FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 27, N° 3, September 2014, pp. 317 - 328
DOI: 10.2298/FUEE1403317S

RAPID EXPLORATION OF COST-PERFORMANCE TRADEOFFS USING DOMINANCE EFFECT DURING DESIGN OF HARDWARE ACCELERATORS

Reza Sedaghat[1], Anirban Sengupta[2]

[1]Electrical and Computer Engineering, Ryerson University, Toronto, Canada
[2]Computer Science and Engineering, Indian Institute of Technology, Indore, India

Abstract. *Modern Very Large Scale Integration (VLSI) designs require a tradeoff between cost efficiency and performance (circuit speed). Furthermore, the Design Space Exploration (DSE) of the cost-performance tradeoffs for the multi objective VLSI designs should also be fast and efficient in nature. This paper presents a novel accelerated DSE approach for the exploration of cost-performance tradeoffs of modular multi (trio parametric. viz. cost, execution time and power consumption) objective VLSI hardware accelerators using hierarchical criterion analysis. The selection of the final design point is made after the tradeoffs are explored using the proposed approach. Results of the proposed approach when applied to various benchmarks yielded significant acceleration in the exploration process compared to current existing approaches with multi parametric objective.*

Key words: *Hardware accelerator, rapid, exploration, performance, cost*

1. INTRODUCTION

The design space exploration process generally takes into account two conflicting situations such as a) accurately searching the optimal design point from the huge design space b) time taken (or number of architectures analyzed) to evaluate the architecture design space in order to select the optimal design point. The second situation is more significant for modern multi-objective heterogeneous VLSI systems because exhaustive exploration of the architecture space is prohibitive due to the massive size of the design space. The architecture exploration process is therefore a battle between the optimal architecture determination and the speed of the exploration process. Furthermore, since present generation VLSI systems are multi-objective in nature, they demand efficient exploration approaches that can satisfy the multi-objective requisite by concurrently reducing the time spent in the architecture evaluation as well as maximizing the opportunity of automating the exploration methodology [1]-[7].

Received January 27, 2014
Corresponding author: Reza Sedaghat
Electrical and Computer Engineering, Ryerson University, Toronto, Canada
(e-mail: rsedagha@ee.ryerson.ca)

2. RELATED WORKS

Exploration has been a subject of research for almost two decades. Many approaches have been proposed in the recent past for fast and efficient evaluation of the design architecture space. The evaluation of the architecture design space has been performed by implementing an Architecture Configuration Graph (ACG) based on the hierarchical criterion factor [8], [9]. After the creation of the ACG, the Pareto optimal analysis is performed to find the optimal architecture. Although the approach seems promising, the major drawback of this approach is the excessive time taken for the framework to build the architecture design space in order to analyze the variants. On the other hand, authors in [10] use an evolutionary algorithm, such as Genetic Algorithm (GA), for efficiently searching the optimal solution. They propose a new encoding scheme to improve the efficiency of GA search for design space exploration. Using chromosome representation, the precedence relationships among the tasks in the input behavioral specification are encoded with a topological order-based representation to specify schedule priorities. Authors in [11] also use GA based on binary encoding of chromosome for efficient design space exploration. Additionally, authors in [12], [13] have developed a model that can assist designers at the system-level DSE stage to explore the utilization of the reconfigurable resources and evaluate the relative impact of certain design choices. All the above mentioned approaches mostly consider dual objective DSE (such as area and delay), but the proposed approach considers multi objective problems (such as cost, delay and power consumption).In addition to the above, a problem space genetic algorithm for design space exploration of data paths have been proposed in [14]. The authors have used the concept of heuristic/problem pair to convert a data flow graph into a valid schedule. The chromosome is encoded based on the 'work remaining' value of each node. One of the problems with approach [14] is that the second special parent chromosome's built in correspondence with the minimum functional units (i.e. serial implementation) does not differ in the work remaining field of the first special chromosome. This may not always lead to the optimal solution. Furthermore, the cost function considers only latency and not total execution time. Authors in [15] describe an approach to solve the DSE problem which is based on GA and weighted sum particle swarm optimization (WSPSO). The authors use crossover between parent and local-best-solution, then parent and global-best-solution to implement particle swarm optimization (PSO) behavior. The authors do not consider velocity to update the position. Moreover in WSPSO, the authors also do not consider user constraints for area and execution time in cost function. In [16], authors describe another approach for DSE in high level systems based on binary encoding of the chromosomes. However, they consider only traditional latency and not execution time constraint for data pipelining. Authors in [17] suggest that identification of a few superior design points from the Pareto set is enough for an excellent design process. The work shown in [18] discusses the optimization of area, delay and power in behavioral synthesis but does not consider execution time during data pipelining. The problem of design space exploration is also addressed in [19] by suggesting order of efficiency, which assists in deciding preferences amongst the different Pareto optimal points. Authors in [20] introduce a tool called SystemCoDesigner that offers rapid design space exploration with rapid prototyping of behavioral systemC models. In [21] evolutionary algorithms such as the Genetic Algorithm (GA) have been suggested to yield better results for the design space exploration process. An automated tool was developed by integrating behavioral

