iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** IV **Month of publication:** April 2018

DOI: http://doi.org/10.22214/ijraset.2018.4277

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

Multi Design Exploration and Register Minimization of Retimed Circuits Using GA in DSP Applications

Madhusudan[1], U. Eranna[2]
[1]*Department of Electronics and Communication Engineering, RRC, Visvesaraya Technology University, Belagavi*
[2]*Department of Electronics and Communication Engineering, BITM, Bellary*

***Abstract:** In this paper, Genetic Algorithm is applied to explore multi design architectures of sequential circuits which are used in DSP applications. Proposed algorithm will provide the different architectures with different iteration bound values and number of registers used in the explored designs. In this algorithm, we used mixed mutation (Gaussian and Cauchy distributed) for faster exploration of solutions. In VLSI implementation, the placement and routing design engineers can have multiple design solutions to a particular design with minimum registers. According to circuit design environment, design engineers can choose best fit circuit design for surrounded circuits or design solutions may be used for suitable placement and routing.*
***Keywords:** High level synthesis, Genetic Algorithm, Multi design exploration Register minimization, Retiming*

I. INTRODUCTION

High Level Synthesis (HLS) generates Register-Transfer Level (RTL) designs from behavioural specifications, in an automatic manner. The inputs to HLS are the behavioural specification of the design, the design constraints such as power, delay and performance etc, a module library which supports components of the design and an optimization function if any. The outputs of HLS are the RTL netlist, finite state machine symbols and schematic of the design. The main aim of HLS is to generate specified RTL that implements the specified behaviour while satisfying the design constraints and optimizing the given cost function.
Retiming is a technique which is used to change the location of delay elements in a sequential circuit without affecting the functional behaviour of the sequential circuits. Retiming has many applications such as critical path minimization (hence increasing clock speed of the circuit), register minimization, power minimization and HLS. This paper is concentrated on finding possible multiple register minimized sequential circuits using Genetic Algorithm (GA).

II. RELATED WORK

C. E. Leiserson and J. B. Saxe first time proposed the synchronous DFG (transformation technique known as *retiming*. In their work they showed that some delay elements can be relocated by maintaining the functionality of the circuit [1]. The issues in implementation of retiming for as large as 10,000 combinational cells and implementation issues to utilize the sparsity of DFGs to allow minimum period retiming and constrained minimum area retiming [2]. In [3], authors proposed the process of retiming for low power. It consists of relocating the delays in sequential circuits, while preserving its functional behaviour and low power characteristic has obtained by retiming.

III.PROBLEM FORMULATION

The focus of this paper is to explore all possible solutions of register minimization and to find minimum register count of a given sequential circuit for given clock period c as constraint.

IV.REGISTER MINIMIZATION

In a sequential circuit, if a node has several output edges and connected to the other nodes then the maximum delay elements required for that out going edge is the maximum delay element of any one edge. This is illustrated in Fig. 1. There are three outgoing edges from node U, each containing one, three and 7 delay elements on individual edges respectively. This can be modified and implemented as shown in Fig. 2. The total number of delay element required in Fig. 2 is only seven which is the maximum number of delay elements in an edge from node U to node V_3.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

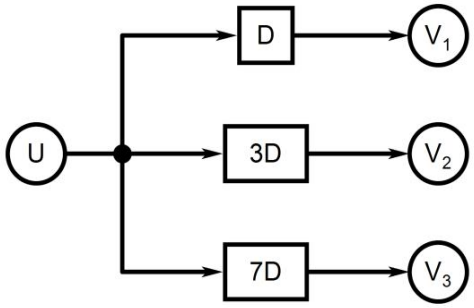


Fig. 1. An example for fan-out from node U.

The number of delay elements required to implement the fan-out edge in the retimed graph is $R_V = \max_{V \xrightarrow{e} ?}\{w_r(e)\}$.

