High Speed Area Efficient 32 Bit Wallace Tree Multiplier

Keshaveni N.
Professor, ECE dept., KVG
College of Engineering Sullia,
Karnataka INDIA

ABSTRACT
A 32 bit high speed area efficient Wallace tree multiplier is designed using verilog HDL and implemented in FPGA. The circuit is designed using carry save adder architecture and finally with one look ahead carry adder. The design is an improved version of tree based Wallace tree multiplier architecture. This paper aims at high speed multiplication and an area efficient 32 bit Wallace tree multiplier. The entire design is coded in Verilog HDL, simulated with Modelsim and synthesized using Xilinx FPGA device. The result shows that the proposed architecture takes very less time for computing the multiplication of two 32 bit numbers. In terms of area also, the proposed multiplier is much efficient than the existing methods. The frequency of operation of the circuit is 200 MHz.

General Terms
Algorithm, Verilog code, Multipliers

Keywords
Carry save adder, FPGA, Modelsim simulator, Wallace tree multiplier

1. INTRODUCTION
Multiplication is the basic arithmetic operation and is widely used everywhere during computation. In digital signal processing, most of the arithmetic operations require the use of multiplications. The performance of three dimensional computer graphics mostly depends on the performance of multiplications. Therefore, there has been much work on advanced multiplication algorithms and design also. Critical factors in the design of multipliers are chip area and speed of multiplication. There is highly demand of high-speed multiplications and require less hardware. The performance of multiplier is affected by the multiplication strategy and type of the multiplier used.

2. MULTIPLICATION METHODOLOGY
The use of carry save adder, to perform multiplication, first calculates the partial products of the multiplication, and then input them to the carry-save adder. For example, consider the multiplication of binary values 011010 and 110101. It generates partial products as shown Figure 1. The partial products are input to two carry-save adders at the top of the Wallace Tree as shown in Figure 2.

The 6-bit Wallace Tree multiplier, with partial products and final results for this example, is also shown in Figure 2. The partial products are fed to the carry-save adders which generates sum and carry outputs. These outputs are combined using additional carry-save adders until only two outputs are left at the end. These values are added using a parallel adder to produce the final product.

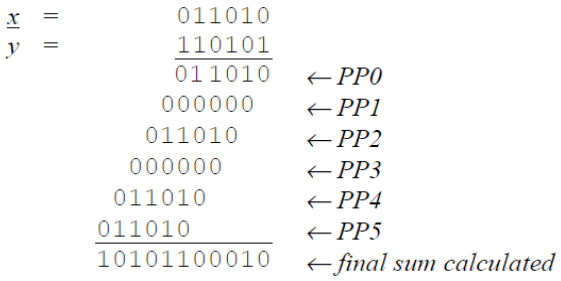

Fig 1 : Generation of partial products

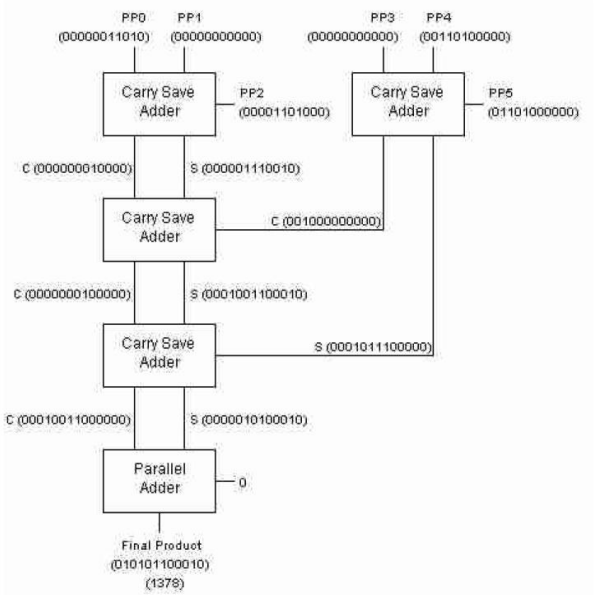

Fig 2 : 6 bit Wallace tree

3. CARRY SAVE ADDER CONCEPT
A carry-save adder can add three values simultaneously, instead of just two. However, it does not output a single result. Instead, it outputs both a sum and a set of carry bits. The carry-save adder is essentially a group of full adders, each of which adds the bits in the same position of its three input sum operands. The full adder that adds bit *i* of its operands outputs bit *Si* and carry bit *Ci+1*. Because the carry bits do not propagate through the adder, it is faster than parallel adders. Unlike the parallel adder, though, it does not produce a final sum of its inputs. A carry-save adder can add three values simultaneously, instead of just two. However, it does not output a single result. Instead, it outputs both a sum and a set of carry bits.

