# Chapter 2
# Hardware/Software Partitioning for Embedded Systems

**M.C. Bhuvaneswari and M. Jagadeeswari**

**Abstract** Current methods for designing embedded systems require specifying and designing hardware and software separately. Hardware/software partitioning is concerned with deciding which function is to be implemented in Hardware (HW) and Software (SW). This type of partitioning process is decided a priori to the design process and is adhered to as much as possible because any changes in this partition may necessitate extensive redesign. As partitioning is an NP hard problem, application of the exact methods tends to be quite slow for bigger dimensions of the problem. Heuristic/evolutionary methods have been proposed for partitioning problems. This chapter deals with multi-objective optimization of minimizing two objectives area and the execution time of the partition. To validate the efficiency of the algorithms, performance metrics are calculated. Experimental results for the HW/SW partition for mediabench and DSP benchmarks are tabulated and analyzed.

**Keywords** Hardware/software partitioning • Embedded systems • Pareto-optimal solutions • Performance metrics • Multi-objective optimization • Multi-objective evolutionary algorithms

## 2.1 Introduction

The complexity of very large scale system integration (VLSI) and embedded systems is increasing and the analysis and design of such complex systems is a nondeterministic polynomial time (NP) hard problem. Hardware/software co

M.C. Bhuvaneswari
Electrical and Electronics Engineering, PSG College of Technology, Coimbatore, India
e-mail: mcb@eee.psgtech.ac.in

M. Jagadeeswari (✉)
Electronics and Communication Engineering, Sri Ramakrishna Engineering College, Coimbatore, India
e-mail: jagadeeswari.m@srec.ac.in

M.C. Bhuvaneswari (ed.), *Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems*, DOI 10.1007/978-81-322-1958-3_2,

design (HSCD) is the discipline of automating the design of complex embedded systems that function using both hardware and software. The central task of HSCD is hardware/software partitioning which is concerned with deciding which function is to be implemented in Hardware (HW) and Software (SW). This implementation aims at finding an optimal trade-off between conflicting parameters such as area and execution time. Deterministic methods for partitioning problems provide exact solutions, but when applied to complex problems they become computationally intractable. As partitioning is an NP hard problem, application of the exact methods tends to be quite slow for bigger dimensions of the problem. Heuristic/evolutionary methods have been proposed for partitioning problems (Schaumont 2013; Wu Jigang et al. 2010; Greg Stit 2008; He Jifeng et al. 2005; Zou Yi et al. 2004; Theerayod Wiangtong 2004; Arato et al. 2003; Dick and Jha 1997; Ernst et al. 1992). This chapter focuses on the application of multi-objective evolutionary algorithms for HW/SW partitioning of embedded systems.

This chapter mainly focuses on the algorithmic aspects of the central phase of HSCD, that is, functional partitioning. Functional partitioning divides a system functional specification into multiple subspecifications. Each subspecification represents the functionality of a system component such as HW or SW processor and is synthesized down to gates or compiled down to machine code. Functional partitioning results in both HW and SW implementations with less number of HW functional blocks. The embedded system to be partitioned is modeled as Directed Acyclic Graph (DAG). A DAG is a graph G (V, E), where $V$ is the set of tasks (functional nodes) and the edge $E$ represents the data dependency between the two tasks with $V \in \{v_1, v_2 \ldots v_i\}$ and $E \in \{e_1, e_2 \ldots e_i\}$. Each task is associated with five integer values: (1) HW-execution time ($t_H$), that is, the time required to execute the task on the HW module; (2) the HW implementation of the task requires area $C_H$ on the HW module; (3) SW-execution time ($t_S$), that is, the time required to execute the task on the processor; (4) the SW implementation of the task requires memory $C_S$ on the software module; and (5) the communication costs ($C_C$), which refers to the delay in transfer of data between HW and SW tasks. This cost is ignored if the transfer of data is between two nodes on the same side (i.e., two HW nodes or two SW nodes). Functional partitioning is done using evolutionary algorithms that decide the tasks to be implemented in HW or in SW. The evolutionary algorithms are iterated until the required number of generations is reached to obtain optimal HW/SW partitions.

In this chapter, three multi-objective optimization algorithms are illustrated: Weighted Sum Genetic Algorithm (WSGA) (Deb 2002), Nondominated Sorting Genetic Algorithm (NSGA-II) (Deb 2002), and Multi-objective Particle Swarm Optimization using Crowding Distance strategy (MOPSO-CD) (Tsou et al. 2006) applied for solving HW/SW partitioning problem. To validate the efficiency of the algorithms, performance metrics are calculated. Experimental results for the HW/SW partition for mediabench and DSP benchmarks are tabulated. The results obtained demonstrate that the NSGA-II is effective in solving HW/SW partitioning problem and provides good results for most of the benchmark circuits.

## 2.2 Prior Work on HW/SW Partitioning

Majority of the proposed partitioning algorithms are heuristic, due to the fact that the partitioning problem is NP-hard. The NP-hardness of the HW/SW partitioning was claimed by several researchers (Eles and Peng 1996; Kalavade and Lee 1994; Binh et al. 1996).

