

# Optimal Resource Allocation and Binding in High-Level Synthesis Using Nature-Inspired Computation



K. C. Shilpa, C. LakshmiNarayana and Manoj Kumar Singh

**Abstract** Allocation of resource and binding it to functional unit at high-level synthesis is an optimal problem to minimize the area and performance in terms of resource sharing and binding is presented in this paper. The paper presents the comparative analysis of nature-inspired computation techniques for resource allocation and binding: 1. Evolutionary-based computation: genetic algorithm. 2. Swarm intelligence-based computation: particle swarm optimization. The comparative analysis of the results shows genetic algorithm surpasses particle swarm optimization in providing the precise mapping between the operation and functional unit sharing with zero errors in resource allocation.

**Keywords** Data flow graph · Genetic algorithm · High-level synthesis · Integer linear programming · Particle swarm optimization · Register-transfer level

## 1 Introduction

High-level synthesis means synthesizing register-transfer level (RTL) formation from the functional explanation. The two distinct tasks in high-level synthesis are scheduling and allocation [1]. Scheduling task describes the distinct start time for every process in the data flow graph (DFG). Scheduling gives the resource usage estimates. Allocation task ensures that sufficient numbers of resources are available for executing the operation.

---

K. C. Shilpa (✉)

Dr. Ambedkar Institute of Technology, Bangalore, India

e-mail: [shilpa.kc2@gmail.com](mailto:shilpa.kc2@gmail.com)

C. LakshmiNarayana

Department of Electrical Engineering Science, BMSCE, Bangalore, India

e-mail: [ln\\_gp@yahoo.co.in](mailto:ln_gp@yahoo.co.in)

M. K. Singh

Manuro Tech Research, Bangalore, India

e-mail: [mksingh@manuroresearch.com](mailto:mksingh@manuroresearch.com)

© Springer Nature Singapore Pte Ltd. 2019

V. Sridhar et al. (eds.), *Emerging Research in Electronics, Computer Science and Technology*, Lecture Notes in Electrical Engineering 545,

[https://doi.org/10.1007/978-981-13-5802-9\\_95](https://doi.org/10.1007/978-981-13-5802-9_95)

The two steps in allocation are resource sharing and resource binding [2]. Resource sharing allows to control on the use of multiple hardware resources for implementing the operation. Resource binding maps between behavioral operations and the resources instances. Resource sharing and binding are NP (nondeterministic polynomial time) complete problem [3], which performs an exhaustive search in finding out the best answer.

## 2 Previous Work

Many novel techniques for resource binding and sharing are reported [4]. A clique partition algorithm for resource sharing and the coloring algorithm are the best method for resource sharing [5]. The ILP formulation [6, 7] for concurrent scheduling and binding provides successful solution of ILP problems for circuits of interesting size. Nature-inspired algorithms act as an optimized technique in solving the complex problem which is flexible in nature. Genetic algorithm is an optimization tool to solve high-complexity computation problems which are based on principles of Charles Darwin. Particle swarm optimization is swarm intelligence computation method-based stochastic algorithm.

Evolutionary-based search techniques are best to solve NP-Complete problem effectively.

## 3 Nature-Inspired Computations Method (Genetic Algorithm (GA), Particle Swarm Optimization (PSO))

Nature-inspired computations (NIC) are a method that is motivated by process focused from natural world. These computing methods led to the growth of working of algorithms so-called nature-inspired computation. The algorithms mainly focused toward the nature computational intelligence.

Nature-inspired computation algorithm is mainly categorized as follows:

Evolutionary Computation (EC): Evolutionary computation is a term used to illustrate an algorithm which was encouraged by ‘survival of the fittest’ or ‘normal selection’ principles proposed by Charles Darwin.

Swarm Intelligence (SI): Swarm intelligence is a phrase used to explain the algorithms and distributed problems-solvers mainly inspired by the cooperative cluster intelligence of swarm.

### 3.1 Genetic Algorithm (GA)

Genetic algorithm (GA) developed by John Holland [8]. The process of genetic algorithm is a search method used to compute to discover accurate or estimated solutions to optimization and search problems based on the rule of regular selection. Genetic algorithm is categorized as universal explore heuristics. Genetic algorithm [9] which is the group of evolutionary algorithms tools method motivated by evolutionary natural science such as inheritance, mutation, selection, and crossover.

