IPHV7I3024X
International Journal Of Advanced Research and Innovation -Vol.7, Issue .III
ISSN Online: 2319 – 9253
Print: 2319 – 9245

DESIGN AND ASIC IMPLEMENTATION OF AUTOMATED WAVE PIPELINED FILTERS

[#1]K.ASHIK REDDY - M.Tech Student,
[#2]K.HARI BABU - Assistant Professor,
Dept of ECE,
MLR INSTITUTE OF TECHNOLOGY, DUNDIGAL, HYD, T.S., INDIA.

Abstract: Multiplication is one of the mostly used operations in all of the devices. This paper presents an efficient implementation of a pipelined multiplier designed with two stage pipelining and performed backend designing using Encounter tool provided by Cadence. This design is compared with shift and adds multiplier, power and timing analysis for this design are performed using Cadence tool. A range of multipliers architectures are available based on applications. The pipelined circuits are designed so that we can achieve high performance for a system as they can be operated at higher frequency. From the implementation results, it is verified that two stage pipelined circuit is faster by a factor of two times than the non pipelined circuit. But we get an area and power overhead. Therefore, this circuit can be used where the performance of system is of major concern.

Keywords: ***Multiplier; Verilog; Pipelining; ASIC; Encounter.***

I.INTRODUCTION

In today's era everybody wants high performance system where power is not considered the major factor because speed and power of the circuits cannot be optimized simultaneously. As if we want to reduce the power consumption then the system would become slow. Therefore power dissipation and speed of the circuits cannot go hand in hand. Hence, in this paper the multiplier is designed which is application specific that is which will be used for high performance system only because the power and area of the circuit are increased as compared to non- pipelined circuits. Now these days, multipliers are being used as one of the fundamental blocks. They play an important role in today's digital signal processing, FIR filters, microprocessors and various other applications. A system's performance is generally determined by the performance of the multiplier. So they are highly preferred, as they offer high speed, low power consumption and regularity in layouts. There are different types of multipliers like booth multiplier, serial multiplier, it depends for which application we want to use. Here we have used shift and add multiplier for non-pipelining circuit and inserted two stages of pipelining in the same multiplier for the comparison in between them. Multipliers are widely used in DSP operations such as convolution for filtering, correlation and filter banks for multi rate signal processing. Without multipliers, no computations can be done in DSP applications. For that reason, multipliers are chosen for pipelining in our proposed design. WAVE-PIPELINING is an example of one of the many methods currently being used in sophisticated VLSI designs. As an alternative to pipelining, it provides a method for significantly reducing clock loads and the associated area, power and latency while retaining the external functionality and timing of a synchronous circuit. It is of particular interest today because it involves design and analysis across a variety of levels (process, layout, circuit, logic, timing, and architecture) which characterize VLSI design. However it also questions some of the fundamental tenets of simplified VLSI design as popularized in the early 1980's.

The idea of wave-pipelining was originally introduced by Cotten , who named it *maximum rate pipelining*. Cotton observed that the rate at which logic can propagate through the circuit depends not on the longest path delay but on the difference between the longest and the shortest path delays. As a result, several computation "waves," i.e., logic signals related to different clock cycles, can propagate through the logic simultaneously. One can also view the wave-pipelining as a virtual pipelining, in which each gate serves as a virtual storage element.

Hardware components in a SOC may include one or more processors, memories and dedicated components for accelerating critical tasks and interfaces to various peripherals . The increased gate count in a complex SOC results in increased power dissipation, clock routing complexity and clock skews between different parts of a synchronous system. These limitations may be partially overcome by adoption of circuit design techniques such as wave-pipelining. Wave pipelining enables a combinational logic circuit to be operated at a higher frequency without the use of registers and may result in lower power dissipation and clock routing complexity compared to a pipelined circuit. However, the maximization of the operating speed of the wave-pipelined circuit requires the following three tasks: adjustment of the clock period, clock skew (δ) and equalization of path delays. The automation of these three tasks are proposed for the first time in this paper. Effectiveness of the automation scheme is studied by a multiplier using dedicated AND gates as well as fast carry logic.