The proposed exact algorithms include branch-and-bound (D'Ambrosio and Hu 1994; Binh et al. 1996), dynamic programming (Madsen et al. 1997; Shrivastava and Kumar 2000), and integer linear programming (Arato et al. 2003). Two partitioning algorithms for HW/SW partitioning were presented by Arato et al. (2003): one based on Integer Linear Programming (ILP) and the other on Genetic Algorithm (GA). It was proved that ILP-based solution works efficiently for smaller graphs with several tens of nodes and yields optimal solutions, whereas GA gives near-optimal solutions on an average and works efficiently with graphs of hundreds of nodes. The performance of GA was found to be uniform, whereas wild oscillations were found in the run time of ILP.

Many researchers have applied general-purpose heuristics to HW/SW partitioning. Particularly, genetic algorithms have been extensively used (Wu Jigang et al. 2010; Greg Stit 2008; He Jifeng et al. 2005; Zou Yi et al. 2004; Theerayod Wiangtong 2004; Arato et al. 2003; Srinivasan et al. 1998; Ernst et al. 1992). Simulated Annealing (SA) was popularized by Kirkpatrick Vecchi (1983) as an alternative to greedy approaches, which are quickly trapped in local minima since they can only make downhill moves. This algorithm was also used by Theerayod Wiangtong (2004), Eles and Peng (1996), Ernst et al. (1992), and Lopez-Vallejo et al. (2000). Other less popular heuristics in this group are tabu search (Theeryod Wiangtong 2004; Eles and Peng 1996) and greedy algorithms (Chatha and Vemuri 2001). Some researchers used custom heuristics to solve HW/SW partitioning (Barros et al. 1993). This includes Global Criticality/Local Phase (GCLP) algorithm (Kalavade and Lee 1994), expert system of Lopez Vallejo et al. (2000), and binary constraint search algorithm by Vahid and Gajski (2001).

Concerning the system model, further distinctions can be made. In particular, many researchers consider scheduling as a part of partitioning (Kalavade and Lee 1994; Niemann and Marwedel 1997; Chatha and vemuri 2001; Lopez-Vallejo et al. 2000), whereas others do not (Eles and Peng 1996; Vahid et al. 1994). The problem of assigning communication events to link between HW and/or SW units was included by Dick and Jha (1997).

In a number of related articles, the target architecture consists of single SW and a single HW unit (Eles and Peng 1996; Gupta and De Micheli 1993; Henkel and Ernst 2001; Lopez-Vallejo and Lopez 2001; Vahid and Gajski 2001). Other researchers do not impose this limitation. Some researchers limit parallelism inside HW or SW (Vahid and Gajski 2001) or between HW and SW (Henkel and Ernst 2001). All of these algorithms aimed at optimization of only one objective. Jagadeeswari and Bhuvaneswari (2009) employed multi-objective optimization techniques for HW/SW partitioning of embedded systems.

## 2.3 Target Architecture

The characteristic of the target system architecture consists of one SW processor (aimed to execute C programs), one HW module (implements the VHDL/Verilog descriptions), and shared bus (communication between HW and SW) as shown in Fig. 2.1.

This target model serves as a platform onto which an HW/SW system is mapped. Both the HW and SW have their local memory and communicate with each other through the shared bus. There is no memory limitation in this model. The SW processor is a uniprocessing system and can execute only one task at a time, while the HW can execute multiple tasks concurrently.

## 2.4 Input Model

The embedded system to be partitioned is represented as a DAG. A DAG is a graph $G(V, E)$, where $V$ is the set of tasks (functional nodes) and $E$, the edge, represents the data dependency between the two tasks with $V \in \{v_1, v_2 \ldots v_i\}$ and $E \in \{e_1, e_2 \ldots e_i\}$ (Blickle 1996; Hidalgo and Lanchares 1997; Vahid 1997; Bakshi and Gajski 1999; Zou Yi et al. 2004). The example of such a DAG is shown in Fig. 2.2.

Each task (node) in the DAG is associated with SW area and SW time. The SW area ($C_S$) represents the SW memory utilized by the task and SW time ($t_S$) represents the execution time of the task if implemented in software processor. The HW implementation of each task is associated with an HW area and an HW time. The HW area ($C_H$) represents the area occupied by the task while implementing in HW and the HW time ($t_H$) is the execution time of the task if implemented in HW. If one of the two communicating tasks is implemented in HW, and the other in SW processor, then the communication cost ($C_C$) between them incurs a significant overhead and is considered during partitioning. If the two tasks are both in HW or both in SW, then the overhead is much lower and the communication cost is neglected. The communication cost in this context refers to the delay time required to transfer the data from HW module to SW module and vice versa.

