International Journal of Electronics Signals and Systems

Volume 3 | Issue 3 Article 7

January 2014

DESIGN OF ALU-64BIT FOR HIGH SPEED USING LOGICAL EFFORT

S.A. JYOTSNAMAYEE
E.C.E Dept., Gitam University, Visakhapatnam, Andhra Pradesh., jyotsnamayeesa@gmail.com

M. MURALI KRISHNA
E.C.E Dept., Gitam University, Visakhapatnam, Andhra Pradesh., madugulamk@gmail.com

DESIGN OF ALU-64BIT FOR HIGH SPEED USING LOGICAL EFFORT

JYOTSNAMAYEE SA[1] & M. MURALI KRISHNA[2]

[1,2]E.C.E Dept., Gitam University, Visakhapatnam, Andhra Pradesh.
E-mail: jyotsnamayeesa@gmail.com, madugulamk@gmail.com

Abstract- Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current.. Conventional methods use repetitive manual testing guided by Logical Effort (LE).In our work, we choose gate widths inside the circuit as parameters to be optimized in order to achieve the target delay, using LE.The main objective of the paper is to calculate the delay using VerilogHDL and synthesized the output driven by it by increasing the speed using optimized paths.

Keywords – *Logical Effort, VerilogHDL, Xilinx 8.1, Cadance tool.*

I. INTRODUCTION

Despite the performance limitations, delay in the VLSI chips should be decreased . Here we design a circuit to achieve the greatest speed or to meet a delay constraint.

Logical effort is a method to make these decisions
–Uses a simple model of delay
–Allows back-of-the-envelope calculations
–Helps make rapid comparisons between alternatives
–Emphasizes remarkable symmetries

The method of logical efforts is to show how many stages of logic are required for fastest implementation of any given logic function.This reveals the proper transistor sizes in each stages to fastest overall implementation. Logical effort involes the sum of the efforts of the driven devices.The logical effort of a logic function depends mainly on its circuit topology and slightly on the electrical properties of the fabrication process used to build it

In this paper ,we discuss about the ALU which consist of adder ,multiplier and shifter and calculate the delay using logical effort and optimize the delay paths.

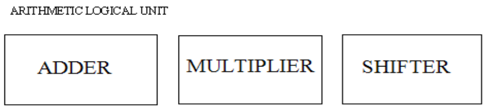


Figure 1.1 Block Diagram of ALU

The above block diagram internally consists of delay units, multipliers etc., which can be modeled by using the difference equations in HDL, implemented in Xilinx and can be dumped into Cadance tool. Several architectural options are available for selecting an appropriate HW block for operations like addition, multiplication and shifting. This paper is presented as follows: Calculation of Logical Effort is presented in Section II. The Design of Adders is presented in section III. Design of Multiplier in IV, Design of Shifter in V. The simulation and results are presented in Section VI. Finally, Section VII presents the conclusions of this paper.

II. CALCULATION OF LOGICAL EFFORT

The method of logical effort (LE) was for fast evaluation and optimization in delay paths by choosing appropriate gate sizes. As the VLSI circuits continue to scale, the contribution of wires to the delay increases and cannot be neglected. This characteristics occurs not only with respect to long wires connecting separate modules but also the interconnect within logic modules where the delays introduced by the wires connecting closely coupled gates approach and can exceed the gate delays. We can select the fastest candidate by comparing delay estimate of different logic structures. This method also specifies the proper number of logic gates on a path and the best transistor sizes for logic gates.

A) *Delays In Logic Gates*

The model describes delay caused by the capacitive load that the logic gate drives and by the topology of the logic gate. As the load increases, the delay increases, but delay also depends on the logic function of the gate.

The first step in modeling delays is to isolate the effects of a particular integrated circuit fabrication process by expressing all delays in terms of a basic delay unit particular to that process.

$$D_{abs}=d\,\tau \qquad (2.1)$$

where τ is the delay of an inverter driving an identical inverter with no parasitic. In a typical 600-nm process τ is about 50 ps. For a 250-nm process, τ

is about 20 ps. For a 180-nm process, τ is about 12ps.In modern 45 nm processes the delay is approximately 4 to 5 ps.