synthesis into their design flow, while authors in [22] describe current state-of-the-art high-level synthesis techniques for dynamically reconfigurable systems. Additionally, authors in [23]-[25] also use a genetic algorithm for scheduling and resource allocation for data path synthesis. Another class of scheduling methods employed previously was probabilistic in nature. For example the simulated annealing (SA) and simulated evolution (SE) based scheduling techniques have been used for the high level synthesis problem. Authors in [26], [27] have proposed a simulated annealing scheduling method called 'SALSA', which uses many probabilistic search operators to enhance the performance of the SA-based technique for high level synthesis problems. In addition, authors have also proposed an extended binding model for handling the scheduling problem in high level synthesis. Simulated evolution has been proposed by authors in [28] to solve the combined problem of scheduling and resource allocation in high level synthesis. Unfortunately, approaches [23]-[28] do not consider execution time, chaining and data pipelining. Authors in [20],[29] proposed alternate approaches based on integer linear programming (ILP).Although they are capable of providing good results, the computational complexity is massive and therefore require and extensive amount of time. Furthermore, the concept of data pipelining based on execution time was not shown during system trade-off. Work shown in [30] for DSE suggests an evolutionary algorithm for successful evaluation of the design for an application specific SoC. Other well known tools for HLS exist, such as GAUT [31]. GAUT inputs a C/C++ behaviour description for automatically generating a RTL structure based on compulsory constraint of throughput (or initiation interval) and clock period. In addition, authors in [32] propose an open-source HLS tool called LegUp for FPGA-based processor/accelerator systems. LegUp is able to synthesize C language to hardware, thereby providing a nice platform for HLS. Different FPGA architectures are supported by this tool, which allows new scheduling algorithms and parallel accelerators. Moreover, ROCCC, proposed in [33], is an open-source HLS tool for generating RTL structure from C. It was designed for kernels that perform computation intensive tasks, such as most DSP applications. Therefore, ROCCC applies to a specific class of applications (streaming-oriented applications) and is not a general C-to-hardware compiler, unlike LegUp [32].

3. THE PROPOSED FRAMEWORK BEHIND DESIGN SPACE EXPLORATION

3.1 The proposed framework for cost model

The model for the cost of the resources is proposed in this section and is an extension of the authors' previous work [3]-[5] on the area model.

Let the area of the resources be given as 'A'. Ri denotes the resources available for system designing; where 1<=i<=n. 'Rclk' refers to the clock oscillator used as a resource providing the necessary clock frequency to the system. The total area can be represented as the sum of all the resources used for designing the system, such as adder, multiplier, divider etc, and clock frequency oscillator. Total area is shown in Equation (1).

$$A = \sum A(Ri) \tag{1}$$

$$A = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) + A(Rclk) \tag{2}$$

Where '*NRi'* represents the number of resource 'Ri',and 'K_{Ri}' represent the area occupied per unit resource '*Ri*'. Let the total cost of all resources in the system be 'C_R'. Further, cost per area unit of the resource (such as adders, multipliers etc) is given as 'C_{Ri}' and the cost per area unit of the clock oscillator is 'C_{Rclk}'. Therefore total cost of the resources is given as:

$$C_R = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + .. + N_{Rn} \cdot K_{Rn}) \cdot C_{Ri} + A(Rclk) \cdot C_{Rclk} \tag{3}$$

