Hindawi Publishing Corporation
VLSI Design
Volume 2012, Article ID 273276, 12 pages
doi:10.1155/2012/273276

Research Article

A Novel Framework for Applying Multiobjective GA and PSO Based Approaches for Simultaneous Area, Delay, and Power Optimization in High Level Synthesis of Datapaths

D. S. Harish Ram,[1] M. C. Bhuvaneswari,[2] and Shanthi S. Prabhu[1]

[1] *Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham University, Coimbatore 641112, India*
[2] *Department of Electrical and Electronics Engineering, PSG College of Technology, Coimbatore 641004, India*

Correspondence should be addressed to D. S. Harish Ram, ds_harishram@cb.amrita.edu

Received 3 February 2012; Revised 19 August 2012; Accepted 23 September 2012

Academic Editor: Dinesh Mehta

Copyright © 2012 D. S. Harish Ram et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

High-Level Synthesis deals with the translation of algorithmic descriptions into an RTL implementation. It is highly multi-objective in nature, necessitating trade-offs between mutually conflicting objectives such as area, power and delay. Thus design space exploration is integral to the High Level Synthesis process for early assessment of the impact of these trade-offs. We propose a methodology for multi-objective optimization of Area, Power and Delay during High Level Synthesis of data paths from Data Flow Graphs (DFGs). The technique performs scheduling and allocation of functional units and registers concurrently. A novel metric based technique is incorporated into the algorithm to estimate the likelihood of a schedule to yield low-power solutions. A true multi-objective evolutionary technique, "Nondominated Sorting Genetic Algorithm II" (NSGA II) is used in this work. Results on standard DFG benchmarks indicate that the NSGA II based approach is much faster than a weighted sum GA approach. It also yields superior solutions in terms of diversity and closeness to the true Pareto front. In addition a framework for applying another evolutionary technique: Weighted Sum Particle Swarm Optimization (WSPSO) is also reported. It is observed that compared to WSGA, WSPSO shows considerable improvement in execution time with comparable solution quality.

1. Introduction

Data path algorithms can be depicted using Data Flow Graphs (DFGs). Each *node* in the DFG represents an operation that is to be executed in the algorithm (Figure 1). The operation may be arithmetic such as addition or multiplication or logical. High Level Synthesis (HLS), also known as Behavioral Synthesis is the process of converting an algorithmic description into a synthesizable Register Transfer Level (RTL) netlist. The high level description may be in the form of a programming language such as C or DFGs. The HLS design flow consists of three subtasks, namely, scheduling, allocation, and binding. *Scheduling* determines the *time step* at which a node in the DFG is executed. For example, in the following DFG, the node *d* is scheduled in the second time step. Some nodes have *mobility* in that they can have several potential execution instances as in the case of node c which can execute in time step 1 or 2 without affecting the schedule.

The term allocation refers to the process of assigning resources or Functional Units (FUs) to execute a particular operation. For instance, one possible allocation for the DFG shown in Figure 1 is three adders and one multiplier. In *binding*, a node is bound to a FU for execution. Thus, node *a* may be bound to Adder 1, node *b* to Adder 2 and node *c* to Adder 3. The HLS sub-tasks can be performed in any order or simultaneously. The result of each sub-task influences the others.

The main purpose of HLS algorithms is to facilitate an early design space exploration of various alternative RTL implementations for a particular algorithm with the primary objective of minimizing area, delay, and power of the final design. This paper investigates the use of evolutionary techniques such as Genetic Algorithms (GAs) and Particle

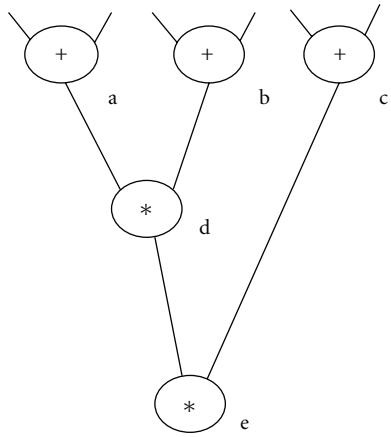


Figure 1: A Data Flow Graph.

Swarm Optimization (PSO) for optimal scheduling of DFGs during High Level Synthesis. Since the problem is multi-objective in nature, the GA framework is adapted for applying true multi-objective algorithms such as NSGA II that yield a population of Pareto optimal solutions with each solution representing a trade-off among the objectives. Also we have adapted the framework developed for applying another multi-objective evolutionary approach, Weighted Sum PSO (WSPSO), for solving the same problem.

This work is based on the technique reported by the authors in Harish Ram et al. [1]. The methodology described in [1] proposes an NSGA II based approach for optimal scheduling during HLS of datapaths. The technique has been validated on additional benchmarks. Also the basic framework used for implementing the GA has been customized for applying WSPSO to the multi-objective HLS problem.

Previous work on multi-objective optimization in HLS has focused primarily on area and delay minimization. In [2], a weighted sum approach has been proposed for area and delay minimization during HLS of DFGs. In Ferrandi et al. [3], the authors have used two different encodings which prioritize scheduling and binding, respectively. The cost function is computed using regression models derived from actual characterizations. A true multi-objective GA (NSGA II) is used to optimize the schedules. We propose an evolutionary framework using the chromosome encoding in [2], with the cost function modified to incorporate power in addition to area and delay. Power cost of a schedule is determined as the likelihood of the schedule to yield low-power bindings, computed from compatibility graph metrics described in [4]. The methodology is evaluated for WSGA and NSGA II on standard benchmarks [2, 5] and the superiority of NSGA II is demonstrated. The framework is extended for applying WSPSO to the HLS scheduling problems.

The rest of the paper is organized as follows. Section 2 describes the new algorithms used in the work. Section 3 gives a review of related literature. Section 4 outlines the evolutionary encoding scheme and cost function computation used in the work. Sections 5, 6, and 7, respectively detail the WSGA, NSGA II, and WSPSO based methodologies. Section 8 discusses the results and Section 9 concludes the paper.

