MACHINE-LEARNING BASED SIMULATED ANNEALER METHOD FOR HIGH LEVEL SYNTHESIS DESIGN SPACE EXPLORATION

SIDDHANT MAHAPATRA

140929350

SECTION B

DEPARTMENT OF MECHATRONICS

TABLE OF CONTENTS

1. INTRODUCTION

The paper proposes an automatic method to explore the alternative design architectures for a specific hardware circuit. It uses search optimization heuristic algorithms such as simulated annealing.This algorithm uses as input certain synthesis directives i.e. attributes or pragmas to insert them as comments in the source code. The source code is a behavioral level C/C++ design that describes the behaviour of the hardware circuit. Using the pragmas in this source code, the proposed system forces specific parts of the input program to be implemented in FPGA/ASIC in various ways. the result of this process results in an exponential number of circuits using certain combinations of the pragmas and executing in iterative manner. Thus, it is important here to use search optimization algorithms as this is a convex optimization problem. The proposed paper improves and speeds up current search techniques by using machine learning algorithm.

2. BACKGROUND

High level synthesis (HLS) is called the process of taking a C level program and mapping it on hardware circuit like FPGA or Application Specific IC. Presently, there are several commercial HLS tools that use pragmas directly inserted and generate a circuit along with a synthesis results file. The advantage of pragma based synthesis is that it allows a very fine level of control over the resulting design.

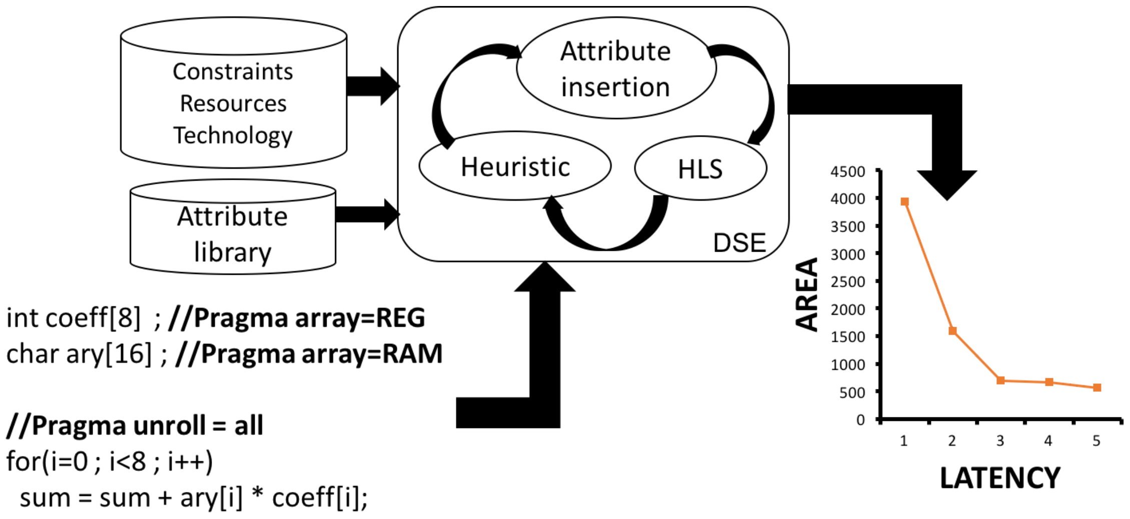
The other part of the tool is the Design Space Exploration (DSE). DSE is the automatic process of inserting a certain combination of pragmas inside the source code and executing different heuristic algorithms to search the optimal combination of those pragmas.

The part of the tool that handles this DSE is the explorer tool. One of the problems with HLS DSE is its exponentially growing design space. The number of explorable constructs for a particular design exponentially increases the complexity of the design space. Among the entire design space, the designer is mainly interested in the most optimal design which suits best the specification requirements. But, the design space is made of different objectives which makes it impossible to obtain one single optimal solution. In these kind of search problems where the objectives are more than one, pareto-optimal solutions are to be found. Solutions are pareto-optimal in nature if each design point on the pareto-optimal curve dominates the designs outside the curve for atleast one objective.

There are many search techniques to solve these multi-objective optimization problems that are mainly classified into: integer linear programming i.e. optimal, and heuristics. The heuristics are mostly used as, in spite of giving the sub-optimal solutions, they are faster than the optimal techniques. ILP techniques suffer from long running times that exponentially increases with the complexity of the designs.