### 3.2 Particle Swarm Optimization (PSO)

Eberhart and Shi [10] developed swarm intelligence method called particle swarm optimization, which emerges as a powerful stochastic optimization technique motivated by the communal activities of organisms such flocks of birds, schools of fish, or swarms of bees, and even human communal behavior, from which the behavior is emerged. Population-based search procedure in which individuals so called particles change their position (state) with time. The standard rule is particle swarm [11] move about toward the best position in explore space, identification based on each particle's best known position and global best known position.

## 4 Methodology

### 4.1 Problem Formulation

The objective of the allocation work aims to locate the resources in finding the appropriate correct number and type of resource sharing of the multiplier resources and ALU resources for each operation so that the design constraint is met.

### 4.2 Benchmark Problem for Resource Allocation

To illustrate the allocation problem, the scheduled sequencing time constraint graph displayed in Fig. 1, the problem based on hardware abstraction layer (HAL). Implementation by resource with type ALU is considered for the operation of type adders, subtractor, and comparator.

The scheduled graph from Fig. 1 indicates the operation for multiplier resources  $\{(o_1), (o_2)\}$  is listed in the first time step, operation for multiplier resources  $\{(o_3), (o_6)\}$  is listed in second time step, operation for multiplier resources  $\{(o_7), (o_8)\}$  scheduled in third time step. Adders, subtractor, and comparator are

generalized as the resource-type ALU. The operation for ALU resources ( $o_{10}$ ) is scheduled in the initial control time step, operation for ALU resources ( $o_{11}$ ) scheduled in second control time step, operation for ALU resources ( $o_4$ ) scheduled in third time control step and operation for ALU resources  $\{o_5, o_9\}$  scheduled in fourth time control step. The obtained scheduled result under time constraint scheduling requires two multiplier operator units and two ALU operator units to fulfill the defined schedule graph.

### 4.3 Integer Linear Programming (ILP) Formulation for Allocation and Binding as a Constraint Optimization

To illustrate the allocation problem, the ILP formulation for operation binding as constraints optimization, for the scheduled data flow graph in Fig. 1 is as follows:

ILP model is a set of binary decision variables with two indices ‘B’ = {‘ $b_{ir}$ ’;  $i = 0, 1, \dots, n$ ; ‘ $r = 1, 2, \dots, a$ }, ‘ $n$ ’ = number of operations, ‘ $a$ ’ = number of resources. {‘ $b_{ir} = 1$ ’} implies the operation ‘ $o_i$ ’ in constraint graph bound to resource ‘ $r$ ’,  $a \leq n$  is an high bound on the numeral of resources to be set. Binary decision constants ‘X’ = {‘ $x_{il}$ ’;  $i = 0, 1, \dots, n$ ;  $l = 1, 2, \dots, \lambda + 1$ , where {‘ $x_{il} = 1$ ’} implies operation  $o_i$  start in the control step ‘ $l$ ’ of the schedule, from the schedule ‘ $l = t_i$ ’.

The different ILP formulation constraints are:

- To obtain a binding, search a set of values of ‘B’, to allocate behavior operation to the resources such that the set of following constraints are met.

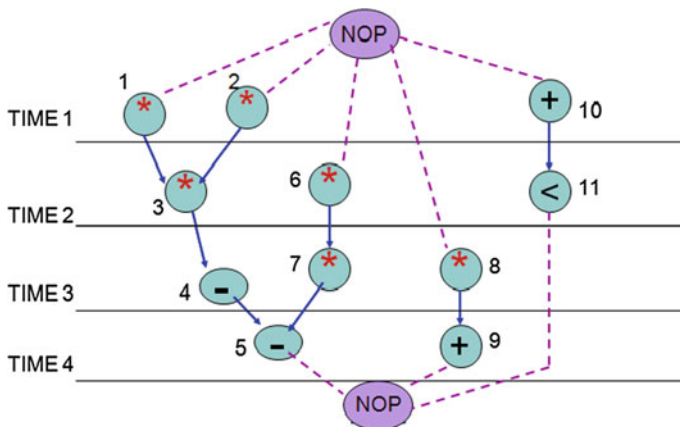


Fig. 1 Scheduled sequencing time constraint HAL benchmark problem

$$\sum_{r=1}^a b_{ir} = 1, \quad i = 1, 2, \dots, n \tag{1}$$

Equation (1) states every operation ‘ $o_i$ ’ should assign to exactly one as well as only one resource of type ‘ $a$ ’.

