

Time-Period Minimization of Circuit Execution in High Level Synthesis

Pavankumar G.V¹, Shilpa K.C²

¹M.Tech Student, Department of ECE- P.G Studies, Dr.AIT, Bangalore, Karnataka, India.

²Assistant Professor, Department of ECE, Dr.AIT, Bangalore, Karnataka, India.

Abstract-The fundamental growth in the VLSI design system based on circuit performance, this can be achieve high speed ,low power, minimum time period of the critical path. Some logic synthesis system deal with circuit by partitioning them into an interconnection of combinational logic component and registers. In this case, the optimization is done on a behavioural model in terms of state diagrams or equivalent representation. This paper focuses on the circuit optimization without partitioning the circuit and without altering the input-output characteristics of the circuit by using a special technique called retiming technique. Retiming technique reduces the computation time of the critical path thereby increasing the speed of the circuit without altering the input-output characteristics of the circuit. The basic idea is to alter the clock period by inserting and deleting register without affecting the circuit characteristic. Solution for retiming can be obtained by using shortest path algorithms such as Bellman-Ford and Floyd-Warshall algorithms. These shortest path algorithms can be used to solve several problems in VLSI. This paper focuses on how to increase the speed of the circuit by minimizing the clock period using retiming techniques and we using mathematical expressions on the basis of a program there by algorithms are implementing by using MATLAB.

Keywords: Circuit Optimisation, Retiming, Shortest Path Algorithms, Bellman-Ford, Critical Path, Floyd-Warshall, MATLAB etc.

1 LITERATURE SURVEY

[1] Charles.E.Leireson and James.B.Saxe, "Retiming Synchronous circuitry", *Algorithmica*.

This referred for determining an equivalent retimed circuit with minimum state. This paper describes a circuit transformation called retiming in which registers are added at some points a circuit and removed from others in such a way that the functional behaviour of the circuit is preserved. This paper shows that retiming can be used to transform a given synchronous circuit into a more efficient circuit under a variety of different cost criteria. This paper show that the problem of determining an equivalent retimed circuit with minimum state is a polynomial-time solvable.

This results yield a polynomial-time optimal solution of the problem of pipelining combinational circuitry with minimum register cost. These papers also give a characterization of optimal retiming based on an efficiently solvable mixed-integer linear programming problem.

[2] P.Y.Calland, Anne Mignotte, Olivier peyran, Yves Robust, Fredric Vivier, "Retiming DAGs", (1998).

It improving complexity of clock minimization problem. The increasing complexity of digital circuitry makes global

optimization no longer possible, a designer will consider only critical parts of a circuit. This paper discusses timing optimization problems when these critical parts can be represented by Directed Acyclic Graphs (DAGs). The main algorithm concerns the determination of the optimal clock period of a given circuit. This paper present an efficient solution, based on retiming technique.

[3] Giovanni De Michelli, "Synchronous logic synthesis algorithms for cycle-time minimization".

A new approach to logic synthesis of a digital synchronous circuit aimed at optimizing circuit performance. The synchronous logic circuit problem can be solved by considering algorithms that operate on the structural specification of a synchronous circuit. This paper introduce the concept of synchronous Boolean network and study transformation on this network that preserve input-output equivalence and that can improve the circuit cycle time or area.

[4] Keshab.K.Parhi, "VLSI Design processing system: Design and Implementation".

To understand retiming and application of retiming. Retiming is a transformation technique used to change the locations of delay elements in a circuit without affecting the input-output characteristics of the circuits. Retiming has many applications in synchronous circuit design. These applications include reducing the clock period of the circuit, reducing the number of registers in the circuit, reducing the power consumption of the circuit and logic synthesis. This book also refers to special case of retiming namely cutest retiming and pipelining.

[5] Rudra Pratap, "Getting started with MATLAB7".

To understand concepts of MATLAB. MATLAB provides an interactive environment with hundreds of built-in functions for technical computation graphical and animation. It also provides easy extensibility with its own high-level programming language. MATLAB built-in functions provide excellent tools for linear algebra computations, data analysis, quadrature, and many other types of scientific computations.

MATLAB is in the project for coding and as a CAD tool.

2 INTRODUCTION

The goal of VLSI design is to improve circuit performance which is measured in terms of speed, area and power without sacrificing the quality of implementation. A common means to achieve the goal is through the use of optimization tools that improve the performance of the circuit. Most digital design is synchronous logic circuits, that are interconnections of logic gate and registers with synchronous clocking. The techniques for synthesizing synchronous logic circuit have been lagging behind, due to additional complexity handling registers and feedback connections. The combinational portion of the circuit is optimized by combinational algorithms. This paper focuses on the circuit optimization without partitioning the circuit and without altering the input-output characteristics of the circuit by using a special technique called *Retiming technique*. Retiming technique reduces the computation time of the *critical path* thereby increasing the speed of the circuit without altering the input-output characteristics of the circuit. The basic idea is to alter the *clock period* by *inserting and deleting* register without affecting the circuit characteristic. Solution for retiming can be obtained by using *shortest path* algorithms such as *Bellman-Ford* and *Flyod-Warshall algorithms*. This paper focuses on how to increase the speed of the circuit by minimizing the clock period using retiming techniques. This can be simulated and synthesized circuit done by *MATLAB* version 7.10, the results are obtain on *MATLAB command window*.

3 HIGH LEVEL SYNTHESIS

High level synthesis is sometimes referred to as *C-synthesis*, *electronic system level synthesis*, behavioural synthesis is an *automated* design process that interpret an algorithmic description of a desired behaviour and creates hardware that implements that behaviour. The synthesis task is to take a specification of the behaviour required of a system and a set of constraints and goals to be satisfied and to find a structure that implements the behaviour while satisfying the goals and constraints. One of the tasks of synthesis is to find the structure that best meets the constraint such as limitation on *cycle time*, *area*, *power*, while minimizing other *costs*.