3. THE DESIGN SPACE EXPLORER OVERVIEW

In figure 1 below, the explorer is shown that takes in inputs as the behavioral description to be explored in ANSI-C or SystemC, a library file containing all the explorable attributes, and uses the simulated annealing heuristic algorithm to output the pareto-optimal curve shown in figure 2. In this system, the simulated annealer uses annealing temperature as input to set its initial parameters. The algorithm then generates a new unique combination of attributes chosen from the library and inserts it into the behavioral description. The newly instrumented source code is parsed and synthesized in order to extract the area of the circuit from the Quality of Report file (QoR). Although the HLS tool used in this work reports the latency of the synthesized circuit, often this value only represents the state count, because the actual latency cannot be determined statically. This is mainly due to data dependent loops or continue/break statements in the behavioral description.



```
int coeff[8]  ; //Pragma array=REG
char ary[16] ; //Pragma array=RAM

//Pragma unroll = all
for(i=0 ; i<8 ; i++)
  sum = sum + ary[i] * coeff[i];
```

Figure 1: Explorer takes in inputs as the behavioral description to be explored.

4. SIMULATED ANNEALING

This search optimization algorithm runs on the concept of the behavior of heated atom elements. The process of slow cooling of the atoms in motion when given a certain initial temperature is called simulated annealing. Given an initial temperature, on every iteration the probability of selecting a solution and its cost function is calculated. If the cost function of the selected solution is better than the previous then the solution is selected and its cost function becomes the present criteria. In the case of it being rejected, the previous cost function is retained and the heuristic continues its search until the temperature condition drops to zero. This algorithm is not a greedy approach and it suffers from some pitfalls that it runs into local minima and gets stuck in that point until it gets better solutions. This wastes time and it exponentially affects the overall running time of DSE systems.

The main features of Simulated annealing used in this paper are:

1. They have evaluated the function for updating of temperature as mentioned below where T_{new} is the updated temperature. They empirically make α to be a constant at 0.95.
2. Finally, the stopping condition to exit the algorithm. In this paper, the exit criterion has been evaluated by counting the number of designs which are not accepted by the default rule or acceptance rule. The algorithm exits when it reaches the given threshold or when the temperature is equal to zero.

5. EXPLORATION METHOD USING SIMULATED ANNEALING

The simulated annealing is executed by generating random designs using random generation function, and then deciding the acceptance of the design using the acceptance probability function, and updating of temperature when its condition is satisfied.

The main objective of this work is to find a trade-off curve at least comparable with the one returned by a standard SA, while reducing the running time. Among the known characteristics of the SA, one of them is its tendency to traverse through down-hill curves and is likely to end up searching local minima thereby increasing the running time. In order to make the SA more efficient, different methods have been proposed in the past as mentioned earlier, mainly aiming at improving its efficiency and run-time. But the use of decision trees has been limited to memory management levels or on various multi-processor exploration uses.

