N-Bit Booth Multiplier using Carry Save Adders

Self Project

Muhammad Shoaib Iqbal
Mtech – EE5
IIT-B

Date : 15 july, 2021

Introduction

The conventional array multiplier computes partial product for every bit of multiplier by using AND gates. To speed up the multiplier, the number of partial products can be reduced by multiplying 2 bits at time. Since we have to add more than two bits at a time, we can design an adder architecture optimized for this. Booth algorithm and carry save adders can significantly speed up the multiplication.

Booth Encoding

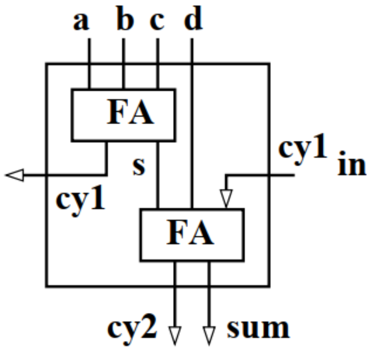
- Let the multiplicand be M and multiplier be Q
- Depending upon the last two bits of Q (00,01,10,11), the partial products will be 0, A, 2A, and 3A
- 0 and A can be produced easily
- 2A and 3A can be generated as 4A-2A and 4A-A
- The task of adding 4A is passed on to the next group of 2 bits of the multiplier
- Since the place value of the next group of 2 bits is 4 times the current one, adding 4A is equivalent to adding 1 to the next group

The following table summarize the effective multiplier for generating the partial products

Current 2-bits	Multiplier of these	Previous MSB	Pending Increment	Total Multiplier
00	0	0	0	0
01	+1	0	0	1
10	-2	0	0	-2
11	-1	0	0	-1
00	0	1	+1	1
01	+1	1	+1	2
10	-2	1	+1	-1
11	-1	1	+1	0

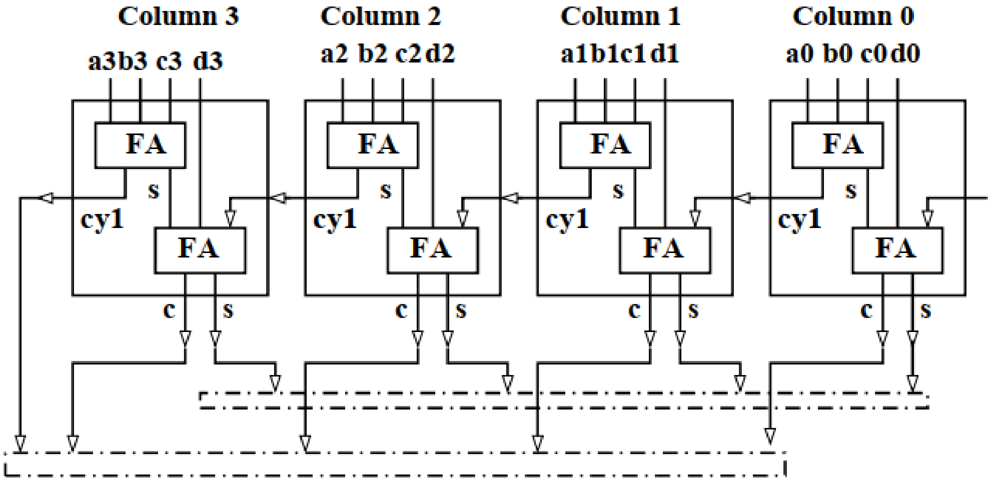
Carry Save Adders

Ripple carry adders are slow due to rippling of carry. CSA can be used for adding 4 bits the same column. A 4 input 2 output CSA is shown in the following figure.



The first full adder takes 3 bits of partial product of same weight. The sum output of first FA goes to next FA. The carry output (cy_1) goes as intermediate input to the CSA used in the column to the left of this. The second FA accept one more bit from partial product column. Even though cy_1 goes to the next CSA it does not ripple all the way horizontally, hence increase speed.