- The resource allocation for Eq. (1) for ‘resource 1’ and ‘resource 2’ of the multiplier is summation of each operational units of multiplier by the ‘resource 1’ and ‘resource 2’ given below as follows:

$$\begin{aligned} b(1, 1) + b(1, 2) &= 1 \\ b(2, 1) + b(2, 2) &= 1 \\ b(3, 1) + b(3, 2) &= 1 \\ b(6, 1) + b(6, 2) &= 1 \\ b(7, 1) + b(7, 2) &= 1 \\ b(8, 1) + b(8, 2) &= 1 \end{aligned}$$

- The resource allocation for Eq. (1) for ‘resource 1’ and ‘resource 2’ of the ALU is summation of each operational units of ALU by the ‘resource 1’ and ‘resource 2’ given below as follows:

$$\begin{aligned} b(10, 1) + b(10, 2) &= 1 \\ b(11, 1) + b(11, 2) &= 1 \\ b(4, 1) + b(4, 2) &= 1 \\ b(5, 1) + b(5, 2) &= 1 \\ b(9, 1) + b(9, 2) &= 1 \end{aligned}$$

- Equation (2) states at each control step, one operation among resource  $r$  can be executed among those allocated.

$$\sum_{i=1}^n b_{ir} * \sum_{m=1-d_i+1}^1 X_{im} \leq 1, \quad i = 1, 2, \dots, \lambda + 1, \quad r = 1, 2, \dots, a \tag{2}$$

$$b_{ir} \in \{0, 1\}, \quad i = 0, 1 \dots n; \quad r = 1, 2, \dots, a \tag{3}$$

- Equation (3) constraint states decision variable ‘ $B$ ’ are binate, either take (0, 1).

The resource allocation for Eq. (2) describes the summation of each multiplier operation by ‘resource 1’ with the product of each multiplier operation at scheduled time step is as follows:

$$b(1, 1) * x(1, 1) + b(2, 1) * x(2, 1) \leq 1$$

$$b(3, 1) * x(3, 2) + b(6, 1) * x(6, 1) \leq 1$$

$$b(7, 1) * x(7, 3) + b(8, 1) * x(8, 3) \leq 1$$

- The resource allocation for Eq. (2) by ‘resource 2’ of multiplier is as follows:

$$b(1, 2) * x(1, 1) + b(2, 2) * x(2, 1) \leq 1$$

$$b(3, 1) * x(3, 2) + b(6, 2) * x(6, 2) \leq 1$$

$$b(7, 1) * x(7, 3) + b(8, 2) * x(8, 3) \leq 1$$

- The resource allocation for Eq. (2) by ‘resource 1’ of ALU is as follows:

$$b(5, 1) * x(5, 4) + b(9, 1) * x(9, 4) \leq 1$$

- The resource allocation for Eq. (2) by ‘resource 2’ of ALU is as follows:

$$b(5, 2) * x(5, 4) + b(9, 2) * x(9, 4) \leq 1$$

## 5 Experimental Analysis and Data

To solve ILP formulation of operation binding as constraints optimization, the nature-inspired computations evolutionary computation genetic algorithm method, and swarm intelligence method are considered to work out the inequalities constraint value. There are ‘12’ different variables for multiplier constraint equation, ‘10’ different variables for ALU constraint equation. Thus, there are total ‘22’ variables which have to be solved.

The allocation optimization specifications are as follows: Integer linear programming can be formulated for the objective function given in Eq. (4) as follows:

$$\text{Minimize } f; \text{ ‘} f \text{’} = \text{resources sharing} \quad (4)$$

$$\text{‘} f \text{’} = \text{sum(abs(‘rs’))} \quad (5)$$

$$\text{‘rs’} = [r \ r2]; \text{ ‘} r \text{’} = \text{multiplier constraint equation,}$$

$$\text{‘} r2 \text{’} = \text{ALU constraint equation}$$

- The algorithm is experienced with regular random numeral for population range (N) = 50.
- Dimension of the search space (D) = 22.

