Performance Analysis of a 64-bit signed Multiplier with a Carry Select Adder Using VHDL

Deepthi, Rani, Manasa

Hyderabad Institute of Technology and Management Gowdavelli village, Medchal, Andhra Pradesh

ABSTRACT:
This paper presents a performance analysis of carry-look-ahead-adder and carry select adder signed data multiplier we are using, one uses a carry-look- ahead adder and the second one uses a carry select adder. The main focus of this paper's on the speed of the multiplication operation on these 64-bit multipliers which are modeled using verilog code, A hardware description language. The multiplier with a carry select adder has shown a better performance over the multiplier with a carry select adder in terms of gate delays. In this paper we are going to prove that the area and delay product of carry select adder gives better performance compare with carry-look-ahead adder signed 64 bit multiplier.

Key Words:
Signed Multiplier, Carry-Look-Ahead Adder, Carry Select Adder, Wallace tree, VHDL Simulation & Synthesis.

I. Introduction

Multipliers are most commonly used in various electronic applications e.g. Digital signal processing in which multipliers are used to perform various algorithms like FIR, IIR etc. Earlier, the major challenge for VLSI designer was to reduce area of chip by using efficient optimization techniques to satisfy MOORE'S law. Then the next phase is to increase the speed of operation to achieve fast calculations like, in today's microprocessors millions of instructions are performed per second. Speed of operation is one of the major constraints in designing DSP processors and today's general-purpose processors. However area and speed are two conflicting constraints. So improving speed results always in larger areas. Now, as most of today's commercial electronic products are portable like Mobile, Laptops etc. that require more battery back-up. Therefore, lot of research is going on to reduce power consumption. So, in this paper it is tried to find out the best solution to achieve low power consumption, less area required and high speed for multiplier operation.

II. CARRY LOOK AHEAD ADDER

Carry Look-ahead Adders (CLAAs) are the fastest adders, but they are the worst from the area point of view. Carry Select Adders have been considered as a compromise solution between RCAs and CSLAs because they offer a good tradeoff between the compact area of RCAs and the short delay of CSLAs.

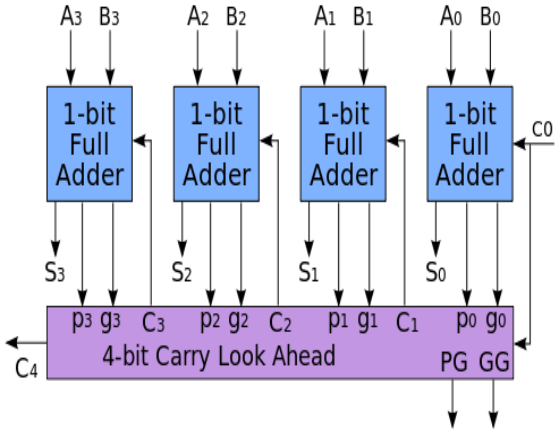

Figure 1: 4-bit carry look ahead adder

Carry look ahead depends on two things:

1. Calculating, for each digit position, whether that position is going to propagate a carry if one comes in from the right.
2. Combining these calculated values to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

Supposing that groups of 4 digits are chosen. Then the sequence of events goes something like this:

3. All 1-bit adders calculate their results. Simultaneously, the look ahead units perform their calculations.
4. Suppose that a carry arises in a particular group. Within at most 5 gate delays, that carry will emerge at the left-hand end of the group and start propagating through the group to its left.
5. If that carry is going to propagate all the way through the next group, the look ahead unit will already have deduced this. Accordingly, before the carry emerges from the next group the look ahead unit is immediately (within 1 gate delay) able to tell the next

Manuscript received November 5, 2015
Manuscript revised November 20, 2015

group to the left that it is going to receive a carry - and, at the same time, to tell the next look ahead unit to the left that a carry is on its way.

The net effect is that the carries start by propagating slowly through each 4-bit group, just as in a ripple-carry system, but then move 4 times as fast, leaping from one look ahead carry unit to the next. Finally, within each group that receives a carry, the carry propagates slowly within the digits in that group.