4 IMPORTENCE OF RETIMING

Retiming is a powerful sequential circuit *optimization technique* for improving the performance of sequential circuits. *Retiming* is a transformation technique used to change the locations of *delay* elements in a circuit without affecting the *input-output characteristics* of the circuit. Retiming has many applications in synchronous circuit design. These applications include reducing the *clock period* of the circuit, reducing the number of registers in the circuit, reducing the *power consumption* of the circuit and *logic synthesis*. Retiming can be used to increase the *clock rate* of a circuit by reducing the *computation time* of the critical path.

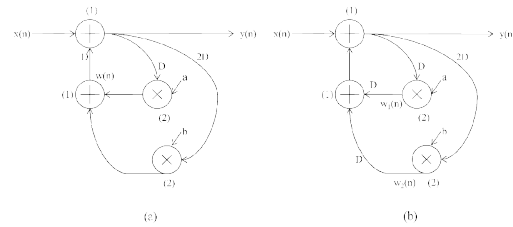


Fig 1: Two IIR Filter with the computation times of the nodes are shown in the parentheses.

Fig(1) where the numbers in brackets indicate the computation time associated with each processing node.

This filter is described by

$$y(n) = w(n - 1) + x(n). \quad (A)$$

However $w(n)$ itself is recursively related to $y(n)$ by

$$w(n) = ay(n - 1) + by(n - 2). \quad (B)$$

Substituting the value of $w(n)$ from (1) into (2) we get the final equation i.e.

$$y(n) = ay(n-2) + by(n-3) + x(n)$$

$$y(n) = w1(n-1) + w2(n-1) + x(n), w1(n) = ay(n-1), w2(n) = by(n-2).$$

Retiming is mainly used to reduce the critical path in synchronous circuits.

The critical path of Fig1(a) passes through 1 multiplier and 1 adder and has a computation time of 3.u.t, so this filter cannot be clocked with a clock period of less than 3.u.t. The retimed filter is shown in the Fig1(b) has a critical path that passes through 2 adders and has a computation time of 2 .u.t, so this filter cannot be clocked with a clock period less than 2.u.t. The clock period has been reduced from 3.u.t to 2.u.t or by 33%.

Advantages: (1) Retiming can be used to decrease the number of *registers* in the circuit Fig1 (a) uses 4 registers while the Fig1 (b) uses 5 registers. Since retiming can affect the clock period and the number of registers. (2) Retiming can be used to reduce the *power consumption* of a circuit by *reducing switching*. (3) Placing registers at the inputs of nodes with large capacitances can reduce the switching activities at these nodes, which can lead to *low-power* resolutions.

5 QUANTITATIVE ANALYSIS OF RETIMING

Retiming maps a circuit G to a retimed circuit G_r . In the context of retiming the terms circuit and graph and DFG are often used interchangeably. A solution is characterized by a value $r(v)$ for each node v in the graph. Let $w(e)$ denotes the weight e in the original graph G and let $w_r(e)$ denote the weight of the edge in the retimed graph G_r . The weight of the edge $U \rightarrow V$ in the retimed graph is computed from the weight of the original graph using.

$$w_r(e) = w(e) + r(v) - r(u) \quad (1)$$

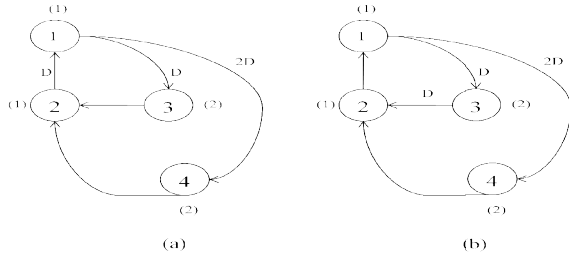


Fig 2: (a) A DFG (b) The retimed DFG obtained using $r(1) = 0, r(2) = 1, r(3) = 0, r(4) = 0$.

Example1: The edge $3 \rightarrow 2$ in the retimed DFG contains

$$w(3 \rightarrow^e 2) = w(3 \rightarrow^e 2) + r(2) - r(3)$$

$$= 0 + 1 - 1 = 0 \text{ delays and the edge } 2 \rightarrow 1 \text{ contains}$$

$$w(2 \rightarrow^e 1) = w(2 \rightarrow^e 1) + r(1) - r(2)$$

$$= 1 + 0 - 1 = 0 \text{ delays}$$

A retiming solution is feasible if $Wr(e) \geq 0$ holds for all edges. While the solution that maps that Fig2(a) to Fig2(b) is feasible because all of the edges in Fig2(b) have nonnegative weights, the solution $r(1) = 0, r(2) = 1, r(3) = 0$ and $r(4) = 0$ in infeasible because, for example, the edge $3 \rightarrow 2$ in the retimed system contains

$$Wr(3 \rightarrow^e 2) = w(3 \rightarrow^e 2) + r(2) - r(3)$$

$$= 0 + (-1) - 0 = -1.$$

6 SOLVING SYSTEMS OF INEQUALITIES

Given a set of M inequalities in N variables, where each inequality has the form $r_i - r_j \leq k$ for integer values of k , one of the shortest path algorithm can be used to determine if a solution exists and to find a solution if one does indeed exist. According to the steps by the design flow is achieve by the flow chart Fig3.

1). Draw a constraint graph.

(a). Draw the node i for each of the N variable $r_i, i = 1, 2, \dots, N$.

(b). Draw the node $N+1$.