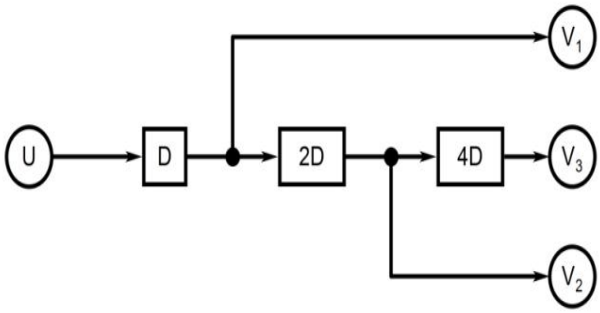


Fig. 2. Fan-out equivalent for Fig. 1 with Max (1, 3, 7) = 7 delay elements.

In the retimed circuit, the total register cost is the summation of all maximum count of delay elements, i.e. $COST = \sum R_V$. The problem formulation of retiming for register minimization under the clock period c as constraint (which will be obtained from retiming for clock period minimization algorithm) is to $Minimize\ COST = \sum R_V$ with subject to

1) $R_V \geq w_r(e)\ \forall\ V$ *and all edges* $V \xrightarrow{e} ?$ (This is Fan-out constraints)

2) $r(U) - r(V) \leq w(e)\ \forall\ U \xrightarrow{e} V$ (**Feasibility constraints**).

3) $r(U) - r(V) \leq w(e) - 1,\ \{\forall$ (U, V in G) $| D(U,V) > c\}$ **(Critical path constraints)**.

The condition 1, i.e. fan-out constraints makes sure that $R_V = \max_{V \xrightarrow{e} ?}\{w_r(e)\}$. In the process of retiming, feasibility and critical path constraints are calculated and verified.

The register minimization problem can be formulated by using "Gadget" model. A gadget is used to represent nodes with multiple nodes. An example of gadget is shown in Fig. 3. Fan out of the node U is k edges. An each edge has weight w (e).

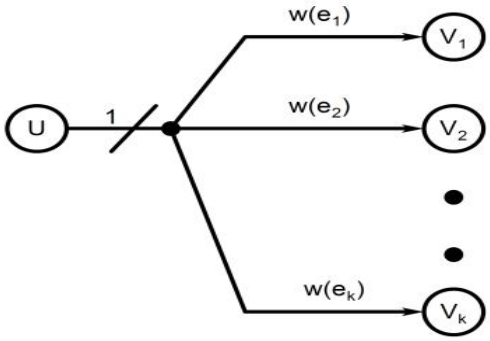


Fig. 3 A node U with k output edges.

A dummy terminating node Û with zero computation time is added to the DFG so that register minimization problem can be converted to linear model. A gadget for node U is shown in Fig. 4.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

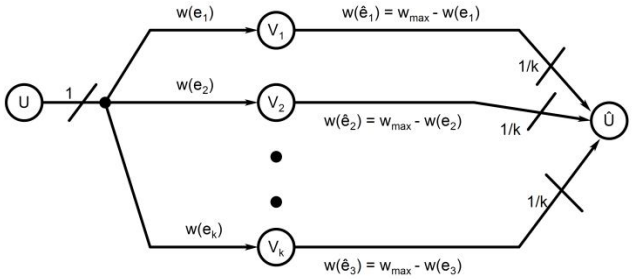


Fig. 4. A Gadget for node U.

V. MULTI DESIGN EXPLORATION

The following algorithm 1 will suggest how to find multi design of retiming for register minimization. The algorithm needs simple input format of circuits which are node computation time matrix, node incidence matrix and minimum clock period c of the circuit which was obtained from retiming for clock minimization algorithm.

