High Throughput Multipliers Using Delay Equalization

Alka Raj
Amrita Vishwa Vidyapeetham

N.Kayalvizhi
Amrita Vishwa Vidyapeetham

ABSTRACT

Pipelining is used for increasing the throughput of the system. Wave pipelining is done by removing the intermediate registers present in the pipelined circuits so that there will be only an input register and an output register. Circuit should be modelled in such a way that all data from one stage should reach the next stage at the same time so that overlapping of data will not occur. In wave pipelined system the clock period should be greater than the difference between maximum delay and minimum delay + clocking overheads such as setup time, hold time, etc. Clock period can be reduced by minimizing the difference between maximum and minimum delay, i.e delay equalization has to be done. Delay equalization can be done by logic restructuring combined with Wong's algorithm and Klass's algorithm. Area can be further decreased by using delay element shifting and delay element sharing.

Keywords

Wave pipelining, Delay equalization, Logic restructuring, Delay element sharing and shifting

1. INTRODUCTION

Pipelining is a technique used for increasing the throughput of the system. Pipelining is done by splitting a task into several subtasks and inserting registers between these subtasks. There will be an increase in area due to the presence of these intermediate registers. In order to boost up the pipelining rate wave pipelining was introduced.

In wave pipelining we remove off the intermediate registers so that there will be only an input register and an output register. Circuit should be modelled in such a way that all datas from one stage should reach the next stage at the same time so that overlapping of data will not occur.

Fig1 shows a combinational logic block with input and output register [1]. During each clock cycle input data is loaded into the input register and computation starts. All the paths will not be having same delay. This is due to the difference in circuit path lengths. Minimum delay path and maximum delay path are represented in fig1.

Between maximum delay and minimum delay the computation takes place. Therefore output data will be unstable in the shaded region. During this period there will be an error in output. Non shaded area represents the stable region .At non shaded area correct output will be obtained. Hence in order to get the correct output, output register should be latched in the stable region. This can be achieved by properly adjusting the clock skew.

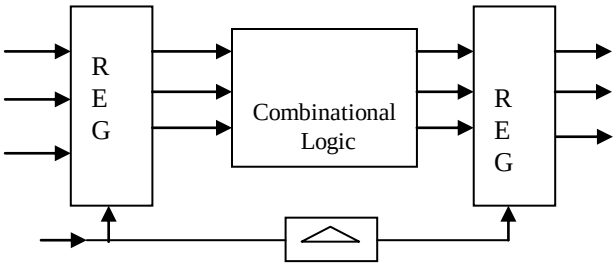


Fig. 1 Data flow through combinational logic circuit.

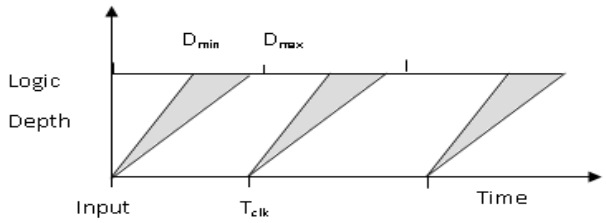


Fig. 2 Temporal/spatial diagram of combinational logic circuits

In conventional system the clock period should be greater than maximum delay and in wave pipelined system the clock period should be greater than the difference between maximum delay and minimum delay + clocking overheads such as setup time, hold time, etc. This shows that in a wave pipelined system there is reduction in clock period [2].

$T_{clk} > D_{max} - D_{min} + T_s + T_h + 2\Delta ck$.

where

T_{clk} = Clock period
D_{max}=Delay of longest path
D_{min}= Delay of shortest path
T_s=Set up time
T_h=Hold time
Δck= worst case clock skew existing over the whole circuit

In wave pipelined system by modifying parameters like maximum delay and minimum delay, minimum value of clock period can be changed. By minimizing the difference between maximum and minimum delay, the clock period can be reduced, i.e. delay equalization has to be done.