(c). For each inequality $r_i - r_j \leq k$, draw the edge $j \rightarrow i$ from node j to node i with length k .

(d). For each node $i, i=1, 2, \dots, n$, draw the edge $N+1 \rightarrow i$ from the node $N+1$ to node i with length 0 .

2). Solve using shortest path algorithm.

(a). The system of inequalities has a solution if and only if the constraint graph contains no negative cycles. If a solution exists, one solution is where r_i is the minimum-length path from node $N+1$ to the node i . The flow chart

construction of constraint is done using following procedure as shown in above and explained.

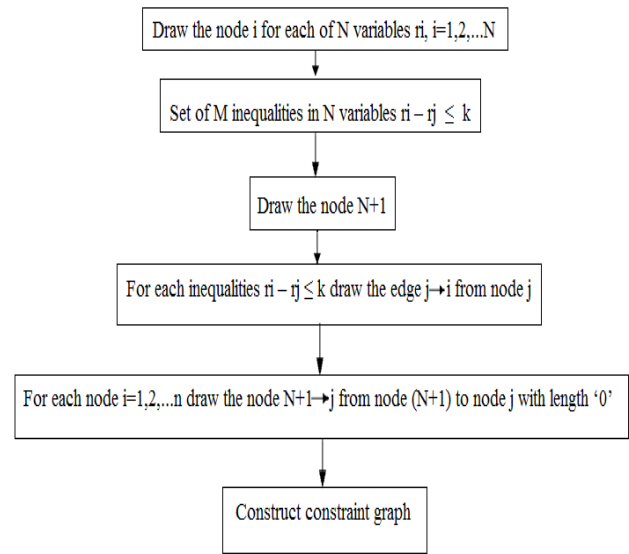


Fig 3: Flow chart for construction of constraint graph.

Example2:

In this example, it is demonstrated how shortest path algorithm can be used to solve a system of $M = 5$ inequalities in $N=4$ variables.

$$r_1 - r_2 \leq 0$$

$$r_3 - r_1 \leq 5$$

$$r_4 - r_1 \leq 4$$

$$r_4 - r_3 \leq -1$$

$$r_3 - r_2 \leq 2$$

Step1: Draw the constraint graph as show in Fig 4.

Step2: solve using shortest path algorithm, here Bellman-Ford shortest path algorithm is used, where the origin u is the node 5, we find that there are no negative cycles.

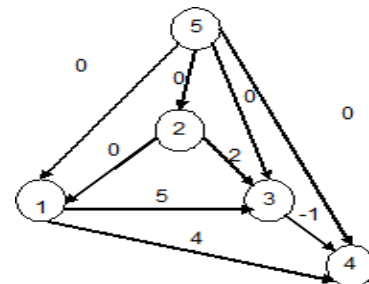


Fig 4: constraint graph for example 2.

So the Retimed solution is as follows below.

$$\begin{aligned} r^{(4)}(1) &= 0 \\ r^{(4)}(2) &= 0 \\ r^{(4)}(3) &= 0 \\ r^{(4)}(4) &= -1 \\ r^{(4)}(5) &= 0 \end{aligned}$$

7 RETIMING FOR CLOCK PERIOD MINIMIZATION

Retiming algorithm for minimizing the clock period of a synchronous circuit. For a circuit G , the minimum feasible clock period is the computation time of the critical paths with no delays. Mathematically, the minimum feasible clock period, $\square(G)$ is defined as

$$\square(G) = \max \{t(p) : w(p)=0\}$$

The two quantities $W(u,v)$ and $D(u,v)$ are used in the algorithm.

- Let $M = t_{\max} * n$, where t_{\max} is the maximum computation time of the nodes in G and n is the number of nodes in G .
- Form a new graph G' which is same as G except the edge weights are replaced by $w'(e) = Mw(e) - t(u)$ for all edges from $u \rightarrow v$.
- Solve the all-pairs shortest path problem G' . Let S'_{uv} be the shortest path from u to v .
- If $u \neq v$, then $w(u,v) = \lfloor S'_{uv}/M \rfloor$ and $D(u,v) = MW(u,v) - S'_{uv} + t(v)$. If $u=v$, then $w(u,v)=0$ and $D(u,v) = t(u)$.

To demonstrate this algorithm, the values of $W(u,v)$ and $D(u,v)$ are computed for the DFG in Fig5.

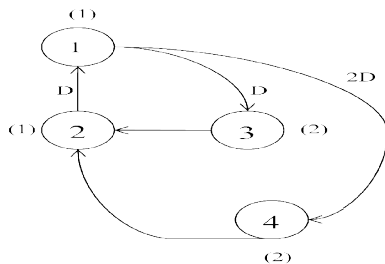


Fig 5: A DFG

In the first step, $t_{\max} = 2$ and $n=4$, so $M=8$. The new graph G' is shown in Fig6.

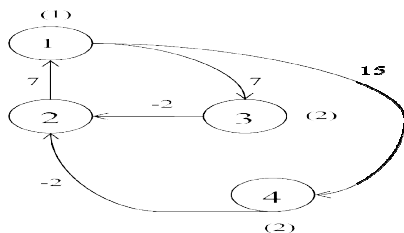


Fig 6: The graph G' used to compute $W(u,v)$ and $D(u,v)$ for the DFG.

Table: 1

S'_{uv}	1	2	3	4
1	12	5	7	15
2	7	12	14	22
3	5	-2	12	20
4	5	-2	12	20

$W(u,v)$	1	2	3	4
1	0	1	1	2
2	1	0	2	3
3	1	0	0	3
4	1	0	2	0

$D(u,v)$	1	2	3	4
1	1	4	3	3
2	2	1	4	4
3	4	3	2	6
4	4	3	6	2

Table 1: Values of S'_{uv} , $W(u,v)$ and $D(u,v)$ for the DFG in the Fig6.