2. New Algorithmic Techniques Used

The application of Genetic Algorithms and similar meta-heuristics in engineering problems has come to stay and is a well researched domain. However, many optimization problems are multi-objective in nature with the candidate solutions trading off one objective in favor of the other [6]. A classic instance in the VLSI domain is the area-delay trade-off. In such problems, the focus is on identifying a population of Pareto optimal solutions rather than a single optimal solution. A simple approach in solving such problems will use a weighted cost function involving all the objectives to be optimized. However this technique is flawed since it yields solutions with limited diversity [6]. This necessitates the development of true multi-objective algorithms that are capable of exploring the entire design space of solutions with different trade-offs among the objectives. Several multi-objective techniques have been proposed to solve engineering problems in the past decade and a half [7–9]. Multi-objective optimization of area and delay during datapath synthesis has been addressed in [2, 3]. However power is not considered in these schemes. In [10] Bright and Arslan had proposed a GA based approach to generate Pareto surfaces for area and delay during synthesis of DSP designs. The paper does not describe any scheme for ensuring the diversity of the solutions. We propose a multi-objective methodology using NSGA II [7] for power, area, and delay optimization of DFGs. NSGA II incorporates a selection technique based on the crowding distance of individuals in a population that preserves the diversity of the solutions when the algorithm evolves. The technique is evaluated on standard benchmarks and fares substantially better than WSGA in terms of solution quality as well runtime. PSO is another recent evolutionary technique that has been widely investigated for solving intractable problems in engineering and science [8, 11]. PSO has been widely applied in scheduling problems such as instruction scheduling and traveling salesman problem (TSP) [12–16]. We have extended the chromosome encoding scheme and cost function computation used in our GA technique for applying Weighted Sum PSO for datapath synthesis.

3. Review of Previous Work

Early work in Behavioral Synthesis focused on *constructive approaches* as in force directed scheduling [2, 17]. These are essentially greedy approaches vulnerable to local minima. *Transformational approaches* [2] start with an initial schedule and apply transformations to improve the initial solution. However, the quality of the solutions largely depends on the transformations used and the heuristics used to select between applicable transformations [2]. *Graphical approaches* have been proposed in which the minimum

cost binding problem is converted into a multiway partitioning of a flow graph. In [18], a graph network is used to model the binding problem with the edge weights representing the switching cost of executing two nodes in succession on the same functional unit (FU) or register. Thus the problem reduces to a multi-way partitioning of the graph for minimal flow cost. This method is reported for optimal bus scheduling and can be extended to FUs also. Scheduling under constraints is not addressed here. In [19], graph based heuristics are proposed to choose an optimal set of resources for a scheduled DFG with more than one architecture to choose from for each functional unit. *Mathematical approaches* formulate the synthesis problem using Integer Linear Programming (ILP) approach or some other mathematical optimization tools such as game theory. Simultaneous binding and scheduling of a DFG using ILP is addressed in [20]. The precedence and time constraints are built into the ILP formulation. The cost function is evaluated based on signal statistics of the inputs obtained through a one-time simulation. The technique suffers from the drawback that ILP methods do not scale well for large circuits and may have to be combined with some heuristics. An ILP approach is combined with retiming for power optimization in [21]. In [22], a game-theoretic approach for power optimization of a scheduled DFG is proposed. The functional units are modeled as bidders for the operations in the DFG with power consumption as the cost. The algorithm does not scale well for larger number of functional units since the complexity increases exponentially with the number of players (bidders).

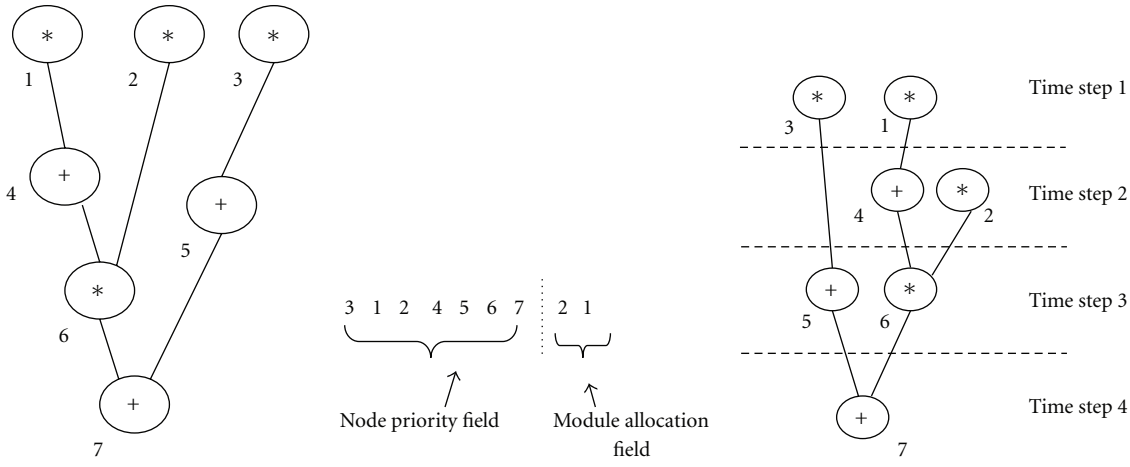
The use of Genetic Algorithms (GAs) for optimal scheduling, allocation, and binding in Behavioral Synthesis has been widely reported in the literature. A survey of GA based methodologies for Behavioral Synthesis is presented in [2]. Genetic Algorithms being population based heuristics are extremely effective in design space exploration. A GA based methodology for datapath synthesis from a DFG for multi-objective area and delay optimization is described in [2]. The authors propose a multichromosome encoding scheme for the DFG scheduling priority and functional unit constraints. List scheduling is carried out based on the priorities coded in the chromosome. A weighted cost function incorporating both area and delay is employed for assessing the fitness of a given schedule. But the weighted sum approach suffers from the drawback that in a sufficiently nonlinear problem; it is likely that the optimal solutions resulting from a uniformly spaced set of weight vectors may not result in a uniformly spaced set of Pareto optimal solutions [6]. Ferrandi et al. [3] have proposed an approach based on Multi-objective GA using the algorithm NSGA II [6, 7]. The authors have used two different encoding schemes. The priority based scheme [2] arranges nodes in the DFG in the order in which they have to be scheduled by a list scheduler whereas the binding based scheme [23] incorporates binding information pertaining to each DFG node. The area and delay costs are extracted from a regression model built using actual characterizations. Power is not included in the fitness evaluation.

The optimal scheduling problem in HLS is somewhat analogous to the traveling salesman problem (TSP). In both cases, an evolutionary approach necessitates a chromosome encoding which specifies the *order* in which scheduling of nodes (cities to be visited in case of TSP) is carried out as described in the next section. These problems differ in their cost functions. TSP seeks to minimize the distance travelled whereas in HLS the objective is to optimize area, delay, or power as the case may be. In addition, the precedence of execution of each node is to be considered during HLS. Wang et al. propose a PSO methodology for TSP in [12]. The authors describe a technique for generating new scheduling strings using the swap operator. An encoding scheme for efficient convergence of the particles during PSO is proposed in [13]. In [14], Jie et al. describe a scheme for reducing the scale of the problem for large number of nodes, without affecting exploration of the search space. To our knowledge, PSO has not been used in multi-objective High Level Datapath Synthesis.

4. Evolutionary Encoding Scheme, Cost Function Computation, Crossover, and Mutation

4.1. Encoding Scheme. A multi-chromosome encoding scheme reported in [2] is used in the proposed technique. The same encoding is used for both GA and PSO. In this scheme a chromosome has a *node scheduling priority field* and a *module allocation field* (Figure 2(b)). The structure of the chromosome is such that simultaneous scheduling of a DFG and functional unit allocation can be carried out. The DFG nodes are scheduled using a list scheduling heuristic [2, 17]. The nodes are taken up for scheduling in the order in which they appear in the chromosome. The module constraint is described in the *module allocation field*. The respective constraints of the number of multipliers and adders are specified in this field. If additional FUs such as subtractors are present, the module allocation field will have more terms.