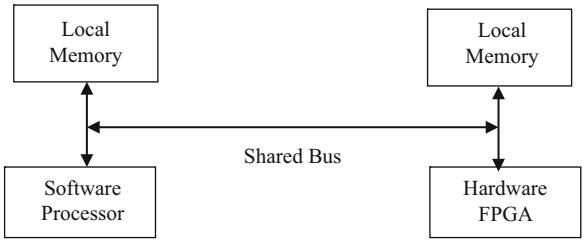


**Fig. 2.1** Target architecture

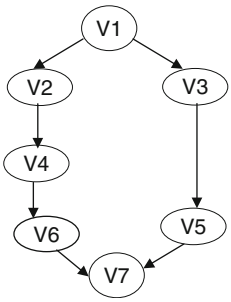


**Fig. 2.2** Directed acyclic graph

## 2.5 Objective Function

During the HW/SW partitioning process, the evolutionary algorithms iterates to move each task mapped in SW to HW, which reduces the total execution time or tries to move task mapped in HW to SW, which reduces the total area. Multi-objective evolutionary algorithms can be employed for optimizing more than one parameter such as area, speed, power, and delay. In this work, the objectives used to guide the evolutionary algorithms through the optimization process are the total execution time ($T$) and total area ($A$), which is calculated using Eqs. 2.1 and 2.2, respectively. Power can be obtained as the sum of HW power and SW power. Optimization of power can be obtained by keeping area and time as constraints.

$$T = \sum_{\forall t_i \in HW} t_H + \sum_{\forall t_i \in SW} t_S + C_C \tag{2.1}$$

$$A = \sum_{\forall t_i \in HW} C_H + \sum_{\forall t_i \in SW} C_S \tag{2.2}$$

The total time is the sum of the execution times of the tasks implemented in the HW ($t_H$) and SW ($t_S$) and the delay in communication ($C_C$) between the HW and SW tasks and vice versa. The total area is the sum of the silicon area of the tasks implemented in the HW ($C_H$) and SW memory utilized ($C_S$) by the tasks when implemented in the SW (Hou and Wolf 1996; Zou Yi et al. 2004).

## 2.6 Encoding Procedure

The chromosomes (individuals) are made up of units called genes. Group of individuals form a population. The individuals are characterized by a chromosome with an amount of genes equal to the number of functional nodes with each gene representing a task in the system. Binary encoding scheme is employed. The chromosome is characterized as an individual defined by $\{t_1, t_2, t_3, t_4 \ldots t_n\}$,

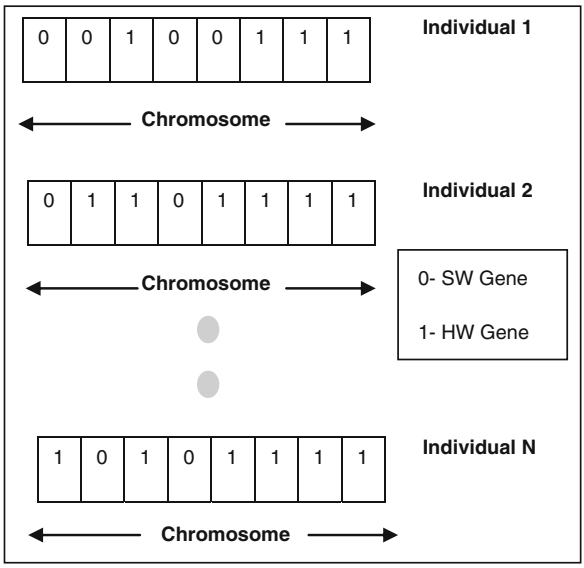


**Fig. 2.3** Sample population for HW/SW partitioning problem

$t_i \in \{1, 0\}, i \in \{1, 2 \ldots n\}$, where "$n$" is the number of tasks in the embedded system. If $t_i = 1$, the corresponding block is implemented in the HW and if $t_i = 0$, then the corresponding block is implemented in the SW as shown in Fig. 2.3.

## 2.7 Performance Metric Evaluation

When different algorithms exist for solving a particular problem, it becomes necessary to compare these in terms of their performance. There are two distinct goals in multi-objective optimization: (1) discover solutions as close to the pareto-optimal solutions as possible and (2) find solutions as diverse as possible in the obtained nondominated front. These two goals are orthogonal to each other as the first one requires a search toward pareto-optimal region while the second goal needs a search along the pareto-optimal front.

### *2.7.1 Metrics Evaluating Closeness to True Pareto-Optimal Front*

These metrics explicitly compute a measure of the closeness of a set of $M$ solutions obtained from the known true pareto-optimal set ($T$) (Deb 2002).

#### 2.7.1.1 Error Ratio (ER)

The metric ER simply counts the number of solutions that are not the members of the true pareto-optimal set $T$. It is calculated using Eq. 2.3.

$$ER = \frac{\sum_{i=1}^{M} E_i}{M} \tag{2.3}$$

$M$ is the total number of solutions obtained. The value of $E_i = 1$ if the solution "$i$" does not belong to the true pareto-optimal solutions and $E_i = 0$ otherwise. The smaller the value of $ER$, the better is the convergence to true pareto-optimal front. The metric takes a value between zero and one. An $ER = 0$ means all the solutions are members of $T$ and $ER = 1$ means no solution is a member of $T$.