- Genetic Algorithm parameter setup: the crossover probability which is two-point factor = '1', mutation probability factor value = 0.01. Selection method tournament is used.
- Particle Swarm Optimization (PSO) parameter setup: Constriction factor ( $cf$ ) value = 0.72, learning factor ( $c1, c2$ ) value of 2.5, inertia weight ' $w$ ' value is reducing from 1.2 to 0.1.

Allocation algorithm problem is solved using MATLAB.

## 6 Results and Discussion

### 6.1 Resource Binding Performance Analysis Using Genetic Algorithm

The 12 variables multiplier allocation results using GA are presented in Table 1. The 10 variables ALU allocation results are been listed as follows in Table 2.

**Table 1** Results of genetic algorithms performance for multiplier allocation and binding

Allocation variables	Binding multiplier for resource type	Resource type
1	$b(1, 1)$	1
2	$b(1, 2)$	0
3	$b(2, 1)$	0
4	$b(2, 2)$	1
5	$b(3, 1)$	1
6	$b(3, 2)$	0
7	$b(6, 1)$	0
8	$b(6, 2)$	1
9	$b(7, 1)$	1
10	$b(7, 2)$	0
11	$b(8, 1)$	0
12	$b(8, 2)$	1

**Table 2** Results of genetic algorithms performance for ALU allocation and binding

Allocation variables	Binding ALU resource type	Resource type
1	$b(10, 1)$	1
2	$b(10, 2)$	0
3	$b(11, 1)$	1
4	$b(11, 2)$	0
5	$b(4, 1)$	1
6	$b(4, 2)$	0
7	$b(5, 1)$	0
8	$b(5, 2)$	1
9	$b(9, 1)$	1
10	$b(9, 2)$	0

## 6.2 Discussion

In Table 1,  $b(1, 1) = b(3, 1) = b(7, 1) = 1$ ; indicates operation for ( $o_1, o_3, o_7$ ) done by ‘resource 1’ and results  $b(2, 2) = b(6, 2) = b(8, 2) = 1$ ; indicates ( $o_2, o_6, o_8$ ) operation done by ‘resource 2’.

The ALU allocation result is tabulated in Table 2. The 10 variables results for ALU resources are presented in Table 2. In Table 2,  $b(10, 1) = b(11, 1) = b(4, 1) = b(9, 1) = 1$ ; indicates operation for ( $o_{10}, o_{11}, o_4, o_9$ ) done by ‘resource 1’ of ALU and  $b(5, 2) = 1$ ; indicate operation ( $o_5$ ) done by ‘resource 2’ of ALU. In Fig. 2, the error in allocation using genetic algorithm is zero; the objective function given in Eq. (4) is minimized to zero which is the obtained optimal value.

The optimal scheduled and resource-bounded sequence graph obtained by solving the ILP constraint using genetic algorithm is shown in Fig. 3. Figure 3 describes multiplier ‘resource 1’ sharing is done for operation ‘ $o_1$ ’, ‘ $o_3$ ’, ‘ $o$ ’ and indicated as (1, 1). ((1, 1) indicate multiplier operations done by multiplier ‘resource 1’).

The multiplier ‘resource 2’ sharing is done for operation ‘ $o$ ’, ‘ $o_6$ ’, ‘ $o_8$ ’ and is indicated as (1, 2); ((1, 2) indicate multiplier operations done by multiplier ‘resource 2’).

ALU ‘resource 1’ sharing is done for operation ‘ $o_{10}$ ’, ‘ $o_{11}$ ’, ‘ $o_4$ ’, ‘ $o_9$ ’, and is indicated as (2, 1); ((2, 1) indicate ALU operations done by ALU ‘resource 1’). The ALU ‘resource 2’ sharing done for  $o_5$  and is indicated as (2, 2); ((2, 2) indicate ALU operations done by ALU ‘resource 2’). Hence, the resource sharing and binding are done simultaneously for a solution under a larger set of constraints.