Applying partial derivative to Equation (3) N_{R1} ….N_{Rn}, N_{RM},and A_{Rclk}yields Equations (4) to (7) respectively as shown below:

$$\frac{\partial C_R}{\partial N_{R1}} = \frac{\partial[(N_{R1} \cdot K_{R1} \cdot C_{R1} + .. + N_{Rn} \cdot K_{Rn} \cdot C_{Rn}) + A(Rclk) \cdot C_{Rclk}]}{N_{R1}} = K_{R1} \cdot C_{R1} \tag{4}$$

$$\frac{\partial C_R}{\partial N_{Rn}} = \frac{\partial[(N_{R1} \cdot K_{R1} \cdot C_{R1} + .. + N_{Rn} \cdot K_{Rn} \cdot C_{Rn}) + A(Rclk) \cdot C_{Rclk}]}{N_{Rn}} = K_{Rn} \cdot C_{Rn} \tag{5}$$

$$\frac{\partial C_R}{\partial A_{Rclk}} = C_{Rclk} \tag{6}$$

According to the theory of approximation by differentials, the change in the total area can be approximated by the following equation:

$$dC_R = \frac{\partial C_R}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial C_R}{\partial N_{Rn}} \cdot \Delta N_{Rn} + \frac{\partial C_R}{\partial A_{Rclk}} \cdot \Delta A_{Rclk} \tag{7}$$

Substituting Equations (4) to (6) into Equation (7) yields Equation (8):

$$dC_R = \underbrace{\Delta N_{Rn} \cdot K_{Rn} \cdot C_{Ri}}_{\text{The change of cost contributed by resource Rn}} + \underbrace{\Delta A(Rclk) \cdot C_{Rclk}}_{\text{The change of cost contributed by resource clock}} \tag{8}$$

Equation (8) represents the change in total cost of resources with a change in the number of all resources and the clock period (clock frequency).
The PF for cost of resources is defined as follows:

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1} \cdot C_{Ri}}{N_{R1}} \tag{9}$$

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn} \cdot C_{Ri}}{N_{Rn}} \tag{10}$$

$$PF(Rclk) = \frac{\Delta A(Rclk) \cdot C_{Rclk}}{N_{Rclk}} \tag{11}$$

Equations (9) and (10) indicate the average deviation of cost with respect to change in resource R1,….Rn. Note: This average deviation of cost helps in finding the dominance

effect of corresponding resource types on cost. Further, Equation (11) indicates the change of cost of the system with respect to change in resource 'Rclk' (i.e. the dominance effect of Rclk).

3.2 The framework used for execution time

This section introduces a new mathematical PF model for clock oscillator resource, thus extending the authors' previous work [3]-[5] on PF model of functional resources. The Priority factor of the resources R1, …Rn (such as adders, multipliers etc) for the execution time is derived from [3]-[5].From [3]-[5], the priority factor for the resources R1,...Rn for execution time, is defined as:

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot T_{Rn}}{N_{Rn}} \cdot (T_p)^{\max} \tag{12}$$

The PF model for the clock oscillator is defined as:

$$PF(Rclk) = \frac{T_{Rclk}^{Max} - T_{Rclk}^{Min}}{N_{Rclk}} \tag{13}$$

In Equation (13), 'T_{Rclk}^{Max}' and 'T_{Rclk}^{Min}' are the maximum and minimum values of 'execution time' and all the available resources have the maximum value. The PF defined in Equations (12) and (13) indicates the average change in execution time with a change in number of a particular resource. This average deviation of execution time depends on various resources to find the dominance effect of corresponding resource types on execution time.

3.3 The framework used for power consumption

PF for power consumption is defined as:

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \cdot (p_c)^{\max} \tag{14}$$

$$PF(RM) = \frac{\Delta N_{RM} \cdot K_{RM}}{N_{RM}} \cdot (p_c)^{\max} \tag{15}$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + .. + N_{Rn} \cdot T_{Rn}}{N_{R_{clk}}} \cdot (\Delta p_c) \tag{16}$$

Similarly as explained above, the priority factors for power consumption defined in Equations (14), (15) and (16) indicate the average change in the total power consumption of the system with the change in number of resources at maximum clock frequency. Therefore, as discussed before, Equations (14),(15),(16) indicate the dominance effect of resource types Rn, RM and Rclk on power metric.