#### 2.7.1.2 Generational Distance (GD)

This metric determines the average distance of the obtained solutions from the set $T$. $GD$ is obtained using Eq. 2.4:

$$GD = \frac{\left(\sum_{i=1}^{M} d_i^T\right)^{1/T}}{M} \tag{2.4}$$

where $d_i$ is the Euclidean distance between the solution $i \in M$ and the nearest member of $T$, calculated using Eq. 2.5:

$$d_i = \min_{j=1}^{|T|} \sqrt{\sum_{n=1}^{N} \left(f_n^{(i)} - f_n^{(j)}\right)^2} \tag{2.5}$$

where $f_n^{(j)}$ is the $n$th objective function value of the $j$th member of the true pareto-optimal solutions ($T$) and $N$ is the total number of objectives used in the problem. An algorithm that generates a small value of GD implies that better solutions are obtained.

#### 2.7.1.3 Maximum Pareto-Optimal Front Error (MFE)

This metric computes the worst distance $d_i$ among all the members of the obtained solutions $M$, that is, the maximum distance among the distances $d_i$ is calculated using Eq. 2.5. A lesser value of this metric indicates that the obtained solutions are closer to set $T$.

### 2.7.2 *Metrics Evaluating Diversity among Nondominated Solutions*

These metrics find the diversity among the obtained nondominated solutions (Deb 2002).

#### 2.7.2.1 Spacing (S)

This metric calculates the relative distance measure between the two consecutive solutions, using Eq. 2.6:

$$S = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(d_i - \overline{d}\right)^2} \tag{2.6}$$

where $d_i$ is calculated using Eq. 2.5 and $\overline{d}$ is the mean value of the Euclidean distance measured as given by Eq. 2.7:

$$\overline{d} = \sum_{i=1}^{M} \frac{d_i}{M} \tag{2.7}$$

A value of zero for the metric spacing indicates that all members of the obtained solutions are equidistantly spaced.

#### 2.7.2.2 Spread (Δ)

This metric measures the spread of the obtained solutions and is calculated using the formula given in Eq. 2.8:

$$\Delta = \frac{\sum_{n=1}^{N} d_n^{\text{es}} + \sum_{i=1}^{M} \left|d_i - \overline{d}\right|}{\sum_{n=1}^{N} d_n^{\text{es}} + M\overline{d}} \tag{2.8}$$

where $d_i$ and $\overline{d}$ are calculated using Eqs. 2.5 and 2.7, respectively. The parameter $d_n^{\text{es}}$ is the distance between the extreme solutions corresponding to the $n$th objective function. This metric takes a value of zero for an ideal distribution only when $d^{\text{es}} = 0$ and all $d_i$ values are identical to their mean $\overline{d}$. $d^{\text{es}} = 0$ means that only true pareto-optimal solutions exist in the obtained solutions and if $d_i$ values are identical to their mean $\overline{d}$, then the distribution of intermediate solutions obtained is uniform. Such a solution set is the ideal outcome of any multi-objective evolutionary algorithm. For an ideal distribution of solutions, the value of **Δ** is zero. In another

case, the distribution of the obtained solution is uniform but they are clustered in one place, then the distribution will make all $|d_i - \overline{d}|$ values zero and will cause nonzero values for $d_n^{es}$. The quantity **Δ** lies within [0, 1]. For bad distributions, the Δ values can be more than one as well.

#### 2.7.2.3 Weighted Metric (W)

One of the convergence metrics and one of the diversity preserving metrics are combined using Eq. 2.9 to form the weighted metric $W$:

$$W = A.GD + B.\Delta \tag{2.9}$$

With $A + B = 1$. In this metric, the GD, which specifies the converging ability, and Δ, which specifies the diversity-preserving ability of the algorithm are combined. GD takes a small value for good convergence and Δ takes a small value for good diversity-preserving algorithm. The algorithm with an overall small value of W means that the algorithm is efficient in both the aspects. To combine the two metrics, the weights A and B are chosen depending on the importance of the performance metric. To give equal weightage to the two metrics, the values are chosen as A = B = 0.5. The performance metrics namely ER, GD, MFE can be determined only when the true pareto-optimal solutions are known for the specified problem. The other metrics S, Δ, and W can be determined even when the true pareto-optimal solutions are unknown.

## 2.8 Experimental Results

The algorithm is programmed in C language and was run using 2.80 GHz, Pentium-IV processor with 1 GB RAM. The algorithms are iterated using the parameter values tabulated in Table 2.1 for the 10-node DAG (Zou Yi et al. 2004). The true pareto-optimal solutions for this system are determined using exhaustive search so that the performance metrics ER, MFE, and GD can be calculated. The adaptation of WSGA, NSGA-II, and MOPSO-CD algorithms for multi-objective optimization of HW/SW partitioning problem for 10-node DAG is obtained and analyzed for minimizing two objectives: area (A) and task execution time (T) (Jagadeeswari and Bhuvaneswari 2009). The algorithms are tested on a 10-node DAG with the input values indicated in Tables 2.1 and 2.2 (Zou Yi et al. 2004). Table 2.1 shows the data available on each node and Table 2.2 shows the delay in communication between each node. The parameter settings used in the algorithms are given in Table 2.3.