III. Carry Select Adder

The carry-select adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one. After the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known.

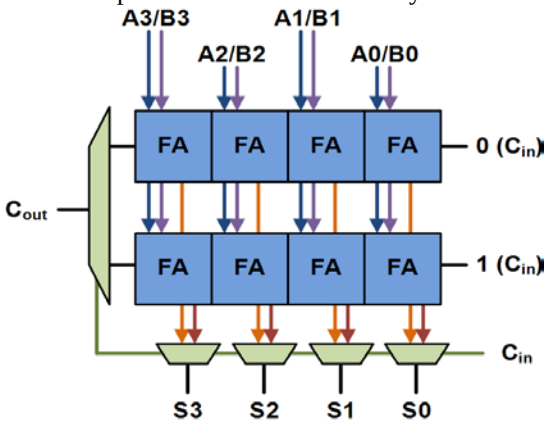

Figure 2: Carry Select Adder

The figure 2 shown basic building block of a carry-select adder, where the block size is 4. Two 4-bit ripple carry adders are multiplexed together, where the resulting carry and sum bits are selected by the carry-in. Since one ripple carry adder assumes a carry-in of 0, and the other assumes a carry-in of 1, selecting which adder had the correct assumption via the actual carry-in yields the desired result.

IV. WALLACE TREE ADDER:

Wallace tree has been used in this project in order to accelerate multiplication by compressing the number of partial products. This design is done using half adders; Carry save adders and the Carry Look Ahead adders to speed up the multiplication. As shown in the figure below, since there are four sign extension values generated namely sign 1E, 2E, 3E and 4E for the partial product PP1, PP2, PP3 and PP4 respectively. The arrangement of total four partial product s is shown in the figure below. The second partial product had to be shifted left by two bits before adding to the first partial product.

Hence the third will be shifted left by four where as for fourth it will be shifted left by six. Hence after proper arrangement all the four partial products will be added along with the sign extension. The multiplier takes in two 8-bit operands: the multiplier (MR) and the multiplicand (MD), and produces the 16-bit multiplication result of the two at its output.

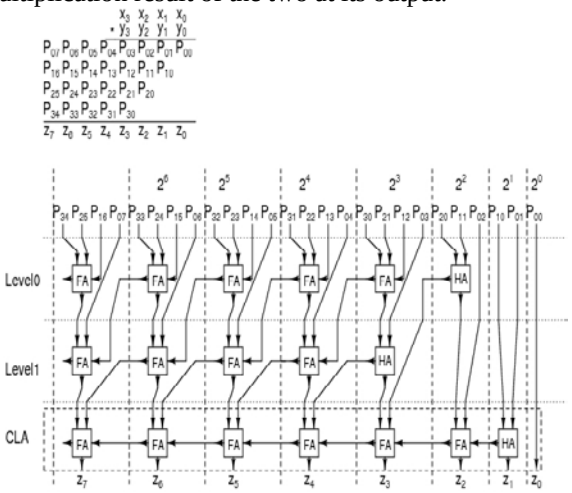

Figure: 3. Partial Product Initial Arrangement

V. BOOTH MULTIPLIER (RADIX-2):

The Booth algorithm was invented by A. D. Booth, forms the base of Signed number multiplication algorithms that are simple to implement at the hardware level, and that have the potential to speed up signed multiplication Considerably. Booth's algorithm is based upon recoding the multiplier, y, to a recoded, value, z, leaving the multiplicand, x, unchanged. In Booth recoding, each digit of the multiplier can assume negative as well as positive and zero values. There is a special notation, called signed digit (SD)en coding, to express these signed digits. In SD encoding +1 and 0 are expressed as 1and 0, but -1 is expressed as 1(Vincent P. Heuring, 2003).The value of a 2s complement integer was defined a by equation 1.

$$y = -y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i$$

This equation says that in order to get the value of a signed 2's complement number, multiply the m – I th digit by -2`-1, and multiply each remaining digit i by +2g.For example, -7, which is 1001 in 2's complement notation, would be, in SD notation, 1001 = -8 + 0 + 0 + 1 = -7.For implementing booth algorithm most important step is booth recoding.