A. Algorithm 1

1) Read node computation matrix
2) Read edge incident matrix
3) Read minimum clock time period in unit time (This we get from retiming algorithm)
4) Construct new graph same as given matrix
5) Add one extra node
6) Create Gadget matrix
7) Find cost from Gadget matrix
8) Let t_{max} = maximum of node computation matrix
9) M = t_{max} × n; where n is number of nodes
10) Create new adjacency matrix
11) Construct sub graph to calculate W and D matrix
12) Calculate W and D matrix; W gives path from each node U to V and D gives maximum computation time for each path from U to V.
13) Calculate shortest path of all nodes
14) Check for negative Cycles
15) If negative cycles exit
16) Print time taken for execution
17) Call Algorithm 2 (GA to get retime vectors)
18) Display the number of retime vector got from GA.
19) Get all unique retime vectors
20) Print how many retiming vectors
21) Print cost values of each retime vector
22) Construct and display all retimed circuits
23) Display number of registers used in each circuit.
24) Compute Minimum clock period required in each circuit and display it.
25) Display time taken by complete algorithm
26) End

VI. GA PROGRAM FOR RETIMING

The GA is a sub program as shown in Algorithm 2. This algorithm is used in retiming to generate different retiming vectors. GA utilized mixed (hybrid) mutation of Cauchy and Gaussian distributed weights for faster exploration of weights. Initial population values are assigned by considering uniform distribution.

A. Algorithm 2

1) *Function GA program*
2) Initialize population
3) Read N (number of node from Algorithm 1)
4) Initialize weights for population (uniformly distributed)
5) Let mutation vector = 0.0001 and

t1 = $\left(\sqrt{2n}\right)^{-1}$ and t2 = $\left(\sqrt{2\sqrt{n}}\right)^{-1}$

6) For termination = 1 to Iter (upper limit, Iter can be varied according to input matrix dimensions)
   a. For k = 1 to Population
   b. Create Gaussian mutation vector weights
   c. Create Cauchy mutation vector weights
   d. End For
7) Mix and sort both weights
8) Call Algorithm 3 (to verify constraints)
9) Get resultant values
10) Take nonzero values from resultant vectors
11) Get minimum value and position of minimum value
12) Initialize minimum value to fitness value
13) For k = 1 to 2 rimes of Population
14) Let r = 100 by 1 pseudorandom integers from 1 to 10
15) Weight score (k) = nonzero values of results (k) equals to values which are less than results (r)
16) End For
17) Sort weight score
18) Let final weights = rounded values of sorted mixed weights
19) Return

VII. CONSTRAINTS VERIFICATION

Algorithm 3 is used to verify constraints for given retiming vectors. In algorithm 3, the final retime vectors are received from Algorithm 2. Constraint matrices are constructed using W and D matrix. Finally the retime vectors are verified through these constraints. If any violation occurs in verification of constraints then algorithm 2 can be re-executed again to get new retime vectors. If all constraints are satisfied then retime vectors are sent to an Algorithm 1 to reconstruct the retimed circuit.

A. Algorithm 3

1) *Function constraints verification*
2) Get final weights from algorithm 2
3) Construct constraint matrix using W and D matrix
4) Verify all constraints by mixed weights
5) Return

VIII. EXPERIMENTAL RESULTS

Testing of the algorithm is done by providing various circuits. Population size of 50 had been taken in GA and medium number of iterations (approximately 20 to 50) has made to obtain the different retime vectors. Dimension of retime vector of problem in each case is equal to one more to the number of nodes available in corresponding circuit. From obtained retime vector retime circuit is generated. It is observed that there are different number of retime circuits in each case which are different in position of delays, critical path and number of delay elements. Following test cases illustrates the results of the algorithm.

A. Test Case 1

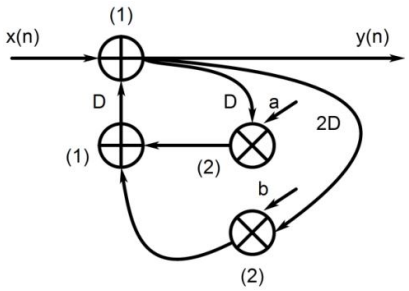


Fig. 5 Original DFG for test case 1

In DFG given in Fig. 5, the execution time of adder circuit is considered as 1 unit of time and for multiplier it is considered as 2 unit of time respectively. The inputs to the algorithm for the DFG shown in Fig. 5 are

1) Execution time of nodes = [1 1 2 2] all in unit time.