Tables 2.4, 2.5, and 2.6 list all the pareto-optimal solutions obtained using the three algorithms WSGA, NSGA-II, and MOPSO-CD, respectively along with the HW/SW implementation for 10-node system. From the results, it is found that both

**Table 2.1** Data table for 10-node system

| Node no. | SW area (KB) | SW time (μs) | HW area (KB) | HW time (μs) |
|---|---|---|---|---|
| V1 | 24 | 526 | 166 | 318 |
| V2 | 45 | 209 | 158 | 133 |
| V3 | 58 | 511 | 201 | 255 |
| V4 | 23 | 366 | 152 | 242 |
| V5 | 80 | 21 | 351 | 6 |
| V6 | 55 | 178 | 194 | 92 |
| V7 | 24 | 92 | 162 | 57 |
| V8 | 237 | 204 | 2342 | 8 |
| V9 | 67 | 277 | 239 | 117 |
| V10 | 59 | 317 | 261 | 122 |

**Table 2.2** Data exchange of 10-node system

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| V1 | – | 5 | 20 | 19 | 0 | 0 | 3 | 16 | 0 | 14 |
| V2 | 0 | 0 | 0 | 7 | 14 | 18 | 0 | 13 | 0 | 2 |
| V3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 19 | 0 |
| V4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 |
| V5 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 |
| V6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 |
| V7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| V8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 |
| V9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 |
| V10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 2.3** Parameter settings used in multi-objective algorithms for 10-node DAG

| S. No. | Parameter | WSGA | NSGA-II | MOPSO-CD |
|---|---|---|---|---|
| 1. | Population size (2N) | 20 | 20 | – |
| 2. | Crossover probability | 0.8 | 0.8 | – |
| 3. | Mutation probability | 0.02 | – | – |
| 4. | Type of crossover | Two point | Two point | – |
| 5. | Number of generations | 100 | 30 | 100 |

NSGA-II and MOPSO-CD are able to determine all the pareto-optimal solutions. MOPSO-CD determines the output with less search time than NSGA-II and WSGA for the 10-node DAG. To compare the performance of the three algorithms and to determine the performance metrics, true pareto-optimal solutions for the 10-node DAG are obtained using exhaustive search method. It is carried out by calculating the nondomination output for all the objectives in $2^{10}$ binary combinations. The plot of such a search is shown in Fig. 2.4.

The quality of the results obtained using WSGA, NSGA-II, and MOPSO-CD is measured using the performance metrics that identify how close and diverse are the obtained pareto-optimal solutions with the true pareto-optimal solutions. The

**Table 2.4** Optimal solutions obtained using WSGA

| Output | Node implementation | Total area (KB) | Total execution time (μs) | Number of process in HW |
|---|---|---|---|---|
| Solution 1 | 1111111111 | 4,226 | 1,556 | 10 |
| Solution 2 | 1111011111 | 3,955 | 1,596 | 9 |
| Solution 3 | 1111010111 | 3,817 | 1,701 | 8 |
| Solution 4 | 1111001111 | 3,816 | 1,798 | 8 |
| Solution 5 | 1111111011 | 2,121 | 1,812 | 9 |
| Solution 6 | 1111011011 | 1,850 | 1,852 | 8 |
| Solution 7 | 1111010011 | 1,712 | 1,957 | 7 |
| Solution 8 | 1111001011 | 1,711 | 2,054 | 7 |
| Solution 9 | 1110011001 | 1,549 | 2,097 | 6 |
| Solution 10 | 1111001010 | 1,509 | 2,144 | 6 |
| Solution 11 | 1110010001 | 1,411 | 2,182 | 5 |
| Solution 12 | 1111000010 | 1,371 | 2,249 | 5 |
| Solution 13 | 1110001000 | 1,208 | 2,327 | 4 |
| Solution 14 | 1000010001 | 1,155 | 2,380 | 3 |
| Solution 15 | 1110000000 | 1,070 | 2,412 | 3 |
| Solution 16 | 0000010001 | 1,013 | 2,539 | 2 |
| Solution 17 | 1000000000 | 814 | 2,570 | 1 |
| Solution 18 | 0000000000 | 672 | 2,701 | 0 |