II. DESCRIPTION OF MULTIPLIERS

We have taken the 8*8 bit shift and add multiplier which is designed by Verilog Code. The algorithm followed in the shift and add multiplier is that first we need to generate partial products, that are added together in order to give final output. After generation of each partial product multiplicand is shifted by one bit in order to perform next bit multiplication operation. We have one control block which checks for existence of multiplicand bit if it is one then only multiplication is performed else we need to set partial product equal to zero as there is no need to perform this bit multiplication.

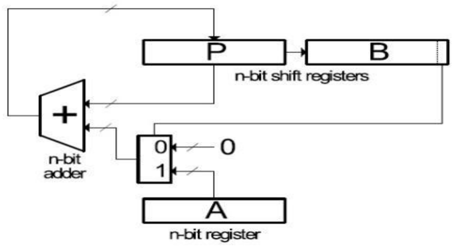
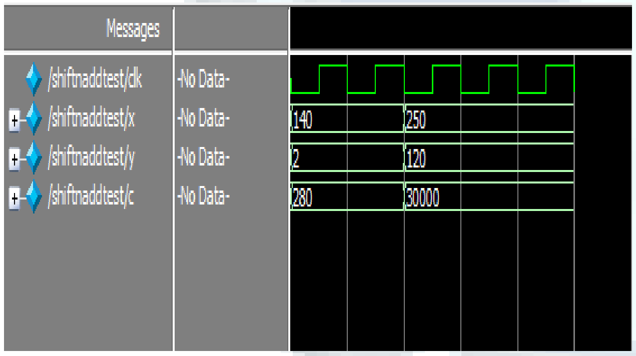
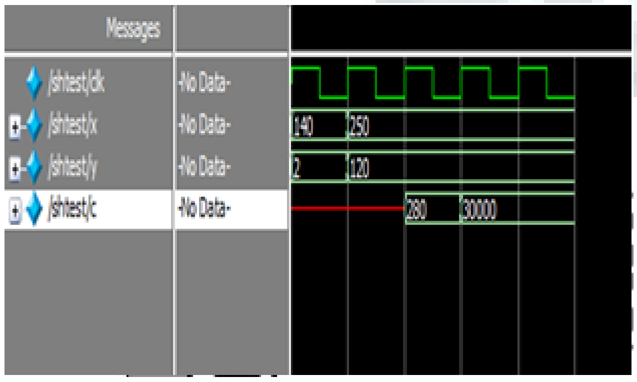


Fig1: Architecture of Shift and Add Multiplier

The architecture of shift and add multiplier is shown in figure1 above, this is most common architecture used to explain the working of a basic multiplier. The two stage pipelining is implemented in this shift and add multiplier for high performance as the basic principle of pipelining is that it divides the work into segments and each segment can execute its operation concurrently with other segments. When a segment completes an operation, it passes the result to the next segment in the pipeline and fetches the next operation from the preceding segment. The final results of each instruction emerge at the end of the pipeline in rapid succession. Pipelining reduces the effective critical path by introducing pipelining latches along the critical data path. It is simple to understand that in same clock period if we need to perform half of the operation than we can decrease the supply voltage or we can change the operating frequency keeping the supply voltage constant. In this paper we kept the supply voltage constant and changed the operating frequency, as this is two stages of pipelining so theoretically the output frequency should be made twice of the previous frequency. After designing this and performing timing analysis it is seen that for the clock period of approximately 1550ns the same performance of the circuit is achieved whereas in case of the non-pipelining circuit the operating frequency was 3100. The pipelining in the circuit is achieved by adding registers in between them and designed in Verilog code. Then the circuit is simulated in order to be verified with a sample test-bench, the results are as follows



(a)



(b)

Fig2: Comparison of outputs of multipliers

III.DESIGN OF MULTIPLIER

The architecture of our designed pipeline multiplier is shown in figure 3, in which the multiplier is divided in two parts. For first four bits it will be calculating the partial products in one clock cycle and in next clock cycle it will calculate the partial products for last four bits and then these partial products are given as input to an adder block which will generate the final output consisting of 16 bits of resultant.