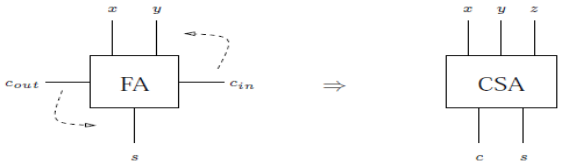

Fig 3: The Carry save adder block resembles the full adder circuit

x:			1	0	0	1	1
y:			1	1	0	0	1
z:	+		0	1	0	1	1
s:			0	0	0	0	1
c:	+	1	1	0	1	1	
sum:		1	1	0	1	1	1

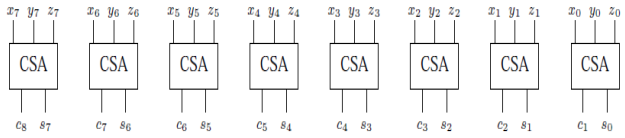

Fig 4: One CSA block is used for each bit. This circuit adds three N = 8 bit numbers together into two new numbers

The computation of sum s and carry c is as follows: It is actually identical to the full adder, but with some of the signals renamed. Figure 3 shows a full adder and a carry save adder. A carry save adder simply is a full adder with the cin input renamed as z, the z output (the original "answer" output) renamed to s, and the cout output renamed to c.

Figure 4 shows how n carry save adders are arranged to add three n bit numbers x, y and z into two numbers c and s. Note that the CSA block in bit position zero generates c1, not c0. Similar to the least significant column when adding numbers by hand (the "blank"), c0 is equal to zero. Note that all of the CSA blocks are independent, thus the entire circuit takes only O (1) time. To get the final sum, we still need a Look ahead carry adder (LCA), which will cost us O (log n) delay. The asymptotic gate delay to add three n-bit numbers is thus the same as adding only two n-bit numbers. Therefore, the numbers that go to the LCA will be at most (n + m – 2) bits long, where m is the total numbers to be added and n is the number of bits in each number. So the final LCA will have a gate delay of O(log (n + m)). Therefore the total gate delay is O(m + log (n + m)) instead of arranging the CSA blocks in a chain, a tree formation can actually be used. In the present work a 32 bit Wallace tree multiplier is designed and coded in verilog, test bench is written and simulated using Modelsim simulator, implemented in FPGA, and the results are compared with the existing methodologies.

4. RESULTS AND CONCLUSION

The entire design of a 32 bit Wallace tree multiplier is coded in verilog and implemented in Xilinx FPGA. The RTL schematic view of the design is presented in figure 5. Test bench is written and different combinations of inputs are taken, simulated using Modelsim simulator. From the simulation results of figure 6, it can be seen that initially, the inputs taken are decimal 0 and 0, next number is 1 and 7, the next is 7 and 7. The design is a synchronous design with respect to clock signal "clk". From the waveform shown in figure 6, it can be observed that, it takes only 5ns time to get the first multiplied output. Similrly from the figure 7 it can be seen that consecutive multiplications takes only 5ns time. Therefore the maximum frequency of operation is 200MHz. The device utilization summary is shown in figure 8. The target FPGA device is Virtex5 xc5vlx330. It can be seen from figure 8, the total gate count of the design is 14,720 and slice LUTs 2044.

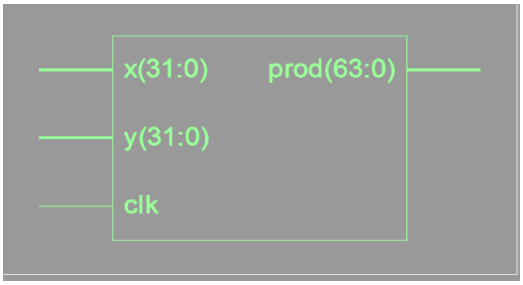

Fig 5: RTL schematic of multiplier

Table 1. Area comparison

Type of multiplier	Width	Slice LUTs used
Modified booth multiplier (Radix 8)	32 bit	2721
Booth recoder multiplier	32 bit	2704
Proposed method	32 bit	2044

Table II. Delay comparison

Type of multiplier	Width	Delay (ns)
Multiplier using Vedic mathematics	16 bit	13.452
Modified Booth multiplier (Radix-8)	32 bit	11.564
Booth recoder multiplier	32 bit	9.536
Proposed method	32 bit	5