2) *Edge incidence matrix:*

TABLE 1: CONNECTION MATRIX FOR TEST CASE 1

Node No.	1	2	3	4
1	∞	∞	**1**	**2**
2	**1**	∞	∞	∞
3	∞	**0**	∞	∞
4	∞	**0**	∞	∞

3) Minimum clock period = 2

Output of the algorithm (First Run) for Test case 1:

a) The new graph is constructed and added a node with zero computation time as shown in Fig. 6.

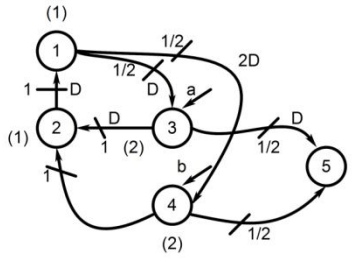


Fig. 6 Fan-out Gadget for given DFG 1

b) Gadget matrix from Fig. 6:

$$\begin{bmatrix} 0 & 0 & 0.5 & 0.5 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

c) Cost of gadget is calculated from Fig. 6:

$$COST = r(2) - r(3) - r(4) + r(5)$$

d) t_{max} = 2 and M = 10

e) New adjacency matrix :

$$\begin{bmatrix} \infty & \infty & 1 & 2 & \infty \\ 1 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

f) Sub graph matrix to calculate W and D:

$$\begin{bmatrix} \infty & \infty & 9 & 19 & \infty \\ 9 & \infty & \infty & \infty & \infty \\ \infty & -2 & \infty & \infty & 8 \\ \infty & -2 & \infty & \infty & -2 \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

g) W matrix from Fig. 6.

$$\begin{bmatrix} 0 & 1 & 2 & 12 & 2 \\ 1 & 0 & 2 & 3 & 3 \\ 1 & 0 & 0 & 3 & 1 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

h) D matrix from Fig. 6:

$$\begin{bmatrix} 1 & 4 & 3 & 3 & 3 \\ 2 & 1 & 4 & 4 & 4 \\ 4 & 3 & 2 & 6 & 2 \\ 4 & 3 & 6 & 2 & 2 \\ 1 & 1 & 2 & 2 & 0 \end{bmatrix}$$

i) Shortest path matrix (S_{UV}):

$$\begin{bmatrix} 16 & 7 & 9 & 19 & 17 \\ 9 & 16 & 18 & 28 & 26 \\ 7 & -2 & 16 & 26 & 8 \\ 7 & -2 & 16 & 26 & -2 \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

j) No negative cycles, Go to genetic algorithm to get retiming vectors

k) Elapsed time is 0.596495 seconds.

l) There are 10 unique retiming vectors in first run of algorithm:

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

m) Corresponding cost values:

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

$$[1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1]$$

n) Solution to retime vector 1:

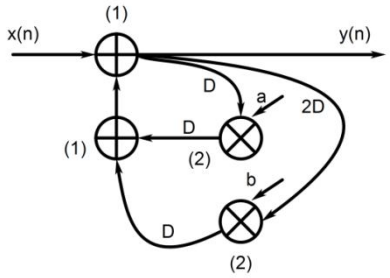


Fig. 7 Retime solution 1

o) Number of registers used in circuit: **4**
(Where a common delays from a single node output to multiple outputs can be shared).

p) Minimum clock period required is **2** unit time

q) Elapsed time is 4.831003 seconds.

r) Solution to retime vector 2:

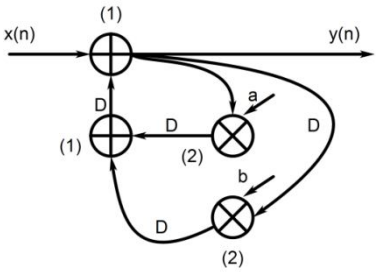


Fig. 8 Retime solution 2

s) Number of registers used in circuit: **4**

t) Minimum clock period required is **3** unit time

u) Elapsed time is 4.875967 seconds.