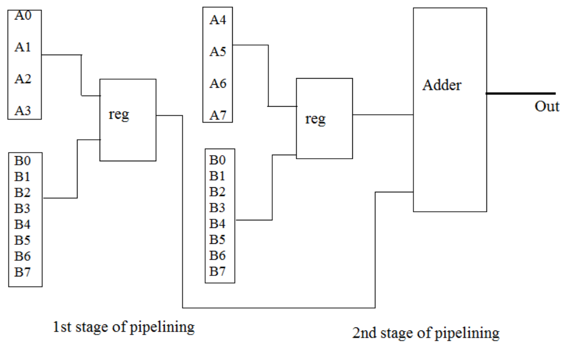


Fig3: Architecture of pipelined multiplier

After this design, we have performed backend operations for synthesizing this design. For that first step is to generate the synthesized netlist, which is given as input to IO file which contains the mapping commands for IO pads and corner pads. In Encounter tool first thing after importing the design is to do Floor planning, in Floor planning we took core ratio as 1 and core utilization as 0.6. Then we have to go for connection of power nets, after that power planning is performed. In power planning we have included rings and strips. After this we go for special route followed by standard cell placement. At last routing is performed in order to connect all the IO pads to internal circuitry. The final Tap out diagram obtained for this design is shown in figure4.

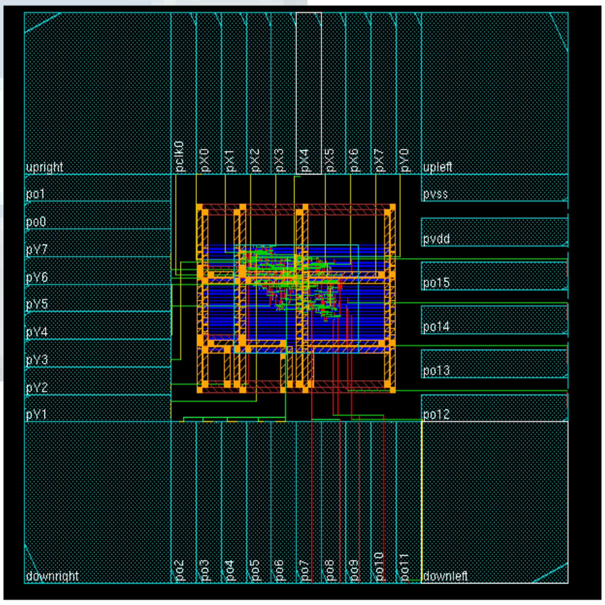


Fig4: Tap out diagram of Pipelined Multiplier

The timing requirements of wave-pipelined circuits will be defined for a single combinational logic block with registers attached to its inputs and outputs. In the case of multistage pipelines, the timing constraints must hold for all stages. In the sequel, the term *conventional pipelining* will be used to refer to a pipeline where only a single set of data propagates between registers at any given time.

The term *constructive clock-skew* will be used to refer to a clock-skew that is intentionally created between two clock signals and that can be adjusted with predictable effects. This is in contrast to an *uncontrolled clock-skew* that exists in the circuit due to delay differences along the clock lines.

Wave-pipelining has been employed for implementing a number of systems on both ASICs and FPGAs . The concept of wave-pipelining has been described in a number of previous works.

The technique of Wave-pipelining is proposed to improve the logic utilization by minimizing the idle time and to allow for maximal rate of operation of the digital circuit. Fig. 1 shows a typical wave-pipelined circuit along with the input, output registers and clocking circuit which include the clock generation and clock skewing circuit.

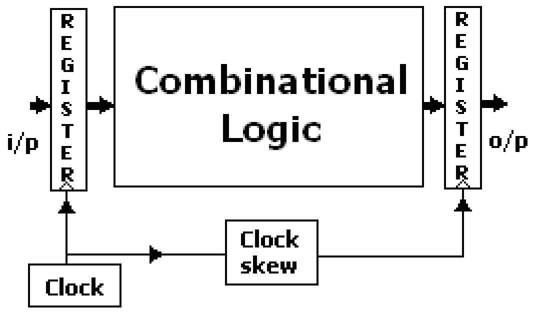


Fig.1. Wave-pipelined circuit

An RTL model of a circuit consists of a combinational logic circuit separated by the input and output registers. The combinational logic circuit may be considered to be a wave pipelined circuit if a number of waves are made to simultaneously propagate through it.

