

Power Minimization using Voltage reduction and Parallel Processing

Sudheer Vemula

Dept. of Electrical and Computer Engineering
Auburn University, Auburn, AL.

Goal of the project:-

To reduce the power consumed by the 32x32 array multiplier by including parallel processing, without any increase in the delay of the critical path.

Problem Statement:-

The array multiplier in itself has a lot of parallelism included, i.e, the processing is done simultaneously in most of the blocks. The initial intuition is that the further inclusion of parallelism in the same circuit may not have any further improvement.

Design Approach:-

Inclusion of parallelism should improve the speed of operation of the circuit. Then we can reduce the voltage supply, which will reduce the power dissipation but will increase the delay of the circuit. The additional delay added in the circuit should be compensated by the included parallelism and the circuit should be able to work at its normal frequency of operation. For including parallelism in the circuit additional circuitry might be added, which will increase the area overhead and also the power consumption. The amount of power dissipated may be high or low depending on the type of the overhead circuitry. The final power consumption of the circuit should be lower than the initial value by adding appropriate amount of overhead.

Introduction:-

Concurrent execution of several programs or several blocks of a program is known as parallel processing [1]. There are two ways of including parallelism in the circuit. They are

- 1) Data Parallelism
- 2) Control Parallelism

Data Parallelism is parallel execution of single expression on data distributed over multiple processors [2]. Control Parallelism is the parallelism that is achieved by the simultaneous execution of multiple threads [3], i.e., performing different operations on same data simultaneously.

Design Techniques:-

There are two ways of including parallelism in the circuits. First, the same core can be replicated several times as shown in Fig.1. This is known as multi-core architecture. By replicating the same core several times the incoming inputs are applied to different cores in sequence. Now the individual cores can be slowed down by Voltage scaling which in turn will reduce the power consumption of the whole circuit. The area overhead is very high in multi-core architecture and this architecture will work for any circuit independent of the combinational logic present in the circuit.

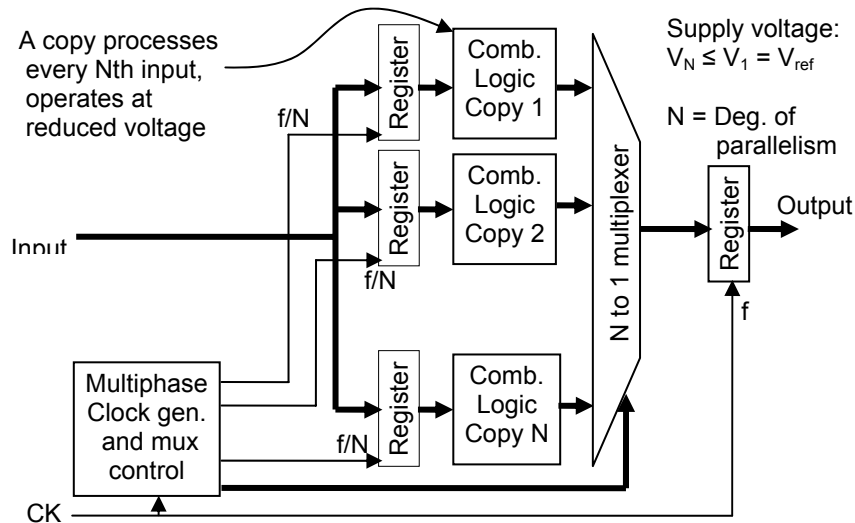


Fig.1 Multi-core parallel Architecture [4]

In second design technique, the big circuit can be partitioned into several small circuits to include parallelism. This is also a type of pipelining but it is different from the pipelining which has been shown for the first design technique. This is similar to the pipelining which is used in the data-path architecture. In this design technique the area overhead is very less compared to the first one.

As design of the circuit by first technique is independent of the combinational logic, it is more of an implementation of the design. The area overhead for the second circuit much lesser than the first one and also there is scope for both the design and implementation. So, I have concentrated mostly on the second design technique.

Architecture of the Design:-

The basic idea used in the design of my pipelined multiplier architecture is to compute partial products rather than computing the whole product at a time. Once the partial products are computed, they can be added by including the respective shift value. It is described in an example showing the operation on two 2 integer numbers

Ex.: A=98 and B=76

$$\begin{aligned}
 A \times B &= (90 \times 76) + (8 \times 76) \\
 &= (9 \times 76) 10 + 8 \times 76 \\
 &= (9 \times 7) 100 + (9 \times 6) 10 + (8 \times 7) 10 + (8 \times 6)
 \end{aligned}$$

As the binary representation takes more digits to represent the same value, this method becomes more effective.