The values of $W(u,v)$ and $D(u,v)$ are used to determine if there is a retiming solution that can achieve a desired clock period. Given a desired clock period c , there is a feasible retiming solution r such that $\square(G_r) \leq c$ if the following constraints hold.

- (Feasibility constraint) $r(u) - r(v) \leq w(e)$ for every edge $u \rightarrow v$ of G , and
- (critical path constraint) $r(u) - r(v) \leq w(u,v) - 1$ for all vertices u,v in G such that $D(u,v) > c$.

If $D(u,v) > c$, then $W(u,v) + r(v) - r(u) \geq 1$ must hold for the critical path to have computation time less than or equal to c . This leads to the critical path constraint. For the DFG shown in Fig6, if c is chosen to be 3, the inequalities

$r(u) - r(v) \leq w(e)$ for every edge $u \rightarrow v$ are

$$\begin{aligned} r(1) - r(3) &\leq 1 \\ r(1) - r(4) &\leq 2 \\ r(2) - r(1) &\leq 1 \\ r(3) - r(1) &\leq 0 \\ r(4) - r(2) &\leq 0 \end{aligned} \tag{A1}$$

And the inequalities $r(u) - r(v) \leq w(u,v) - 1$ for all vertices u,v in G such that $D(u,v) > 3$, found using table1 are.

$$\begin{aligned} r(1) - r(2) &\leq 0 \\ r(2) - r(3) &\leq 1 \\ r(2) - r(4) &\leq 2 \\ r(3) - r(1) &\leq 0 \end{aligned} \tag{A2}$$

$$\begin{aligned} r(3)-r(4) &\leq 2 \\ r(4)-r(1) &\leq 0 \\ r(4)-r(3) &\leq 1 \end{aligned}$$

If there is a solution to the 12 inequalities in (A1) & (A2) is shown in the Fig7, then the solution is a feasible retiming solution such that the circuit can be clocked with period $c=3$.

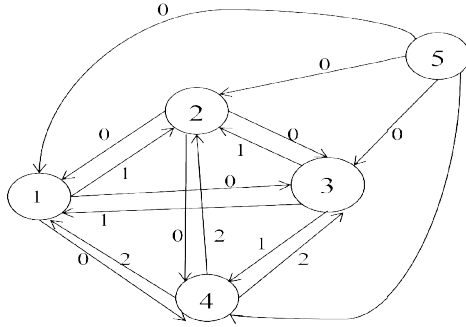


Fig 7: constraint graph for inequalities in (A1) and (A2).

For $c=2$, the inequalities $r(u)-r(v) \leq w(e)$ for every edge $u \rightarrow v$ are given in (A2) and the inequalities $r(u)-r(v) \leq w(u,v)-1$ for all vertices u,v in G

Such that $D(u,v) > 2$ are.

$$\begin{aligned} r(1)-r(2) &\leq 0 \\ r(1)-r(3) &\leq 0 \\ r(1)-r(4) &\leq 1 \\ r(2)-r(3) &\leq 1 \\ r(2)-r(4) &\leq 2 \\ r(3)-r(1) &\leq 0 \\ r(3)-r(2) &\leq -1 \\ r(3)-r(4) &\leq 2 \\ r(4)-r(1) &\leq 0 \\ r(4)-r(2) &\leq -1 \\ r(4)-r(3) &\leq 1 \end{aligned} \tag{A3}$$

The equations for a feasible retiming solution with clock period $c= 2$ are given in (A2) and (A3). Combining the equations in (A2) and (A3) that the identical left-hand sides results in the set of equations.

$$\begin{aligned} r(1)-r(3) &\leq 0 \\ r(1)-r(4) &\leq 1 \\ r(2)-r(1) &\leq 1 \\ r(3)-r(2) &\leq -1 \\ r(4)-r(2) &\leq -1 \\ r(1)-r(2) &\leq 0 \\ r(2)-r(3) &\leq 1 \\ r(2)-r(4) &\leq 2 \\ r(3)-r(1) &\leq 0 \\ r(3)-r(4) &\leq 2 \\ r(4)-r(1) &\leq 0 \\ r(4)-r(3) &\leq 1 \end{aligned} \tag{A4}$$

The constraint graph for 12 inequalities in (A4) is shown in Fig8.

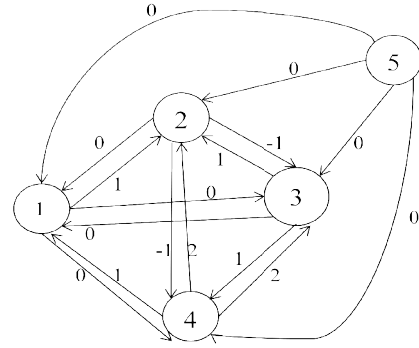


Fig 8: constraint graph for the inequalities in (A4).

Using Bellman-Ford algorithm. if no negative cycles for $c=2$, node at 5, single source shortest path which is $r(1) = -1, r(2) = 0, r(3) = -1$ and $r(4) = -1$. The solution, which has clock period $\phi(G_r) = 2$, is the retimed version of Fig6 as shown in Fig9.

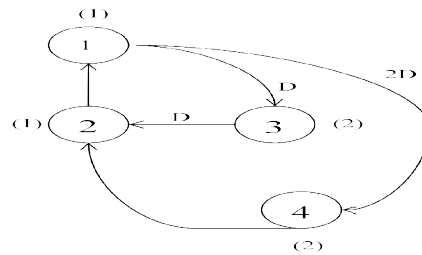


Fig 9: Retimed DFG

a) Types of Retiming:

The different types of retiming are, (a) Clock Period, (b) Area, (c) Power, (d) Testability.

b) Properties of retiming:

- Retiming does not change the number of delays in a cycle.
- Retiming does not alter the iteration bound in a DFG as the number of delays in a cycle does not change.
- Adding the constant value j to the retiming value of each node does not alter the number of delays in the edges of the retimed graph.