**Table 2.5** Optimal solutions obtained using NSGA-II

| Output | Node implementation | Total area (KB) | Total execution Time (μs) | Number of Process in HW |
|---|---|---|---|---|
| Solution 1 | 1111110111 | 4,088 | 1,406 | 9 |
| Solution 2 | 1011110011 | 1,870 | 1,407 | 7 |
| Solution 3 | 1011010011 | 1,599 | 1,419 | 6 |
| Solution 4 | 1011000011 | 1,460 | 1,463 | 5 |
| Solution 5 | 1011110011 | 1,331 | 1,606 | 7 |
| Solution 6 | 1011000010 | 1,258 | 1,671 | 7 |
| Solution 7 | 1010000010 | 1,129 | 1,800 | 3 |
| Solution 8 | 1011000000 | 1,086 | 1,933 | 3 |
| Solution 9 | 0010000010 | 987 | 1,971 | 2 |
| Solution 10 | 0011000000 | 944 | 2,142 | 2 |
| Solution 11 | 0010000000 | 815 | 2,233 | 1 |
| Solution 12 | 1000000000 | 814 | 2,570 | 1 |
| Solution 14 | 0001000000 | 801 | 2,610 | 1 |
| Solution 15 | 0000000000 | 672 | 2,701 | 0 |

different performance metrics such as Error Ratio (ER), Generational Distance (GD) or Standard Deviation ($\gamma$), Maximum Pareto-Optimal Front Error (MFE), Spacing (S), Spread ($\Delta$), and Weighted Metric (W) for the 10-node DAG using WSGA, NSGA-II, and MOPSO-CD are calculated using the formulas mentioned

**Table 2.6** Optimal solutions obtained using MOPSO-CD

| Output | Node implementation | Total area (KB) | Total execution time (μs) | Number of process in HW |
|---|---|---|---|---|
| Solution 1 | 1111110111 | 4,088 | 1,406 | 9 |
| Solution 2 | 1011110011 | 1,870 | 1,407 | 7 |
| Solution 3 | 1011010011 | 1,599 | 1,419 | 6 |
| Solution 4 | 1011000011 | 1,460 | 1,463 | 5 |
| Solution 5 | 1011110011 | 1,331 | 1,606 | 7 |
| Solution 6 | 1011000010 | 1,258 | 1,671 | 7 |
| Solution 7 | 1010000010 | 1,129 | 1,800 | 3 |
| Solution 8 | 1011000000 | 1,086 | 1,933 | 3 |
| Solution 9 | 0010000010 | 987 | 1,971 | 2 |
| Solution 10 | 0011000000 | 944 | 2,142 | 2 |
| Solution 11 | 0010000000 | 815 | 2,233 | 1 |
| Solution 12 | 1000000000 | 814 | 2,570 | 1 |
| Solution 14 | 0001000000 | 801 | 2,610 | 1 |
| Solution 15 | 0000000000 | 672 | 2,701 | 0 |

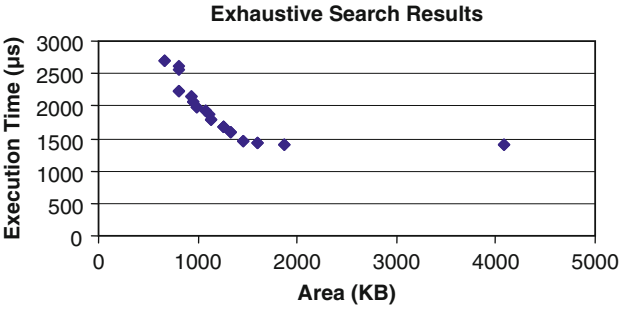


**Fig. 2.4** True pareto-optimal solutions

**Table 2.7** Comparison of performance metrics for 10-node system

| Method | ER | GD(γ) | S | MFE | Δ | W | Run time (s) |
|---|---|---|---|---|---|---|---|
| MOPSO-CD | **0.0** | **0.0** | **74.9** | **0.0** | **0.5** | **0.20** | **0.34** |
| NSGA-II | **0.0** | **0.0** | **74.9** | **0.0** | **0.5** | **0.20** | 1.05 |
| WSGA | 0.22 | 34.55 | 102.69 | 93.60 | 3.84 | 20.19 | 9.62 |

and tabulated in Table 2.7. The first three metrics are calculated based on the true pareto-optimal solutions obtained with the exhaustive search method. The other three metrics can be determined without the knowledge of true pareto-optimal solutions. It is noted that the performance metrics obtained by using NSGA-II and MOPSO-CD are the same, since both the algorithms find the same pareto-optimal solutions. The performance of MOPSO-CD algorithm with respect to run time is better than NSGA-II and WSGA for the 10-node DAG.