The given 32x32 array multiplier can be partitioned in two different ways to include parallelism

- 1) Horizontal Partition and
- 2) Vertical Partition.

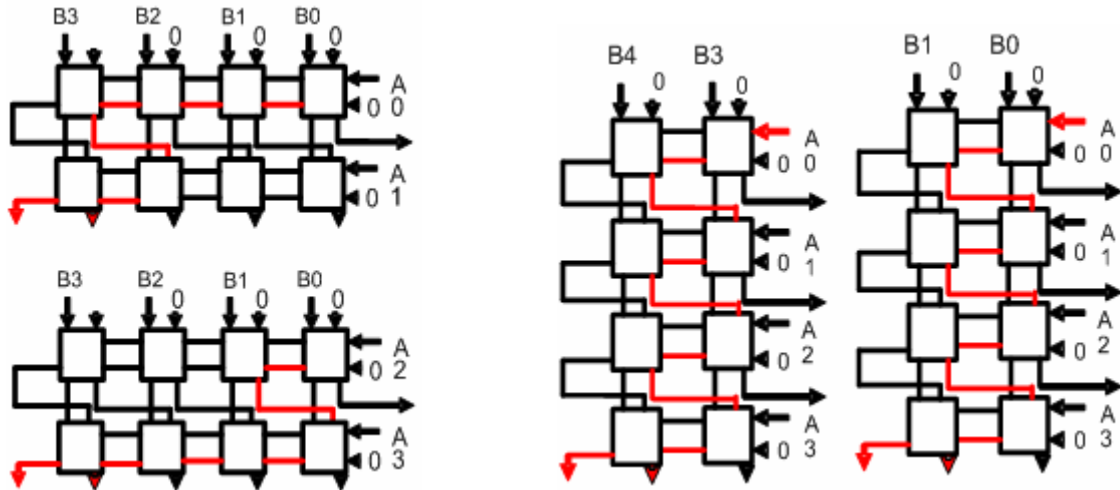


Fig.2 Horizontal and Vertical Partitions of the circuit

By doing the vertical partition, the delay of the critical path can be reduced by a larger amount than by doing the horizontal partition. The red lines show one of the possible critical paths in the circuit. Both vertical and horizontal partitions can be included on same circuit to further decrease the delay of the critical path.

Delay Overhead:-

A 32x32 multiplier with four 16x16 multipliers is shown in Fig. 3. There is only one ‘full adder delay’ overhead due to the 16 bit full adder because the ripple carry adder performs the addition in sequence as soon as it receives the values. And the outputs of the multiplier are always available to the full adder. And the delay overhead due to 16bit Carry Look Ahead (CLA) adder and the 16 bit Ripple Carry (RC) adder is equal to the delay of the 16 bit CLA adder because the inputs to the RC adder arrive in sequence before the inputs to the CLA adder. And both the inputs to CLA adder arrive at the same time.

The delay of the critical path can be further reduced by implementing the 16x16 multiplier with four 8x8 bit multipliers. And it can be further reduced by implementing 8x8 multiplier with four 4x4 multipliers.

RC adders have been chosen because they have the lowest power consumption of all the adders. And wherever the speed is important CLA adders have been used [5].

For a 32x32 multiplier designed with 4x4 multipliers the delay is supposed to get reduced by 68% with only 17% area overhead. The detailed calculations of delay and area overhead can be found in the presentation slides.

Implementation:-

First a generic NxM array multiplier has been designed in VHDL. The circuit has been simulated and verified using the Mentor Graphics, Modelsim simulation tool. ELDO, a circuit level simulation tool has been used to find the actual delay of the circuit. ELDO accepts only verilog files. So, the VHDL source code has to be converted to verilog. This conversion is done using Leonardo, a synthesis tool. Leonardo performs the synthesis and also it can provide the output in several different formats.