As an example, an unscheduled DFG and a corresponding chromosome encoding are shown respectively in Figures 2(a) and 2(b). The scheduling starts with the first time step. Node 3 which appears first in the chromosome is scheduled in the first time step itself. Also node 1 is scheduled to be executed in time step 1 since it appears next in the chromosome. Node 2 which is next cannot be scheduled in the first time step because only two multipliers are available as specified in the module allocation field. Therefore Node 2 has to wait till the next time step. Node 4 which performs addition cannot be scheduled in this time step since its predecessor node, that is, Node 1, has just been scheduled. The result of the execution of this node will be available only in the next time step. Similarly nodes 5, 6, and 7 cannot be scheduled since the results of their predecessor nodes are not ready. Thus we move to the second time step since no further scheduling is possible in the current time step. In this step, nodes 2 and 4 which were not scheduled in the previous time step are scheduled for execution. This sequence continues till



(a) Unscheduled DFG (b) Multi-chromosome (c) Scheduled DFG

Figure 2: Multichromosome and corresponding schedule.

all nodes are scheduled. It has to be noted that in a valid string the precedence relationship between the nodes has to be maintained. For instance, in the chromosome shown in Figure 2(b), the nodes 1 and 2 which are the predecessors for node 6 appear before node 6 in the priority list. The scheduled DFG is shown in Figure 2(c).

4.2. Cost Function Computation. The fitness evaluation of each chromosome is carried out as described in [2] for area and delay. The potential of a given schedule to yield low-power bindings is determined from the compatibility graphs for the FUs and registers extracted from the schedule using a method described in [4]. Actual power numbers are not computed.

4.2.1. Area and Delay Cost. The delay or length L is the number of time steps or clock cycles needed to execute the schedule. For example $L = 4$ for the schedule in Figure 2(c). The area cost A of the schedule is based on the total FUs and registers required to bind all the nodes in the DFG. The former is known from the chromosome itself (e.g., 2 multipliers and 1 adder for the DFG in Figure 2(c)). The latter is obtained by determining the total number of registers required for the DFG implementation using the left edge algorithm [17]. The total area is expressed as the total number of transistors required to synthesize the FUs and registers to a generic library.

4.2.2. Power Cost. The potential of an individual schedule to yield a low-power binding solution is found out using a set of metrics described in [4]. Here a *compatibility graph* (CG) is extracted from the scheduled DFG corresponding to the schedule obtained from the node priority field in each chromosome of the population. The compatibility graph will have an edge between two nodes if they are *compatible.* Two nodes are said to be compatible if they can be bound to the same FU or register. For instance, nodes 1 and 2 in the DFG in Figure 2(c) are FU compatible since they execute the same

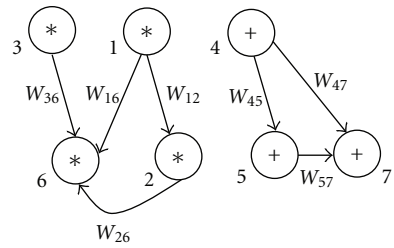


Figure 3: Compatibility graph.

operation (multiplication) and their lifetimes do not overlap. On the other hand, nodes 3 and 2 though executing the same operation are not compatible since their lifetimes overlap. The output of node 3 is required by node 5 in the third time step whereas node 2 has to be executed in the second time-step itself. The compatibility graph for the schedule given in Figure 2(c), is given in Figure 3. Separate CGs are created for the registers and FUs. The edge weights between two nodes represent the *switching cost* when these two nodes are executed one after the other in the same FU [24].

The power-related metrics [4] are based on the edge weights of the compatibility graph and are defined as follows:

m_1 = total number of edges in the graph,

m_2 = average of edge weights for the lowest k% in the value range for each node where k is user-defined

m_3 = average of all edge weights (i.e., m_2 for k = 100%). A DFG may yield a low-power binding if m_1 is high and m_2 and m_3 are low. The power cost function designed based on the compatibility graph metrics is given below

$$P = \left(\frac{n}{m_1}\right) + m_2 + m_3, \tag{1}$$

where n is a tuning parameter, is chosen depending on the value of m_1, and has a value greater than m_1. The rationale

for the choice of the power metrics is as follows. Higher number of edges (m_1) in the CG indicates higher number of possible bindings and hence more likelihood of the schedule yielding low-power bindings. Thus the power metric is made *inversely proportional* to m_1. Smaller values of CG edge weights indicate lesser switching cost and hence lower power dissipation when a pair of nodes execute on an FU. Thus a schedule having a CG with smaller average edge weight has better potential to yield more power-aware bindings. This dependency is encoded in the second and third terms of the cost function, m_2 and m_3.

4.3. Crossover. We have used the single-point crossover technique reported in [2]. In this method, the precedence relationships between the nodes in the node priority field of the multi-chromosome are maintained. The technique is illustrated in Figure 9.

The parent string is retained intact till the crossover point. For instance, in the example shown in Figure 9, crossover is carried out from the fifth position in the string. Thus, in offspring 1, the substring 3 1 2 4 is replicated from Parent 1. From location 5 onwards, the Parent 2 string is traversed and those nodes in the list that are *not present* are filled in the same order in which they appear in Parent 2. For instance, the nodes 1, 2, and 4 are already present in the substring 3 1 2 4 taken from Parent 1, but 6 is not present and hence is filled in the location 5. The node 3 which appears next is already present whereas 5, which appears next to 3, is not present and is filled in. The procedure is continued till the entire string is generated. The same sequence is repeated with Parent 2 for generating offspring 2.

For the module allocation field, the module allocations which appear after the crossover point are swapped. In Figure 9, the module allocation strings are 2 1 and 2 2 respectively for parent *Parent 1* and *Parent 2*. After crossover, the module allocation strings become 2 2 and 2 1, respectively.

4.4. Mutation. Mutation also is based on the technique described in [2]. For the node priority field, a node in the list is chosen at random and is moved without affecting the node precedence constraints. As an example, consider the same chromosome described previously, that is, the string 3 1 2 4 6 5 7 | 2 1. Let node 5 in location 6 be chosen for the mutation. Its predecessor is node 3 in the first slot and its successor is node 7 in the seventh slot. Thus the node can be moved to any slot from slot 2 to slot 6 without violating the precedence constraints. The actual slot to which the node is to be moved is chosen by generating a random number between 2 and 6. If the random number generated is, say, 3, then the node 5 is moved to slot 3 and the new chromosome after mutation becomes 3 1 **5** 2 4 6 7 | 2 1 (mutated node is shown in bold).