B) Delay Components

The delay incurred by the logic gate is comprised of two components:

- Parasitic delay (p)
- Effort delay or stage delay (f)

Parasitic delay (*p*) is a fixed part which is an intrinsic delay of the gate and can be found by considering the gate driving no load. It is estimated by counting the total transistor width on the output node, assuming diffusion and the gate capacitances are approximately equal. These scale with transistor width so parasitic delay independent of transistor sizes.

Effort delay or stage delay (*f*) which is dependent on the load on the gate's output.

The total delay is sum of parasitic delay and effort delay, and measured in units of τ

$$d=f+p \qquad (2.2)$$

The Effort delay depends on the load and the properties of the logic gate driving the load. The stage delay or effort delay is divided into two components:

Logical effort (*g)* captures intrinsic properties of logic gate, which is the ratio of the input capacitance of a given gate to that of an inverter capable of delivering the same output current .It is independent of the sizes of the transistors.

Electrical effort (*h)* categorizes the load, which is the ratio of the input capacitance of the load to that of the gate. It describes how the electrical environment of the logic gate affects performance and how the size of the transistors in the gate determines its load-driving capability. The electrical effort is defined by

$$h = \frac{C_{out}}{C_{in}} \qquad (2.3)$$

It is also called *fan-out*, which depend on the load capacitances.The stage effort is then simply describes as

$$f = gh \qquad (2.4)$$

Combining Equation 2.2 and 2.3, we obtain the delay through logic gate as

$$d=gh+p \qquad (2.5)$$

Logical effort And The Parasitic Delay In The Full Adder in Table below:

GATE	NO.OF INPUTS	LE	PD
INVERTER	1	1	1
BUFFER	1	1	1
NAND	2	4/3	2
NOR	2	4/3	2

C)Calculating The Logical Efforts

First, assume that all transistors are of minimum length, so that a transistor's size is completely captured by its width *w.* The capacitance of the transistor's gate is proportional to *w* and its ability to produce output current, or conductance, is also proportional to *w* .In most CMOS processes, pull-up transistors must be wider than pull down transistors to have the same conductance $\mu=\mu_n/\mu_p$ is the ratio of PMOS to NMOS width in an inverter for equal conductance. γ is the actual ratio of PMOS to NMOS width in an inverter. For simplicity, we will often assume that $\gamma= \mu=2$ 2. Under this assumption, an inverter will have a pull down transistor of width *w* and a pull-up transistor of width2 *w*, so the total input capacitance can be said to be 3*w.*

Now design a 2-input NAND gate so that it has the same drive characteristics as an inverter with a pull down of width 1 and a pull-up of width 2.Because the two pull down transistors of the NAND gate are in series, each must have twice the conductance of the inverter pull down transistor so that the series connection has a conductance equal to that of the inverter pull down transistor. Therefore, these transistors are twice as wide as the inverter pull down transistor. This reasoning assumes that transistors in series each of the two pull-up transistors in parallel need be only as large as the inverter pull-up transistor to achieve the same drive as the reference inverter. Here we assume that if either input to the NAND gate is LOW, the output must be pulled HIGH, and so the output drive of them NAND gate must match that of the inverter even if only one of the two pull-ups is conducting. By extracting capacitances from the circuit schematic. The input capacitance of one input signal is the sum of the width of the pull down transistor and the pull-up transistor, or 2+2=4. The input capacitance of the inverter with identical output drive is C_{inv}=1+2=3 The logical effort per input of the 2-input NAND gate is therefore g =4/3.Observe that both inputs of the NAND gate have identical logical efforts .Similarly for NOR gate, to obtain the same pull-up drive, transistors four units wide are required, since two of them in series must be equivalent to one transistor two units wide in the inverter. Summing the input capacitance on one input, we find that the NOR gate has logical effort $g=5/3$. This is larger than the logical effort of the NAND gate because pull-up transistors are less effective at generating output current than pull down transistors. Were the two types of transistors similar, i.e., γ =1 both AND and NOR gates would both have a logical effort of 1.5.