4. PROPOSED DEMONSTRATION

4.1 System specifications

The case study of a selected benchmark has been provided for demonstration of the proposed method based on multiple real system specifications (as shown in Table 1). The function of the selected second order digital IIR Chebyshev filter benchmark is given in (17).

$$y(n) = 0.041x(n) + 0.082x(n-1) + 0.041x(n-2) - 0.6743y(n-2) + 1.4418y(n-1) \quad (17)$$

x(n), x(n-1) and x(n-2) are the input vector variables for the function. The previous outputs are given by y (n-1) and y(n-2), while the present output is y(n).

Table 1 System specifications and constraints

1) Maximum cost of resources**:** 1588 area units
2) Maximum time of execution: 200μs (for D =1000 sets of data)
3) Power consumption**:** Minimum
4) Maximum resources available for the system design:
   a) 3 Adder/subtractor units.
   b) 3 Multiplier units
   c) 3 clock frequency oscillators: : 24 MHz, 100 MHz and 400 MHz
5) No. of clock cycles needed for multiplier and adder/subtractor to finish each operation: 4 cc and 2cc
6) Area occupied by each adder/subtractor and multiplier: 12 area units (a.u), and 65a.u on the chip (e.g. 12 CLBs on FPGA for adder/subtractor)
7) Area occupied by the 24 MHz, 100 MHz and 400 MHz clock oscillator: 6 a.u., 10 a.u. and 14 a.u.
8) Power consumed at 24MHz, 100MHz and 400 MHz: 10mW/a.u., 32 mW/a.u. and 100mW/a.u. respectively.
9) Cost per area unit resource (C_{Ri}) = 10 units and Cost per area unit clock oscillator = 8 units

4.2 Arrangement of the design space (consisting of resources) in increasing orders of magnitude in the form of Architecture tree for cost model

This paper proposes the use of a hierarchical tree topology for arrangement of design points in sorted orders and exploration of the optimal design point. Unlike the authors' previous works [3]-[5] using vector design space, this approach uses a more convenient topology for exploration. The tree structure is easy to construct and does not require a special algorithm to order the design space in increasing/decreasing order. The PF of the different resources for cost model is given in equations below:

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1} \cdot C_{Ri}}{N_{R1}} = \frac{(3-1) \cdot 12 \cdot 10}{3} = 80 \quad (18)$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot K_{R2} \cdot C_{Ri}}{N_{R2}} = \frac{(3-1) \cdot 65 \cdot 10}{3} = 433.33 \quad (19)$$

$$PF(Rclk) = \frac{\Delta A(Rclk) \cdot C_{Rclk}}{N_{Rclk}} = \frac{(14-6) \cdot 8}{3} = 21.36 \quad (20)$$

Based on the PF calculated for cost model, the architecture tree for cost can be constructed. The tree is constructed in such a way, so that the resource with the highest PF is assigned level (l1) in the tree, followed by level (L2) being assigned to the resources with next highest PF and finally the last level being assigned to the resource with the lowest PF. The resource with the highest PF influences the cost of the system the most compared to the resource with the least PF. After the assigning the levels, the architecture tree comprising of the design space is automatically arranged in increasing orders of magnitude for the cost model. The architecture tree for the cost model is shown in Fig. 1. After the design space is sorted in increasing order of magnitude, searching is applied on the design space. A mixed searching approach is proposed in this work by extracting the advantages of two different well known searching algorithms viz. interpolation search and binary search. Previous works [3]-[5] employed a mono binary searching procedure. However, as highlighted in Fig. 1, a mixed searching approach is proposed to further enhance the speed of the exploration process. Interpolation search is used with the cost model in order to search for the border variant for cost, while for the execution time model binary search is used to find the border variant. The interpolation search performs faster than binary search in cases of uniformly sorted models, such as design space for cost (cost is an increasingly linear function of the number of resources, i.e. cost of the system increases with increase in number of resources). On the other hand, binary search exploits the 'divide and conquer' approach. Hence, it works faster on non-uniformly linear sorted models, such as execution time (execution time being a non-uniformly decreasing linear function of the number of resources i.e. increase in number of resources does not always decrease execution time, but remains same). Therefore applying interpolation search on the sorted design space for cost, shown in Fig.1 yields the border variant in just 2 comparisons (cost is calculated according to eqn.(3)). The border variant for cost is the last variant in the design space (in Fig.1) which satisfies the constraint for cost specified. The border variant obtained for cost is 'V11'.