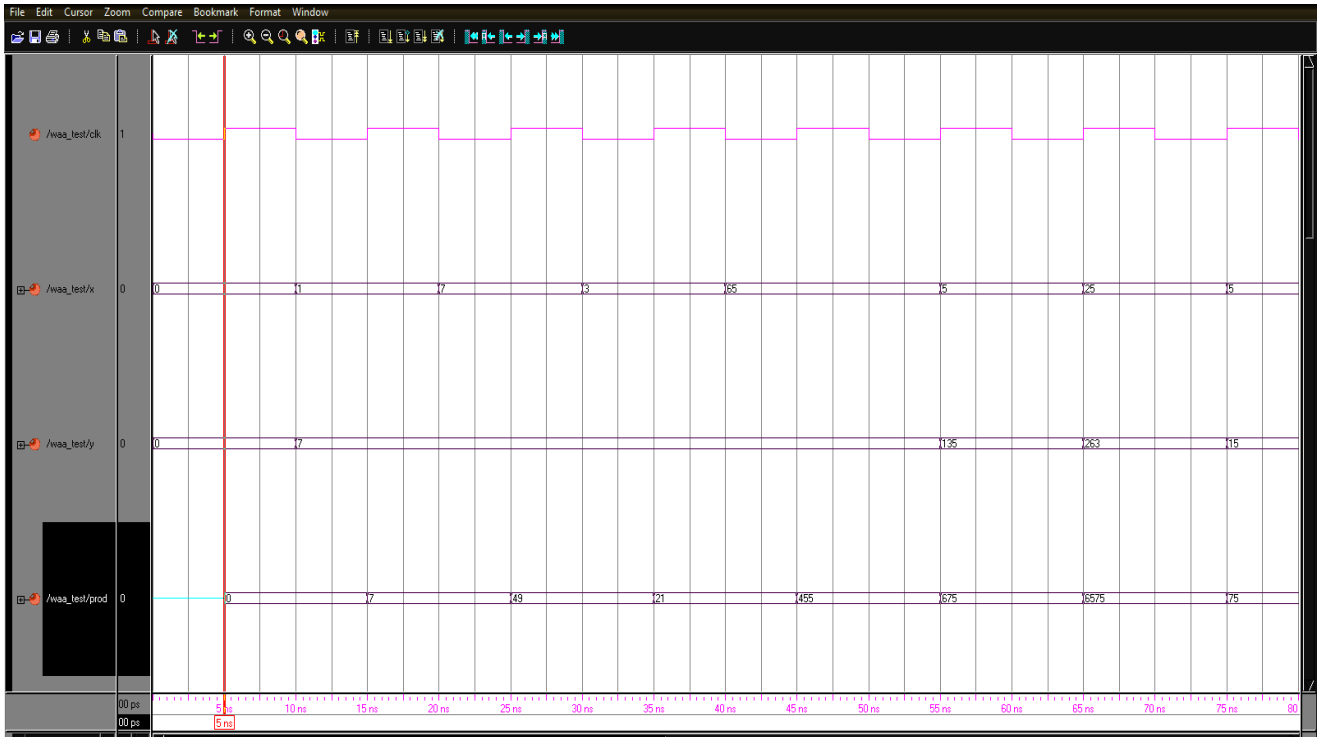

Fig 6: Simulation waveform of multiplier

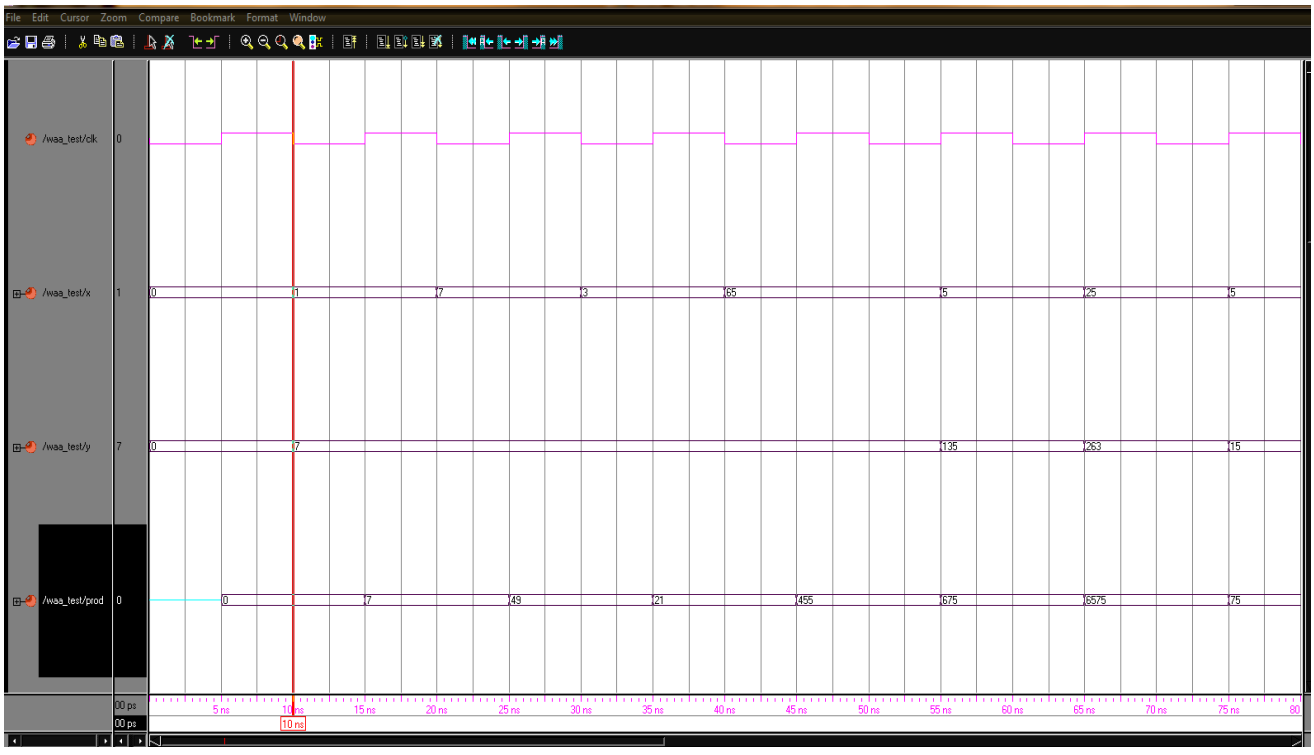

Fig 7: Simulation waveform of multiplier

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	2,044	207,360	1%
Number used as logic	2,044	207,360	1%
Number using O6 output only	2,040		
Number using O5 and O6	4		
Slice Logic Distribution			
Number of occupied Slices	848	51,840	1%
Number of LUT Flip Flop pairs used	2,044		
Number with an unused Flip Flop	2,044	2,044	100%
Number with an unused LUT	0	2,044	0%
Number of fully used LUT-FF pairs	0	2,044	0%
IO Utilization			
Number of bonded IOBs	129	1,202	10%
IOB Flip Flops	64		
Specific Feature Utilization			
Number of BUFG/BUFGCTRLs	1	32	3%
Number used as BUFGs	1		
Total equivalent gate count for design	14,720		
Additional JTAG gate count for IOBs	6,192		

Fig 8: Device utilization summary

The present work is area efficient and faster than the existing methods. One of the authors [6] reported that the total number of slice LUTs for a 32 bit Wallace tree multiplier is 2704. Table I shows the comparison of the area of various types of Wallace tree multipliers and Table II shows the comparison in terms of delay of the various Wallace tree multipliers. Therefore comparing these results with the present work, it can be concluded that the present work is area efficient and faster than the existing methods. Also, it can be seen that, in the present work, the time required for 32 bit Wallace tree multiplication, 5 ns, is not yet reported in the literature. In future, the design can also be focused on floating point multiplication.