III. DESIGN FOR ADDERS

The speed of execution is the most important factor that needs to be considered for apprising the quality of an adder.Traditional CLA is constructed by XOR,AND,OR gate due to which the speed increases.Adders are used in addition The adder mainly used for simple addition is ripple carry adder,Carry look –ahead adder and etc.

A)Carry look –ahead adders:

In a carry look ahead adder the carries entering all the bit positions of the adder are generated simultaneously by a carry look ahead (CLA) generator; that is, computation of carries takes place in parallel with sum calculation. As the word length increases, the hardware organization of the addition technique gets complicated. Hence adders with a large number of elements may require two or three levels of carry look ahead stages.The result is reduced carry propagation time

Here the carry propagate, carry generate are being calculated for each and every bit and then carry is propagated from first bit to the last. The architecture of the 64 bit carry look ahead adder is given below and then verilog model is implemented and output waveform is observed.

The carry propagation logic provides fast paths for the carry to go from one block to another. This helps the designer to implement fast RCA. The user can still implement parallel adders,when compared with the simplest RCA performance.

It uses the concept of generating and propagating carries.

$$C_{i+1}=G_i+(P_iC_i)$$
$$G_i=A_i*B_i$$
$$P_i=A_i\odot B_i$$

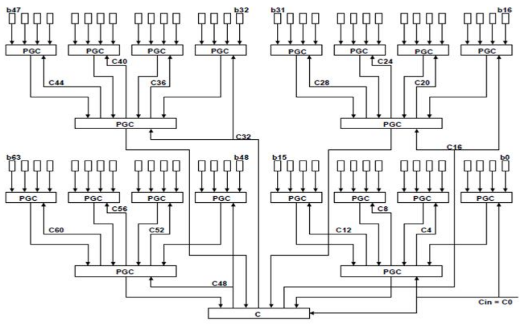


Figure: 3.1 Block Diagram of Carry look-ahead adders

IV. DESIGN FOR MULTIPLIER

Multiplier are most commonly used in various electronic application .Multiplication circuit used for a high speed multiplier in a computer system is constituted by a multiplier and a carry propagating adder . The multiplier obtains a sum and carry per each bit by using carry save adders trees having a plurality of carry save adders ,and generates a carry generation function and carry propagation function based on sum and carry by using a generation /propagation unit.

Multipliers and their associated circuits (adders, subtracters, and accumulators) consume a significant portion in ALU. Therefore, it makes sense to increase their performance/size efficiency by customization. The partial products are first reduced to two products .The reduction trees can reduce the number of partial products to three instead of two to make full use of this block. Any parallel multiplier architecture consists of three basic operations like partial product generation, partial product reduction, and computation of the final sum using a CPA. For each of these operations, several techniques are used to optimize the HW of a multiplier.

Carry Save Addition Saves the Carry at the next bit location. The architecture of a multiplier is shown below in which it comprises of multiplicand, multiplier, partial product reduction, partial product generation, and a carry propagation adder.

Partial product generated in parallel in carry save addition results in faster array multiplier.

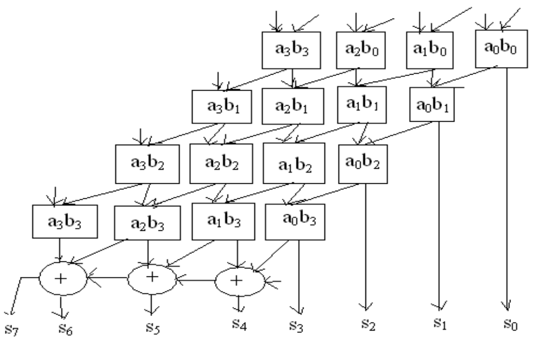


Figure 4.1 Block Diagram of Carry Save multiplier

For a general N1*N2 multiplier, the following four techniques are generally used to reduce N1 layers of the partial products to two layers for their final addition using any CPA namely Carry Save Reduction and this technique techniques is used for partial product reduction