v) Solution to retime vector 3 is

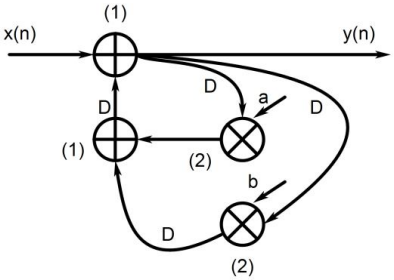


Fig. 9 Retime solution 3

w) Number of registers used in circuit: **3 ,** Minimum clock period required is **3**

x) Elapsed time is 4.917866 seconds, Solution to retime vector 5 is same as Fig. 7. So solution is not shown again.

y) Solution to retime vector 6 is

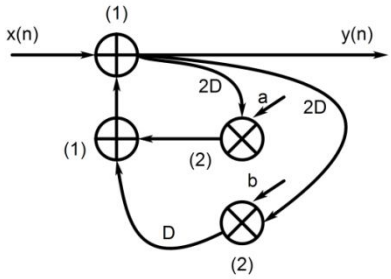


Fig. 10 Retime solution 6

z) Number of registers used in circuit: **3,** Minimum clock period required is **4** unit time

aa) Elapsed time is 5.041701 seconds; Retime vector 7 produced output same as Fig. 10. So the solution may be skipped.

bb) Solution to retime vector 8 is

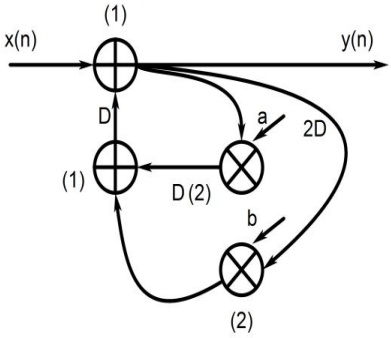


Fig. 11 Retime solution 8

cc) Number of registers used in circuit: **4,** Minimum clock period required is **3** unit time

dd) Elapsed time is 5.107259 seconds.

ee) Solution to retime vector 9 and 10 are same as Fig. 8. So solutions are not shown again.

Total elapsed time of the algorithm to produce all 10 solutions is 5.110290 seconds. Within 5.110290 seconds, the algorithm produced **5** different solutions with different architecture of sequential circuits and minimum clock time required by the designed circuit. The design engineers can choose any of above shown circuits according the surrounding design requirements.

TABLE 2 OUTPUT SUMMARY TO DFG

Sl. No	Solution circuit	Number of delay elements used	Minimum clock period required	Successive Elapsed time (sec)
1	1	**4**	**2**	4.831003
2	2	4	3	4.917866
3	3	3	3	4.917866
4	6	3	4	5.041701
5	8	4	3	5.107259

From Table 2 it is observed that the solution circuit 1 uses 4 delay elements and minimum clock period is 2, which is optimum value for given test case 1 DFG. Retime vector 4 will produce original circuit because all retime vector elements are zeros. If any retime vectors contains all elements as similar values, such as all zeros, all ones or all -1s, then those solution vectors can be skipped from producing retimed circuits. Due to generation of retime vectors by random numbers, other results can be expected from the algorithm when it is re-executed again. The algorithm was re-executed 2-3 times for test case 1 DFG and then it produced the similar solutions as shown above diagrams. Sometimes number of re-execution (by increasing iteration) of the algorithm can also be increased to cover all possible solutions.