By booth recoding we can replace string of 1s by 0s. For example the value of strings of five 1s,11111 = 2□ - 1 = 100001□ = 32 – 1 = 31. Hence if this number were to be used as the multiplier in a multiplication, we could replace five additions by one addition and one subtraction.

The Booth recoding procedure, then, is as follows:

1. Working from lsb to msb, replace each 0 digit of the original number with a 0 in the recoded number until a 1 is encountered.
2. When a 1 is encountered, insert a 1 at that position in the recoded number, andskip over any succeeding I's until a 0 is encountered.
3. Replace that 0 with a 1 and continue .This algorithm is expressed in tabular form in Table 1, considering pairs of numbers.

Table: 1. Booth recoding table for radix-2.

y_i	y_{i-1}	z_{i-i}	Multiplier Value	Situation
0	0	0	0	String of 0s
0	1	1	+1	End of string of 1 s
1	0	1	-1	Begin string of 1s
1	1	0	0	String of 1s

Vi. Array Multiplier Using Cla and Csa

Though Wallace Tree multipliers were faster than the traditional Carry Save Method, it also was very irregular and hence was complicated while drawing the Layouts. Slowly when multiplier bits gets beyond 32-bits large numbers of logic gates are required and hence also more interconnecting wires which makes chip design large and slows down operating speed Booth multiplier can be used in different modes such as radix-2, radix-4, radix-8 etc. But we decided to use Radix-4 Booth's Algorithm because of number of Partial products is reduced to n/2.

Multipliers are key components of many high performance systems such as FIR filters, Microprocessor, digital signal processors, etc.(Hsin-Lei Lin , 2004). Signed multiplication is a careful process. With compared to unsigned multiplication

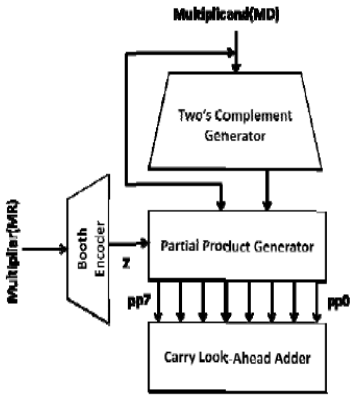

Figure: 4. Architecture of signed multiplier

VII. Simulation Results

The VHDL simulation of the two multiplier is presented in this section. By using Xilinx's 14.2E software we done 64 bit CLA and CSA simulation results with time delay as shown in figure 5 and Figure 6.

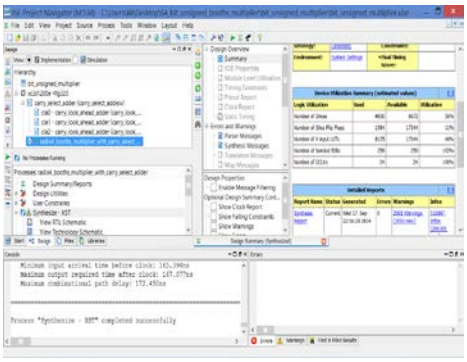

Figure 5: Carry Look Ahead Adder Simulation results

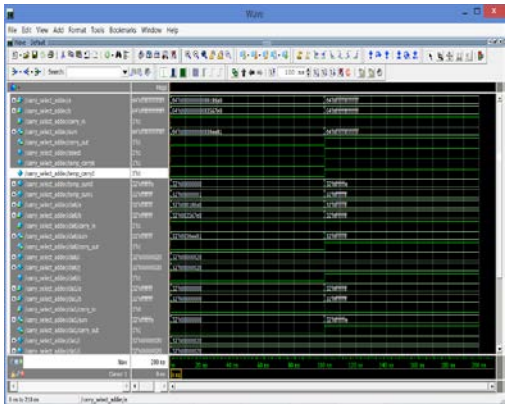

Figure 6: Carry Select Adder Results

VIII. Results and Discussion

In this section, the results obtained from Synthesis and Simulation reports are presented. The aim of this experiment is to evaluate the performance of two Array multipliers (one by using CLA and second by using CSA) on the basis of Area required, Speed of operation and power consumption. As shown in table I, figure 5 and 6, multiplier with CSA has shown better results than with CLA. Area results are presented in terms of number of CLBs and gate count required for implementing design on FPGA. Multiplier with CSA requires less CLBs because it requires less number of full adders than multiplier with CLA.