V. DESIGN FOR SHIFTER

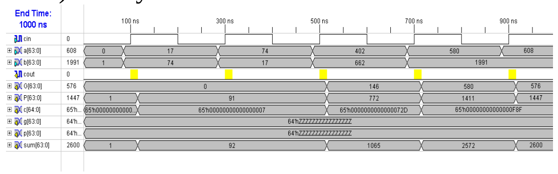
Shifting is required in several applications including arithmetic operations like bit-indexing. Shifters can be useful as efficient ways of performing multiplication or division unsigned integers by powers of two. Shifting left by *n* bits on a signed or unsigned binary number has the effect of multiplying it by 2^n. Shifting right by *n* bits on an *unsigned* binary number has the effect of dividing it by 2^n (rounding towards 0).

A)Barrel Shifter:

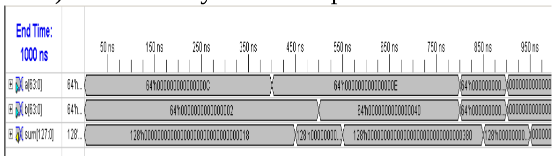
A common usage of a barrel shifter is in the hardware implementation of floating point arithmetic. For a floating-point add or subtract operation, the significant of the two numbers must be aligned, which requires shifting the smaller number to the right, increasing its exponent, until it matches the exponent of the larger number. Barrel Shifter is used to perform shift right and shift left.

VI.SIMULATION AND RESULTS

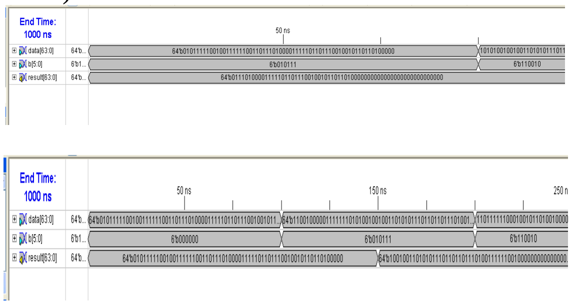
A) Carry look-ahead adder:



B) 64 Bit carry save multiplier:



C) 64 Bit Barrel Shifter:



Comparision of Delays:

ALU COMPONENT	DELAY (XILINX)	DELAY (CADANCE)	DELAY (LOGICAL EFFORT)
Carry Look-Ahead Adder	82.90ns	82.79ns	79.92ns
Carry Save Multiplier	122.665 ns	125.665ns	116.958ns
Barrel Shifter	11.93ns	12.68ns	11.19ns

VII. CONCLUSION

High speed ALU which consists of adders ,multipliers and shifters is designed and implemented using VerilogHDL and the results are verified and optimized by optimizing the paths. building blocks. The delay of the basic building block has been calculated and output is verified by VerilogHDL and Cadance tool.

ACKNOWLEDGEMENT

The authors are very grateful to M.Murali Krishna and Sharma from Gitam University for the study of Logical Effort explained in this study and valuable discussions on 64-bit design of ALU issues

REFERENCES

[1]. I. Sutherland, B. Sproull, D. Harris, “Logical Effort: Designing Fast CMOS Circuits,”Morgan Kaufmann Publisher, 1999.

[2]. V. G. Oklobdzija, E. R. Barnes, “Some Optimal Schemes for ALU ImplementationinVLSITechnology”, Proceedings of 7th Symposium on Computer Arithmetic, June 4-6,1985,

[3]. University of Illinois, Urbana, Illinois.

[4]. V. G. Oklobdzija, “High-Performance System Design: Circuits and Logic”, IEEE Press,1999. H. Q. Dao, V. G. Oklobdzija, “Application of Logical Effort Techniques for Speed Optimization and Analysis of Representative Adders,” 35th Annual Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, November 4–7, 2001

[5]. J. Park et al., “470ps 64-Bit Parallel Binary Adder,” 2000 Symposium on VLSI Circuits Digest of Technical Papers.

❖❖❖