ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 12, December 2015

Performance Evaluation of Genetic Algorithm on Scheduling Systolic Processors

C. Bagavathi Shivakumar

***Abstract*—Systolic systems provide remarkable scope for scalable architecture and regular data flow in an array of cells. An algorithm or design such as FIR filter requires iterative computations and hence is suitable for systolic implementation through systolic mapping. For mapping, the search space has to be searched for the optimal space-time mapping (allocation of resources). Many algorithms have been implemented for this purpose. This paper gives a brief idea of convergence and resource allocation for a 4-tap FIR filter through genetic algorithm in systolic architecture.**

***Index Terms*—Systolic array, Genetic Algorithm, Mapping.**

I. INTRODUCTION

With the accelerated pace of miniaturization forcing the designers to keep Moore's law still valid, implementation of computationally demanding tasks like real time processing, image processing algorithms onto FPGAs can pose a serious threat to designing methodology. A change in design methodology to improve the number of iterations and reuse the same hardware can be welcomed with a modest sacrifice in the speed. To make the FPGAs run faster and with lesser number of logic gates, systolic architecture is preferred.

Systolic array is a set of simple processing elements with habitual connections, which takes external inputs and processes them in a predestined, pipelined fashion. Data flow into the computation points (processing elements) and the results flows out of these points. For processing an algorithm in VLSI systolic architecture, systolic mapping is essential.

Mapping is a process of assigning each point in iteration space a processing element for the operation at discrete time. For searching the processor space for an efficient, optimal matrix, we require search techniques. The search optimal matrix is generally through heuristic [7].

The rest of the paper is organised as follows: Section II explains the systolic architecture in brief and its applications. Section III explains about the genetic algorithm.

II. SYSTOLIC ARCHITECTURE

Systolic array is a network of processors that rhythmically compute and pass the data through the system. To design the systolic array, the application of architecture, the algorithm to be executed and the technology should be known. Each processor is assigned to do a specific operation such as minimum distance, addition, multiplication, etc. The power of systolic arrays arises from concurrent use of cells with simple processing power and pipelining [10].

A systolic system consists of the set of processing elements (PE). Each cell performs a sequence of operations on the data that flows between them. The overall operation of the array is determined by the processor allocation matrix and scheduling matrix. The advantage of the architecture is mainly the reduced design of hardware components employed compared to the design without systolic concepts. If an algorithm requires 1000 processors to process a set of data in a second, systolic architecture can implement the same design with 100 processors. But there is an increase in execution time from 1 second to 10 seconds. When the data of 1000 processors is to be executed in 100 processors, the space and time constraints are to be accounted for. This accounting is done using the mapping process. In mapping process, the algorithm requires processor allocation matrix and scheduling matrix. The processor allocation matrix allocates a processor for each data. The scheduling matrix determines the time at which the next data should reach the processor taking into account the time required to process the previously scheduled data. These matrices have certain constraints on space based on the projection vector, where projection vector is the vector which determines which are the data to be processed by the same processor. The search technique is applied to search for the optimal matrices that minimises a cost function. The cost function is designed based on many factors. A simple cost function includes the number of processors and number of cycles required to finish the algorithm for the specified image size. The number of processors N_{PE} is given by

$$N_{PE} = PE_{\max} - PE_{\min} + 1 \quad \text{... (1)}$$

$$PE_{\max} = \max\left(\vec{p}^{\,t}.\vec{q} / \vec{q} \in J^n\right) \quad \text{... (2)}$$

$$PE_{\min} = \min\left(\vec{p}^{\,t}.\vec{q} / \vec{q} \in J^n\right) \quad \text{... (3)}$$

where p is the processors allocation matrix and q is the index vector representing the space. Space refers to the location of processors arranged in the Cartesian co-ordinates. The number of cycles N_{cycle} is given by

$$N_{cycle} = \max\{\vec{s}^{\,t}.(\vec{p} - \vec{q}) + 1\} \quad \text{... (4)}$$

where p and q are index space vectors and s is the scheduling vector. The cost function can be decided based on the problem. The cost function is given as

$$cost\mathit{function} = 0.9*(N_{PE}) + 0.1*(N_{cycle}) \quad \text{... (5)}$$

The main advantages of systolic architecture are as follows as

Manuscript received Dec 10, 2015.

