I J C T A, 9(20), 2016, pp. 123-136

Analysis of Different Selection Scheme for Scheduling in High-Level Synthesis

Atul Kumar Srivastava*, Himanshu Shishodia and Himanshu*****

ABSTRACT

Very Large Scale Integrated Circuit (VLSI) technology caters to densities of several million gates of random logic per chip. Chips of such complexity are very tedious, if not impossible, to design using the normal capture and simulate design methods. The industry has begun looking at the product development cycle comprehensively to reduce the design time and to have a competitive edge in the time-to-market race. Automation of the entire design process from conceptualization to silicon or describe & synthesize design methodology has become necessary. Finally, if synthesis algorithms are fine construed, design automation tools may out-perform average human designers in meeting most of the design constraints and requisites.

Keyword: High level synthesis, Scheduling, SAST, Genetic Algorithm

1. INTRODUCTION

Synthesis, is the process of interconnecting primitive components at a certain level of abstraction (target level) to perform a specification at high level of abstraction (source level). Synthesis, sometimes defined as design refinement, adds an additional piece of detail that provides information required for the next level of synthesis or for manufacturing the design. The high level synthesis (HLS) step takes an algorithmic or high level behaviour as input and give as output the register transfer level (RTL) behavior consisting of functional units, storage and interconnection units. The logic synthesis step inputs Boolean equations and generates a gate level design after doing logic optimizations of the input. The layout synthesis step takes the gate level specification and outputs the physical layout performing the gate level specifications.

1.1. High Level Synthesis

High-level synthesis (HLS), sometimes also called as C synthesis, is an automated design process that interprets an algorithmic detail of a desired behavior and forms digital hardware that realizes that behavior. A behavioral description is used as the initializing point for HLS. It specifies the behavior in terms of operations, assignment statements & controls the construct in a hardware description language (HDL) (e.g., VHDL [2] or Verilog [3]). The output from a high-level synthesizer comprises of two parts: a data path structure at the register-transfer level (RTL) and a specifics of the finite state machine to control the data path. At the RTL level, a data path is composed of three kind of components viz functional units (e.g, ALUs, multipliers, and shifters), storage units (e.g., registers and memory) & interconnection units for e.g., buses and multiplexors. The finite state machine specifies every set of micro-operations for the data path to be performed during every control step. In the first step of HLS, the behavioral description is designed into an internal representation.

* Department of Electronics and Communication Engineering Jaypee Institute of Information Technology, Noida-(U.P.), INDIA, *Email: atul.srivastava@jiit.ac.in*

** Department of Electronics and Communication Engineering Jaypee Institute of Information Technology, Noida-(U.P.), INDIA, *Email: himanshushishodia@gmail.com*

*** Department of Electronics and Communication Engineering Jaypee Institute of Information Technology, Noida-(U.P.), INDIA, *Email: himanshu243243@gmail.com*

1.2. Scheduling in High-Level Synthesis

The goal of high-level synthesis is to produce a structure of the digital system that satisfies the constraints. In order to have an efficient design, a system should perform operation scheduling and hardware allocation simultaneously [4]. However, due to complexity in terms of time, many systems perform them separately in chunks [5].

According to Gajski[1], that says "perhaps the most important step during the structure synthesis."

Two basic scheduling problems [1] are:

(PI) Time-Constrained Scheduling: Given that there is a constraint on the maximum number of time steps, find the cheapest schedule which best meets constraints.

(P2) Resource-Constrained Scheduling: Given the constraints on the resources provided, find the fastest scheduling which satisfies the constraints. Some basic scheduling techniques are ASAP (As Soon As Possible), ALAP (As Late As Possible), Force-Directed Scheduling, List-based Scheduling, etc. The technique used for scheduling is known as Structured Architectural Synthesis Technique (SAST)[7]. This tool uses the genetic algorithm as an application to solve the scheduling problem.

1.3. Structured Architecture Synthesis Tool (SAST)

A number of systems such as HAL[8], STAR[9], SAM[10] and GABIND[11] support the high-level synthesis of digital systems. Most of the current synthesis generate data paths with random interconnections between data path elements, which may lead to use of greater routing area during physical design. SAST on the other side supports the synthesis of structured data paths, specifically avoiding the random interconnections. This conserves the on-chip wiring resources.

SAST essentially accept as input, precedence constraints between the operations represented as a partial order, and outputs a schedule of operations & transfers, and a data path to perform the schedule. The generated data path is organized as architectural blocks (A-block)[7], and optional global memory blocks as shown in figure 1.

1.4. The Application of Genetic Algorithms to High-Level Synthesis

Genetic algorithms [6] [7] are probabilistic search algorithms which are inspired on the principle of "survival of the fittest" attributed to theory of evolution by Charles Darwin in The Origin of Species. Genetic algorithms maintain a collection of potential solutions, which evolve according to a measure reflecting the quality of solutions.