In other words, at any point of time, a sequence of data is processed in the combinational logic block. In the case of pipelining, only one data is processed in the combinational logic block at a time. Further, the maximum data rate in the pipelined circuit depends only on Dmax, the maximum propagation delay in the combinational logic block. If Dmin denotes the minimum propagation delay of the signal through the combinational logic block, the maximum data rate of the wave-pipelined circuit depends on (Dmax – Dmin). Traditionally, in a wave pipelined circuit, higher speeds are achieved by equalizing the Dmax and Dmin.

The output of the wave-pipelined circuit alternates between unstable and stable states. The stable period decreases with the increase in the logic depth. By adjusting the latching instant at the output register to lie in the stable period, the wave-pipelined circuit can be made to work properly. But, for large logic depths, there may not be any stable period. Hence adjusting the latching instant by itself may not be adequate for storing the correct result at the output register. For such cases, the clock period has to be increased to increase the stable period.

ADVANTAGES

- Occupies Less Area

DISADAVNTAGES

- Low Speed
- High Power Consumption
- High Delay
- Storing Capacity Less

PROPOSED MODEL

As discussed in Earlier, critical speed-limiting factors in wave-pipelining are the uncontrolled clock-skew, the sampling time of registers, and the worst case transition time at the logic outputs. While the minimization of these factors has been a major challenge in the design of conventional highspeed pipelined systems as well, the equalization of path delays comes as a new challenge for the design of wave-pipelined systems. While in theory the path-delay equalization problem has been solved, the real challenge is to accomplish it in the presence of a variety of static and dynamic delay tolerances, some of which are listed below.

1) Gate-Delay Data-Dependence

Gate-delay independence on the input pattern, i.e., constant gate delay, is not guaranteed in general. It depends on the particular technology and the structure of logic gates. Some input patterns may cause significant delay variations.

2) Coupling Capacitance Effects

The effective capacitance seen by a gate depends on the capacitive coupling between adjacent wires. This may introduce significant changes in the gate delay, especially in advanced multilevel metal processes.

3) Power-Supply Induced Noise

Power supply noise is attributed mainly to IR drops, capacitive coupling between the interconnection wires and the power supply lines, and the inductance of bonding wires, package trace, and pins. Noise in the power-supply voltage can cause accumulative delay dispersions as waves propagate through several logic layers.

4) Process Parameter Variations

Variations of process parameters during manufacture may result in substantial gate delay variations. Equivalent circuits from different fabrication runs may have different propagation delays. This delay dispersion must be accounted for to establish the minimum separation between waves.

5) Temperature-Induced Delay Changes

Some process parameters, such as carrier surface mobility of metal–oxide–semiconductor (MOS) transistors, are highly thermally sensitive. Changes in the operational temperature result in changes in the gate-delay.

A. Data Dependencies

One of the major obstacles initially found in static complementary metal–oxide–semiconductor (CMOS) is the strong dependence of the gate-delay on the input data. Consider for example a static two-input CMOS NAND gate. Depending on whether one or both of the parallel PMOS devices are switching on, the delay of the gate can vary by a factor of two.
For gates with several inputs, this factor is even larger. Clearly, this feature is undesired for wave-pipelining; constant logic gate delay is a requirement for the equalization of path-delays.

To overcome this problem the use of biased CMOS gates, also known as pseudo-NMOS gates, has been proposed. In those devices parallel pullup devices are replaced by a single device whose gate is connected to a bias voltage [28]. While this approach reduces the delay dependence problem and makes CMOS better suited for wave-pipelining, it is achieved at the expense of increased power dissipation.