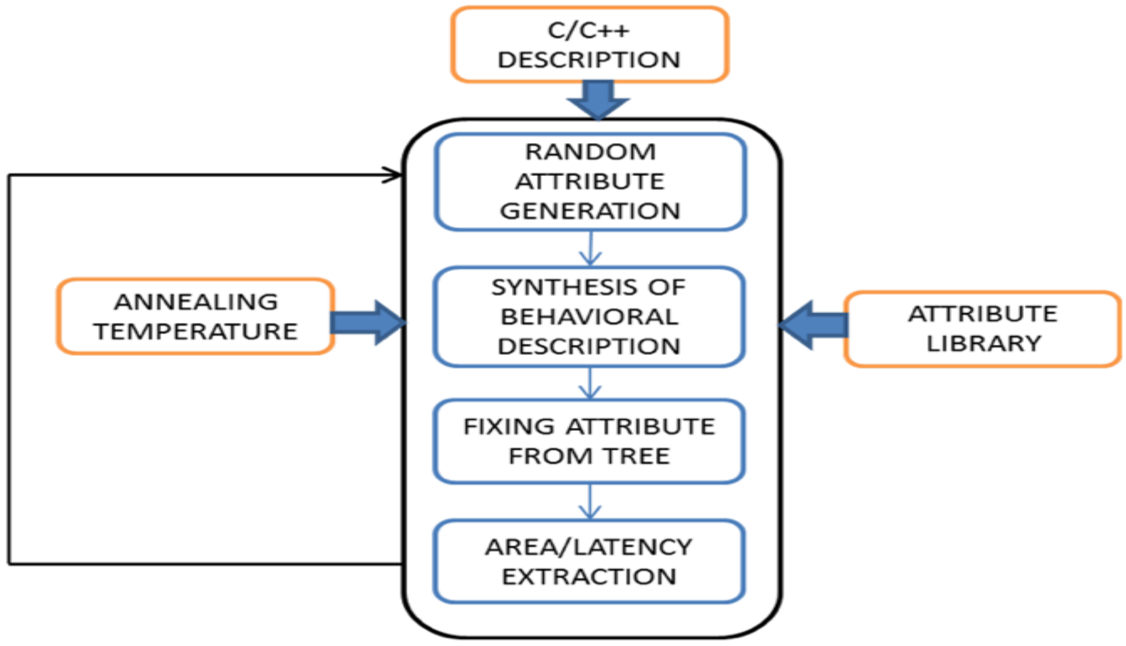

C/C++ DESCRIPTION
RANDOM ATTRIBUTE GENERATION
SYNTHESIS OF BEHAVIORAL DESCRIPTION
FIXING ATTRIBUTE FROM TREE
AREA/LATENCY EXTRACTION
ANNEALING TEMPERATURE
ATTRIBUTE LIBRARY

Figure 2: Overall flow of DSE using machine learning with simulated annealing

Using the above figure, the DSE process first selects a random combination of attributes or pragmas. This combination is inserted in the source code and synthesized using a commercial tool. From the results, the area and latency of the circuit is chosen to compute the design's cost function. This value is used in the simulated annealing to decide on the quality of the circuit generated. In this paper, a fixed set of circuits are generated using random combination of attributes in SA algorithm. After this, this set of designs are used as training set to train a decision tree and the model is therefore used to predict the forth coming design solutions iteratively selected by the SA algorithm.

6. DECISION TREE LEARNING

Decision tree learning (ID3 algorithm) is one of the most widely used and practical methods for inductive inference. In our case, it is required to infer the impact of different attribute combinations in the training set by fixing some of the attributes of the design having the highest impact on a design metric (area or latency) and letting the remaining attributes to be generated randomly during the exploration. The impact of the attributes on the quality of designs is predicted by the decision tree.

The basic function of decision trees is to classify instances from the training set and sort them in the tree on a top-down basis from the root till the leaf node as shown in figure 3.

Once the training set is created, our method builds a tree of attributes based on the ID3 algorithm. In our paper, the term attribute (v) refers to the specific attribute option inserted into the source code. The term, attribute type (A), refers to the category of attributes which perform a specific family (loop, array, function) of function under which the attributes are categorized. Each attribute is evaluated using a statistical test to determine its impacts on the quality of the training set. In ID3 algorithm the term information gain is defined as: as the statistical property for the purpose of sorting the attributes (v) based on their capacity to maximize/minimize one of the cost function objectives (Area or Latency). The entropy (E), is used to characterize the purity or impurity of a specific attribute for a collection of samples, S.

$$Entropy(S) = -p_{+}log_{2}p_{+} - p_{-}log_{2}p_{-}$$

Here, p_{+} represents the positive proportion of training examples where as p_{-} are the negative labeled proportion of training examples.

The term positive proportion refers to the fraction of the training designs satisfying the minimization of an objective and negative proportion is the fraction not contributing to minimizing of objectives. With reference to DSE in HLS, the process of classifying the training designs into a tree is different from its standard implementation used by ID3 algorithm. For every design objective, the preliminary training set is initially sorted in ascending order, and then the value of the objective of the first example in the sorted set with the addition of 20% of its value is considered for setting the range of threshold. For our work, all designs with objective values below this threshold are positive. Empirically, it has been shown that the indicated range achieves the best results. The design space to be explored can be easily controlled by changing the threshold value. Higher threshold value will fix more attributes and hence reduce the design space to be explored reducing the running time, while specifying a lower threshold value would fix fewer attributes and hence increase the design space and thus also the running time. The user can therefore set this parameter in our system.

Using this entropy as a measure the importance of each attribute, the information gain (IG) of each attribute type is measured as :

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

In the ID3 algorithm, this IG helps decide the best attribute type to be the root node that will be effective for classifying the designs. In the above equation, values (A) is the set containing all the attributes under attribute type A and S_v is the subset of for which the attribute type A has attribute v. The first term in the equation is the entropy of the original collection of the examples S, but the second term is simply the summation of the expected values of entropies of each specific attribute under the attribute type A. Thus, Gain(S,A) is the information provided about the classification of the sets, given the values of the attribute type A.