For the module allocation field, the mutation is effected by simply incrementing or decrementing the number of functional units. For example, we may decrement the number of FU in the above multi-chromosome. Thus after mutation, the final chromosome string becomes 3 1 **5** 2 4 6 7 | **1 1** (mutated locations are shown in bold).

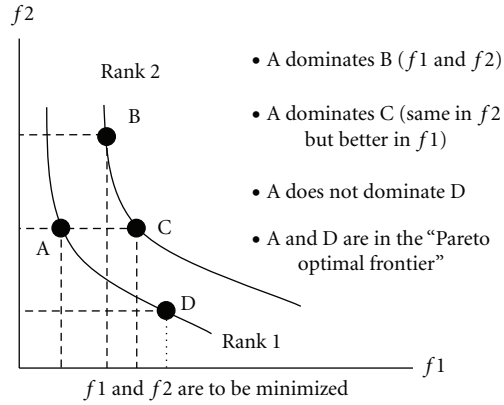


Figure 4: Multi-objective solution space and Pareto Front.

5. Weighted Sum GA

The weighted sum approach [6] used in this paper converts a multi-objective problem into a single objective GA. Each term in the cost function of this GA represents the cost of one objective to be optimized and is multiplied by a weight factor between 0 and 1. This weight is chosen depending upon the extent to which a particular objective is to be optimized. This approach is computationally simple. The fitness evaluation of each individual is based on a cost function value calculated using a weighted sum approach. The cost function described in [2] is modified to include a power dissipation term described in the previous section and is given below:

$$C = \frac{w_1 L_s}{L_{\max}} + \frac{w_2 A}{A_{\max}} + w_3 P, \tag{2}$$

where w_1, w_2, and w_3 are the weights of the area, delay, and power terms, respectively, with w_1, w_2, and w_3 always obeying the relation $w_1 + w_2 + w_3 = 1$. The terms L_s and A are the length and area, respectively, of the implementation of the given schedule and P is the power metric. $A_{\max}$ and $L_{\max}$ are the maximum area and delay, respectively, among solutions in the current population. A population size of 100 is used in the GA implementation. Elitism is used for preserving the best solutions in a generation. Binary tournament selection is used to identify parents for crossover. The crossover probability used is 0.9 and mutation probability is 0.2. The WSGA run was repeated with 10 different initial seed populations.

6. NSGA II

In a *true multi-objective* algorithm, the solutions converge to the *true* Pareto front of *non-dominated* solutions. A solution is said to be *nondominated* if no other solution can be found in the population that is better for *all the objectives* (Figure 4). A multi-objective solution space for a minimization problem involving two objectives and the Pareto front is depicted in Figure 4. The weighted sum approach described in the previous section suffers from lack of diversity in that uniformly spaced weights do not result in a uniform spread of solutions as shown in Figure 5. The

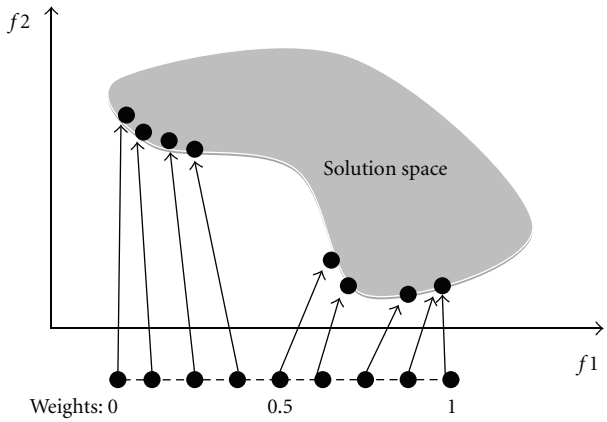


Figure 5: Solutions for WSGA [6].

figure depicts a solution space for a minimization problem with two objectives, $f1$ and $f2$. The x-axis represents different values of $f1$ for uniformly spaced values for the weight for $f2$ (say, w_2). For $w_2 = 0$, $f1$ has the most optimal value and $f2$ the worst. For higher values of w_2, the objective $f2$ shows progressively better values by trading off $f1$. But it can be observed that the values of $f2$ obtained by trading off $f1$ are not uniformly spaced though the weight for $f2$ (w_2) is increased in equal steps. Besides, proper assignment of weights and, hence, solution quality depend on knowledge of the solution space [6].

Deb proposed the "Non-dominated Sorting GA-II" or NSGA II [7], which is a true multi-objective GA. It uses the notion of crowding distance to ensure diversity among the solutions in a population. Initially a random seed is created. The cost of each objective for all the solutions is determined and they are classified into ranks based on nondomination. The Rank I individuals are fully non-dominated whereas those in Rank 2 are dominated by the Rank I individuals and so on. Each solution is assigned a fitness based on its rank and crowding distance. The crowding distance is a measure of the uniqueness of a solution. Crossover and mutation are performed on the individuals using the method described in [2]. The parents and offspring in a particular generation are merged and the individuals for the next generation are selected based on the crowding distance metric [6, 7]. Selection of individuals with higher crowding distance is favored for better diversity among solutions. The population size in a generation was 100. The GA was run for 200 generations. A flow diagram depicting the NSGA II methodology is shown in Figure 6. A detailed discussion of the results obtained for this method on standard DFG benchmarks is given in Section 8.

7. Weighted Sum Particle Swarm Optimization (WSPSO)

7.1. Introduction. Particle Swarm Optimization (PSO) is an evolutionary approach proposed by Eberhart and Kennedy, inspired by the behavior of bird flocks or schools of fish. PSO, like GA, operates on a population of candidate solutions referred to as a *swarm*. Each solution in a swarm is called a *particle*. The swarm is made to evolve by applying a *velocity* to each particle. The best solution in each swarm is called the "particle best" or *pbest* and the best solution among all the swarms so far is the "global best" or *gbest*. When the swarm moves, each particle is subjected to a velocity which tends to propel it in the direction of *pbest* as well *gbest* with each direction being assigned a weight as modeled by the equation given below:

$$v^i_{j,t+1} = wv^i_{j,t} + \eta_1 R_1\left(p^i_{j,t} - x^i_{j,t}\right) + \eta_2 R_2\left(g^i_{j,t} - x^i_{j,t}\right), \quad (3)$$

where $v^i_{j,t+1}$ = velocity of the ith particle for the $t+1$th iteration, and $v^i_{j,t}$ = velocity of the ith particle for the tth iteration.

The weight w assigns some *inertia* to the particle. The parameters η_1 and η_2 assign weights to the *pbest* and *gbest* variables, that is, the extent to which these positions influence the movement of the particle. The random values R_1 and R_2 introduce a degree of perturbation in the particle, for better exploration of the search space.