**Table 2.8** Simulation results for benchmark DAG

| Bench mark DAG | Node/ edges | Algorithm used | Mean area (KB) | Mean execution time (μs) | Run time (s) |
|---|---|---|---|---|---|
| HAL FILTER | 11/8 | GA | 381.88 | 154.53 | 0.17 |
| | | WSGA | 379.24 | 135.23 | 8.33 |
| | | NSGA-II | 354.50 | 124.74 | 1.42 |
| | | MOPSO-CD | **323.70** | **124.74** | **0.16** |
| MPEG MOTION VECTOR | 32/29 | GA | 979.29 | 543.67 | 1.48 |
| | | WSGA | 889.54 | 423.59 | 12.55 |
| | | NSGA-II | 849.61 | 398.70 | 2.81 |
| | | MOPSO-CD | **840.33** | **343.22** | **0.59** |
| JPEG DOWN SAMPLE | 49/52 | GA | 1032.79 | 888.53 | 5.11 |
| | | WSGA | 1015.25 | 723.85 | 15.68 |
| | | NSGA-II | **885.14** | **459.43** | 4.06 |
| | | MOPSO-CD | 965.11 | 601.22 | **3.77** |
| MPEG | 114/164 | GA | 4551.35 | 1579.56 | 20.64 |
| | | WSGA | 3971.25 | 1497.47 | 34.23 |
| | | NSGA-II | **3312.50** | **981.50** | 11.52 |
| | | MOPSO-CD | 3523.50 | 1281.50 | **10.71** |
| JPEG IDCT | 120/162 | GA | 4339.88 | 3543.50 | 21.26 |
| | | WSGA | 3455.58 | 2563.98 | 37.09 |
| | | NSGA-II | **3010.31** | **1969.46** | **11.23** |
| | | MOPSO-CD | 3145.38 | 2368.69 | 11.66 |

An effective HW/SW partition for the mediabench and DSP benchmark DAGs was obtained using WSGA, NSGA-II, and MOPSO-CD multi-objective optimization algorithms and compared with single-objective GA. All the algorithms are run for 100 generations. The results are analyzed and tabulated in Table 2.8. It is observed that MOPSO-CD algorithm performed better than NSGA-II, WSGA, and GA for HAL FILTER and MPEG MOTION VECTOR benchmarks with less mean area and mean task execution time. The run time of MOPSO-CD algorithm is less than GA, WSGA, and NSGA-II.

For all other benchmark DAGs applied, NSGA-II performs better than MOPSO-CD, WSGA, and GA with lesser mean area and mean task execution time. The run time of MOPSO-CD is comparable to NSGA-II and less when compared to GA and WSGA. From the experimental results it is observed that MOPSO-CD performs better for DAGs with lesser nodes (less than 50 nodes) and NSGA-II is found to perform better than MOPSO-CD, GA and WSGA for DAGs with larger number of nodes (more than 50 nodes).

## 2.9 Summary

In this chapter, three multi-objective evolutionary algorithms, WSGA, NSGA-II and MOPSO-CD, are dealt for HW/SW partitioning of embedded systems. Multioptimal solutions for the functional partitioning of embedded systems using

WSGA, NSGA-II, and MOPSO-CD was obtained. The pareto-optimal solutions obtained for a 10-node DAG are analyzed. The closeness to true pareto-optimal solutions and the diversity among the pareto-optimal solutions found is determined by calculating the performance metrics. It is seen that both NSGA-II and MOPSO-CD algorithms performed better than WSGA for 10-node DAG. The reason may be that crowding distance strategy retains the optimal solutions obtained in each generation until the final generation is reached. The algorithm was tested on several mediabench and DSP benchmark DAGs. For the mediabench and DSP benchmark applications, MOPSO-CD was found to perform better for graphs less than 50 nodes. NSGA-II was found to perform better compared to GA, WSGA, and MOPSO-CD, for nodes greater than 50 and HW/SW implementations with lesser mean area and mean execution time are obtained.

## References

Arato P, Juhasz S, Mann ZA, Orban A, Papp D (2003) Hardware/software partitioning in embedded system design. In: Proceedings of the IEEE international symposium on intelligent signal processing, 4–6 Sept 2003, Budapest, Hungary, pp 197–202

Bakshi S, Gajski D (1999) Partitioning and pipelining for performance-constrained hardware/software system. IEEE Trans Very Large Scale Integr Syst 7(4):419–432

Barros E, Rosenstiel W, Xiong X (1993) Hardware/software partitioning with UNITY. In: Proceedings of 2nd international workshop on hardware-software codesign- CODES/CASHE '93, Austria, 24–27 May 1993, pp 210–217

Blickle T (1996) Theory of evolutionary algorithms and applications to system synthesis. Dissertation, Swiss Federal Institute of Technology, Zurich

Binh NN, Imai M, Shiomi A, Hikichi N (1996) A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate counts. In: Proceedings of the 33rd annual conference on design automation, 3–7 June 1996, Las Vegas, pp 527–532

Chatha KS, Vemuri R (2001) MAGELLAN: Multiway hardware-software partitioning and scheduling for latency minimization of hierarchical control-dataflow task graphs. In: Proceedings of CODES'01, 25–27 Apr 2001, Copenhagen, pp 42–47

D'Ambrosio JD, Hu X (1994) Configuration-level hardware/software partitioning for real-time embedded systems. In: Proceedings of third international workshop on hardware/software codesign, 22–24 Sept 1994, Grenoble, pp 34–41

Deb K (2002) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester

Dick RP, Jha NK (1997) MOGAC: a multi-objective genetic algorithm for the co-synthesis of hardware-software embedded systems. In: Proceedings of IEEE/ACM international conference on computer-aided design, 9–13 Nov 1997, San Jose, pp 522–529