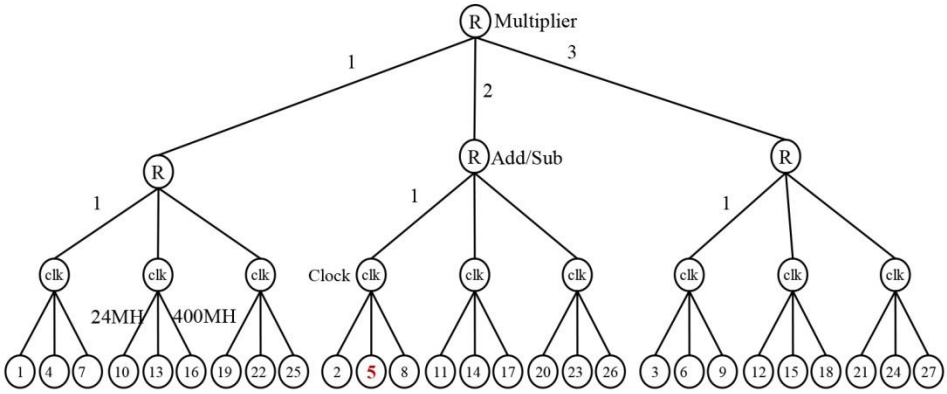

Fig. 1 Architecture tree representing the design space for cost arranged in increasing order

4.3 Arrangement of the design space in decreasing orders of magnitude in the form of Architecture tree for execution time model

The PF of the different resources used in system design for execution time model is given below equations:

$$PF(R1) = \frac{\Delta N_{R1} \cdot T_{R1}}{N_{R1}} \cdot (T_p)^{\max} = \frac{(3-1)\cdot 2}{3} \cdot 0.0416 = 0.055 \quad (21)$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot T_{R2}}{N_{R2}} \cdot (T_p)^{\max} = \frac{(3-1)\cdot 4}{3} \cdot 0.0416 = 0.111 \quad (22)$$

$$PF(Rclk) = \frac{T_{Rclk}^{Max} - T_{Rclk}^{Min}}{N_{Rclk}} = \frac{333.5 - 20.01}{3} = 104.50 \quad (23)$$

Similarly, as described in section II.B, the architecture tree for execution time is constructed based on the PF calculated for execution time. Thus, the architecture tree obtained after construction is now also automatically arranged (sorted) in decreasing orders of magnitude. After arrangement, binary searching is applied in order to find the border variant for execution time (execution time is calculated according to the model of execution time shown in [4]). The border variant for execution time is the first variant in the design space, which satisfies the constraint for cost specified. The border variant obtained is variant 'V5'. After the border variants for both cost and execution time are found, the Pareto optimal set is derived as explained in [3]-[5]. The architecture tree for power consumption is constructed similarly in increasing orders of magnitude for power consumption. Among the variants of the Pareto set, the one which appears first in the ascending ordered sorted design space (in the tree), is the one with the minimum power consumption. It concurrently satisfies the constraints for cost, execution time and power consumption (specified in Table1) for the design problem. Therefore the optimal variant obtained, which satisfies all the specified constraints, is Variant 'V5' (marked bold red in Fig.1).

5. Analysis and Results

The results of the proposed approach using PF and mixed searching scheme for rapid exploration of cost performance tradeoffs are verified for a number of benchmarks. Compared to the authors' previous works [3]-[5], the proposed approach is capable of further enhancing the speedup of the exploration process.