7.2. Weighted Sum PSO (WSPSO) for DFG Scheduling. Scheduling of DFGs during datapath synthesis is a good candidate for application of PSO. This problem is somewhat analogous to TSP [12, 16], albeit with precedence and module constraints. The multi-chromosome encoding introduced in Section 4 can be used for PSO also. In discrete problems, the notion of velocity is implemented by adapting the crossover technique [2, 16] for the PSO problem. A particle in the swarm is represented by a single multi-chromosome string representing a given DFG schedule and allocation. This particle is crossed over with the current particle that represents the *gbest* (or *pbest*) to implement the velocity function. The particle is crossed over with both the *gbest* and *pbest* using the procedure outlined in Section 4. This is illustrated in Figure 10.

The same approach is followed for *pbest* also. It can be seen that the degree of shift of the particle towards the *pbest* or *gbest* can be controlled by appropriately choosing the crossover location. If the location is in the beginning of the string then the shift is more.

Area, delay, and power cost of the schedule represented by the chromosome is computed using the method described in Section 5. The weighted sum approach outlined in Section 6 is used to determine the fitness of a particle represented by the multi-chromosome. A swarm size of 100 individuals was used. The total number of generations (swarms) was 70. The flowchart shown in Figure 7 summarizes the WSPSO based methodology used.

The PSO methodology developed was tested on various benchmarks. It was found to exhibit better runtimes than WSGA with comparable solution quality.

8. Results and Discussion

All the results reported below were obtained from executing the algorithms in an Intel i5-2400 CPU with a clock frequency of 3.10 GHz and 4 GB RAM.

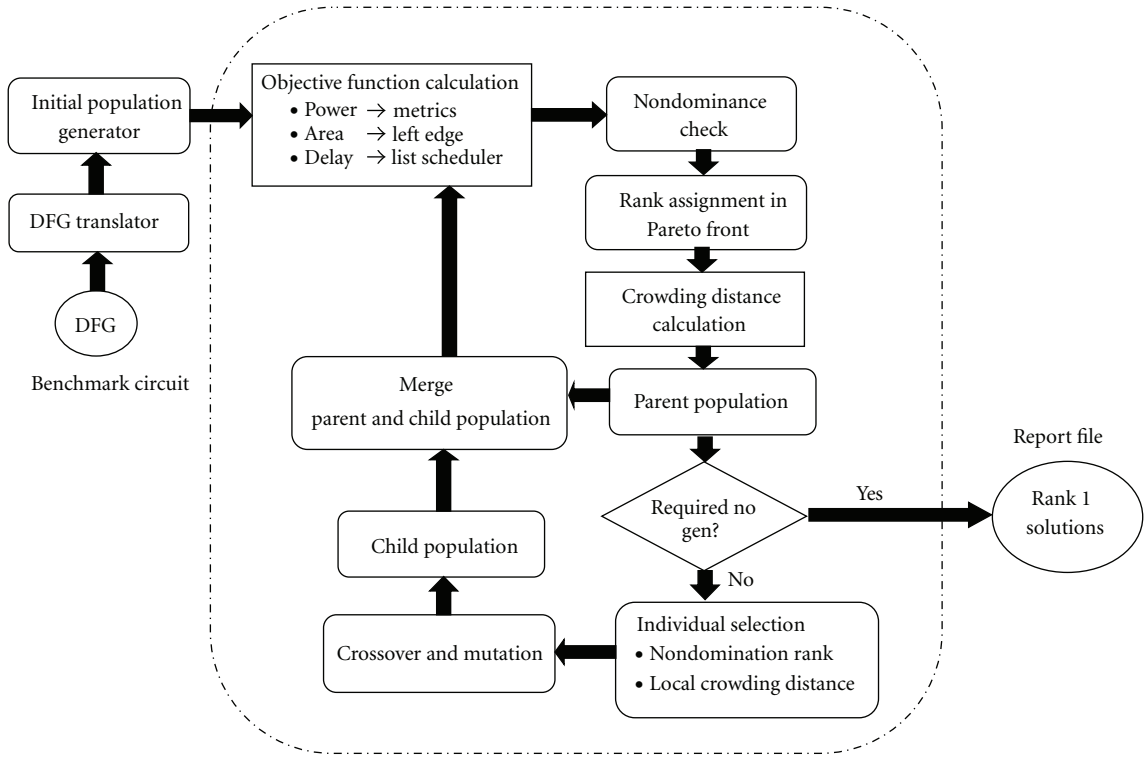


Figure 6: NSGA II based methodology.

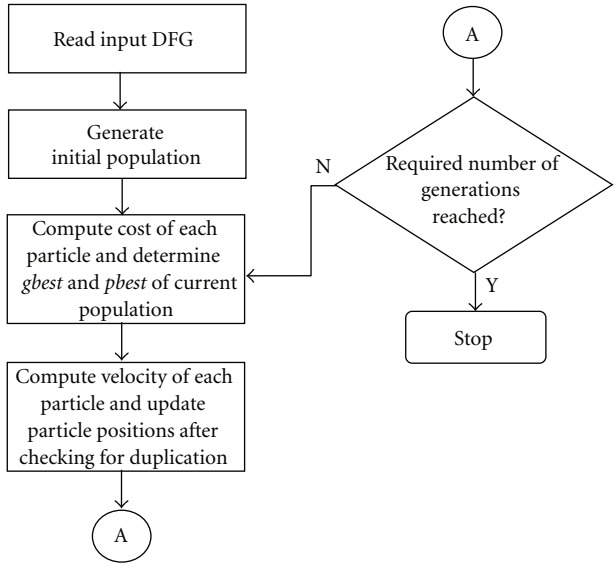
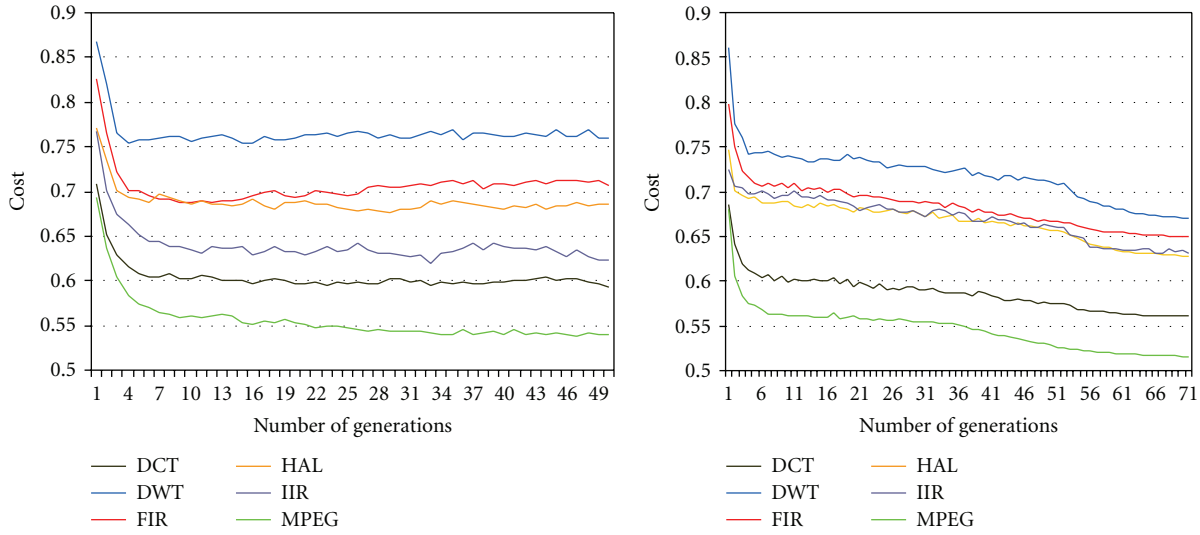


Figure 7: PSO methodology for DFG scheduling.