The working of the ID3 algorithms is described as follows:

1. Run the SA algorithm for X iterations – Training set
2. Start the Decision tree .
3. While the temperature is larger than zero,
    a. Calculate the entropy of each attribute using the negative proportion and positive proportion from training set.
    b. Using the entropy, the overall information gain of the attribute type is calculated.
    c. The attribute type with highest IG is chosen to create the new parent node in the tree.
    d. The parent node is created and their attributes are created as children nodes.
    e. If all attributes under A have not been tested, then decision to select the node under which the next attribute type is to be selected and placed is taken.

f. Then continue with SA using the attribute tree so that only good quality design solutions are selected.

The ID3 algorithm uses a greedy, hill- climbing approach, beginning with an empty tree and subsequently progressing by searching through the entire design space for a tree which would completely classify the training data. In this work, we have a binary target function for classifying each attribute in the library. For each attribute, the decision is either true or false. A leaf node in the tree is declared only when the decision about the attribute has been taken i.e. if the attribute has either zero true values or zero false values in the subset S_V. At each node of the tree, the attribute type with the highest information gain is chosen for being tested at that level. The procedure continues until all the attribute categories have been completely exhausted or all the nodes in the tree have been classified as leaf nodes.

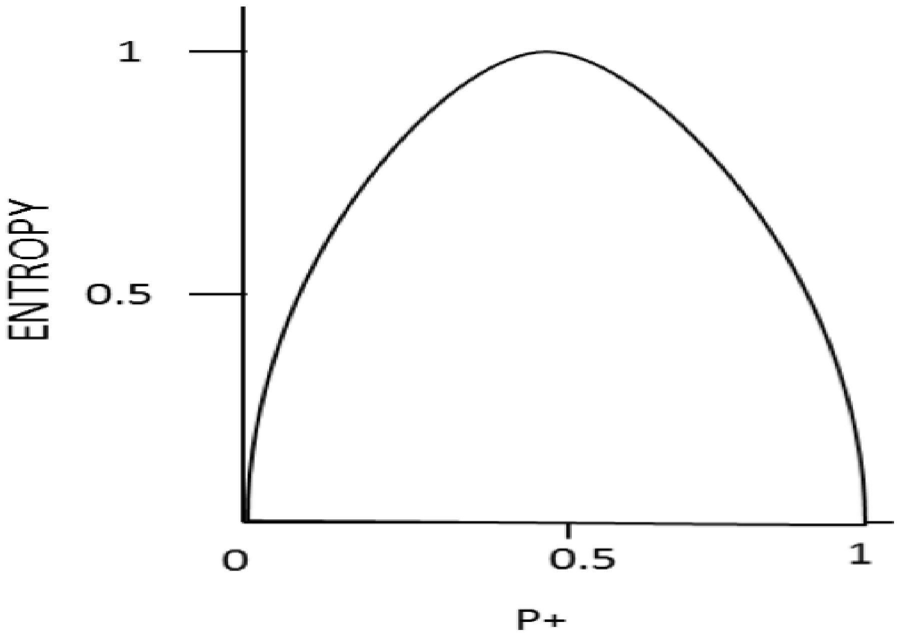


Figure 3: Entropy curve with increasing positive proportions

A brief summary of the implementation in Fig. 4 shows an example of a generated tree . A partially created tree consisting of roots representing attribute types and nodes representing the attributes is shown in figure 4. During the tree search at each level, the attribute declared as a leaf node and having a “true” value, is accepted for fixing, otherwise the attribute having the maximum entropy is selected for fixing in the newly generated design. The rest of the attributes are left for random selection. A “true” value indicates that the attribute satisfies the particular design objective.

