los límites en la función de pertenencia.

Referencias

- M.B. Abdelhalim and S.E.-D. Habib. An integrated high-level hardware/software partitioning methodology. Design Automation for Embedded Systems, 15(1):19–50, 2011. ISSN 0929-5585. doi: 10.1007 / s10617-010-9068-9. URL http://dx.doi.org/10.1007/s10617-010-9068-9.
- F.-F. Amin, K. Mehdi, F. Sied Mehdi, and S. Saeed. Hw/sw partitioning using discrete particle swarm. En: 17th ACM Great Lakes symposium on VLSI. Proceedings of the 17th ACM Great Lakes Symposium on VLSI. ACM, New York, NY, USA: 2007, 359–364.
- P. Arató, S. Zuhász, Z. A. Mann, A. Orbán, and D. Papp. Hardware/software partitioning in embedded system design. En: IEEE International Symposium on Intelligent Signal Processing. IEEE: 2003, 197–202.
- Sébastien Le Beux, Guy Bois, Gabriela Nicolescu, Youcef Bouchebaba, Michel Langevin, and Pierre Paulin. Combining mapping and partitioning exploration for noc-based embedded systems. Journal of Systems Architecture, 56(7):223 – 232, 2010. ISSN 1383-7621. doi: http://dx.doi.org/10.1016/j.sysarc.2010.03.005.
- Humberto Díaz Pando. Modelo y estrategias de particio´n de componentes hardware/software en el co-diseño de sistemas embebidos. PhD thesis, Universidad de Alicante, España, 2014.
- Humberto Díaz-Pando, Sergio Cuenca Asensi, Roberto Sepúlveda Lima, Jenny Fajardo Calderín, and Alejandro Rosete Suárez. An application of fuzzy logic for hardware/software partitioning on embedded systems. Computación y Sistemas, 17(1):30–40, marzo 2013 a.
- Giovanni De Micheli and Rajesh K. Gupta. Hardware/software co-design. Proceedings of the IEEE, 85(3): 349–365, 1997.
- P. Eles, Z. Peng, K. Kuchcinski, and A. Doboli. System level hardware/software partitioning based on simulated annealing and tabu search. Des Autom Embed Syst, 2(1):5–32, 1997.

- Rolf Ernst, Jorg Henkel, and Thomas Benner. Hardware-software cosynthesis for microcontrollers. IEEE Des. Test, 10(4):64–75, October 1993. ISSN 0740-7475. doi: 10.1109/54.245964. URL http://dx.doi.org/10.1109/54.245964.
- Wu Jigang, Thambipillai Srikanthan, and Guang-Wei Zou. New model and algorithm for hardware/software partitioning. Journal of Computer Science and Technology, 23(4):644–651, July 2008b. ISSN 1000-9000. doi: 10.1007/s11390-008-9160-9.
- Terry Jones. Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico, 1995.
- Trong-Yen Lee, Yang-Hsin Fan, Yu-Min Cheng, Chia-Chun Tsai, and Rong-Shue Hsiao. An efficiently hardware-software partitioning for embedded multiprocessor fpga system. En: International multiconference of engineers and computer scientists. Hong Kong: 2007b, 346–351.
- Yang Liu and Qing Cheng Li. Hardware software partitioning using immune algorithm based on pareto. En: International Conference on Artificial Intelligence and Computational Intelligence, volume 2. IEEE: 2009 , 176–180.
- Marisa López-Vallejo and Juan Carlos López. On the hardware-software partitioning problem: System modeling and partitioning techniques. ACM Trans. Des. Autom. Electron. Syst. (TODAES), 8(3):269–297, July 2003. ISSN 1084-4309. doi: 10.1145/785411.785412.
- Zoltán Adám Mann. Partitioning algorithms for hardware/software co-design. PhD thesis, Budapest Univerity of Technology and Economics, Department of Control Engineering and Information Technology, 2004.
- Pankaj Kumar Nath and Dilip Datta. Multi-objective hardware-software partitioning of embedded systems: A case study of {JPEG} encoder. Applied Soft Computing, 15(0):30 – 41, 2014. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2013.10.032. URL http://www.sciencedirect.com/science/article/pii/S1568494613003669.
- Ralf Niemann. Hardware/Software co-design for data flow dominated embedded systems. Kluwer Academic Publishers, Boston, 1998.
- M. Schwiegershausen, H. Kropp, and P. Pirsch. A system level hw/sw partitioning and optimization tool. En: Proceedings of the conference on European design automation. IEEE Computer Society Press: 1996, 120–125.
- Qiaoling Tong, Xuecheng Zou, Qiao Zhang, Fei Gao, and Hengqing Tong. The hardware/software partitioning in embedded system by improved particle swarm optimization algorithm. En: Fifth IEEE International Symposium on Embedded Computing. IEEE Computer Society Press: 2008, 43–46.

- Frank Vahid and Tony Givargis. Embedded System Design: A Unified Hardware/Software Introduction. J. Wiley and Sons, 2002.
- Wayne Wolf. A decade of hardware/software codesign. Computer, 36(4):38–43, April 2003. ISSN 0018-9162.