8.1. WSGA and WSPSO Convergence. The convergence of WSGA and WSPSO is verified by plotting the cost function with respect to the number of generations for each DFG benchmark (Figures 8(a) and 8(b)). For WSGA, the number of runs was 50 and for WSPSO the algorithms were run for 70 generations. The convergence plots for each benchmark are presented in Figures 8(a) and 8(b) respectively for WSGA and WSPSO. It is observed from the cost function values at convergence that the extent of optimization observed for all the benchmarks is comparable for both WSGA and WSPSO. This is further discussed in connection with the results in Table 6.

8.2. Comparison of Power Aware WSGA with GA without Power in the Cost Function [2]. One of the contributions of this work is the introduction of a power metric that indicates at an early stage the *likelihood* of a schedule to yield a low power binding solution during the binding phase. This metric is added to the weighted sum cost function for area and delay optimization proposed in [2]. The efficacy of the modified GA with power is verified by evaluating the modified WSGA on various benchmarks and comparing with the results of WSGA method reported in [2] which does not incorporate power in the cost function. The weight assigned to the power cost is 0.7 and the other objectives are assigned equal weights of 0.15 each.

It was observed that the modified GA methodology yielded schedules that had improved values of the power metric indicating higher likelihood of obtaining low power bindings. The results of the comparison are listed in Table 1. An average reduction of around 9% is observed in the power metric for the power aware GA run. The reduction is not significant for the IIR benchmark which has only 9 nodes. Hence the search space of feasible schedules is limited. The HAL benchmark though having only 10 nodes yields a better power number since it has higher mobility for the nodes and hence more number of feasible schedules. The power



(a) WSGA convergence

(b) WSPSO convergence

Figure 8: Convergence plot for WSGA and WSPSO.

Table 1: Comparison of WSGA and power-aware WSGA.

Benchmark	WSGA without power optimization [2]			Power-aware WSGA			Reduction in power cost metric
	Power metric	Area (register units)	Delay (time steps)	Power metric	Area (register units)	Delay (time steps)	
IIR	0.9062	42.9	5.44	0.8996	48.48	5.15	0.73%
HAL	0.9911	38.33	4.74	0.8719	37.81	4.98	12%
FIR	0.6487	42.73	11.63	0.6029	54.48	11.32	7%
MPEG	0.6317	81.5	8.50	0.6046	107.5	9.00	17%

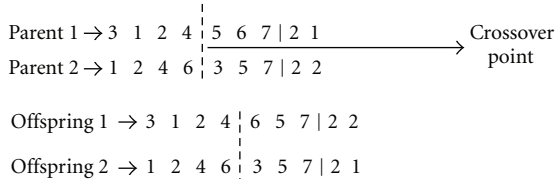


Figure 9

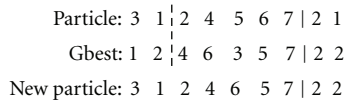


Figure 10

aware WSGA run yields better power optimal solutions for the FIR and MPEG benchmarks which have 23 and 28 nodes, respectively, obviously due to the higher number of nodes and also more degrees of freedom in moving the nodes for generating different schedules.

Table 2: Metrics evaluating closeness to the Pareto optimal front for IIR benchmark for NSGA II.

Performance metric	WSGA	NSGA II
Error ratio	0.7143	0
Generational distance	0.02708	0
Maximum Pareto optimal front error (MFE)	0.1833	0
Spacing	0.1046	0
Spread	1.0026	0
Weighted metric	0.5148	0

Table 3: Metrics evaluating closeness to the Pareto optimal front for HAL benchmark for NSGA II.

Performance metric	WSGA	NSGA II
Error ratio	0.7826	0
Generational distance	0.0050	0
Maximum Pareto optimal front error	0.1832	0
Spacing	0.0516	0
Spread	1.8600	0
Weighted metric	0.9360	0

8.3. NSGA II Results and Analysis. The quality of solutions obtained using a multi-objective algorithm must be assessed

Table 4: Comparison of WSGA and NSGA II on standard benchmarks.

Benchmark	WSGA			NSGA II		
	Power metric	Area (register units)	Delay (time steps)	Power metric	Area (register units)	Delay (time steps)
IIR	0.9019	48.48	5.148	0.9063	35.41	5.88
HAL	0.9901	37.81	4.981	0.9923	39.91	4.75
DWT	0.7126	50.92	10	0.6821	48.95	10
FIR	0.6416	54.48	11.32	0.6312	43.58	11.23
MPEG motion vector	0.6714	107.5	9	0.6561	63.13	10.71
DCT	0.6086	79.06	12.18	0.6087	77.44	12.12

Table 5: Execution times in seconds.

Benchmark circuit	WSGA	NSGA	% reduction
IIR	6	4	33.33%
HAL	11	11	0%
DWT	26	5	80.77%
FIR	112	9	91.96%
MPEG motion vector	48	23	52.08%

in terms of the spread of the solutions and the closeness of the individuals to the true Pareto front. The metrics described in [6] for diversity, spread and Pareto front errors were computed for the Rank I individuals obtained for the IIR filter and HAL benchmarks [2, 5] using WSGA and NSGA II. For computing the quality metrics, the true Pareto front was obtained for these two benchmarks by exhaustive search. The metric values obtained for the WSGA and NSGA runs are listed in Tables 2 and 3. The *error ratio* is the fraction of solutions in the population of Rank I individuals, which are not members of the true Pareto-optimal set, P^*. The *generational distance* metric gives a measure of the average distance of the solutions from P^*. The *maximum Pareto front error* gives the distance of the solution which is farthest from the Pareto front. The values of the metrics, *spacing* and *spread* indicate the diversity of solutions obtained. The *weighted metric* is the weighted sum of the closeness and diversity measures. These metrics should have small values for a good multi-objective algorithm. The results of comparing NSGA II and WSGA in terms of the quality metrics are given in Tables 2 and 3, respectively, for the IIR and HAL benchmarks.