1.5. Selection Schemes used in Genetic Algorithms

Genetic Algorithms is a basic probabilistic optimization method based on the model of natural evolution. One important operator in these algorithms is the selection method. There are many selection methods [12-14] out of which following have been added to the tool and results are analyzed and compared:-

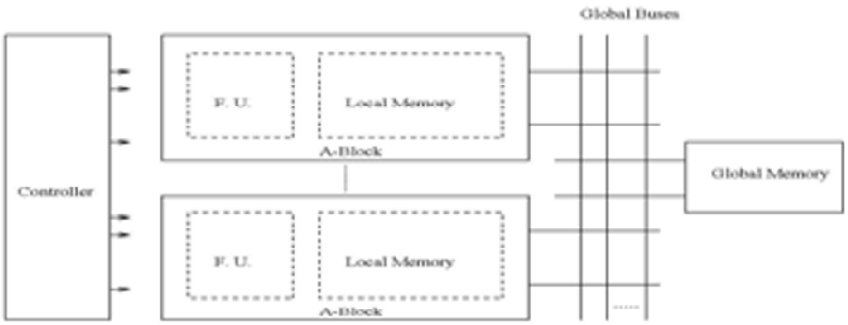

Figure 1: Schematic of Structured Architecture

- Roulette Wheel Selection
- Stochastic Remainder Roulette Wheel
- Tournament Selection
- Truncation Selection

1.6. Literature Survey

As per our work, the survey is divided into two parts that are explained below.

1.6.1. Scheduling in High-Level Synthesis

A recent survey of the synthesis task is by Paulin [8], where the concentration is on scheduling techniques. In Balakrishnan et al. [15], the basic scheme to schedule an operation and then bind it, along with its associate's source and destination operands is given. Estimation of design cost, at this stage, is costly. SAST is the tool or technique which has adopted this approach. It permits better control over the structure of synthesized data path, by means of architectural parameters.

1.6.2. Genetic Algorithm and Selection Methods

SAST tool uses genetic algorithm for solving the scheduling problem [7]. This tool uses Roulette Wheel selection method for selecting the parents on the basis of their cost. This method suffers from the disadvantage of being a high-variance process with the outcomes that there are seldom large differences between the actual and expected numbers of copies made - there is no surety that the best solution will be copied. De Jong [16] tested a scheme, which gave just such a guarantee by snowballing the population to include a copy of the best solution if it hadn't been retained. He found that on problems with just single maximum (or minimum) the algorithm performance was much improved, but on multimodal problems it was degraded. In roulette wheel selection method, since the computation of the average fitness requires the fitness of all population members, this selection operator is slow compared to tournament selection [13]. This is also noisy in the sense of introducing a large variance in its realizations. Stochastic remainder roulette wheel selection [14] and tournament selection are better when compared to it.

1.7. Motivation and Objective of the Present Work

Roulette Wheel Selection method used by SAST tool has few drawbacks due to randomness in its behavior when compared with other selection methods like tournament selection method [13]. They are:-

- Takes more time to schedule operations as compared to tournament selection. i.e. execution time is more.
- Noisy, in the sense of large variance in its realizations.
- When a population contains only individuals with scores of large, nearly equal, absolute value, the selection probability of all individuals approaches to be identical to each other, which works against the basic idea of genetic algorithms.

In this work, three more selection methods have been added in the SAST tool. The main objective of adding these methods primarily is to do the Performance analysis of SAST tool by applying different selection schemes in the genetic programming for improved schedule. The selection schemes added in the tool are:

- Stochastic Remainder Roulette Wheel Selection
- Tournament Selection
- Truncation Selection

These selection methods have been applied on different high-level synthesis benchmarks as follows:-

- Differential Equation Solver
- Elliptical Wave Filter
- Discrete Cosine Transform

Finally, the results of different benchmark are analyzed and compared.

2. SCHEDULING IN HIGH LEVEL SYNTHESIS

2.1. Classification of Scheduling Algorithms

Scheduling algorithms can be divided into time constrained and resource constrained scheduling, based on the objective of the scheduling problem. Further, scheduling algorithms are distributed in the following categories:

i. Unconstrained Scheduling

a. ASAP

b. ALAP

ii. Time Constrained Scheduling

a. Force Directed Scheduling

b. ILP

iii. Resource Constrained Scheduling

a. List-based Scheduling

2.1.1. ASAP (As Soon As Possible)

As-Soon-As-Possible (ASAP)[1] scheduling is one of the simplest scheduling algorithms used in HLS. In ASAP scheduling, first the highest number of control steps that are allowed is determined. Following that, the algorithm schedules each operation one by one into the earliest possible control step. An example of ASAP is shown in figure 2(b) of the data flow graph shown in figure 2(a).