5. REFERENCES

[1] King Fai Pang, IEEE 1990, Architecture for pipelined Wallace tree multiplier- accumulators.

[2] Akther S, European conference oncircuit theory and design, August 2007, VHDL implementation of fast NxN multiplier based on vedic mathematics.

[3] C Vinoth, V S Kanchana Bhaskaran, IEEE 2011, A novel low power and high speed wallace tree multiplier for RISC Processor.

[4] N Surekha, R Porselvi, K K Kumuthapriya, 2012, An efficient high speed wallace tree multiplier.

[5] Monika Vaishnav, October 2012, Volume 1, No.1, IJISCS, Design of multi-precision reconfigurable Wallace Tree Multiplier for high performance applications.

[6] Jagadeshwar Rao M, Sanjay Dubey, Asia Pacific conference 2012, A high speed and area efficient booth encoded Wallace tree multiplier for fast arithmetic circuits.

[7] Rahul D Kshirasagar, Aishwarya.E.V, Ahire Shashank Vishwanath, P Jayakrishnan, IEEE 2013, Implementation of pipelined booth encoded Wallace tree multiplier architecture.

[8] Damarla Paradhasaradhi, N Prashanti, N Vivek, IEEE 2013, Modified Wallace tree multiplier using efficient square root carry select adder.

[9] M Ravindra Kumar, August 2013, International journal of innovative research and studies, ISSN 2319-9725, Design and implementation of 32*32 bit high level Wallace tree multiplier.

[10] Kartikeya Bhardwaj, Praveen S, IEEE 2014, Power and area efficient approximate wallace tree multiplier for error resilient systems.

.