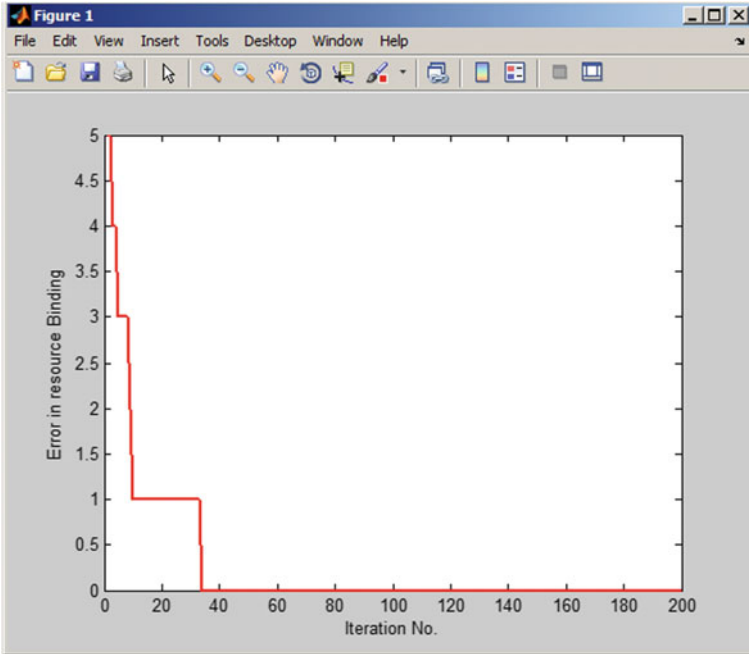


Fig. 2 Error in resource binding analysis for genetic algorithm

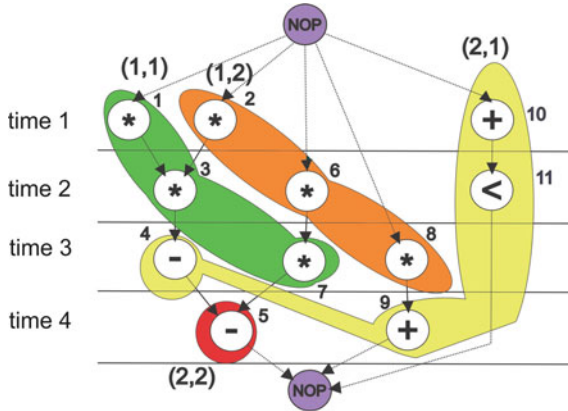


Fig. 3 Scheduled and resource-bounded graph obtained from GA

### 6.3 Resource Binding Performance Analysis Using Particle Swarm Optimization

The performances of multiplier allocation using PSO are shown in Table 3 and ALU allocation using particle swarm optimization is shown in Table 4.

### 6.4 Discussion

The multiplier allocation results are tabulated in Table 3. The 12 variables results allocating multiplier resources are presented in Table 3. Table 3 result  $b(2, 1) = b(7, 1) = 1$ ; indicates  $(o_2, o_7)$  operation done by ‘resource 1’, and  $(o_6, o_8, o_1)$  operation have done by ‘resource 2’ and PSO fails to allocate resources sharing for operation  $(o_3)$ .

The ALU allocation result using PSO is tabulated in Table 4. The 10 variables results for ALU resources are presented in Table 4. In Table 4,  $b(10, 1) = b(4, 1) = 1$ ; indicates operation for  $(o_{10}, o_4)$  done by ‘resource 1’ of ALU and  $b(5, 2) = b(11, 2) = 1$ ; indicates operation  $(o_5, o_{11})$  done by ‘resource 2’ of ALU, but PSO fails to deliver the results for resources sharing for the operation  $(o_9)$ .