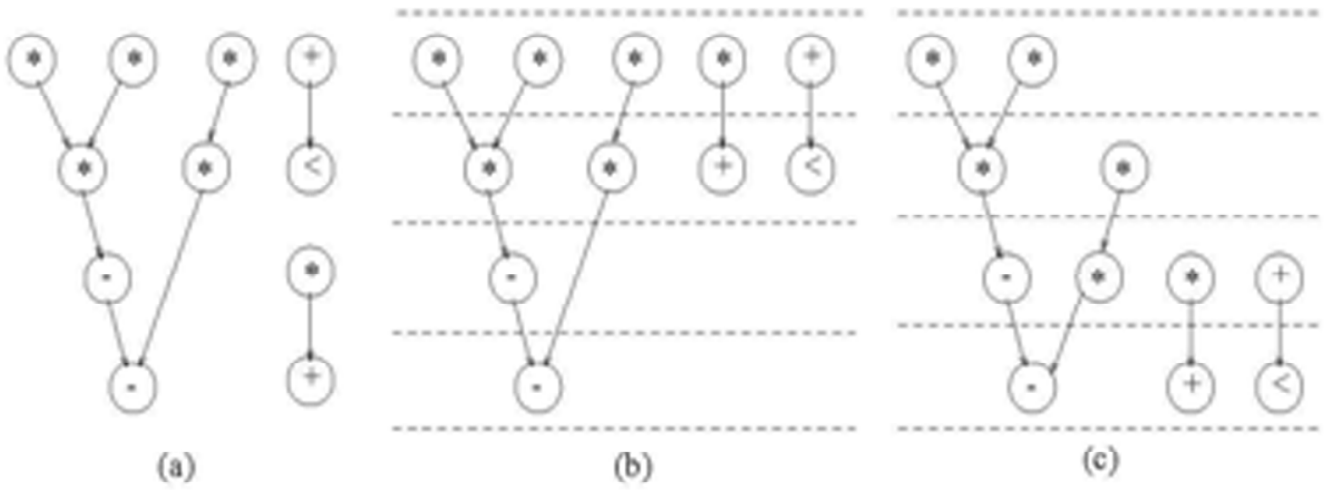

Figure 2: (a) Data Flow Graph (b) ASAP Scheduling (c) ALAP Scheduling

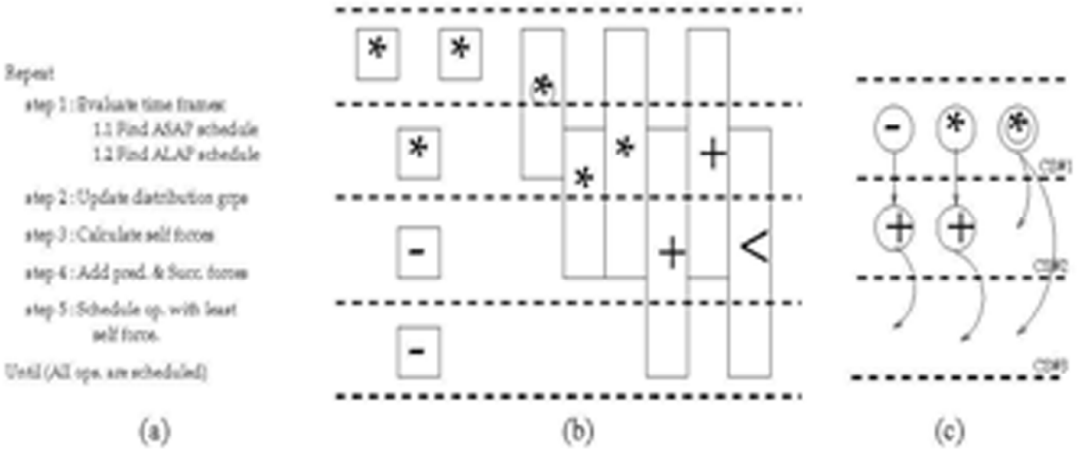

Figure 3: (a) FDS Outline (b) Distribution Graph (c) Sub-opt schedule example

2.1.2. ALAP (As Late As Possible)

As-Late-As-Possible (ALAP) scheduling is almost similar to ASAP, but instead of scheduling operations to early control steps, in ALAP, first the maximum number of control steps that are allowed is determined. Following that, the algorithm schedules each operation, one at a time, into the latest possible control step. An example of ALAP is shown in figure 2(c) of the data flow graph shown in figure 2(a).

2.1.3. Force-directed Scheduling

The Force-directed scheduling (FDS) is a heuristic method[8] that is a very popular scheduling technique for time constrained scheduling. The main goal of the algorithm is to reduce the total number of FUs used. This algorithm achieves its goal by uniformly distributing the operations of same type over the available control steps. This algorithm is briefly explained in figure 3.

2.1.4. List-based Scheduling

Unlike ASAP, ALAP or FDS scheduling, which process operations individually in a fixed order, list scheduling handles each control step individually (in increasing order). List scheduling works by trying to schedule “maximum” number of operations in the control step, subject to resource constraints and data dependency. During the scheduling process, list scheduling uses a ready list (hence the name) to keep a note of data-ready operations subject to data dependency.