2. RELATED WORK

Different algorithms are proposed for delay equalization. In [3] wave pipelining is done by using CAD algorithms like logic restructuring, delay buffer insertion and by clock buffer insertion. By using rough tuning delay equalization is done with minimum area and by using fine tuning delay equalization is done with minimum power [4]. By including delay element shifting and delay element sharing area can be further reduced [5]. If LUTs are used for delay equalization it gives a circuit with high throughput and minimum latency but power consumption will be increased [6]. Linear Programming techniques were used to minimize the overall power dissipation and over all circuit delay can be reduced by using Integer Programming techniques using branch and bound algorithm [7].

3. DELAY EQUALIZATION

The main aim of this paper is to equalize all the path delays and to reduce the area. The delay equalization can be achieved by Logic restructuring combined with Wong's algorithm and Klass's algorithm. For performing delay equalization the circuit has to be converted to directed acyclic graph (DAG).

For converting a circuit to DAG each node represents a gate and is denoted by un, where n ranges from 1 to g. g is the total number of gates. Delay of each gate is written inside its corresponding node and it is represented by w(un).Connection between gates are represented by edges with weight w(e). w(e) corresponds to the wire delays between the corresponding gates. To the graph a source node and sink node are added. u0 is the source node and uN is the sink node. u0 is connected to all the primary inputs and all the primary outputs are connected to uN. Sequence of nodes and edges are called path. A path from u0 to uN can be represented as u0→uN. Delay of the path is the sum of all the edge weights and node weights that exist between u0 and uN, i.e. the sum of all the gate delays and wire delays that are present in the path.

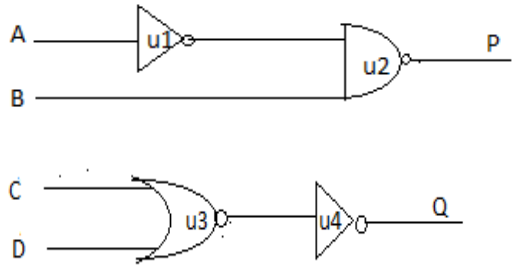
In the circuit given in fig 3(a) has four gates- two NOT gates, one NOR gate and one NAND gate [5]. While converting the circuit to DAG as shown in fig 3(b) NOT gates are represented by nodes u1 and u4, NAND gate by u2 and NOR gate by u3. Edges are drawn wherever there is a connection between the gates. u1, u2 and u3 have primary inputs and u4 and u2 have primary outputs. Source node u0 is connected to u1, u2 and u3. u4 and u2 are connected to sink node uN. Gap is the delay difference between the two paths having same starting and ending node.

3.1. Logic Restructuring

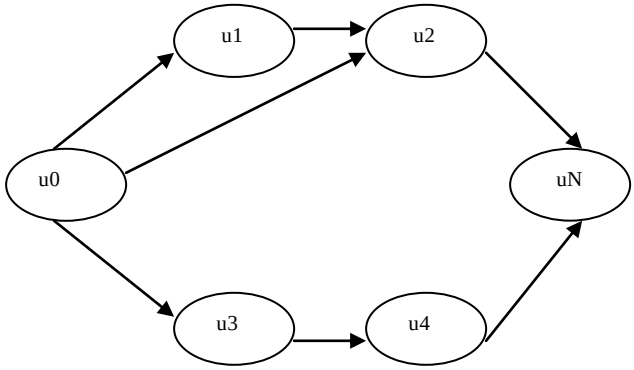
By logic restructuring delay equalization is done by changing the structure of the circuit without changing the functionality of the
circuit. For logic restructuring the circuit has to be decomposed to canonical form. In canonical form the input should have only two inputs. This is followed by node collapsing and decomposition. In node collapsing several nodes are collapsed into single node. Decomposition of collapsed node is done by kernel division method[3]. Delay equalization is further done by Wong's algorithm combined with Klass's algorithm.

3.2. Wong's Algorithm

In Wong's algorithm the gaps are reduced by inserting delay elements along fastest path. In this first we find out the critical path and delay of all the non critical path is made equal to the critical path delay.



(a)



(b)

Fig. 3(a,b) Conversion of a circuit to DAG

3.3. Klass's Algorithm

Construct the longest path by including all the nodes. That means longest spanning tree is constructed. The edges which are not present in the longest spanning tree belongs to set S. Close every loop in the spanning tree by taking edges from set S. Find out the gaps of the loop. If two sides of the loop are having same delay the gap will be 0(gap is balanced). If a gap exists between two sides of the loop, insert delay elements in fastest path inorder to reduce the gap. The length of the padding delay element can be varied between Bmin and Bmax. If the delay difference is less than Bmin the gap cannot be balanced properly.