The flow of design is steps in flow chart below. The above flow chart is described above examples 7.

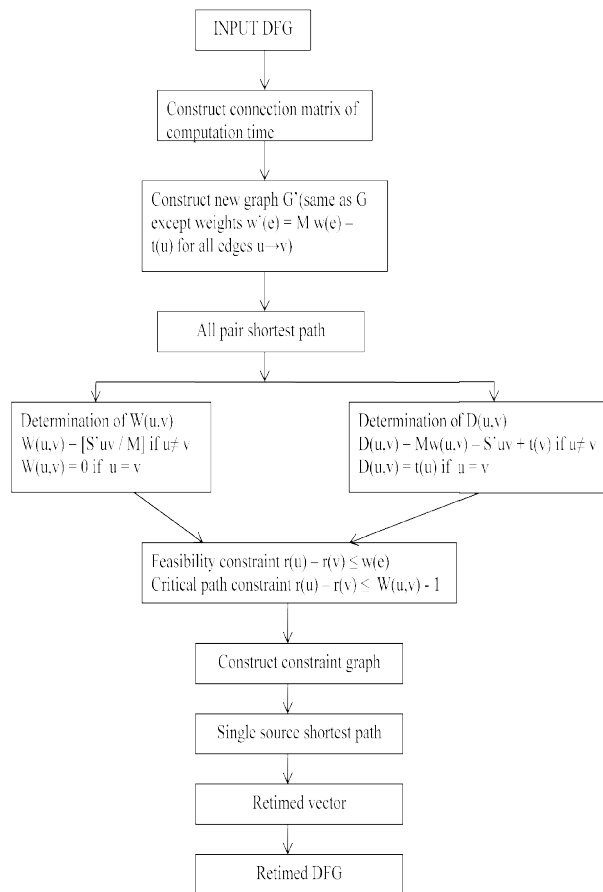


Fig 10: Flow chart of retimed DFG from the input DFG.

8 PROPOSED DESIGN

MATLAB is a software package for high-performance numerical computation & visualization. It provides an interactive environment with hundreds of built in functions for technical computation, graphics & animation. It also provides extensibility with its own high-level programming language. The name MATLAB stands for Matrix Laboratory Matlab's built-in functions provide excellent tools for linear algebra computations, data analysis, signal processing, optimization, numerical solution of ordinary differential equations (ODEs), quadrature & many other types of scientific computations. There are also several optional 'toolboxes' available from the developers of MATLAB. These toolboxes are collections of functions written for special applications such as symbolic computation, image processing, statistics, and control system design, neural networks etc. The basic building block of MATLAB is the matrix. The fundamental data type is the array. Vectors, scalars, real matrices & complex matrices are

all automatically handled as special cases of the basic data types.

MATLAB Windows: MATLAB Desktop basically works on the following sub windows.

(a) Command window: This is the main window. It is characterized by MATLAB command prompt (>>), when you launch the application program, MATLAB puts you in this window. In the command window, we have current directory where all our files are listed.

(b) Figure window: The output of all graphics commands typed in the command window is flushed to the graphics or figure window.

(c) Editor window: This is where we write, edit, create & save your own programs in files called as M-files.

9 SYNTHESIS AND SIMULATION RESULTS

Retiming is a transforming technique used to change the location of delay elements of the circuit without affecting input-output characteristics of the circuit.

9.1 Design 1

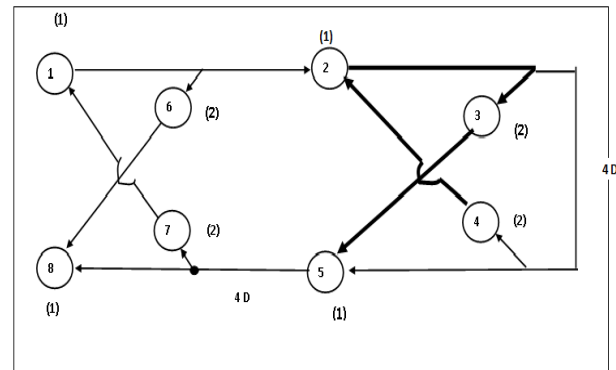


Figure 11: Input DFG1

9.2 Matrix form of the input DFG:

The matrix is given by:

fckt =

-1	0	-1	-1	-1	0	-1	-1
-1	-1	0	4	4	-1	-1	-1
-1	-1	-1	-1	0	-1	-1	-1
-1	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	4	4
0	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

- The computation time matrix is given by,

$$n+ = [1 \ 1 \ 2 \ 2 \ 1 \ 2 \ 2 \ 1]$$

9.3 Output of DFG 1:

When we Debug and Run, it requests for execution giving the possible clock periods. From those clock periods, we have to select. If for the clock period which we have selected, there exist negative cycle, then it displays 'negative cycle exists' and output will not be displayed. Then we have to select the other options for which the output will be displayed DFG.

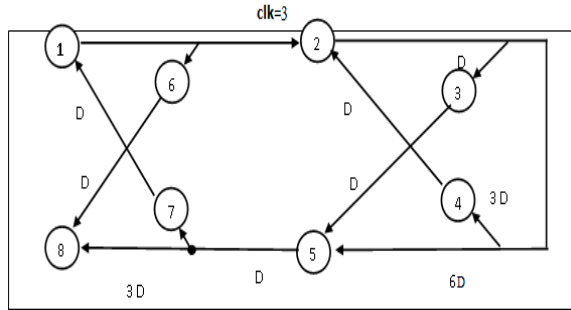


Fig 12: Retimed DFG of input DFG1 for CLK=2.