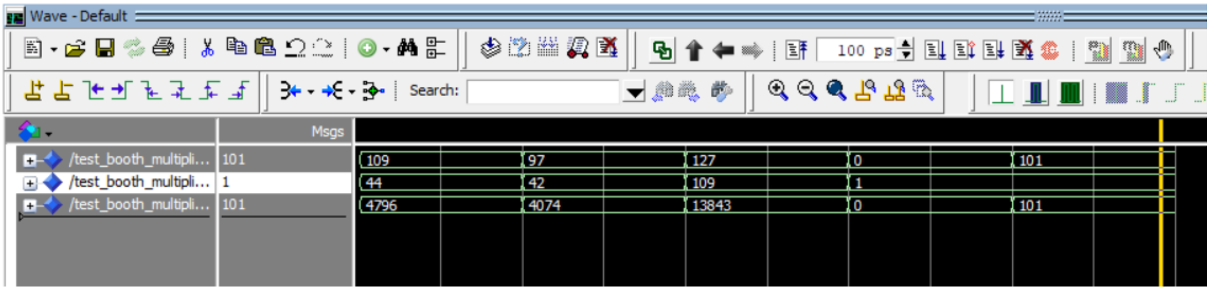
The following figure shows addition of 4 columns of 4 bits each. Rows are labelled as a, b, c, and d. Columns are 0, 1, 2, and 3.



Outputs are collected in two separate registers shown in dotted lines known as sum and carry registers. The contents are added using conventional ripple carry adder.

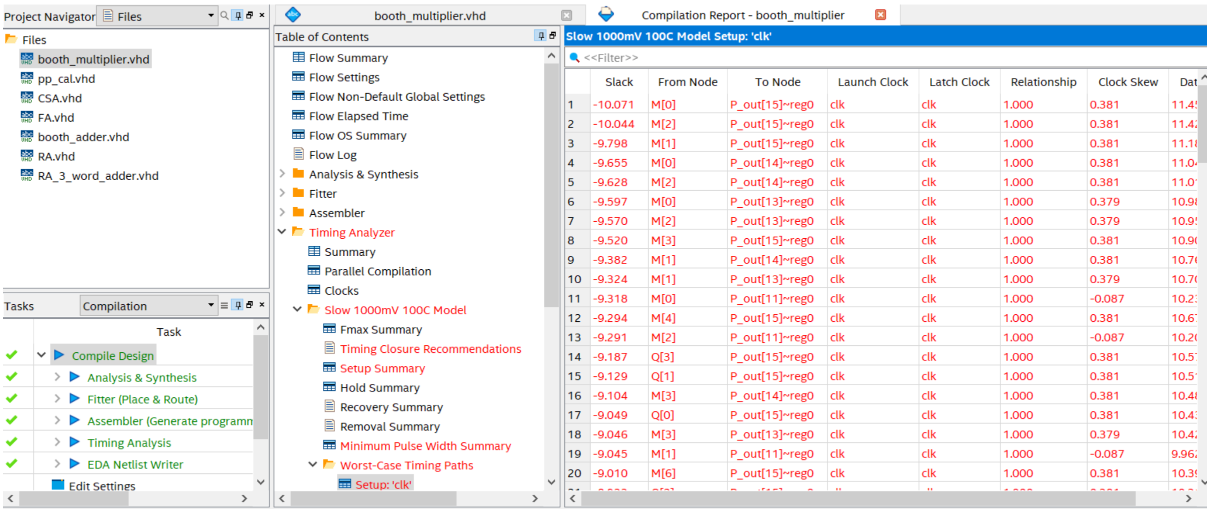
Simulation and Results

A VHDL code is written in Quartus and simulated in Modelsim. Following figure is taken from Modelsim window showing result of multiplication of two 8 bits numbers.


Wave - Default
Search:
100 ps
Msgs
/test_booth_multipli... 101
/test_booth_multipli... 1
/test_booth_multipli... 101
109
97
127
0
101
44
42
109
1
4796
4074
13843
0
101