C. Bagavathi Shivakumar, Department of ECE, Sri Krishna college of Engineering and Technology, Coimbatore, Tamil Nadu, India.

- ✓ Systolic array takes less processing time.
- ✓ Systolic arrays have easily scalable architecture.
- ✓ It can do many tasks that single processor machines cannot attain.
- ✓ It turns some exponential problems into linear or polynomial time.
- ✓ Hardware utilization efficiency is high.
- ✓ It allows extremely high throughput with multidimensional arrays.
- ✓ Implementation of algorithm directly requires several thousands of gates but systolic architecture reduces the hardware used.

The applications of systolic architecture are matrix algorithms, DSP algorithms, data structures, graph algorithms, language recognition, dynamic programming, etc.

III. GENETIC ALGORITHM

Mapping transformations requires search for scheduling and processor allocation matrices which are time and space mapping respectively. Search techniques are adopted to find the optimal matrices among all possible combinations with required results such as maximum hardware utilisation efficiency and minimum processing time. The common search technique for mapping is heuristics [13]. Search problems can often benefit from an effective use of parallelism, in which many different possibilities are explored simultaneously in an efficient way. One way of such parallel evaluation of candidate solutions is Genetic Algorithm.

Genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics. Genetic Algorithm is a guided random search method where individuals in a set of population are randomly combined and modified until some termination condition like maximum number of iterations or a satisfactory fitness function value is reached. Genetic algorithms are implemented in a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals or phenotypes) to an optimization problem evolves toward better solutions. Genetic algorithms are robust and efficient. The three main operators: Reproduction selection, Crossover, and Mutation.

The reproduction selection operator may be implemented in a number of ways. The most common way is to create a biased roulette wheel where each current string in the population has a roulette wheel slot sized in proportion to its fitness.

Crossover proceeds after reproduction in two steps. First, member of the newly reproduced strings in the mating pool are mated at random. Second, each pair of strings undergoes crossing over. The pair exchanges their string parts along the crossover site to produce new candidates for next generation.

Mutation is the occasional (with small probability) random alteration of the value of the string position. This simply means changing a 1 to 0 and vice versa. Mutation never allows the probability of finding an optimal solution to be zero. The binary coded genetic algorithm is a probabilistic search algorithm that iteratively transforms population of binary coded individuals, each with an associated fitness value, into a new population of offspring objects using the principle of natural selection and using these operations. Following the model of evolution, they establish a population of individuals, where each individual corresponds to a point in the search space.

The elitist operator insures the GA will not get worse as it progresses. The elitist operator copies the best chromosome to the next generation bypassing the crossover and mutation operators. This guarantees the best chromosome will never decrease in fitness. The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

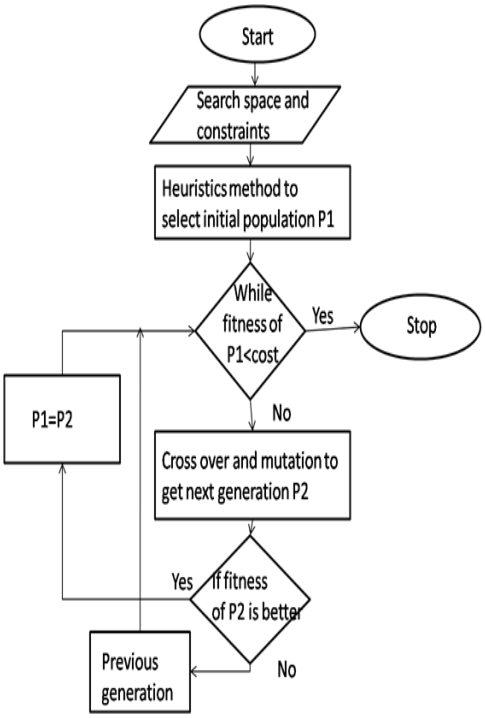

Fig. 1 Flowchart of Genetic Algorithm

Based on the problem, the operators can be tailored to meet the specifications. The termination criteria can be set using the number of iterations or when the algorithm halts in a particular value for more generations. GA is different from more normal optimization [12] and search procedures in five ways:

1. GA works with a coding of the parameter set, not the parameters themselves.
2. GA searches from a population of points, not a single point.
3. GA uses payoff (objective function) information, not derivatives or other auxiliary knowledge.
4. GA uses probabilistic transition rules, not deterministic rules.
5. GA coding can be easily adapted to any kind of problem.