Table 2: 64 bit signed multiplier

Adders	Delay(ns)	Area(logic cells)
CLAA	172.4	4936
CSLA	172.02	4018

Further, simulation result shows that multiplier with CSA takes less time to generate final product than with CLA because addition is performed in parallel without waiting for the previous result in case of CSA. Similarly, result shows slight improvement in power consumption in case of multiplier using CSA. Power consumption depends on the switching activities. Therefore power consumption is directly proportional to area covered by the design on chip. Here we take dynamic power consumption for performance analysis.

Conclusion and Future Work

Use booth's multiplier with CSLA if area is critical. Use booth's multiplier without CSA if area is critical and a bit of compromise on timing can be made. The Design of high speed bit signed multiplier using adders is proposed. Simulation and synthesis of high speed Bit signed multiplier using CLAA and CSLA has been done in Xilinx 10.2 E using Verilog Hardware Description Language. The CSLA increases the performance of the multiplier.

This radix-4 algorithm can be extended to radix-16 algorithms to get an high speed and efficient multiplication This 64 bit multiplier can be further extended to 128 bit multiplier and 256 bit multiplier using the proposed method for multiplication operation can be done as future work.

REFERENCES

[1] B. Parhami, Computer Arithmetic, Algorithm and Hardware Design, Oxford University Press, New York, pp. 91-119, 2000.

[2] Stephen Brown and Zvonko Vranesic, Fundamentals of Digital Logic with VHDL De sign.2nd Edn. McGraw-Hill Higher Education, USA.ISBN: 0072499389, 2005.

[3] Wakerly, J.F., 2006. Digital Design-Principles and Practices. 4th Edn. Pearson Prentice Hall, USA.ISBN: 0131733494.

[4] Pong P. Chu "RTL Hardware Design Using VHDL: coding for Efficiency, Portability and Scalability" Wiley-IEEE Press, New Jercy, 2006

[5] Hasan Krad and Aws Yousif Al-Taie, "Performance Analysis of a 32-Bit Multiplier with a Carry-Look-Ahead Adder and a 32-bit Multiplier with a Ripple Adder using VHDL", Journal of Computer Science 4 (4): 305-308, 2008

[6] Asadi, P. and K. Navi "A novel high-speed 54-54-bit multiplier", Am. J. Applied Sci., 4 (9): 666-672, 2007.

[7] Z. Abid, H. El-Razouk and D.A. El-Dib, "Low power multipliers based on new hybrid full adders", Microelectronics Journal, Volume 39, Issue 12, Pages 1509-1515, 2008.

[8] Nagendra, C.; Irwin, M.J.; Owens, R.M .,"Area-time-power tradeoffs in parallel adders", Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on Volume 43, Issue 10, Page(s): 689 – 702, 1996.

[9] Sertbas, A. and R.S. Özbey, 2004. A performance analysis of classified binary adde r architectures and the VHDL simulations. J. Elect. Electro n. Eng., Istanbul, Turkey, 4: 1025-1030.

[10] Fonseca, M.; da Costa, E. et al, "Design of a Radix-2m Hybrid Array Multiplier Using Carry Save Adder"Sept. 2005 Page(s): 172-177.

E.Deepthi is working as asst.profesor in Hyderabad institute of technology and mangement and her areas of interest are VLSI design,DSP, Antenna's

V.Moshe Rani is working as asst.profesor in Hyderabad institute of technology and mangement and her areas of interest are VLSI design,DSP, Antenna's

K.Manasa is working as asst.profesor in Hyderabad institute of technology and mangement and her areas of interest are VLSI design,DSP, Antenna's