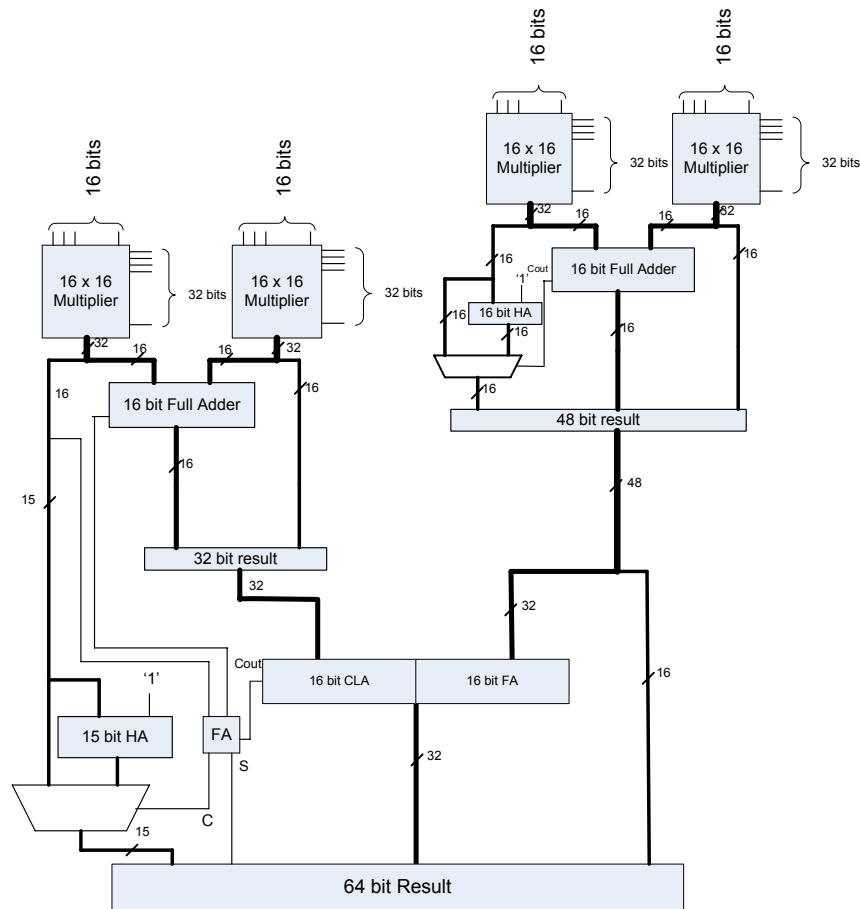


Fig.3 32x32 multiplier with four 16x16 multipliers

Next the proposed circuit with parallelism has been designed using VHDL and the same procedure as explained above has been done to import the circuit to ELDO. For designing the circuit with parallelism Full Adders, Half Adders and Multiplexers are designed separately and then combined.

Comparison of Delay using the Leonardo:-

Circuit	Normal Circuit Delay	Parallelized Circuit Delay
8x8	7.42 units	6.54 units
16x16	16.7 units	12.73 units
32x32	35.27 units	25.13 units

When the simulation is performed using ELDO, the circuit has been simulated for all 0's to all 1's transition. For a 32x32 normal multiplier the delay was 8.7ns and for a parallelized circuit the delay was 12.07ns. The delay was actually higher because the delay due to different inputs will be different for different design of the multipliers. The actual delay of the critical path is supposed to be smaller.

Here each multiplier has been implemented with 4 mini multipliers of respective dimensions. And the CLA adder has not been implemented, so it is replaced by a RC adder.

The power consumption for a 16x16 multiplier was around 1 watt (peak power for different vectors).

Points of Interest:-

- 1) ELDO simulation tool hasn't been totally developed. It was requiring some modifications in the verilog file even when the source code is totally correct.
- 2) The documentation for the tool is not really good. It takes some time to figure out things.

Lessons Learned:-

- 1) I was mainly concentrating on improvement of the delay and I didn't note the area values provided by the synthesis tool. So, I couldn't provide any results on area overhead.
- 2) Didn't find the vector to activate the critical path, so couldn't provide any valuable results of the work done at the circuit level simulation.

Future Work:-

The parallelism in the circuit has been implemented at only four levels. The number of levels can be improved to get good results. CLA can be implemented and used to improve the results. Still there are quite a few improvements that can be made in this circuit to provide better results. An LFSR can be used to provide the inputs required for power estimation.

Conclusion:-

The Leonardo synthesis tool showed considerable improvement in delay. So, power can be reduced by implementing parallelism in the array multiplier.

References:-

- [1] dspvillage.ti.com/docs/catalog/dspplatform/details.jhtml
- [2] www.llnl.gov/CASC/Overture/henshaw/documentation/App/manual/node160.html
- [3] books.nap.edu/html/up_to_spedd/appD.html
- [4] Class Slides
- [5] J. M. Rabey & M. Pedram, *Low power Design Metodologies*, Kluwer Academic Publishers, Boston MA, 1996.