The above metrics indicate the superior quality of the NSGA II solutions over those obtained by the Weighted Sum GA approach. All metrics have zero values for NSGA indicating that the solutions are *fully coincident* with the true Pareto front individuals.

The NSGA II methodology was evaluated on several standard DFG benchmarks and the results are summarized in Tables 4 and 5. Table 4 shows the average values of the delay, area, and power metrics obtained for the WSGA and NSGA II runs. Power dissipation is quantified in terms of the potential of the schedule to yield low power bindings as described in Section 4. Area is computed as the total gate count of the FUs and registers when synthesized to a generic library and is normalized to the register gate count as register units. Delay is expressed in terms of the total time steps required for completing each schedule. The average values for the NSGA II run are either better than WSGA or comparable in most of the cases. For the smaller DFGs (IIR and HAL), the NSGA II results do not exhibit an appreciable improvement in the *average* values. This is because of the fewer number of feasible solutions. Since NSGA searches the solution space more efficiently the effect of trade-offs between the three objectives is more pronounced. Another aberration that is noticeable is in the case of the MPEG motion vector benchmark where a substantial reduction in average area is noticed at the expense of higher average delay. This DFG is rich in multiplication nodes whose implementation is area intensive. Hence an effective search of the solution space will yield solutions that exhibit tremendous trade-offs between delay and area. Thus a considerable reduction in average area is observed at the expense of delay. Thus it can be concluded that NSGA II, being a true Multi-Objective GA, is highly effective in more intensive exploration of the design space. This is evident from the quality metrics as well as the results on the various DFG benchmarks.

Table 5 lists out the comparison of the execution times. The NSGA II algorithm shows appreciable improvement over WSGA with an average reduction of 51.63% in execution time. As expected, the improvement is more noticeable in the case of the larger benchmarks which necessitate exploration of a much larger search space. The search space is multidimensional and nonlinear and hence increase in number of DFG nodes will add exponentially large complexity to the search process.

8.4. WSPSO—Results and Analysis. The WSPSO-based weighted sum approach outlined in Section 7 was evaluated on the DFG benchmarks and compared with WSGA. A particle size of 100 was used and the WSPSO was run for 70 generations. The average power, area, and delay values of 10 runs of the algorithm with different initial seed populations are tabulated for various DFG benchmarks. The quality of the solutions was also assessed against the results from exhaustive search. The results are shown in Tables 6 and 7. Initial results are comparable to the corresponding results for WSGA. However, the quality and run times can be improved upon optimizing the swarm size and introducing additional operators in the velocity function used. Further investigations need to be carried out in this direction.

Tables 6 and 7 present the comparison of WSGA and WSPSO on standard DFG benchmarks. Whereas Table 6

Table 6: Comparison of WSGA and WSPSO on DFG benchmarks.

Benchmark	WSGA				WSPSO				Reduction in execution times%
	Power	Area (register units)	Delay (time steps)	Execution time (seconds)	Power	Area (register units)	Delay (time steps)	Execution time (seconds)	
IIR	0.9019	48.48	5.148	6	0.9083	35.93	5.86	4	33.33%
HAL	0.9901	37.81	4.981	11	0.9960	33.75	4.98	8	27.27%
DWT	0.7126	50.92	10	26	0.7242	26.57	10.11	14	46.15%
FIR	0.6416	54.48	11.32	112	0.6595	44.68	10.59	97	13.39%
MPEG motion vector	0.6714	107.5	9	48	0.6984	95.30	7.69	34	29.17%
DCT	0.6086	79.06	12.18	1528	0.6120	75.98	11.30	1408	7.85%

Table 7: Metrics evaluating closeness to the Pareto optimal front for IIR benchmark for WSGA and WSPSO.

Performance metrics	WSGA	WSPSO
Error ratio	1	1
Generational distance	2.09	3.02
Maximum pareto optimal front error (MFE)	81	43.02
Spacing	2.73	0
Spread	1.56	1
Weighted metric	1.82	2.04

compares average values of power, area, and delay and execution times, Table 7 lists out the performance metrics [6] that indicate the quality of the solutions obtained in terms of diversity and closeness to the true Pareto front. This is assessed by generating the true Pareto front using exhaustive search and comparing the Rank I solutions obtained from the algorithm with the true Pareto front as described in Section 8.3. Low values for each metric indicate a higher degree of diversity and closeness to the Pareto front.

From Table 6, it can be observed that the results for WSPSO are comparable with WSGA for the area, delay, and power values whereas WSPSO exhibits an average improvement of 26.19% in the execution times. Thus there is scope for improvement in the WSPSO methodology by enhancing the search method without an adverse impact on execution times. However WSPSO, as in the case of WSGA, fares poorer than NSGA II in terms of solution spread, diversity, closeness to Pareto front, and execution times.

Average measurements are not indicative of the ability of the algorithm to search the entire multi-objective solution space. The diversity of solutions and optimality in terms of closeness to the true Pareto front are also indicators of algorithm efficiency. The performance metrics [6] described in Section 8.2 evaluated for WSGA as well as WSPSO are listed in Table 7. The error ratio is the same for both algorithms. WSPSO fares slightly better in yielding uniformly spaced solutions with better diversity as evidenced by the lower values of *spacing* and *spread* for WSPSO. However, the *generational distance* metric is higher for WSPSO. Thus a higher *generational distance* metric coupled with a lower *MFE* indicates that Rank I individuals have more or less uniform separation from the true Pareto front. This is corroborated by the lower values for spacing and spread mentioned earlier.

9. Conclusions and Future Work

A framework for applying multi-objective evolutionary techniques for multi-objective power, area, and delay optimization for the datapath scheduling problem in High Level Synthesis is presented. The methodology has been applied to the NSGA II algorithm and considerable improvement in runtimes and solution quality is demonstrated compared to a Weighted Sum GA approach by evaluating the technique on standard DFG benchmarks. Thus the technique will be effective as a tool for design space exploration during DFG synthesis. The methodology has been extended to Weighted Sum PSO and preliminary results indicate that WSPSO is faster than WSGA with potential for improvement in solution convergence and quality. The following conclusions can be drawn from the analysis of the WSGA, NSGA II, and WSPSO algorithms.

(i) NSGA II exhibits the best solution quality among all the techniques analyzed and fastest execution times and emerges as the best approach among the three methods compared from the point of view of efficient design space exploration. This is in line with expectations since NSGA II incorporates features for ensuring diversity and convergence to the Pareto front. The method for identifying non-dominated solutions is computationally fast thus resulting in the reduced runtime.

(ii) WSPSO is computationally simple and exhibits lesser run time compared to WSGA since the evolution process does not involve crossover and mutation which are computation intensive.

(iii) However, WSPSO may not be an alternative to NSGA II unless the solution quality is improved by incorporating some hybrid approaches like interweaving with Simulated Annealing as reported in [16]. This aspect needs to be investigated and will be part of the future work.