2.1.5. Integer Linear Programming based Scheduling

Integer Linear Programming (ILP), have been used to solve a wide range of constraint based optimization problems. In this section, we will discuss an ILP formulation for optimally solving the synthesis problem. The biggest advantage of using ILP is the quality of the solution; unlike the heuristics based scheduling algorithms, described earlier, an ILP solver is guaranteed to find an optimal schedule from these formulations. However, this guarantee of quality comes at a price — ILPs cannot, in general, be solved in polynomial time. Thus, the trade-off is between the guarantee of solution quality and a guarantee of quickly finding a solution.

3. BASICS OF GENETIC ALGORITHMS

Definition 3.1 (Chromosome).

Let A be an alphabet (in other words a set of symbols). A chromosome (x) is a string of symbols from alphabet A. The number of symbols of (x) is called the length of $|(x)|$, denoted by $|(x)|$. The set A′, with I

∈ N, consists of all possible chromosomes ð with | (ϰ) | = l. (ϰ) (i) denotes the ith symbol, with 0 ≤ i < l of chromosome (ϰ).

Definition 3.2 (Encoding Enc, decoding Dec).

Let (F, c) be a search problem. Let A be a set of chromosomes. The function Dec: A′ → F is called a decoding. The function Enc: F → A′ is called an encoding. The encoding Enc(f) of an element f ∈ F is defined as an element of {(ϰ) A′| Dec((ϰ)) = f}. Hence, for each element f∈F, one or more encodings (ϰ) A′ exist. If for all f ∈ F.

The set of possible encodings consists of exactly one element, the encoding is called one-to-one.

Classical genetic algorithms as described in [6] use bit-strings as encodings, in other words alphabet A = {0 , 1}.

Definition 3.3 (Fitness s, scaling function S).

Let (F , c) be an instance of a combinatorial optimization problem. Let Al be a set of chromosomes, and let Dec: A′ → F be an onto function. The fitness s: A′ → R is a function, with s((ϰ)) the fitness (or score) of chromosome ð ∈ A′. Fitness s is related to cost function c by use of a scaling function S: R → R, given by s((ϰ)) = S((ϰ) (Dec(ð))).

During the run of a genetic algorithm, it keeps track of a collection of chromosomes, called a population.

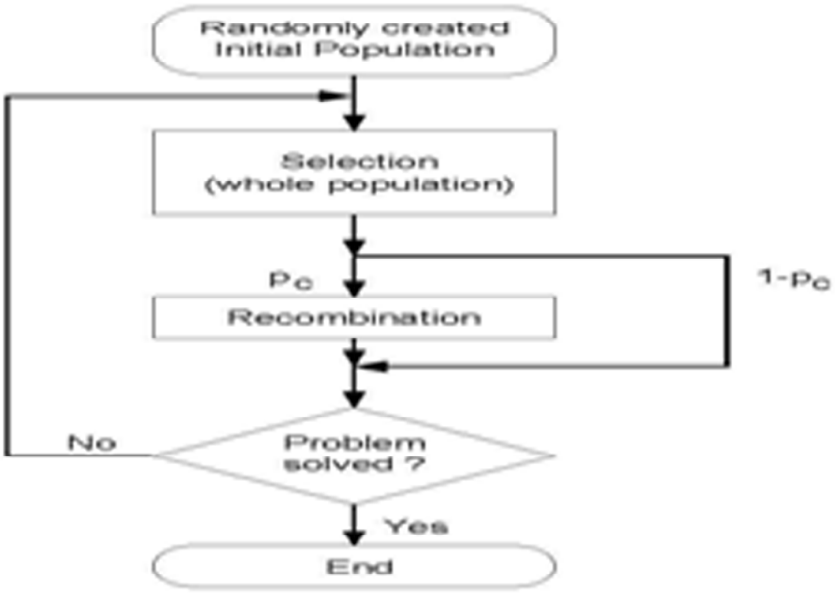

Figure 4: Flowchart of Genetic Algorithm

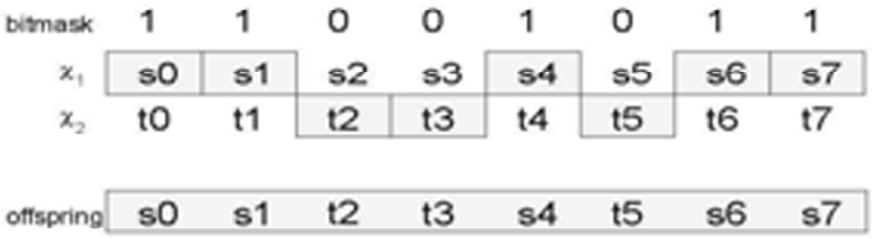

Figure 5: Example of Uniform Crossover

Definition 3.4 (Population P, population size |P|, individuals).

A population P is a bag (also called collection), the elements of which are taken from the set of chromosomes A′. The elements of P are called individuals. The size of the population P, denoted by |P| is called the population size of P.

In a genetic algorithm the initial population Po is created by randomly selecting |Po| individuals from the set of chromosomes Al. A genetic algorithm iteratively tries to improve the average fitness of a population by the construction of new populations, using selection and recombination mechanisms. Recombination of individuals is performed by so-called operators. An operator accepts a set of chromosomes (sometimes called parents), and constructs new chromosomes (called offsprings or children) by copying information from the parents.