The majority of optimization methods move from a single

point in the decision space to the next using some transition rule to determine the next point. This point-to point method is dangerous as it can locate false peaks in multimodal (many-peaked) search spaces. GAs overcome this by working from a database of points simultaneously (a population of strings), climbing many peaks in parallel. The probability of finding a false peak is reduced compared to methods that go point to point [13]. In specific, binary coded GA, gets an answer which improves with time. It is a good algorithm for a noisy environment.

Genetic algorithms are a very effective way of quickly finding a reasonable solution to a complex problem. Genetic algorithms are most effective in a search space for which little is known. They produce solutions that solve the problem in ways you may never have even considered.

IV. RESULTS

Consider a systolic architecture with 9 processors. The dependency matrix is chosen as a 4 x 9 matrix which represents a 4-dimensional filter problem to be implemented with 9 systolic processing elements. If systolic architecture is not to be employed, there would be 3 processors in 4 different directions. The total number of processors required will be 3^4. But using systolic system, 100 processors can be used to get the results. The results are obtained in MATLAB R2013b.

The processor allocation matrix (p) is of the size 2 x 4 and the scheduling matrix (s) is of the size 1 x 4. The space-time mapping gives the information of the processing element (PE) on which it is mapped and clock cycle at which it is executed. It is obtained using from p and s matrices. The delay edge matrix E gives the architectural details. It is obtained by product of space time matrix T and dependence matrix D.

$$E = T * D \quad \text{... (6)}$$

$$T = \begin{pmatrix} p^T \\ s^T \end{pmatrix} \quad \text{... (7)}$$

Genetic Search method is applied with five strings as initial population selected randomly.

The parameters of GA are set as follows: Population size: 5

Mutation probability is 2 bits

Crossover type is one-point crossover.

Crossover is between 1 and 2 pairs.

Cost function is varied to analyze the performance of algorithm for various criteria such as area and speed.

The fitness of the individual can be decided based on the parameter that is more of a constraint to the designer. If number of processing elements is limited, the more importance can be given to number of processors by giving higher proportion to that component in cost function as shown in Table I and Table II.

Case 1: Cost function is

$$cost function = 0.9*(N_{PE}) + 0.1*(N_{cycle}) \quad \text{-(8)}$$

$$p_1^T = \begin{bmatrix} 6 & 3 & 0 & 5 \\ 4 & 7 & 4 & 4 \end{bmatrix}$$

$$s_1^T = [8 \quad 10 \quad 13 \quad 2]$$

Best		Worst		Cost function	Generation of convergence
N_{PE}	N_{cycle}	N_{PE}	N_{cycle}		
1	175	100	493	261.2	2
100	50	194	173	317	3
100	25	240	296	408.1	4

Table I: Set of values indicating the travel of algorithm at different iterations. The cost function used for evaluation is given by equation (8)

Case 2: Cost function is

$$cost function = 0.75*(N_{PE}) + 0.25*(N_{cycle}) \quad \text{-(9)}$$

$$p_1^T = \begin{bmatrix} 10 & 1 & 14 & 1 \\ 5 & 11 & 0 & 9 \end{bmatrix}$$

$$s_1^T = [1 \quad 12 \quad 8 \quad 13]$$

Best		Worst		Cost function	Generation of convergence
N_{PE}	N_{cycle}	N_{PE}	N_{cycle}		
100	43	100	337	306.5	7
100	12	100	369	1102.7	7
100	20	100	312	275.5	9

Table II: Set of values with cost function defined in equation (9).

From Table I and II, it is obvious that there is an increase in cost function when there is less number of cycles with constant number of processors. The number of generations needed to reach the optimum value is also less.

If time of execution is a constraint, then importance can be given to number of cycles as given in table III and Table IV.