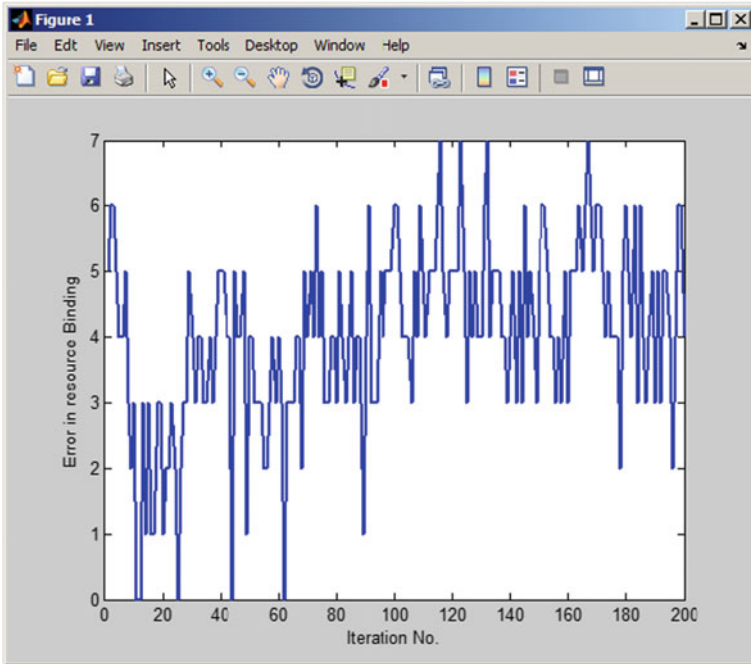
The objective function is minimized to the value four as shown in Fig. 4. The error in allocation using particle swarm optimization is four; PSO fails to minimize the objective function to the optimal value of zero; and struck at local minima as shown in Fig. 4.

**Table 3** Particle swarm performance for multiplier allocation

Allocation variables	Binding multiplier for resource type	Resource type
1	$b(1, 1)$	0
2	$b(1, 2)$	1
3	$b(2, 1)$	1
4	$b(2, 2)$	0
5	$b(3, 1)$	0
6	$b(3, 2)$	0
7	$b(6, 1)$	0
8	$b(6, 2)$	1
9	$b(7, 1)$	1
10	$b(7, 2)$	0
11	$b(8, 1)$	0
12	$b(8, 2)$	1

**Table 4** Particle swarm performance for ALU allocation

Allocation variables	Binding ALU for resource type	Resource type
1	b(10, 1)	1
2	b(10, 2)	0
3	b(11, 1)	0
4	b(11, 2)	1
5	b(4, 1)	1
6	b(4, 2)	0
7	b(5, 1)	0
8	b(5, 2)	1
9	b(9, 1)	1
10	b(9, 2)	1



**Fig. 4** Error in resource binding analysis for particle swarm optimization

## 7 Conclusion

Comparative analysis of resource allocation and binding of multiplier and ALU functional units is obtained by means of genetic algorithm which is one of the evolutionary computation methods and swarm intelligence method.

Genetic algorithm, an evolutionary method to find optimal solution for resource allocation and binding, achieves minimum objective function; i.e., 'f' = 0; is successfully with zero error in allocation. Particle swarm optimization, a swarm intelligence method, poorly gets struck at local minima and fails to obtain minimum objective function. Error in allocation exists in PSO method. The comparison results from GA and PSO show, GA is foremost in determining the binding of allocation problem in with zero errors.

## References

1. Micheli GD (1994) Synthesis and optimization of digital circuits. McGraw-Hill, USA
2. Gajski D, Dutt ND, Wu A, Lin S (1992) High level synthesis: introduction to chip and system design. Kluwer Academic Publisher, USA
3. Ku D, DeMicheli G (1992) High level synthesis of ASICS under timing and synchronization constraints. Kluwer Academic Publishers, USA
4. Hafer L, Parker A (1992) Automated synthesis of digital hardware. *IEEE Trans Comput Aided Des Integr Circuits Syst* 31(2):365–370
5. Pangrle B, Gajski D (1987) A state synthesizer for intelligent silicon compilation. *IEEE Trans Comput Aided Des* 6(3):42–45
6. Devadas S, Newton AR (1989) Algorithms for allocation in data path synthesis. *IEEE Trans Comput Aided Des Integr Circuits Syst* 8(2):768–781
7. Thomas DE (1983) The automatic synthesis of digital systems. *IEEE Trans Comput Aided Des Integr Circuits* 69(10):1200–1211
8. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing Co, Reading, MA
9. Mandal C, Chakrabarti PP, Ghose S (2000) GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 8(6)
10. Eberhart RC, Shi Y (1998) Comparison between genetic algorithms and particle swarm optimization. In: Proceedings of seventh annual conference on evolutionary programming, pp 611–616
11. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings IEEE international conference on neural networks (Perth, Australia), Piscataway, IEEE Service Center