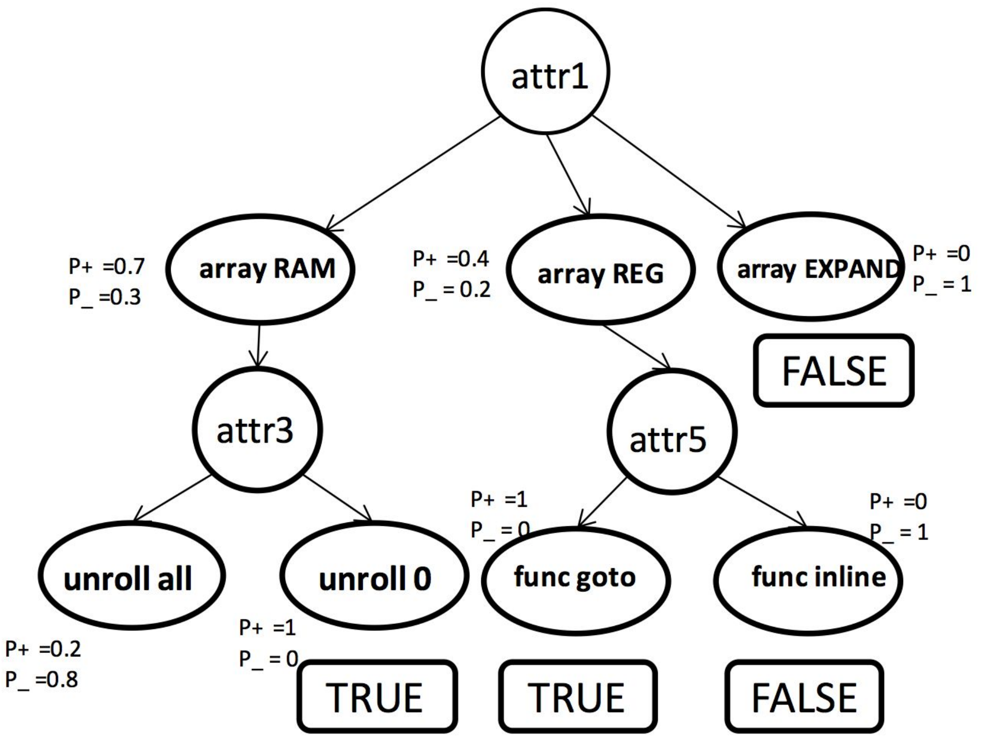


Figure 4: Example of a decision tree built with attributes and attribute types

The major importance of this decision tree comes when the root node at any level in the tree does not have any leaf node with either a “true” or “false” value. In these situations, we have developed a unique strategy in order to find the exact attribute combination which contributes to minimization of the design objective. The relation of entropy of an attribute and the positive proportion of examples related to that attribute represents an inverted bell curve as shown in Fig 5. If p_+ of an attribute is less than 0.5, then the entropy is directly proportional to p_+ otherwise the entropy is inversely proportional to p_+.

Depending on the characteristics of the nodes under a particular root, if there are no attributes with p_+ more than 0.5, then the direct proportionality rule is applied to check for the attribute with maximum entropy. If a single node exists with entropy more than 0.5, then the inverse proportionality rule is applied to check for the attribute having the minimum entropy since with the p_+ more than 0.5, the lesser the entropy, higher is the positive proportion. Thus, in this way, in the absence of true leaf nodes, the attributes with highest p_+ prove that they contribute to relatively more number of designs having true values for the target function and those attributes are chosen.

Here two measures, Information gain and entropy both help in creating the tree. While the IG helps in deciding the root at each level of the tree, the entropy helps the decision for which attribute shall be used for adding the next root node in the tree. The mode of selection of attributes from the tree is a depth-first search. At each level of the tree, only if the attribute is a leaf node with all positive proportions, then that attribute is selected otherwise the attribute with highest entropy is selected.