4. AREA MINIMIZATION

To further decrease the delay elements added use delay element sharing and delay element shifting along with this so that area can be minimized

4.1. Delay Element Sharing

Delay element sharing can be done to all nodes starting from a single node.

Steps involved in delay element sharing are:

1. Draw the gate which has maximum gap
2. The gaps of other gates are obtained from the intermediate points in the longest path

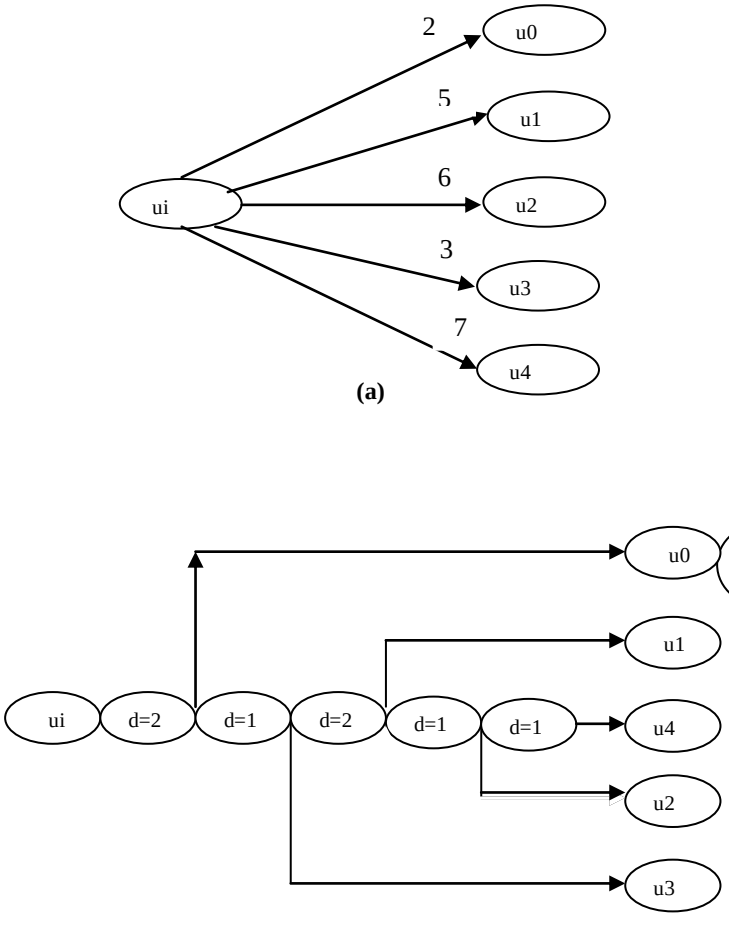


Fig. 4(a,b) Example of delay elements sharing

In the DFG shown in fig 4(a) ui has five fan outs – u0, u1, u2, u3 and u4. Five gaps starting from ui are 2, 5, 6, 3 and 7. Implement the maximum delay gap, 7. Then all other delay gaps are taken from the intermediate points. Fig 4(a) has 23 delay elements. By performing delay element sharing the delay elements are reduced to 7 as shown in fig 4(b).

4.2. Delay Element Shifting

If there is no gap for single fan out edge from a gate, delay element shifting cannot be performed. Delay element shifting can be done only if all the fan outs edges of the gate have gap and the minimum gap can be shifted from the output side to the input side.

In fig 5(a) uk has two fan outs - uk+1 and uk+2. Fan out edge of uk has a gap of two and five. The minimum gap, two is shifted from output side to input side. Fig 5(a) has 13 delay elements. By performing delay element shifting the delay elements are reduced to 12 as shown in fig 5(b).