(iv) True Multi-objective PSO (MOPSO) techniques have been proposed such as the works reported in [9, 11]. This will be adapted for the HLS scheduling problem and the performance vis-à-vis NSGA II will be studied.

(v) The solutions obtained by WSPSO, WSGA, and NSGA II will be synthesized to a target library and the trends predicted by the DFG metrics used in the cost functions with actual estimates from postbinding synthesis results will be compared.

References

[1] D. S. Harish Ram, M. C. Bhuvaneswari, and S. M. Logesh, "A novel evolutionary technique for multi-objective power, area and delay optimization in high level synthesis of datapaths ," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '11)*, pp. 290–295, Chennai, India, July 2011.

[2] V. Krishnan and S. Katkoori, "A genetic algorithm for the design space exploration of datapaths during high-level synthesis," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 213–229, 2006.

[3] F. Ferrandi, P. L. Lanzi, D. Loiacono, C. Pilato, and D. Sciuto, "A multi-objective genetic algorithm for design space exploration in high-level synthesis," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '08)*, pp. 417–422, Montpellier, France, April 2008.

[4] E. Kursun, R. Mukherjee, and S. Ogrenci Memik, "Early quality assessment for low power behavioral synthesis," *Journal of Low Power Electronics*, vol. 1, no. 3, pp. 1–13, 2005.

[5] Mediabench benchmark, express.ece.ucsb.edu/benchmark/.

[6] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley and Sons, 2003.

[7] K. Deb, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[8] M. Reyes-Sierra and C. A. Coello Coello, "Multi-objective particle swarm optimizers: a survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[9] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—a comparative case study," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pp. 292–304, Springer, London, UK, 1998.

[10] M. S. Bright and T. Arslan, "Multi-objective design strategy for high-level low power design of DSP systems," in *Proceedings of the ISCAS '99*, June 1999.

[11] N. Padhye, J. Branke, and S. Mostaghim, "Comprehensive comparison of MOPSO methods: study of convergence and diversity—survey of state of the art," Tech. Rep., CEC, 2009.

[12] K. P. Wang, L. Huang, C. G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the 2nd International Conference on Machine Learning and Cybernetics*, November 2003.

[13] D. Lin, S. Qiu, and D. Wang, "Particle swarm optimization based on neighborhood encoding for traveling salesman problem," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, October 2008.

[14] J. Jie, H. Ji, M. Wang, and M. Zhao, "Improved discrete particle swarm optimization based on edge coding and multilevel reduction strategy for larger scale TSP," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '2010)*, August 2010.

[15] R. F. Abdel-Kader, "Particle swarm optimization for constrained instruction scheduling," *VLSI Design*, vol. 2008, Article ID 930610, 7 pages, 2008.

[16] L. Fang, P. Chen, and S. Liu, "Particle swarm optimization with simulated annealing for TSP," in *Proceedings of the 6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, Corfu Island, Greece, February 2007.

[17] S. H. Gerez, *Algorithms for VLSI Design Automation*, John Wiley and Sons, 2000.

[18] C. G. Lyuh and T. Kim, "High-level synthesis for low power based on network flow method," *IEEE Transactions on VLSI Systems*, vol. 11, no. 3, pp. 364–375, 2003.

[19] A. Davoodi and A. Srivastava, "Effective techniques for the generalized low-power binding problem," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 1, pp. 52–69, 2006.

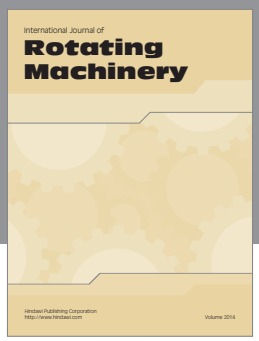
[20] X. Tang, T. Jiang, A. Jones, and P. Banerjee, "Behavioral synthesis of data-dominated circuits for minimal energy implementation," in *Proceedings of the International Conference on VLSI Design: Power Aware Design of VLSI Systems*, January 2005.

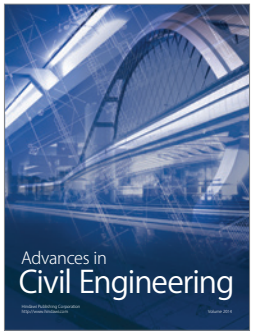
[21] N. Chabini and W. Wolf, "Unification of scheduling, binding, and retiming to reduce power consumption under timings and resources constraints," *IEEE Transactions on VLSI Systems*, vol. 13, no. 10, pp. 1113–1126, 2005.

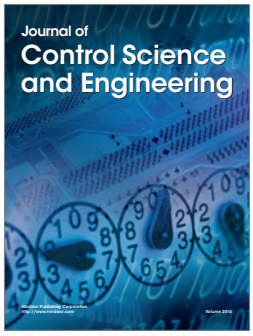
[22] A. K. Murugavel and N. Ranganathan, "A game theoretic approach for power optimization during behavioral synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1031–1043, 2003.

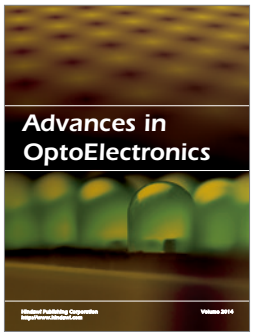
[23] C. Mandal, P. P. Chakrabarti, and S. Ghose, "GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 6, pp. 747–750, 2000.

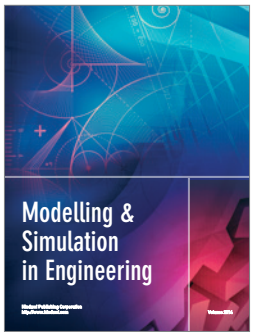
[24] J.-M. Chang and M. Pedram, "Register allocation and binding for low power," in *Proceedings of the Design Automation Conference (DAC '95)*, June 1995.

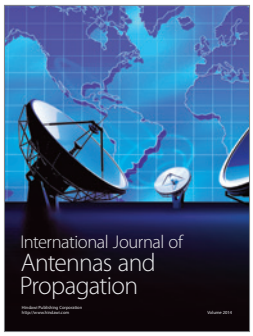


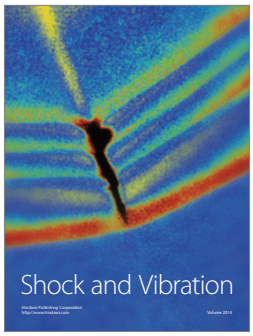


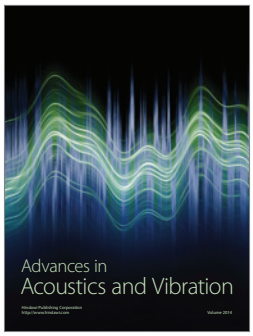












Hindawi

Submit your manuscripts at
http://www.hindawi.com