The search of the border architecture in the case of execution time (using binary search) requires only $\log_2 \prod_{i=1}^{n} v_{Ri}$ where 'n' = number of type of resources and 'v_{Ri}' is the number of variants of resource 'Ri'. The search of the border architecture (using interpolation search) for cost parameter requires $\log_2 \log_2 \log_2 \prod_{i=1}^{n} v_{Ri}$. In the design space exploration approach presented here, three objective parameters have been used; execution time and cost are the parametric constraints and power consumption is the optimization parameter. The total number of architecture evaluations performed during searching using the proposed method is given as:

$$\log_2 \log_2 \prod_{i=1}^{n} v_{Ri} + \log_2 \prod_{i=1}^{n} v_{Ri}$$

When applied on various benchmarks, the proposed approach indicated massive acceleration in the speedup compared to the exhaustive approach. The proposed method was also compared with a current approach in [8], [9]. The acceleration obtained, compared to the [8], [9], for both small and large size benchmarks is shown in Tables2 and 3 respectively.

Moreover, the proposed approach has also been compared with a heuristic approach (WSPSO) [15]. As evident from Tables 4 and 5, the proposed approach performs lower architecture evaluations than [15] for both small and large benchmarks respectively. For example, in case of MPEG MMV (shown in Table 5) the proposed approach performs only 14 evaluations, while [15] perform 53 evaluations to search a final solution.

Table 2 Experimental results of comparison between proposed DSE approach with the current approach [8], [9] for small benchmarks

Benchmarks [2],[34],[35]	Total possible architecture in the design space for one parameter	Architecture evaluation using proposed approach (Number of variants analyzed)			Architecture evaluation using approach [8],[9] (Number of variants analyzed)	Percentage speed up using proposed approach compared to [8],[9]	Average speedup using proposed approach compared to [8],[9]
		Cost	Execution time	Total			
IIR Chebyshev Filter	27	4	6	10	18	44.44 %	
Mesa Horner	36	5	6	11	19	42.10 %	
Elliptic Wave Filter	78	5	7	12	19	36.84 %	**41. 85 %**
Differential Equation Solver (HAL)	90	5	7	12	19	47.82 %	
BPF	100	5	8	13	21	38.09 %	

Table 3 Experimental results of comparison between proposed DSE approach with the current approach [8], [9] for large benchmarks

Benchmarks [2],[34],[35]	Total possible architecture in the design space for two parameters	Architecture evaluation using proposed approach (Number of variants analyzed)			Architecture evaluation using approach [8],[9] (Number of variants analyzed)	Percentage speed up using proposed approach compared to [8],[9]	Average speedup using proposed approach compared to [8],[9]
		Cost	Execution time	Total			
Auto Regressive Filter	144	5	8	13	21	38.09 %	
MPEG MMV	200	5	9	14	23	39.13 %	**37.56 %**
Matrix multiplication	400	6	10	16	25	36 %	
JPEG_IDCT	900	6	11	17	27	37.03 %	

Table 4 Experimental results of comparison between proposed DSE approach and the current approach [15] for small benchmarks

Benchmarks [2],[34],[35]	Total possible architecture in the design space for one parameter	Architecture evaluation using proposed approach (Number of variants analyzed)			Architecture evaluation using approach [15] (Number of variants analyzed)	Percentage speed up using proposed approach compared to [15]	Average speedup using proposed approach compared to [15]
		Cost	Execution time	Total			
IIR Chebyshev Filter	27	4	6	10	17	41%	**48.7%**
Mesa Horner	36	5	6	11	21	47%	
Elliptic Wave Filter	78	5	7	12	31	61%	
Differential Equation Solver (HAL)	90	5	7	12	32	62.5%	
BPF	100	5	8	13	35	62%	

Table 5 Experimental results of comparison between proposed DSE approach and the current approach [15] for large benchmarks

Benchmarks [2][34][35]	Total possible architecture in the design space for two parameters	Architecture evaluation using proposed approach (Number of variants analyzed)			Architecture evaluation using approach [15] (Number of variants analyzed)	Percentage speed up using proposed approach compared to [15]	Average speedup using proposed approach compared to [15]
		Cost	Execution time	Total			
Auto Regressive Filter	144	5	8	13	52	75%	**75%**
MPEG MMV	200	5	9	14	53	73.5%	
Matrix multiplication	400	6	10	16	65	75.3%	
JPEG_IDCT	900	6	11	17	72	76.3%	

6. CONCLUSIONS

This paper presented a novel framework for rapid exploration of the cost-performance tradeoffs for modular multi-objective hardware accelerators. Once the design space for the cost-performance is explored, the final design point with minimum power consumption is searched from the obtained small Pareto optimal set. The proposed DSE approach for different benchmarks yielded superior results in terms of acceleration obtained compared to the current existing approaches.