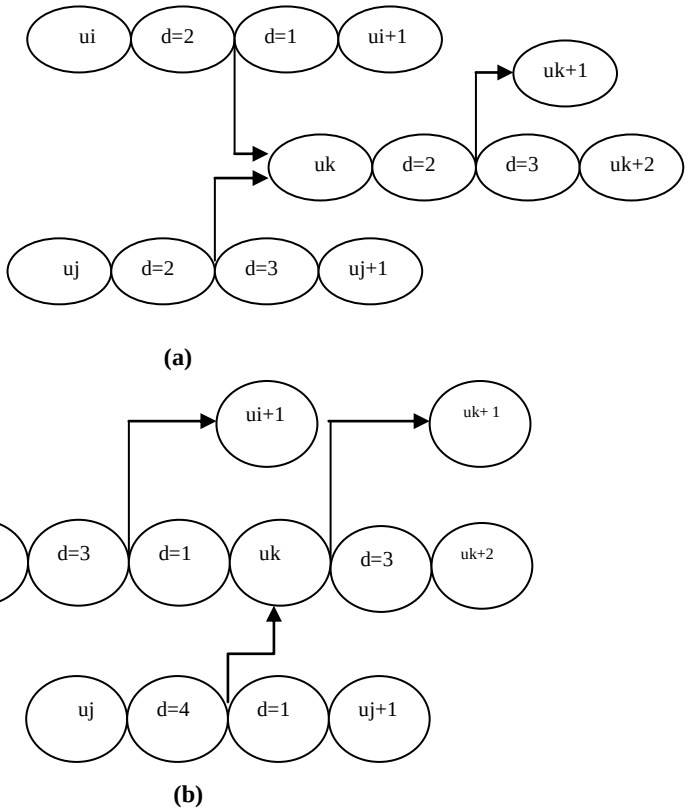


Fig. 5(a,b) Example of delay elements shifting

5. MULTIPLIERS

There are different types of multipliers like Shift and add multiplier, Serial/Parallel multiplier, Array multiplier etc. For Shift and add multiplier and Serial/Parallel multiplier adders with feedbacks are used and for array multipliers there is no feedback. Delay equalization task will be difficult in a wave pipelined circuit if a feedback exists. So in order to make delay equalization simple we use array multipliers. Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one

multiplier bit. The partial product are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. Fig 6 shows an array multiplier. Y and X are the inputs. The block shaded with blue colour has two input bits. These two bits are multiplied. The block shaded with black colour has three input bits. Two input bits are multiplied and added with the third input bit. The block shaded with grey colour has four input bits. Two input bits are multiplied and added with the other two input bits and the block shaded with brown colour represents half adder. The operation of array multiplier are splitted into 8 stages and each stage is represented by numbers 1-8.

6. IMPLEMENTATION

The proposed algorithm is used for implementing high throughput low area wave pipelined multiplier. Steps involved during implementation are as follows. Write the verilog code for multiplier . Construct the corresponding direct acyclic graph by