9.3 MATLAB Output Command Window

```

MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work\proj_retime

DFG BEFORE RETIMING
oclk =

    Inf     0     Inf     Inf     Inf     0     Inf     Inf
    Inf     Inf     0     4     4     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     0     Inf     Inf     Inf
    Inf     0     Inf     Inf     Inf     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     Inf     Inf     4     4
    Inf     Inf     Inf     Inf     Inf     Inf     Inf     0
    0     Inf     Inf     Inf     Inf     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     Inf     Inf     Inf     Inf

DESIRED CLOCK PERIOD
ckp =

    3

RETIME SOLUTION
rtv =

    -1
    -1
     0
    -1
     0
     0
    -2
     0

DFG AFTER RETIMING
retcir =

    Inf     0     Inf     Inf     Inf     1     Inf     Inf
    Inf     Inf     1     4     5     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     0     Inf     Inf     Inf
    Inf     0     Inf     Inf     Inf     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     Inf     Inf     2     4
    Inf     Inf     Inf     Inf     Inf     Inf     Inf     0
    1     Inf     Inf     Inf     Inf     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     Inf     Inf     Inf     Inf
    
```

In this command window showing output which contains input of the given circuit such that source to destination with matrix form. In which we should give computational time of the given circuit need to process the critical path. Then after run the program with desired clock period that is clk=3, this circuit cannot clock with clock period 3u.t.this is the minimum clock period after that we got the result for retiming solution this is for single node shortest path using Bellman Ford algorithm from origin node to particular node i.e. shortest path from the source to the destination is as shown in the command window. To solve the mathematical expression of possible constraints and critical path constraints we got the retimed circuit of the graph as shown in the command window. The clock period can be reduced from 7 u.t. i.e. 33%.

```

MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work\proj_retime

0     Inf     Inf     Inf     Inf     Inf     Inf     Inf
Inf     Inf     Inf     Inf     Inf     Inf     Inf     Inf

DESIRED CLOCK PERIOD
ckp =

    3

RETIME SOLUTION
rtv =

    -1
    -1
     0
    -1
     0
     0
    -2
     0

DFG AFTER RETIMING
retcir =

    Inf     0     Inf     Inf     Inf     1     Inf     Inf
    Inf     Inf     1     4     5     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     0     Inf     Inf     Inf
    Inf     0     Inf     Inf     Inf     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     Inf     Inf     2     4
    Inf     Inf     Inf     Inf     Inf     Inf     Inf     0
    1     Inf     Inf     Inf     Inf     Inf     Inf     Inf
    Inf     Inf     Inf     Inf     Inf     Inf     Inf     Inf
    
```

Fig 13: All output of input DFG1 in matrix form MATLAB command window.

9.4 Design 2

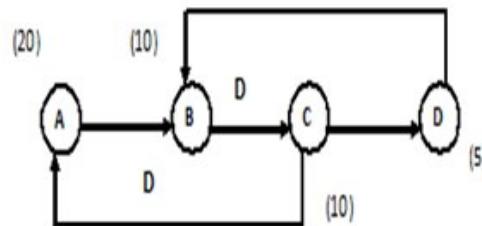


Fig 14: Input DFG2

9.5 Matrix form of input DFG2.

Similarly, Input should be give matrix form for any DFG. The input matrix is given by:

$$f_{ckt} = \begin{bmatrix} -1 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 0 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

- The computation time matrix is given by,

$$n+ = [20 \ 10 \ 10 \ 5]$$

9.6 Output of DFG2:

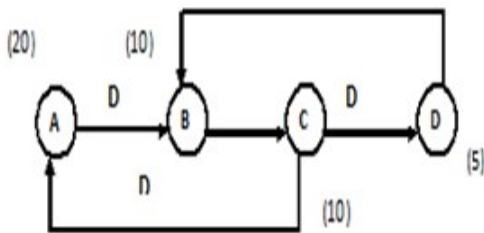


Fig 15: Retimed DFG of input DFG2 for CLK=25

9.6 MATLAB Output Command Window

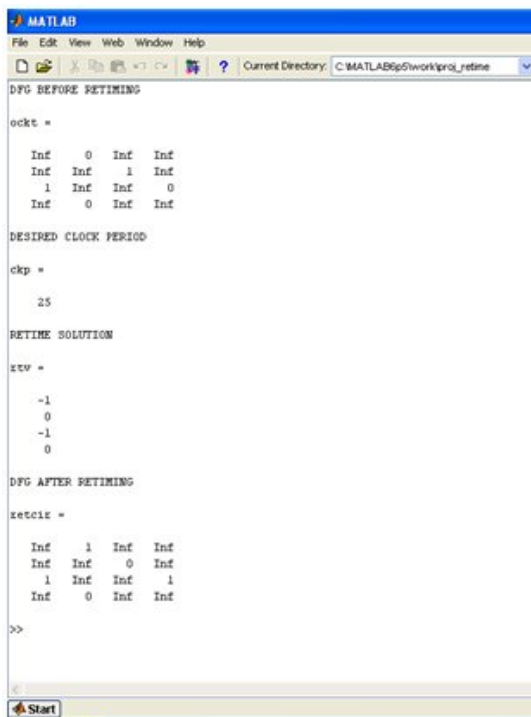


Fig 16: All outputs of DFG2 in matrix form MATLAB command window.