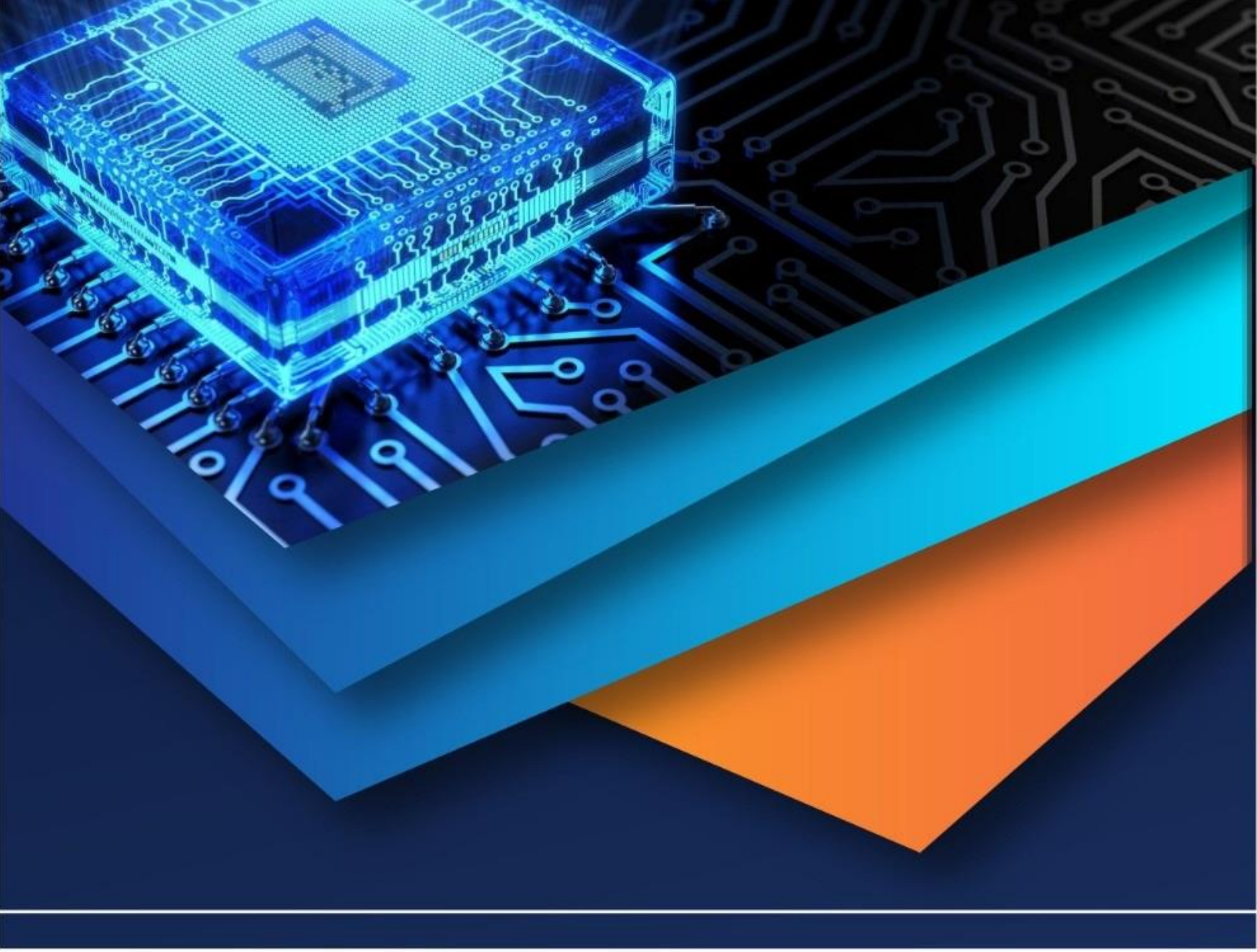
IX. CONCLUSION

In this paper, we have generated multiple retimed architecture circuits from a sequential circuit for DSP. These are useful in the process of placement and routing. Designer can choose whichever is useful for a particular design. In VLSI Design, clock period is

the most critical parameter which must have minimum value. In high level synthesis when proto type or initial circuit is designed, it will be satisfying the functional characteristics. There is no attention given to optimization of the designed sequential circuit. In this work, for a given minimum clock period c, we optimized number of delay elements by applying genetic algorithm. By this, the different circuits are explored with number of registers used and minimum clock period it required. In the retiming process, the role of genetic algorithm provides the multiple solution designs for the same circuit. In result, designer can have multiple choices to place the optimal architecture as it needed by surrounding environment. Hybridization of Gaussian and Cauchy mutation has given the faster exploration of solution space in result very short period of time required to get the optimal architecture. This work or algorithm can be merged with existing CAD tool for optimization or used as independently as algorithm for clock time minimization.