B. Process and Environmental Delay Variations

Changes in temperature, supply levels, and process parameters can have a substantial effect on the delay of a CMOS circuit. How such changes affect the operation of wave pipelining is discussed below. Although the following analysis is for temperature, it applies to process and power supply changes as well.

In an automation technique for tuning the clock period is proposed. This technique uses the system clock: First, it applies the system clock to the circuit; if the circuit is not working with system clock then it doubles the period and applies it to the circuit. This doubling process is continued until system works properly. However, this approach cannot ensure that the circuit works at the highest possible frequency. This may be either because the circuit operating frequency is greater than system clock frequency or because the highest operating frequency is not exactly equal to clock frequency /2n where n is an integer. Equalization of path delays, adjustment of the clock period and clock skew are the three tasks carried out for maximizing the operating speed of the wave-pipelined circuit. All the three tasks require the delays to be measured and altered if required.

IV.SIMULATION RESULTS

These circuits are implemented using 180nm technology in ASIC. Verilog HDL language is used to describe the functionality of the circuit and for the timing, area and power analysis the Cadence tool is used.

Schemes	Area (cell)	Power (µW)	Frequency(MHz)
Pipelining	312	549.650	645.16
Non-Pipelining	169	465.784	322.580

Table1. Implementation Results of Multipliers

From table1, it is noticed that for the pipelining circuit the frequency is twice as compared with the non pipelining circuit whereas the area and power of the non-pipelined circuit is less as compared to the pipelining circuit because the area increases in the pipelined circuit as we have introduced the registers in between them. Therefore for high speed of the system we have to go with area and power overhead.

V.CONCLUSION

The proposed scheme is implemented for the pipelined circuit and it is tested using 8*8 shift and add multiplier. From the implementation results, it is verified that the pipelined multipliers are faster by a factor of two as compared to non-pipelined multipliers.

REFERENCES

[1] Flavio R. Wagner, Wander O. Cesario, Luigi Carro and Ahmed A. Jerraya, “Strategies for the integration of hardware and software IP components in embedded systems-on-chip,” *Elsevier Integration, The VLSI Journal*, pp. 1-31, Nov. 2003.
[2] J. Nyathi and J. G. Delgado-Frias, “A hybrid wave pipelined network router,” *IEEE Transactions on Circuits and Systems I*: Fundamental Theory and Applications, vol. 49, no. 12, pp. 1764 – 1772, Dec. 2002.
[3] O. Hauck., A. Katoch and S. A. Huss, “VLSI system design using asynchronous wave pipelines: a 0.35 µm CMOS 1.5 GHz elliptic curve public key cryptosystem chip,” *Proc. of Sixth Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000, (ASYNC 2000)*, pp. 188 –197, April 2000.
[4] W. P. Burleson, M. Ciesielski, F. Klass, and Liu, “Wavepipelining: a tutorial and research survey,”
[5] *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 3, pp. 464 474, Sep.1998.
[6] WooKim, YongKim, “Automating Wave-pipelined Circuit Design”, IEEE Design & Test of Computers, Vol. 20, Nov. 2003.
[7] C. Thomas Gray, W. Liu and R. Cavin, “Wave Pipelining: Theory and Implementation,” *Kluwer Academic Publishers*, 1993.
[8] E. I. Boemo, S. Lopez-Buedo and J. M. Meneses, “Wave pipelines via look-up tables,” *IEEE International Symposium on Circuits and Systems ISCAS '96*, vol. 4, pp. 185 -188, 1996.
[9] M. J. S. Smith, “Application Specific Integrated Circuits,” *Pearson Education Asia Pvt. Ltd*, Singapore, 2003.
[10]G.Seetharaman, B.Venkataramani and G.Lakshminarayanan, “Design and FPGA implementation of self-tuned wave pipelined filters,” *IETE journal of research*, vol 52, no. 4, pp. 305-313, July-August 2006.
[11] Steven Elzinga, Jeffrey Lin, and Vinita Singhal. 2000. “Design Tips for HDL Implementation of Arithmetic Functions,” *Xilinx Application notes*, XAPP-215 (v1.0), June.
[12] Altera documentation library- 2003, Altera corporation, USA.