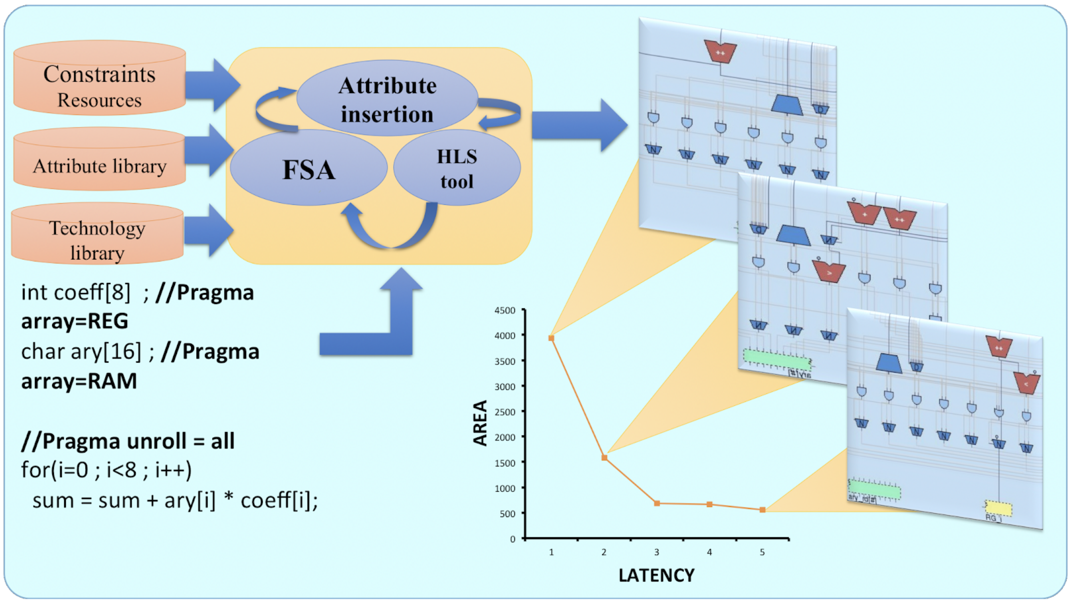


Figure 5: The graphical view of DSE using machine learning

In figure 4, the pragma values are the attribute values and the type of the pragmas are the attribute types. On analyzing the resultant graph, each design point indicates a separate micro-architecture described in registers, wires etc. This circuit is directly generated by the commercial HLS tool on each iterative exploration. The positive proportions of each attribute play the most important role in deciding the number of attributes to be fixed from the tree. For satisfying the design objectives, it is important to choose the attribute having the largest positive proportion, or having entropy as zero. It may be noted that while creating the decision tree, the impact of the attributes on the quality of designs decreases as we go downwards the tree. Thus, the priority of the attribute types is sorted in descending order throughout the tree. Thus, the upper half of the tree shall prove to be more effective. According to our implementation strategy, the decision tree shall be parsed only once. During this parsing, a unique list is created by selecting attributes from each level within the upper half of the tree. Thus, the decision tree is used best for deciding on the attributes with higher priority and process continues until the temperature reaches zero.

7. EXPERIMENTS

The paper uses around 8 benchmark programs written in C/SystemC. These programs describe filter, encryption algorithms etc.

Two quality indicators were used in order to compare the method FSA against a standard SA: (i) Pareto dominance and (ii) cardinality. Pareto Dominance is equal to the ratio between the total numbers of points in the Pareto set being evaluated, also present in the reference Pareto set. The higher the value, the better the Pareto set is. The cardinality indicates the number of dominating designs found by each method. A high cardinality indicates a larger number of solutions to choose from, which should be considered to be positive, although it needs to be interpreted carefully with the rest of the results. The results reported were computed by comparing the Pareto-front of each method compared to the reference Pareto front (the combination of the best non-dominated results of each method). The running times for each benchmark are used for quantitative analysis.

In the results, it is shown that the running times of FSA is faster than the system of SA by an average of 36% and runs upto a maximum of 48%. The quantitative part of analysis is by measuring the running time whereas the qualitative part of the analysis is by measuring the cardinality and dominance. Thus if both factors are better for FSA then the proposed method is robust and validated. According to our qualitative analysis, our proposed FSA algorithm produces comparable sets of dominating designs compared to SA while running almost twice as fast.

8. CONCLUSION

Since HLS is becoming necessary for VLSI in order to increase its design productivity FSA explorer is enabling the industry to produce a better trade-off curve while running much faster compared to regular design space exploration using SA. Since our algorithm statically learns the combination of attributes which minimizes the given cost function, it can reduce the search space considerably and hence reduce the running time of the exploration.

9. References

[1] Dr. Anushree Mahapatra and Dr. Benjamin C. Schafer. "Machine-learning based simulated annealer method for high level synthesis design space exploration" Electronic System Level Synthesis Conference (ESLsyn), IEEE Proceedings of the 2014, IEEE 14432521

[2] Mitchell , Tom. "Decision Tree Learning". Machine Learning, Mc Graw Publishers, 1997

[3] Cristina Silvano , William Fornaciari, Eugenio Villar. "Multi-objective Design Space Exploration of Multiprocessor SoC Architectures", Springer-Verlag New York.

[4] Philippe Coussy , Adam Morawiec. "High-Level Synthesis", Springer Netherlands.

[5] Dr. Anushree Mahapatra and Dr. Benjamin C. Schafer, Department of Electronics and Information Engineering, The Hong Kong Polytechnic University, Hong Kong.

[Secondary References]

[1] Lunman Deng, Song Jeong-Young. "Decision tree classification algorithm based on cost and benefit dual-sensitive", Electro/Information Technology (EIT), 2014 IEEE International Conference,Beijing, China.

[2]Fangfang Yuan, Fusheng Lian, Xingjian Xu, Zhaohua Ji. "Decision tree algorithm optimization research based on MapReduce", Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference, Beijing, China .