In this command window showing output this contains input of the given circuit such that source to destination with matrix form. In which we should give computational time of the given circuit need to process the critical path. Then after run the program with desired clock period that is $clk=3$, this circuit cannot clock with clock period 3u.t.this is the minimum clock period after that we got the result for retiming solution this is for single node shortest path using Bellman Ford algorithm form origin node to particular node i.e. shortest path from the source to the destination is as shown in the command window. To solve the mathematical expression of possible constraints and critical path constraints we got the retimed circuit of the graph as shown in the command window. The clock period can be reduced from 30 u.t to 25u.t. i.e. 4%.

10 CONCLUSIONS

From VLSI Design perspective, speed is one of the most important factors which are always expected to have maximum values. With the raw or initial circuit which has been designed by the circuit designer just satisfying the input-output desired characteristics, where no attention has been given to optimize the parameter. With the retimed concept, which has been implemented in my project work can be utilized to optimize speed parameter. This will be very helpful to get the optimal design without involving the human expertise to increase the speed of the circuit. In the implementation of retiming concept, the shortest path algorithms having also some good contribution to find the solution of system of inequalities. The developed solution can be utilized individually as a CAD tool for any defined circuit to increase the speed or can be integrated with the other existing CAD tool.

10 DIRECTION OF THE FUTURE WORK

Although significant amount of research has been performed on retiming, some key issues need to be addressed before retiming is widely accepted by the design community. We now present some of these issues, and thoughts on them.

- a). **Restriction on design styles:** The traditional retiming methods impose some design style restrictions on the circuits they can handle. Many of these styles are very popular in high performance designs, and these restrictions need to be relaxed before retiming can be applied to a large section of designs. Some of these are,
 - a). Gated clocks
 - b). Multi-cycle paths
 - c). Registers with logic
 - d). Mix of register types
 - e). Don't care timing assertions.
- b). **Verification:** One of the main road-blocks in the use of retiming is the problem of verifying the correctness of retimed circuits. Unfortunately sequential verification is a hard problem.
- c). **Position in design flow:** The sequential nature of retiming makes it hard for the designer to recognize the retimed circuit.
- d). **Improve delay models:** One of the main drawbacks of the current retiming algorithms is that they assume constant

gate delays. The delay of a gate does not depend on the number of its fan-outs. Although retiming does not change the topology of a circuit, the sharing of FF's at the output of a gate can change the number of fan-outs for that gate.

e). Retiming and logic synthesis: The sequential nature of retiming makes it possible to improve the quality of results obtained by subsequent combinational logic optimization and also improves global sequential transform.

APPENDIX

SHORTEST PATH ALGORITHMS

In shortest path problem [5], given a weighted, directed graph $G(V,E)$ with weight functions $w:E \rightarrow R$, mapping edges to real-valued weights. The weight of the path $P = \langle v_0, v_1, \dots, v_n \rangle$ is the sum of the weight of the constituent edges. We define shortest path from u to v by,

$$S(u,v) = \begin{cases} \min \{w(p): u \rightarrow v\} & \text{if there is path from } u \text{ to } v \\ \text{Infinite} & \text{otherwise} \end{cases}$$

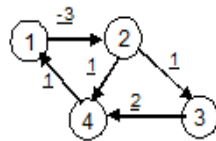
The shortest path from u to v is denoted by S_{uv} .

Variants of shortest path algorithm

1. Single-destination shortest path algorithm.
Finds a shortest path to a given destination vertex v from vertex u .
2. Single-pair shortest path algorithm.
Finds a shortest path from u to v for a given vertex (u,v) .
3. All-pairs shortest path algorithm.
Finds a shortest path from u to v for every pair of vertices u to v .

A1 Negative cycles

A negative cycle is a directed graph in a directed cycle such that the sum of the lengths on the edges of the cycle is negative.



FigureA.1: Graph containing negative cycles

The above graph contains 2 cycles:

1. $2 \rightarrow 4 \rightarrow 1 \rightarrow 2$.
2. $2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2$

The first cycle $2 \rightarrow 4 \rightarrow 1 \rightarrow 2$ is a negative cycle because the sum of the edge weight is negative.
 $1+1+(-3) = -1$.

The second cycle $2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2$ is not a negative cycle because the sum of the edge weight is not negative.
 $1+2+1+(-3) = 1$.

If there is no negative cycles the algorithm returns true and provide some information about the length of the shortest path between nodes.

If there exist atleast one negative cycle in the graph, the algorithm returns false and no shortest path exist.

A2 Bellman-Ford Algorithm

The algorithm is a single-point or single-destination algorithm. If no negative cycle exists in the graph it finds the shortest path from an arbitrarily chosen node u (called origin) to each node in the graph. For a graph with n nodes, the algorithm constructs $n-1$ vectors $r^{(k)}$ where $k=1,2,\dots,n-1$ which are of size $n \times 1$. If True is returned, then $r^{(n-1)}(v)$ is the shortest path from node u to node v . If false is returned then there exist a negative cycle. The $r^{(k)}(u)$ represent the u -th entry in column vector $r^{(k)}$.

Algorithm

```

R(1)(u) = 0
For k = 1 to n
  If k ≠ u
  R(1)(k) = w(u → k)
For k = 1 to n-2
  For v = 1 to n
  R(k+1)(v) = r(k)(v)
  For W = 1 to n
  If r(k+1)(v) > r(k)(w) + w(W → v)
  r(k+1)(v) = r(k)(w) + w(W → v)
  For v = 1 to n
  For W = 1 to n
  If r(n-1)(v) > r(n-1)(w) + w(W → v)
  Return FALSE and exit
  Return TRUE and exit
    
```

EXAMPLE 1

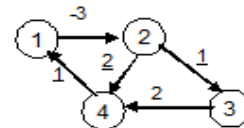


Figure A.2: Graph

In example1, the Bellman-Ford is used to find the shortest path from the node $u = 2$ to the nodes in the graph in figA.2. The $r^{(k)}(v)$ $k = 1,2,3$ and $v = 1,2,3,4$ are shown in TableA.1.