Definition 3.5 (Operator O).

An operator O is a mapping O: (A′)^m → (A′)^n, with n, m ∈ N. It accepts m chromosomes (also called parents), and, using a particular mechanism, generates n chromosomes (called children or offsprings).

The most popular operators are called crossover and mutation. Mutation takes one chromosome, changes its contents, and returns the modified chromosome as a result. Crossover takes two chromosomes ð1, ð2∈A′, exchanges information between these chromosomes to create new chromosomes, and returns one or two chromosomes as a result.

Definition 3.6 (Selection probability sel).

Let P be a population, and let c ∈ P. The selection probability is a function sel: P → [0 , 1] ∈ R, with sel(c) the probability that individual c is chosen from the population as a parent for a particular operator that is calculated by (i). A well-known way of performing selection is by using so-called roulette wheel selection (also called proportionate selection), in which for a chromosome ð the selection probability sel is defined as follows:

$$sel(x) \quad \frac{s(x)}{\sum_{(x \in P)} s(x)} \tag{i}$$

4. GA BASED SCHEDULING ALGORITHM

4.1. Solution Representation

Each solution contains several decisions which are required for the proper implementation of the design. For each operation the moment at which it is to be scheduled and where it has to be scheduled are stored. Similarly, a multiplication could be implemented by a combinatorial multiplier or by a pipelined multiplier. However, the information of module requires to be stored explicitly. Thus there are three types of information to be represented which elaborated below:

1) Information immediately related to the scheduling of operations.
2) Information that indicates the scheduling of variable transfers, and
3) Information relating the composition of FUs.

4.2. Parent Selection

In SAST tool, the parents are selected on the basis of their costs using the Roulette Wheel technique. This being a minimization problem, the selection probability of a parent is computed taking into account the maximum cost of solutions in the population.

4.3. Crossover

First and foremost two parent solutions are chosen. These go through process of mutation and then the actual crossover takes place that results in a raw offspring. Inheritance of the features from either of the two parents proceeds in the (inverse) ratio of their solution costs. The solution at this stage is in general not feasible. The completion algorithm explained next is applied to this raw solution to generate a feasible solution.

4.4. Solution Completion

A methodology for solution completion is applied to the raw solution resulting from features inheritance during crossover. Solution completion is also applied at the time of generating new solutions because the randomly generated features used to make the initial solutions may not correspond to feasible solutions either. The programming complexity to support the various features.

4.5. Replacement

The replacement policy is design to ascertain that all solutions arrived at stay in the population for at least one iteration. This is done by virtue of introducing all the new solutions generated by crossover during one generation of the GA into the population and replacing the equal number of existing solutions. To overcome this, a scheme has been used simultaneously to maintain the solutions with more efficient costs and also retain a diversity of FU configurations within the population.

5. SELECTION SCHEMES IN GENETIC ALGORITHM

5.1. Introduction

Genetic Algorithms (GA) are nothing but probabilistic search algorithms characterized by the fact that a number N of potential solutions (called individuals Ji∈J, where J represents the space of all possible individuals) of the optimization problem at the same time sample the search space. This population P = {J1, J2,....,JN} is modified according to the natural evolutionary process: after initialization, selection w : JN → JN and recombination R : JN → JN are executed in a loop until some criteria for termination is reached. Each iteration of loop is called a generation and P(τ) denotes the population of generation.

5.2. Roulette Wheel Selection Scheme

Parents are selected in accordance with their fitness. The better the chromosomes are, the more chances to be selected they have. It is also known as proportionate selection method. In this solutions are assigned copies, the number of which increase or decrease in proportion to their fitness values. If the average fitness of all the population members is favg, a solution with a fitness fi gets an expected fi/favg number of copies.

In SAST tool, for fitness, probability of each solution is calculated with the help of (ii)

$$p_{si} = \frac{C_{\max} + \delta - C_i}{N_{sols}\left(C_{\max} + \delta\right) - \sum_i C_i} \tag{ii}$$

Thereafter, the cumulative probability of each solution is calculated by adding the probabilities from the top of the list. Thus, the most bottom string in the population possesses a cumulative probability equal to 1.

5.3. Stochastic Remainder Roulette Wheel Selection Scheme

In this selection scheme, the probabilities pi are multiplied by the population size and the expected number of copies is calculated for all the solutions. Thereafter, each solution is first assigned a number of copies

which is equivalent to integer part of the expected number. Thereafter, the usual roulette-wheel selection (RWS) operator is applied along with the fractional part of the expected number of all solutions to other copies also.

In this, the expected number of copies of each solution is calculated as

$$Ei = Pi + NSi$$

Where, N is the number of solutions,Pi is the probability of the ith solution.

Each solution is then copied Ii times, Ii being the integer part of Ei. The fractional remainder

$$Ri = Ei-Ii$$

is treated as the probability of further duplication.