Eles P, Peng Z (1996) System level hardware/software partitioning based on simulated annealing and Tabu Search. Des Autom Embed Syst 2(1):5–32

Ernst J, Henkel J, Benner T (1992) Hardware software co synthesis for microcontrollers. IEEE Des Test Comput 10(4):64–75

Gupta RK, De Micheli G (1993) Hardware-software co-synthesis for digital systems. IEEE Des Test Comput 10(3):29–41

Greg Stit (2008) Hardware/software partitioning with multi-version implementation exploration. In: Proceedings of 18th ACM Great Lakes symposium on VLSI, 04–06 May 2008, USA, pp 143–146

He Jifeng, Huang, Dang, Pu Geguang, Qiu, Zong Yan Yi, Wang (2005) Exploring optimal solution to hardware/software partitioning for synchronous model. J Formal Aspects Comput 17(4):587–611

Jagadeeswari M, Bhuvaneswari MC (2009) An efficient multi-objective genetic algorithm for hardware-software partitioning in embedded system design: ENGA. Int J Comput Appl Technol 36(3/4):181–190

Kalavade A, Lee E (1994) A global criticality/local phase driven algorithm for the constrained hardware/software partitioning problem. In: Proceedings of third international workshop on hardware/software co design, 22–24 Sept 1994, Grenoble, pp 42–48

Kirkpatrick S Jr, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Comput Sci J 220(4598):671–680

Henkel J, Ernst R (2001) An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques. IEEE Trans VLSI Syst 9(2):273–289

Hidalgo JI, Lanchares J (1997) Functional partitioning for hardware-software codesign using genetic algorithms. In: Proceedings of 23rd EUROMICRO conference, 01–04 Sept 1997, Budapest, Hungary, pp 631–638

Hou J, Wolf W (1996) Process partitioning for distributed embedded systems. In: Proceedings of fourth international workshop on hardware/software co-design, Mar 1996, Pittsburgh, Pennsylvania, pp 70–76

Lopez Vallejo M, Grajal J, Lopez JC (2000) Constraint-driven system partitioning. In: Proceedings of design automation and test in Europe, Jan 2000, Paris, France, pp 411–416

Lopez Vallejo M, Lopez JC (2001) Multi-way clustering techniques for system level partitioning. In: Proceedings of the 14th IEEE ASIC/SOC conference, 12–15 Sept 2011, Arlington, pp 242–247

Madsen J, Grode J, Knudsen P (1997) Hardware/software partitioning using the LYCOS system. Hardware/software codesign: principles and practices, Springer, US

Niemann R, Marwedel P (1997) An algorithm for hardware/software partitioning using mixed linear programming. Des Autom Embed Syst 2(2):165–193

Schaumont P (2013) A practical introduction to hardware/software codesign, 2nd edn. Springer Science+Business media, New York

Shrivastava A, Kumar M (2000) Optimal hardware/software partitioning for concurrent specification using dynamic programming. In: Proceedings of thirteenth international conference on VLSI design, 03–07 Jan 2000, Calcutta, India, pp 110–113

Srinivasan V, Radhakrishnan S, Vemuri R (1998) Hardware software partitioning with integrated hardware design space exploration. In: Proceedings of design automation and test in Europe, Feb 1998, Paris, France, pp 28–35

Theerayod Wiangtong (2004) Hardware/software partitioning and scheduling for reconfigurable systems. Dissertation, Imperial College, London

Tsou CS, Fang HH, Chang HH, Kao CH (2006) An improved particle swarm pareto optimizer with local search and clustering, vol 4247, Lecture notes in computer science. Springer-Verlag, Berlin, Heidelberg, pp 400–406

Vahid F (1997) Modifying min-cut for hardware and software functional partitioning. In: Proceedings of fifth international workshop on hardware/software co-design, 24–26 Mar 1997, Germany, p 43

Vahid F, Gajski DD (2001) Incremental hardware estimation during hardware/software functional partitioning. IEEE Trans VLSI Syst 3:516–521

Vahid F, Jie Gong, Daniel D Gajski (1994) A binary constraint search algorithm for minimizing hardware during hardware/software partitioning. In: Proceedings of the conference on European design automation EURO-DAC '94, Grenoble, pp 214–219

Wu Jigang, Qiqiang Sun, Thambipillai Srikanthan (2010) Multiple-choice hardware/software partitioning: computing models and algorithms. In: Proceedings of international conference on computer engineering and technology, April 2010, China, pp 61–65

Zou Yi, Zhenquan Zhuang, Huanhuan Chen (2004) HW/SW partitioning based on genetic algorithm. In: Proceedings of the congress on evolutionary computation, 19–23 June 2004, pp 628–633

Springer

http://www.springer.com/978-81-322-1957-6

Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems

Bhuvaneswari, M.C. (Ed.)

2015, XI, 174 p. 63 illus., 8 illus. in color., Hardcover
ISBN: 978-81-322-1957-6