The Bellman-Ford algorithm returns true in this example, so the value of $r^{(3)}(v)$ is the shortest path from nodeA.2 to node v for $v=1,2,3,4$.

$R^{(k)}(v)$	$k = 1$	$k = 2$	$k = 3$
$v = 1$	Infinite	3	3
$v = 2$	0	0	0
$v = 3$	1	1	1
$v = 4$	2	2	2

TableA.1: values for $r^{(k)}(v)$ for example 1

A3 Floyd-warshall algorithm

The floyd-warshall algorithm is an all-points shortest path algorithm. If no negative cycles exist in the graph, the algorithm finds the shortest path between all possible nodes in the graph. The algorithm construct n+1 matrices $R^{(k)}$, $k = 1, 2, \dots, n+1$, which are each of size $n \times n$. If TRUE is returned, then $R^{(n+1)}(u,v)$ is the shortest path from u to v. If False is returned the graph contains negative cycles.

Algorithm

```

For v = 1 to n
For u = 1 to n
 $r^{(1)}(u,v) = w(u \rightarrow v)$ 
for k = 1 to n
for v = 1 to n
for u = 1 to n
 $r^{(k+1)}(u,v) = r^{(k)}(u,v)$ 
if  $r^{(k+1)}(u,v) > r^{(k)}(U,k) + r^{(k)}(k,V)$ 
 $r^{(k+1)}(u,v) = r^{(k)}(U,k) + r^{(k)}(k,V)$ 
for k = 1 to n
for u = 1 to n
if  $r^{(k)}(u,u) < 0$ 
return FALSE and exit
return TRUE and exit
    
```

EXAMPLE2:

In this example, the floyd-warshall algorithm is used to solve all pair shortest path problem for the graph in fig A.2. The values $r^{(k)}(u,v)$ for $u,v \in \{1,2,3,4\}$ for $k = 5$ are given in TableA.2 where $r^{(k)}(u,v)$ is the element u,v in the matrix $R^{(5)}$. The floyd-warshall algorithm return TRUE because all of the diagonal elements in the matrices $R^{(k)}$, are nonnegative, so $r^{(5)}(u,v)$, is the shortest path from node u to node v.



$R^{(1)}$ [8 -3 8 8 8 8 1 2 8 8 8 2 1 8 8 8]	$R^{(2)}$ [8 -3 8 8 8 8 1 2 8 8 8 2 1 -2 8 8]
$R^{(3)}$ [8 -3 -2 -1 8 8 1 2 8 8 8 2 1 -2 -1 0]	$R^{(4)}$ [8 -3 -2 -1 8 8 1 2 8 8 8 2 1 -2 -1 0]
$R^{(5)}$ [0 -3 -2 -1 3 0 1 2 3 0 1 2 1 -2 -1 0]	

TableA.2: Values of $r^{(k)}(u,v)$ for example2.

REFERENCES

- [1] Charles.E.Leiserson and James.B.Saxe "Retiming synchronous circuitry", *Algorithmica*.
- [2] P.Y.Calland, Anne Mignotte, Olivier Peyran, Yves Robert, Frederic Vivien "Retiming DAGs", *IEEE Transaction on Computer-Aided design of Integrated Circuits and systems*, vol 17, NO 12, December 1998. pp 1319-1323.
- [3] Giovanni De Micheli, "Synchronous Logic Synthesis: Algorithms for cycle-time Minimization", *IEEE Transcation on CAD/ICAS*.
- [4] Jose Monterio, Srinivas Devdas, Abhijit Ghosh, "Retiming sequential circuits for low power", *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*.
- [5] Keshab.K.Parhi, *VLSI Design processing system: Design and Implementation*, Wiley, India, pp 91-105.
- [6] Rudra Pratap, "Getting started with MATLAB7," Indian Edition, *Oxford University press*, ISBN-10-0-19-568001-4, pp-3-101.
- [7] Krishna.M.Kauli, Bill.P.Buckles, U.Narayan Bhat,"A formal definition of data flow graph models", *IEEE Transactions on Computers*, Vol C-35, NO 11, November 1986, pp 940-948.
- [8] Ntsibane Ntlatlapa, "High level synthesis using dependence flow graphs as the intermediate form", Department of Computer science and engineering, Auburn University.
- [9] Narendra Shenoy, Richard Rudell," Efficient implementation of Retiming", *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided designs*, pp 226-233.
- [10] Naresh Maheshwari, Sachin.S. Sapatnekar, "Retiming control logic", *Integration, VLSI Journal*, volume 28, issue 1, September 1999, pages 33-53.

AUTHOR'S PROFILE:

	Pavankumar G.V B.E from Dr.TTIT,KGF, Kolar, (Karnataka). M.Tech. VLSI design & Embedded System, (pusing), Department of ECE-P.G Studies,Dr.Ambedkar Institute of Technology, Bangalore, Karnataka. Area of interest includes VLSI design, Network Analysis, Verilog, Control System and Signal & System. Email: pawankumar05035@gmail.com
	Shilpa K.C Completed M.Tech degree in Electronics from Visvesvarya University of Technology (Karnataka, India) in 2007. Now, Presently Pursuing Ph.D in the field od VLSI, Evolutionary Computation from Visvesvarya University of Technology (Karnataka, India). The object of her interest is the High level Synthesis, Evolutionary Computation, and Neural Networks. Email: shilpa.kc2@gmail.com