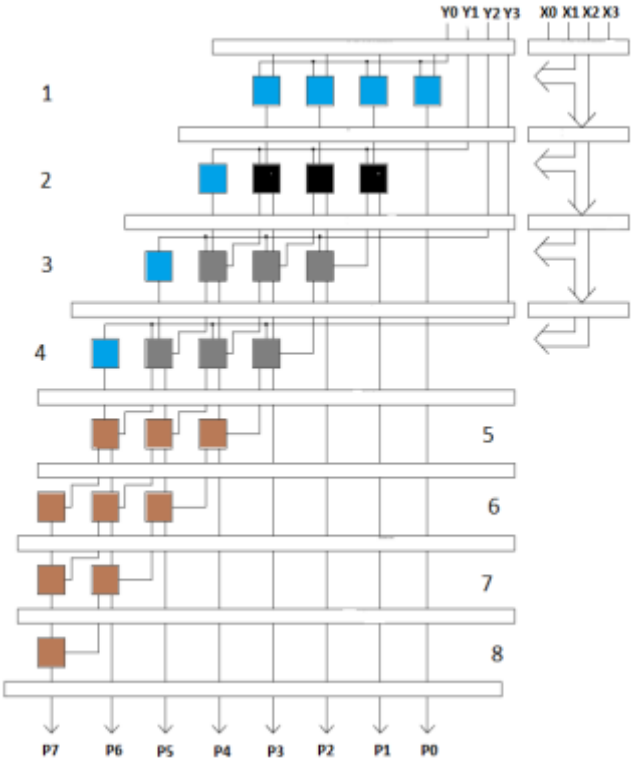


Fig. 6 Array multiplier

using all the required information of the gate obtained from the net-list and connection between the gates. From the net-list gate's type, inputs, outputs, internal delay, fan-ins, fan-outs are obtained. Perform Wong's algorithm for each block and Klass's algorithm for each stage with and without logic restructuring This gives rise to a new DAG with all the path delays equal by inserting minimum delay elements. Write new verilog code for the modified DAG.

As mentioned earlier for proper working of wave pipelined circuits output should be latched in the stable region. This can be done by only trial-and-error. Manual procedures are adopted for the choice of the optimum value of clock frequency and clock skew between the input and output registers of wave-pipelined circuits. Clock frequency and clock skew are manually adjusted until we get correct output. This is a time consuming procedure. Automating the above procedure can make this task easier.

7. RESULTS

4-bit array, carry save, wallace tree and carry ripple multipliers are used for conducting experiments. Table1 shows the number of delay elements added for each block by Wong's algorithm and Table 2 shows the number of delay elements added for each stage by Klass's algorithm for array multiplier. Table 3 and 4 shows the total number of delay elements added for multipliers along with the reduction of delay elements by using sharing + shifting algorithm without and with logic restructuring.

Table 1. Delay elements added for each block by wong's algorithm without logic restructuring for array multiplier

Block	Number of delay elements added
Blue	0
Black	4
Grey	17
Brown	0

Table 2. Delay elements added for each stage by klass's algorithm without logic restructuring for array multiplier

Stage	Number of delay elements added
1	10
2	31
3	55
4	10
5	0
6	0
7	0
8	0

Table 3. Delay elements added for multipliers without logic restructuring

Type of multiplier	Combining Wong's and Klass's algorithm	After delay element sharing and shifting
Array	220	148
Carry save	342	284
Carry ripple	289	248
Wallace tree	383	354

Table 4. Delay elements added for multipliers with logic restructuring

Type of multiplier	Combining Wong's and Klass's algorithm	After delay element sharing and shifting
Array	135	59
Carry save	203	179
Carry ripple	189	114
Wallace tree	304	228

8. CONCLUSION

By wave pipelining the multiplier clock period can be reduced and throughput can be increased. But due to the addition of delay elements there will be an increase in area. By using the method mentioned here area of the array multiplier can be decreased. Thus a highly efficient multiplier is obtained.

9. REFERENCES

[1] G.Lakshminarayanan,B.Venkataramani, "Optimization techniques for FPGA based wave-pipelined DSP blocks", in Proc IEEE Transactions on VLSI Systems,vol.13,number7,pp.783-793,July 2005 .

[2] K. K. Parhi, VLSI Signal Processing Systems. New York: Wiley, 1999.

[3] W. P. Burleson, M. Ciesielski, F. Klass, and F. Liu, "Wave-pipelining: a tutorial and research survey" in IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 6, no. 3, pp. 464–474, Sep. 1998.

[4] D.C Wong,G. DeMicheli,and M.J Flynn, "Designing High-Performance Digital Circuits Using Wave Pipelining: Algorithms and Practical Experiences", in IEEE Trans Comput Aided Des Integr . circuits syst.,vol. 12, no. 1, pp 25-46, Jan 1993.

[5] Rui Tang,Yong-Bin Kim, "A novel delay balancing methodology for wave pipelined circuits", 48th midwest symposium on Circuit and systems, pp. 1035-1038, vol 2 ,Aug 2005 IEEE

[6] E.I.Boemo, S.Lopez-Buedo, J.M.Meneses, "Wave pipelining via look-up tables" in Proc IEEE Int.Symp.circuits systems,vol 4,1996, pp185-1884.

[7] Srivastav Sethupathy,Nohpill Park,Marcin Paprzycki , "Logic restructuring for delay balancing in wave-pipelined circuits: an integer programming approach", in proceedings of the seventh international symposium on symbolic and numeric algorithms for scientific computing,2005 IEEE.