REFERENCES

[1] E. Leiserson, James B. Saxe, “Retiming synchronous circuitry”, Algorithmica, June 1991, Volume 6, Issue 1–6.
[2] N. Shenoy, R. Rudell, “Efficient Implementation of Retiming”, the best of ICCAD, pp 615-630, 2003.
[3] José Monteiro, Srinivas Devadas, “Retiming for Low Power”, Computer-Aided Design Techniques for Low Power Sequential Logic Circuits pp 97-110, 1997
[4] Xun Liu, M.C. Papaefthymiou, E.G. Friedman, “Retiming and clock scheduling for digital circuit optimization”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 21, Issue: 2, Feb 2002)
[5] T. W. O'Neil, E. H. M. Sha, “Retiming synchronous data-flow graphs to reduce execution time”, IEEE Transactions on Signal Processing, Volume 49, Issue 10, Oct 2001.
[6] T.C Denk, K. K. Parhi, “Two Dimensional Retiming”, IEEE Transactions on VLSI Systems, Vol. 7, Issue 2, June 1999.
[7] Pramod Kumar Meher, “On Efficient Retiming of Fixed-Point Circuits”, IEEE transactions on VLSI Systems, Volume: 24, number 4, April 2016.
[8] Weiqiang Liu, Liang Lu, Maire Máire, Earl E. Swartzlander, Roger Woods, “Design of quantum-dot cellular automata circuits using cut-set retiming”, IEEE Transactions on Nanotechnology , Volume 10, Issue 5, Sept. 2011.
[9] Vinita Pandey, S.C. Yadav, Priya Arora, “Retiming technique for clock period minimization using shortest path algorithm”, International Conference on Computing, Communication and Automation (ICCCA), 2016.
[10] Jingsong He, Zhenyu Yang, Xin Yao, Fellow, IEEE, “Hybridization of Evolutionary Programming and Machine Learning with k-Nearest Neighbor Estimation”, IEEE Congress on Evolutionary Computation (CEC 2007).
[11] P. Y. Calland, Anne Mignotte, Olivier Peyran, Yves Robert, Frederic Vivien “Retiming DAGs”, IEEE Transaction on Computer-Aided design of Integrated Circuits and systems, volume 17, number 12, December 1998. pp 1319-1323.
[12] Keshab. K. Parhi, “VLSI Digital Signal processing systems: Design and Implementation”, Wiley, India, pp 91-105.
[13] Rudra Pratap, “Getting started with MATLAB 7”, Indian Edition, Oxford University press, ISBN-10-0-19-568001-4.
[14] Krishna. M. Kauli, Bill. P. Buckles, U.Narayan Bhat, “A formal definition of data flow graph models”, IEEE Transactions on Computers, Volume C-35, NO 11, November 1986, pp 940-948.
[15] Naresh Maheshwari, Sachin. S. Sapatnekar, “Retiming control logic”, Integration, VLSI Journal, volume 28, issue 1, September 1999, pages 33-53.
[16] David E. Goldberg, “Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley Publishing Company, Inc.1989.
[17] Lance Chambers, “The Practical Handbook of Genetic Algorithms Application”, Second Edition, Chapman & Hall / CRC, 2001.
[18] Randy L. Haupt, Sue Ellen Haupt, “Practical Genetic Algorithms”, Second Edition, John Wiley & Sons, Inc. 2004.
[19] Pavankumar G.V, Shilpa K.C, “Time-Period Minimization of Circuit Execution in High Level Synthesis”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. 3 Issue 7, July 2014.
[20] Xin Yao, Yong Liu and Guangming Lin, “Evolutionary Programming Made Faster”, IEEE Transactions On Evolutionary Computation, Vol. 3, No. 2, July 1999.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY