Comparison of 4-bits Unsigned Multipliers

Wen Bo Zhang
School of Electrical and
Computer Engineering
McGill University

Simon Dang Khoa Ho
School of Electrical and
Computer Engineering
McGill University

Zixuan Yin
School of Electrical and
Computer Engineering
McGill University

***Abstract*—Two 4-bits unsigned multiplier are designed and implemented for comparison purposes. The array multiplier and the Booth encoding Radix-4 multiplier were designed and implemented in Electric. The criterion for evaluation included delay analysis, power comsumption and layout size of the implementations. IRSIM was the simulation software used for delay and power simulations and the Electric layout implementations were used for size comparison.**

I. Introduction

Hardware multiplication can be implemented in many different ways. Each of the different algorithms have its own advantages and draw back in term of the VLSI design. A way to compare the different kinds of multiplication algorithm is to implement them in layout using CAD tools such as Electric and then use simulation softwares to evaluate their performance. For simplicity's sake, two 4-bits unsigned multipliers are compared and evaluated in the scope of this project. The goal is to determine if the Radix-4 Booth multiplier has a sizeable advantage over the simple array multiplier.

A. Booth Algorithm

The idea is to reduce the number of partial products by half, by using the technique of radix 4 Booth recoding. Instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by 1, 2, or 0, to obtain the same results.

Booth
PP0 = Multiplicand * -1, shifted left 0 bits (x-1)
PP1 = Multiplicand * 2, shifted left 2 bits (x 8)

Array
PP0 = Multiplicand * 1, shifted left 0 bits (x 1)
PP1 = Multiplicand * 1, shifted left 1 bits (x 2)
PP2 = Multiplicand * 1, shifted left 2 bits (x 4)
PP3 = Multiplicand * 0, shifted left 3 bits (x 0)

To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit.

B. Encoder Design

At the beginning of our design, a preliminary schematic of our multiplier was drawn based on the calculation example. The encoder cells for each partial product and selector cell are designed based on the Booths multiplication algorithm introduced in CMOS VLSI design [1]. The 8-bit binary adder are provided in standard cell library wordlib8.

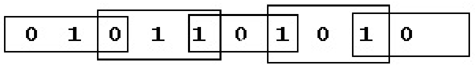

Figure 1. Booth encoding grouping

Block	Partial Product
000	0
001	1 * Multiplicand
010	1 * Multiplicand
011	2 * Multiplicand
100	-2 * Multiplicand
101	-1 * Multiplicand
110	-1 * Multiplicand
111	0

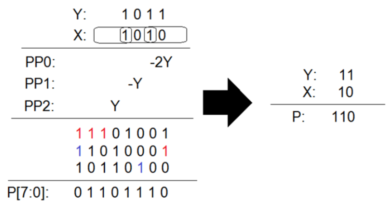

Figure 2. Calculation example

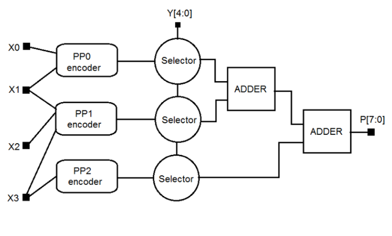

Figure 3. High level schematic

The encoder cells produce control signals SINGLE, DOUBLE and NEG to indicate the value of each partial product.

Partial product PP2 is a special case that only have two possible value: 0 and Y. Therefore, the PP2 encoder and selector is implemented using AND gates in high-level schematic.

Inputs		Partial Product	Booth selector		
X_1	X_0	PP0	SINGLE0	DOUBLE0	NEG0
0	0	0	0	0	0
0	1	Y	1	0	0
1	0	-2Y	0	1	1
1	1	-Y	1	0	1

$$SINGLE0 = x_0$$
$$DOUBLE0 = x_1 x_0'$$
$$NEG0 = x_1$$

Figure 4. PP0 Encoder logic

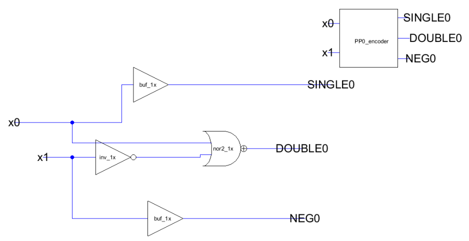

Figure 5. PP0 Encoder schematic

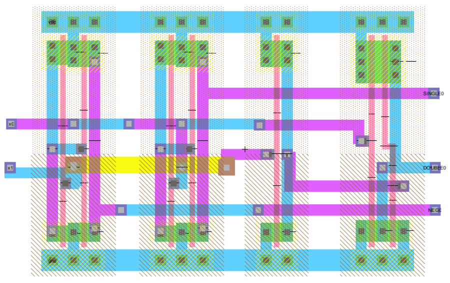

Figure 6. PP0 Encoder layout

Inputs			Partial Product	Booth Selector		
X_{2i+1}	X_{2i}	X_{2i-1}	PP_i	$SINGLE_i$	$DOUBLE_i$	NEG_i
0	0	0	0	0	0	0
0	0	1	Y	1	0	0
0	1	0	Y	1	0	0
0	1	1	2Y	0	1	0
1	0	0	-2Y	0	1	1
1	0	1	-Y	1	0	1
1	1	0	-Y	1	0	1
1	1	1	0	0	0	0

Figure 7. PP1 Encoder logic

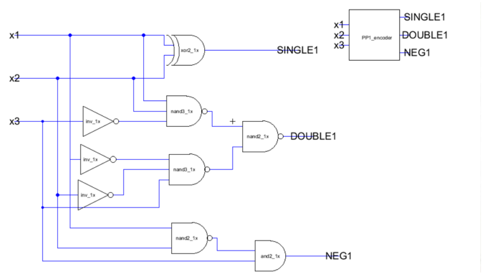

Figure 8. PP1 Encoder schematic

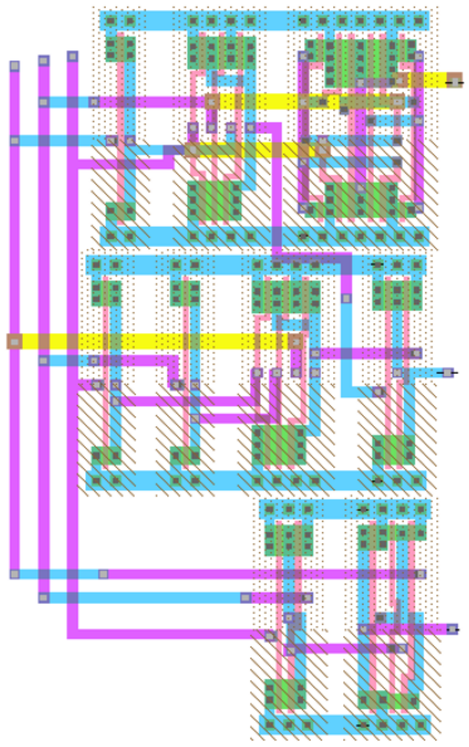

Figure 9. PP1 Encoder layout

X_3	PP_2
0	0
1	Y

$$PP2_i = AND(x_3, Y_i)$$

Figure 10. PP2 Encoder logic

C. Selector Design

The booth selector cell produce a 5-bit partial product based on control signals from encoder and 4-bit multiplicand Y. The logical function of Booth selector is shown below.

$$PP_i = XOR(y_i * SINGLE + y_{i-1} * DOUBLE, NEG)$$

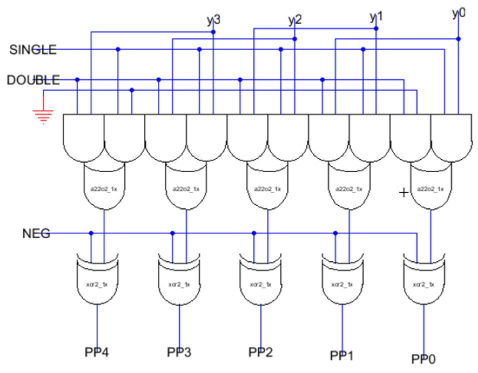

Figure 11. Booth selector schematic

D. Sign extension and 2s complement

In order to produce the correct answer, each partial product from previous stage needs to be shifted by certain amount and assigned with corresponding sign extension. The addition of 1 for negative partial products is implemented in the next partial product. The bitwise implementation for each partial product is shown in the table below. NEG0 and NEG1 correspond to the NEG control signal from Booth encoders.

bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
NEG0	NEG0	NEG0	PP0[4]	PP0[3]	PP0[2]	PP0[1]	PP0[0]
NEG1	PP1[4]	PP1[3]	PP1[2]	PP1[1]	PP1[0]	0	NEG0
PP2[3]	PP2[2]	PP2[1]	PP2[0]	0	NEG1	0	0

E. Multiplier Schematic and layout

After implementing each fundamental cells, the high-level schematic and layout of Booth multiplier was put together. Power and ground wires are provided on both side of the circuit. All input and output exports are provided at the top of the layout. The cell passed NCC, ERC and DRC verifications in Eletric.

F. Performance Evaluation

As a metric to compare the array and Booth multiplier, the power, area and delay of both implementation were analyzed. A stronger emphasis was given for timing measurement since we evaluated that the main drawback of an array multiplier is its long critical path, which has a stronger influence on the propagation delay difference. The testing simulations for power and delay were performed using IRSIM, a tool for simulating digital circuits. [2] It is a ”switch-level” simulator; that is, it treats transistors as ideal switches. Extracted capacitance

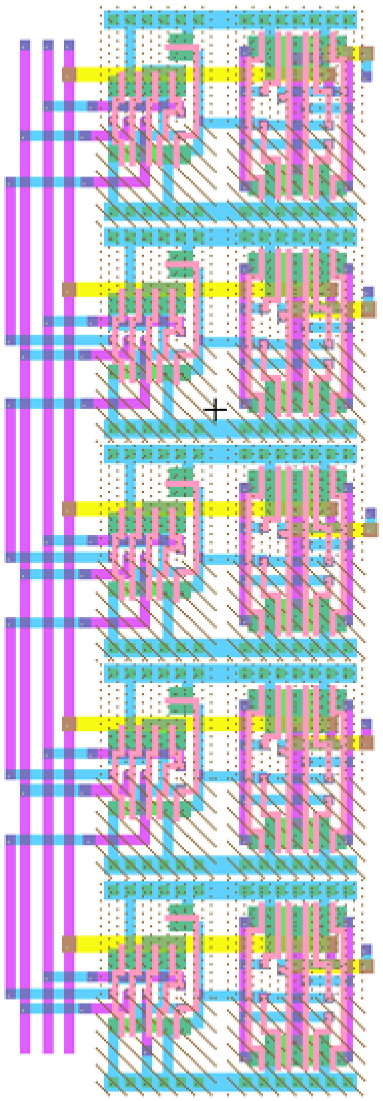

Figure 12. Booth selector layout

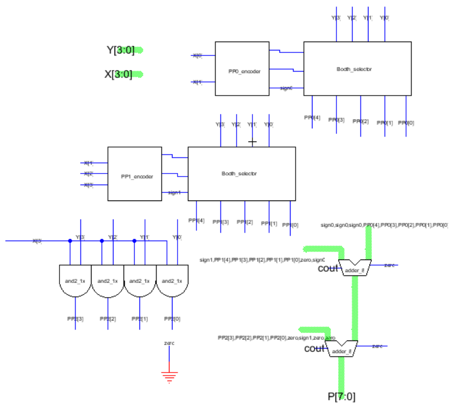

Figure 13. Booth multiplier schematic

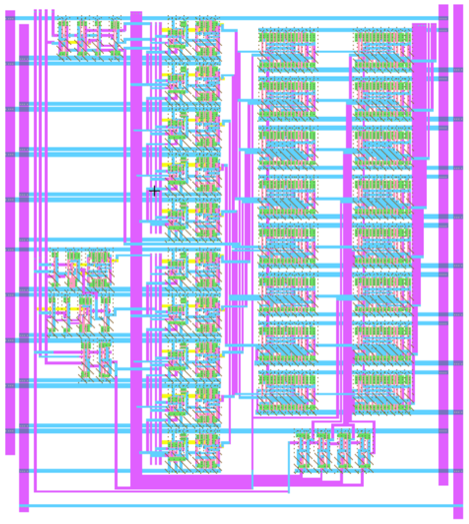

Figure 14. Booth multiplier layout

and lumped resistance values are used to make the switch a little bit more realistic than the ideal, using the RC time constants to predict the relative timing of events. In these simulations we have used Vdd= 5V and f=1MHz frequency operation.

G. Area

For the area measurement, the Electric VLSI designs system software was used to count the number of transistors as well as to evaluate the width and height of the layouts. As observed in Table 1, the Booth multiplier has a lower number of transistors but covers a larger space area than the array multiplier. This can be explained by how the array multiplier is structured, which can be easily compacted and optimized in space since it's composed of the same building structure: carry save adders. For Booth multiplier, as seen in Figure 9, it is more difficult to use similar tricks because of the way the encoders and booth selectors are placed. For example, the partial product 0 encoder on the left significantly increase the design width and leaves a lot of unused white space below it. Power rails, which are absent in the array design, also increase the area. A smaller Booth implementation is definitely possible and is subject to further optimizations if given more time.

Parameter	Booth	Array	Diff(%)
Number of transistors	2542	3418	+34.5%
Width(um)	1029	656	-44.27%
Height(um)	1155	518.5	-76.06%
Total area(mm)	0.06781	0.03401	- 66.39%

H. Timing

For the delay, a testbench with every 8 bit input combinations (256 inputs) was tested to verify the design functionality. The file testbench.cmd in the deliverable presents the script used for the multipliers and without any issues, the outputs were exactly the same. After the functionality has been ensured, the delay from 00000000 to every single input was measured. Figure 10 shows the inputs x0, x1, x2 ,x3, y0, y1, y2, y3 and outputs p0, p1, p2, p3, p4, p5, p6, p7 were displayed with the delay displayed at the top. The multiplier.xml excel file presents the data and IRSIM delays generated. As predicted, booth delay is faster, with an average delay and worst case delay of 15.133 ns, 23.08 ns and 20.047 ns, 29.76 ns respectively for Booth and array. This represents a 25.28

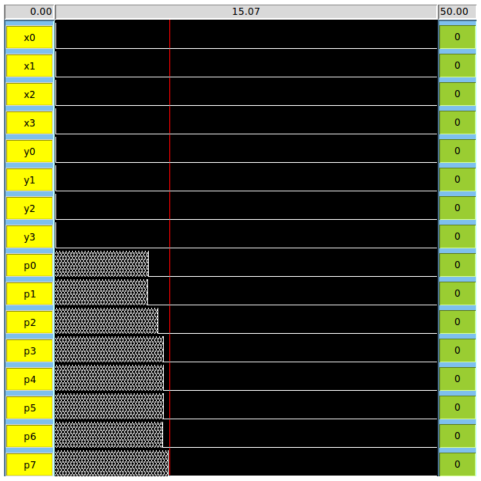

Figure 15. IRSIM graphical analyzer

I. Power

An initial dynamic power estimation calculation was performed with Equation 1. An activity factor of 0.1 was used which is standard constant used for the fraction of the circuit that is switching, a supply voltage of 5V and clock frequency of 1 Mhz. The total nodal capacitance was found to be 33.61 pF and 28.87 pF for Booth and array respectively which gives a theoretical dynamic power of 0.0168 mW and 0.0144 mW. This is very close to the IRSIM power given at 0.0835 mW and 0.0297 mW, which considers transistors as ideal switches as mentioned earlier. As seen in Table 2, the Booth multiplier consumes much more power than array multiplier, 95.05

$$P = \alpha C V^2 f$$

Parameter	Booth	Array	Diff(%)
Nodal capacitance	33.61	28.87pF	-15.17
Dynamic power (theory)	0.0168mW	0.0144mW	-15.38
Dynamic power (practical)	0.0835mW	0.0297mW	-95.05

J. Test Simulation Analysis

The method of Booth recording reduces the numbers of adders and hence the delay required to produce the partial sums by examining three bits at a time. The high performance of booth multiplier comes with the drawback of power consumption. The reason for this is the large number of adder cells that consume power. For the array multiplier, it gives better power consumption as well as optimum number of components required, but delay for this multiplier is larger. It also requires larger number of gates because of which area is also increased; due to this array multiplier is less economical . Thus, it is a fast multiplier but hardware complexity is high.

II. Conclusion

Comparison of the logic and electrical level estimates were done and support that the Booth radix-4 multiplier consumes more power and has less delay. If given more time, a sign extension corrector would have been added to our design to enhance the ability of the Booth multiplier to multiply not only the unsigned number but as well the signed number. Additionally, as originally planned in the project proposal, the three multipliers designs (Wallace and Dadda trees, and SFQ multiplier) would have been implemented to compare more implementations.

References

[1] N. Weste and P. D. Harris, *CMOS VLSI design*, 1st ed. Boston: Pearson/Addison-Wesley, 2005.

[2] http://opencircuitdesign.com/irsim/

[3] T. Callaway and E. Swartzlander. *Optimizing multipliers for WSI*, In Fifth Annual IEEE International Conference on Wafer Scale Integration, pages 85-94, 1993.