Acknowledgement: *This work is supported by the Optimization and Algorithm Research Lab (OPRAL), Ryerson University, Canadian Microelectronics Corporation (CMC), Motorola, NSERC CRSNG, Ontario Innovation Trust and Sun Microsystems. Additionally, This work acknowledges the assistance provided by Science and Engineering Research Board (SERB), Department of Science and Technology, Govt. of India.*

REFERENCES

[1] G. De Micheli, "Synthesis and Optimization of Digital Circuits". *McGraw-Hill: New York*, 1994.

[2] Saraju P. Mohanty, Nagarajan Ranganathan, Elias Kougianos and Priyadarsan Patra, "Low-Power High-Level Synthesis for Nanoscale CMOS Circuits" Chapter- High-Level Synthesis Fundamentals, Springer US, 2008

[3] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, "A High Level Synthesis design flow with a novel approach for Efficient Design Space Exploration in case of multi parametric optimization objective", Microelectronics Reliability, Science Direct, Elsevier, Volume 50, Issue 3, March 2010, pp. 424-437.

[4] Zhipeng Zeng, Reza Sedaghat, Anirban Sengupta, "A Framework for Fast Design Space Exploration using Fuzzy search for VLSI Computing Architectures", Accepted to Appear in the Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), June 2, 2010.

[5] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, "Rapid Design Space Exploration for multi parametric optimization of VLSI designs", Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), June 2, 2010, Paris, France, Article # 2016 (Session: Logic & High-level Synthesis, C2L-F).

[6] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, "Hardware Efficient Design of speed optimized Power stringent Application Specific Processor", Proceedings of IEEE 21st International Conference on Microelectronics (ICM), Morocco, December 22, 2009, pp. 167-170.

[7] D. Gajski, N. Dutt, A.Wu, and S. Lin, "High Level Synthesis: Introduction to Chip and System Design". *Kluwer: Norwell, MA*, 1992.

[8] Kirischian, L;Geurkov, V., Kirischian, V. and Terterian, I. 'Multi-parametric optimisation of the modular computer architecture', Int. J.Technology, Policy and Management, Vol. 6, No. 3,2006, pp.327–346.

[9] Kirischian, L. 'Optimization of parallel task execution on the adaptive reconfigurable group organized computing system', Proc. of International Conference PARELEC 2000, Canada, pp.150–154.

[10] Vyas Krishnan and SrinivasKatkoori, "A Genetic Algorithm for the Design Space Exploration of Datapaths During High-Level Synthesis, IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, June 2006, pp.229-313.

[11] E. Torbey and J. Knight, "Performing scheduling and storage optimization simultaneously using genetic algorithms," in Proc. IEEE Midwest Symp. Circuits Systems, 1998, pp. 284–287.

[12] Giuseppe Ascia, Vincenzo Catania, Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, "Efficient design space exploration for application specific systems-on-a-chip" Journal of Systems Architecture 53 (2007) pp. 733–750.

[13] C. H. Gebotys and M. I. Elmasry, "Global optimization approach for architectural synthesis," IEEE Trans. Comput.-Aided Des., vol. 12, 1993, pp. 1266–1278.

[14] M. K. Dhodhi, F. H. Hielscher, R. H. Storer, and J. Bhasker, "Datapath synthesis using a problem-space genetic algorithm," in IEEE Trans.Comput.-Aided Des., vol. 14, 1995, pp. 934–944.

[15] Harish Ram D. S., M. C. Bhuvaneswari, and Shanthi S. Prabhu, (2012) A Novel Framework for Applying Multiobjective GA and PSO Based Approaches for Simultaneous Area, Delay, and Power Optimization in High Level Synthesis of Datapaths, VLSI Design Hindawi, Article ID 273276, 12 pages

[16] E. Torbey and J. Knight, "High-level synthesis of digital circuits using genetic algorithms," in Proc. Int. Conf. Evol. Comput, May 1998, pp.224–229.