Case 3: Cost function is

$$cost function = 0.1*(N_{PE}) + 0.9*(N_{cycle}) \quad \text{-(10)}$$

$$p_1^T = \begin{bmatrix} 8 & 15 & 7 & 2 \\ 14 & 11 & 5 & 1 \end{bmatrix}$$

$$s_1^T = [6 \quad 12 \quad 14 \quad 6]$$

Best		Worst		Cost function	Generation of convergence
N_{PE}	N_{cycle}	N_{PE}	N_{cycle}		
100	42	100	458	777.2	4
89	78	209	474	826.2	8
100	22	100	525	749	3

Table III: Set of values with cost function defined in equation (10)

Case 4: Cost function is

$$cost function = 0.25*(N_{PE}) + 0.75*(N_{cycle}) \quad \text{-(11)}$$

$$p_1^T = \begin{bmatrix} 3 & 1 & 5 & 13 \\ 8 & 1 & 5 & 9 \end{bmatrix}$$

$$s_1^T = [5 \quad 6 \quad 5 \quad 9]$$

Best		Worst		Cost function	Generation of convergence
N_{PE}	N_{cycle}	N_{PE}	N_{cycle}		
100	41	100	410	598.25	2
100	41	10	302	445.25	7
100	38	100	271	455	6

Table IV: Set of values with cost function defined in equation (11)

By comprehending the values given in tables 1, 2, 3 and 4, we can understand that the range of values can be extremes with only one processor being used again and again. The cost function can also be decided based on the available resources.

V. Conclusion and Future Work

In this above mentioned problem, genetic algorithm is implemented for a 4-tap FIR filter in systolic architecture. GA is employed in mapping the processor with data for efficient reduction in hardware complexity and execution time. Hardware complexity is given by the number of processors and execution time is comprehended through number of cycles. Efficiency of Genetic Algorithm can be explained better by implementing in dimension greater than 6. This work can be extended as a comparison of various genetic algorithms such as hybrid and interval genetic algorithm.

REFERENCES

[1] Omar S. Al-Kadi and D. Watson, "Texture Analysis of Aggressive and Nonaggressive Lung Tumor CE CT Images", *IEEE Transactions on Biomedical Engineering*, Vol. 55, No. 7, July 2008.
[2] Dongming Peng and Ming Lu, "On Exploring inter-Iteration Parallelism Within Rate-Balanced Multirate Multidimensional DSP Algorithms" *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol 13, No. 1, January 2005. -
[3] Surin Kittitornkun and Yu Hen Hu, "Mapping Deep Nested Do-Loop DSP Algorithms to Large Scale FPGA Array Structures", *IEEE Transactions on very large scale integration (VLSI) systems*, Vol. 11, No. 2, April 2003
[4] Albert Y. Zomaya, Chris Ward, Ben Macey "Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues" *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 8, August 1999.
[5] Ricardo C. CorreA, Afonso Ferreira and Pascal Rebreyend, "Scheduling Multiprocessor Tasks with Genetic Algorithms", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 8, August 1999.
[6] H.F.Li, C.N.Zhang, R. Jayakumar, "Space-time mapping, latency of data flow and concurrent error detection in systolic arrays", *IEEE Proceedings – E*, Vol. 140, No. 1, January 1993.
[7] L. D. Whitley, A. E. Howe, S. Rana, J. P. Watson, L. Barbulescu, "Comparing Heuristic Search Methods and Genetic Algorithms for Warehouse Scheduling"
[8] Claudio Lopes De Souza, "Comparisons of intra-, inter population and modified recurrent selection methods", *Rev. Brazil. Genetics*, Vol. 16, No. 1, 1993.
[9] Swagatam Das, Amit Konar, and Ajith Abraham, "Particle Swarm Optimisation and Differential Evolution Algorithms: Technical Analysis, Application and Hybridisation Perspectives", *Studies in Computational Intelligence*, 116, 1-38, 2008.
[10] Keshab.K.Parhi, VLSI digital signal processing systems: Design and Implementation
[11] Taha, Operations Research, An introduction.
[12] Melanie Mitchell, An Introduction to Genetic Algorithms, 1999.
[13] David Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1989.

C. Bagavathi Shivakumar completed B.Tech ECE and M.Tech VLSI Design from Amrita University, Coimbatore, India. She has published over 10 papers in International Journals. Her areas of interests are VLSI Testing, artificial intelligence and programming.