5.4. Tournament Selection Scheme

Tournament selection can be construed as the natural process of individuals competing with each other to mate. When moose rut, the competition in between the bucks decides which will mate. So too, the chromosomes in the population are chosen for competition, usually in a random manner, with the victor maximizing its expected incidence in the mating pool. Note that selection is the one step of the process where we select individuals among the population that acts as parents of new offspring. To do same with tournament selection, you have to choose certain number of possible parents, after that select the best one as the winner. How many possible parents should be allowed to compete taken as value of k. The algorithm for the tournament selection is as given under:

Let k = 1. Looking at the pseudocode, this gives sheer random selection. You pick one individual randomly and return it.

Let k = 10*N. Now we have much higher probability of choosing every member of the population at least for once, so almost for every time, we are terminating process of returning the best individual among the population.

None of these options would work efficiently. Instead, you want something that give back good individuals more than bad ones, but not so dominantly that it keeps choosing the same few individuals time and again. Binary tournament selection (k=2) is most often used.

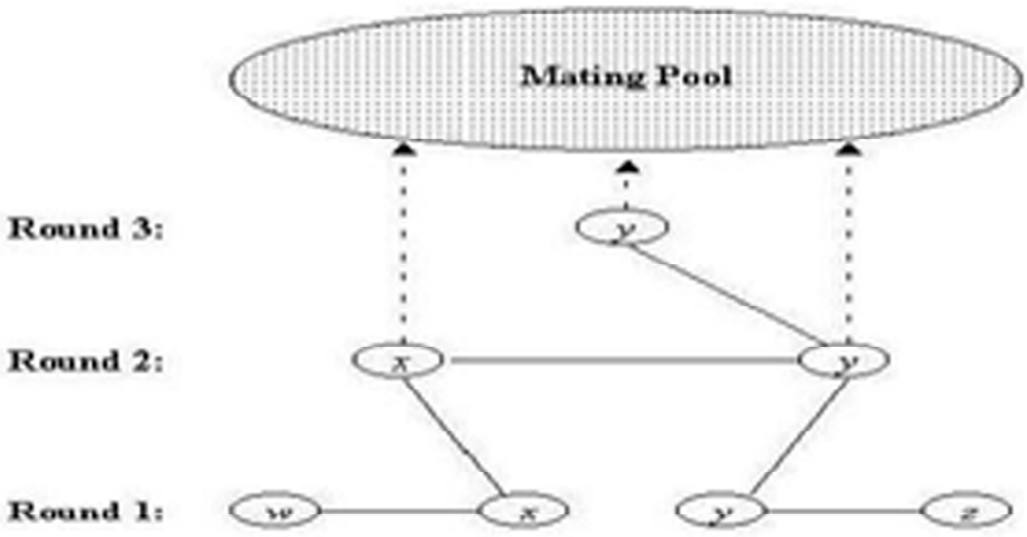

Figure 6 : Tournament Competition between Solutions

6. EXPERIMENTAL RESULT

The plot shown below compares the execution time of EWF by applying different selection schemes

As per the plot, truncation and tournament takes the least time as compared to others. Hence, the execution time has been optimized. For mathematical values, please refer to the Table no. 1.

6.1. Comparison of Number of Control Steps

The plot shown below compares the number of Control Steps of EWF by applying different selection schemes.

As per the plot, roulette-wheel reduces the control steps from 31 to 28 and on the other hand, tournament selection is reducing it to 27. Hence, control steps are reduced along with the execution time in case of tournament selection scheme.

6.2. Comparison of Cost Estimation Steps

Number of cost estimation steps should be as low as possible. In case of roulette wheel, selection is noisy in the sense of introducing a large variance in its realizations. To compare the number of steps,

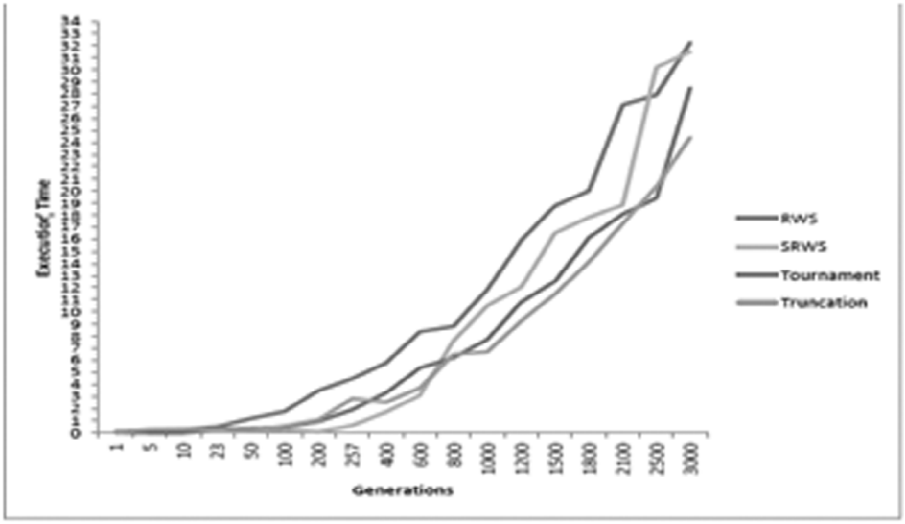

Figure 7: Plot of Execution Time(in seconds) vs No. of Generations (EWF)