[17] Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, Fuzzy Decision Making in Embedded System Design," Proceedings of the 4th International Conference on Hardware/Software Codesign and System synthesis, October 2006,pp. 223-228.

[18] A.C.Williams, A.D.Brown and M. Zwolinski,"Simultaneous optimisation of dynamic power, area and delay in behavioural synthesis", IEE Proc.-Comput. Digit. Tech, Vol. 147, No. 6, 2000, pp. 383-390.

[19] I. Das. A preference ordering among various Pareto optimal alternatives. Structural and Multidisciplinary Optimization, 18(1):Aug. 1999, pp.30–35.

[20] Christian Haubelt, Thomas Schlichter, Joachim Keinert, Mike Meredith, "SystemCoDesigner: automatic design space exploration and rapid prototyping from behavioral models", Proceedings of the 45th annual ACM IEEE Design Automation Conference, 2008, pp. 580-585.

[21] J. C. Gallagher, S. Vigraham, and G. Kramer,"A family of compact genetic algorithms for intrinsic evolvable hardware," IEEE Trans. Evolutionary Computation., vol. 8, no. 2 , Apr. 2004, pp. 111–126.

[22] Xuejie Zhang and Kam W. Ng, "A review of high-level synthesis for dynamically reconfigurable FPGAs", Microprocessors and Microsystems, Elsevier, Volume 24, Issue 4, 2000, pp. 199-211.

[23] R. M. San and J. P. Knoght, "Genetic algorithms for optimization of integrated circuit synthesis," in Proc. 5th Int. Conf. Genetic Algorithms, San Mateo, CA, 1993., pp. 432–438.

[24] R. J. Cloutier and D. E. Thomas, “The combination of scheduling, allocation and mapping in a single algorithm,” in Proc. 27th Design Automation Conf., Jun. 1990, pp. 71–76.
[25] N. Wehn et al., “A novel scheduling and allocation approach to datapath synthesis based on genetic paradigms,” in Proc. IFIPWorking Conf. Logic Architecture Synthesis, 1991, pp. 47–56.
[26] G. Krishnamoorthy and J. A. Nestor, “Data path allocation using extended binding model,” in Proc. 32nd ACM/IEEE Design Automation Conf.1992, pp. 279–284.
[27] J. A. Nestor and G. Krishnamoorthy, “SALSA: A new approach to scheduling with timing constraints,” IEEE Trans. Comput.-Aided Des., vol. 12, 1993, pp. 1107–1122.
[28] T. A. Ly and J. T. Mowchenko, “Applying simulated evolution to high level synthesis,” IEEE Trans. Comput.-Aided Des., vol. 12, no. 2, Feb. 1993, pp.389–409.
[29] C. T. Hwang, J. H. Lee, Y. C. Hsu, and Y. L. Lin, “A formal approach to the scheduling problem in high-level synthesis,” IEEE Trans. Comput.- Aided Des., vol. 10, no. 2, Feb1991, pp. 464–475.
[30] Giuseppe Ascia, Vincenzo Catania, Alessandro G. Di Nuovo, Maurizio Palesi, Davide Patti, “Efficient design space exploration for application specific systems-on-a-chip” Journal of Systems Architecture 53, 2007, pp. 733–750.
[31] GAUT: A High-Level Synthesis Tool for DSP Applications”, P. Coussy, C. Chavet, P. Bomel et al., in High-Level Synthesis: From Algorithm to Digital Circuits, Springer, 2008, pp. 147-169.
[32] Canis, A., Choi, J., Aldham, M., Zhang, V., Kammoona, A., Czajkowski, T., Brown, S. D., and Anderson, J. H. 2013. LegUp: An open-source high-level synthesis tool for FPGA-based processor/accelerator systems. ACM Trans. Embedd. Comput. Syst. 13, 2, Article 24 (September 2013), 27 pages.
[33] Villarreal, J., Park, A., Najjar, W., and Halstead, R. 2010. “Designing modular hardware accelerators in C with ROCCC 2.0”. In Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines. 2010, pp. 127–134.
[34] http://www.cbl.ncsu.edu/benchmarks/.
[35] http://express.ece.ucsb.edu/benchmark/