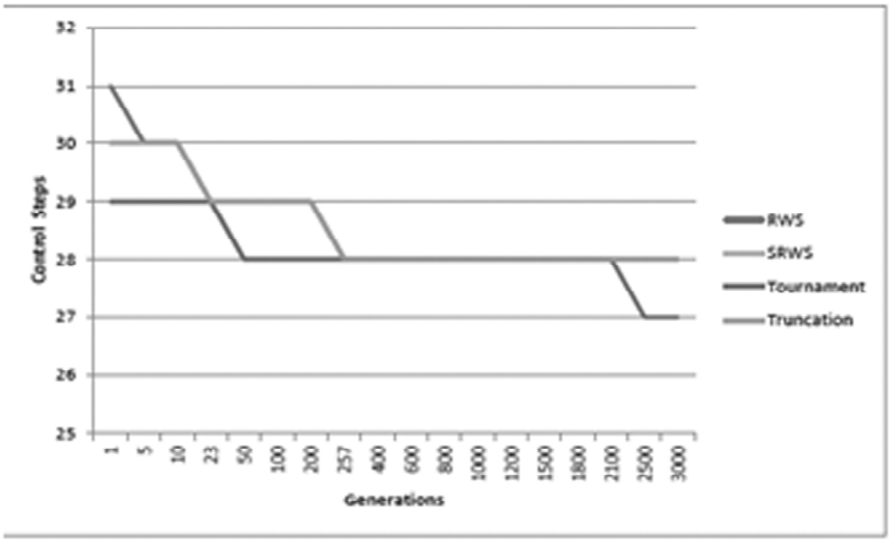

Figure 8: Plot of No. of Control Steps vs Generation

roulette wheel is compared with tournament selection scheme and have been applied on three benchmarks with varying number of solutions given as input. For experimental, please refer to table no. 2(a), 2(b) and 2(c).

6.3. Experimental Data

The selection schemes are applied to three different circuits. In this section, the results of each are shown individually.

Table 1(a), 1(b) and 1(c) shows the Execution time, Number of Control Steps and Number of registers after 3000 generations for all the three benchmark circuits.

Table 2(a), 2(b) and 2(c) shows the evaluation of steps for cost estimation for roulette wheel and tournament selection. This has been applied on DCT, DIFFEQ and EWF with varying number of solutions. (i.e. 150, 250 and 350).

Table 1(a) : Experimental Results for DIFFEQ.

Selection Schemes	Differential Equation Solver (DIFFEQ)		
	Execution Time (sec)	No. Of Control Steps	No. Of Registers
Roulette Wheel	14.206	14	12
Stochastic Roulette Wheel	19.28	14	13
Tournament	13.331	14	13
Truncation	9.79	14	12

Table 1(b) : Experimental Results for EWF.

Selection Schemes	Elliptical Wave Filter (EWF)		
	Execution Time (sec)	No. Of Control Steps	No. Of Registers
Roulette Wheel	32.277	28	16
Stochastic Roulette Wheel	31.5	28	16
Tournament	28.48	27	17
Truncation	24.4	28	17

Table 1(c) : Experimental Results for DCT.

Selection Schemes	Discrete Cosine Transform (DCT)		
	Execution Time (sec)	No. Of Control Steps	No. Of Registers
Roulette Wheel	14.296	14	15
Stochastic Roulette Wheel	14.1151	14	15
Tournament	13.9141	14	17
Truncation	13.037	14	15

Table 2(a) : Cost Estimation Steps for DCT

Cost Estimation Steps for DCT		
No. Of Solutions	Roulette Wheel	Tournament Selection
150	14	9
250	12	10
350	7	7

Table 2(b) : Cost Estimation Steps for DIFFEQ

Cost Estimation Steps for DIFFEQ		
No. Of Solutions	Roulette Wheel	Tournament Selection
150	4	11
250	9	6
350	24	8

Table 2(c) : Cost Estimation Steps for EWF

Cost Estimation Steps for EWF		
No. Of Solutions	Roulette Wheel	Tournament Selection
150	15	5
250	16	8
350	11	14

7. CONCLUSION AND FUTURE SCOPE OF WORK

7.1. Conclusion

Due to the enormous growth of the design complexity, the gap between the electronic system level and the register transfer level must be filled. A complete design methodology is described in this thesis, which allows automation of the high level synthesis design flow. The main objective of this thesis was to do the Performance analysis of SAST tool by applying different selection schemes in the genetic programming for improved schedule. Selection is a stage in Genetic Algorithm in which individual solutions are chosen from the population on the basis of its fitness for crossover. Three selection schemes have been added successfully in the SAST tool. They are:-

- Tournament Selection
- Stochastic Remainder Selection
- Truncation Selection

Effects on different parameters are as follows:-

7.1.1. Execution Time

In figure 7, execution time for the Elliptical Wave Filter (EWF) is compared for all the four selection schemes. As the number of generations increases, execution time of the genetic algorithm increases. From the plot, it is clear that execution time is maximum in case of roulette-wheel (from table 1(b), execution time for roulette-wheel selection scheme is 32.277 seconds) and minimum in case of truncation selection (from table 1(b), execution time for truncation selection scheme is 24.4 seconds). Hence, there is an optimization of 24.4% in execution time

In case of tournament selection, execution time is 28.48 seconds which is second best but is still preferred over truncation selection scheme. This is because, truncation selection is less sophisticated.

Even, stochastic remainder roulette wheel-selection which is more deterministic as compared to basic roulette-wheel selection takes lesser time to execute.

7.1.2. Number of Control steps

In figure 8, number of control steps for the Elliptical Wave Filter (EWF) is compared for all the four selection schemes. As the generation proceeds, control steps are optimized. From the plot, it can be seen that in 1st generation, maximum control steps was 31 and they reduced to 27 in case of tournament

selection (which is minimum) as maximum generation was reached. Hence, tournament selection is again better than roulette wheel. One more thing was observed in this case, as the number of control steps is reduced, number of registers increases and then SAST tool tries to optimize them as far as possible.

Not much difference was observed for other selection schemes

7.1.3. Evaluation of steps required for Cost Estimation

It was compared between roulette wheel and tournament selection as shown in table 2(a), 2(b) and 2(c). These selection schemes were applied on DCT,DIFFEQ and EWF by keeping generations equal to 3000 and varying the number of solutions (i.e. 150,250 and 350).

From the tables 2(a), 2(b) and 2(c) it is clear that number of steps is high in case of roulette wheel selection when compared with tournament. This makes it very noisy and randomness is high.

7.2. Future Scope of Work

There is much potential in the area of high level synthesis to improve the search time for finding the optimal design architecture, and thereby accelerate the speedup of the exploration process.

The developed exploration approach for high level synthesis can be improved further by including other selection schemes and other properties of selection scheme.

Another aspect of high level synthesis, which also has significant potential for improvement, is the optimization of many other parameters such as reliability, temperature etc., which stills lies in the nascent stage of development.

REFERENCES

[1] D. D. Gajski, N. D. Dutt, A. C. Wu, and S. Y. Lin, High-Level Synthesis: Introduction to Chip and System Design. Kluwer Academic Publishers, 1992.

[2] M. Shadad, .An overview of VHDL language and technology,. Procs. of the 23rd Design Automation Conference, 1986.

[3] D. E. Thomas and P. Moorby, The Verilog Hardware Description Language. Kluwer Academic Publishers, 1991.

[4] M. C. McFarland, A. C. Parker, and R. Camposano, "Tutorial on high-level synthesis,'' in Proc. 25th Design Automation Conf, June 1988, pp. 330-336.

[5] Hwang, Cheng-Tsung, J-H. Lee, and Yu-Chin Hsu. "A formal approach to the scheduling problem in high level synthesis." Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 10.4 (1991): 464-475.

[6] Heijligers, Marcus Josephus Maria. "The application of genetic algorithms to high-level synthesis." (1996).

[7] Mandal, C., and R. M. Zimmer. "A genetic algorithm for the synthesis of structured data paths." VLSI Design, 2000. Thirteenth International Conference on. IEEE, 2000.

[8] Paulin, Pierre G., and John P. Knight. "Force directed scheduling for the behavioral synthesis of ASIC's." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 8.6 (1989): 661-679.

[9] Tsai, Fur-Shing, and Yu-Chin Hsu. "STAR: An automatic data path allocator."Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 11.9 (1992): 1053-1064.

[10] Cloutier, Richard J., and Donald E. Thomas. "The combination of scheduling, allocation, and mapping in a single algorithm." Proceedings of the 27th ACM/IEEE Design Automation Conference. ACM, 1991.

[11] Mandal, Chittaranjan A., P. P. Chakrabarti, and Sujoy Ghose. "Allocation and binding in data path synthesis using a genetic algorithm approach." VLSI Design, 1996. Proceedings., Ninth International Conference on. IEEE, 1996.

[12] Blickle, Tobias, and Lothar Thiele. "A comparison of selection schemes used in genetic algorithms." (1995).

[13] Goldberg, David E., and Kalyanmoy Deb. "A comparative analysis of selection schemes used in genetic algorithms." Urbana 51 (1991): 61801-2996.

[14] Deb, Kalyanmoy. Multi-objective optimization using evolutionary algorithms. Vol. 2012. Chichester: John Wiley & Sons, 2001.

[15] Balakrishnan, M., and Peter Marwedel. “Integrated scheduling and binding: a synthesis approach for design space exploration.” Design Automation, 1989. 26th Conference on. IEEE, 1989.

[16] De Jong, K. A., An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Thesis, University of Michigan, Ann Arbor, MI., 1975.