Comparison with Array Multiplier using Timing Analysis

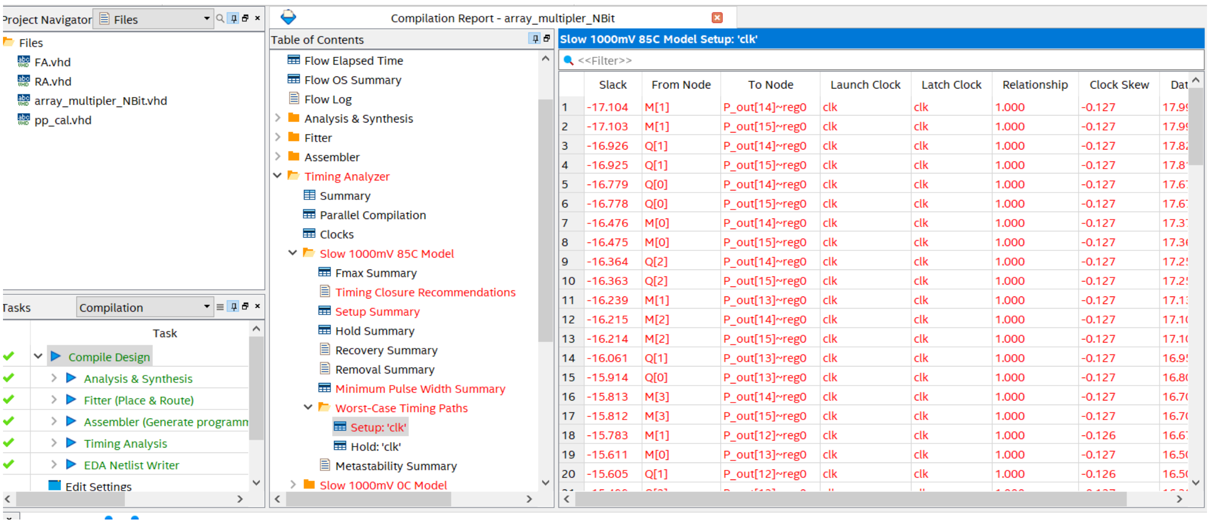
Timing analysis for 8-bit Booth multiplier and 8-bit array multiplier is done in Quartus. The result for worst case timing path for both the multipliers is shown in following figures.



Slow 1000mV 100C Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Dat
1	-10.071	M[0]	P_out[15]~reg0	clk	clk	1.000	0.381	11.4
2	-10.044	M[2]	P_out[15]~reg0	clk	clk	1.000	0.381	11.4
3	-9.798	M[1]	P_out[15]~reg0	clk	clk	1.000	0.381	11.1
4	-9.655	M[0]	P_out[14]~reg0	clk	clk	1.000	0.381	11.0
5	-9.628	M[2]	P_out[14]~reg0	clk	clk	1.000	0.381	11.0
6	-9.597	M[0]	P_out[13]~reg0	clk	clk	1.000	0.379	10.9
7	-9.570	M[2]	P_out[13]~reg0	clk	clk	1.000	0.379	10.9
8	-9.520	M[3]	P_out[15]~reg0	clk	clk	1.000	0.381	10.9
9	-9.382	M[1]	P_out[14]~reg0	clk	clk	1.000	0.381	10.7
10	-9.324	M[1]	P_out[13]~reg0	clk	clk	1.000	0.379	10.7
11	-9.318	M[0]	P_out[11]~reg0	clk	clk	1.000	-0.087	10.2
12	-9.294	M[4]	P_out[15]~reg0	clk	clk	1.000	0.381	10.6
13	-9.291	M[2]	P_out[11]~reg0	clk	clk	1.000	-0.087	10.2
14	-9.187	Q[3]	P_out[15]~reg0	clk	clk	1.000	0.381	10.5
15	-9.129	Q[1]	P_out[15]~reg0	clk	clk	1.000	0.381	10.5
16	-9.104	M[3]	P_out[14]~reg0	clk	clk	1.000	0.381	10.4
17	-9.049	Q[0]	P_out[15]~reg0	clk	clk	1.000	0.381	10.4
18	-9.046	M[3]	P_out[13]~reg0	clk	clk	1.000	0.379	10.4
19	-9.045	M[1]	P_out[11]~reg0	clk	clk	1.000	-0.087	9.96
20	-9.010	M[6]	P_out[15]~reg0	clk	clk	1.000	0.381	10.3

Timing Analysis for Booth Multiplier



Slow 1000mV 85C Model Setup: 'clk'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Dat
1	-17.104	M[1]	P_out[14]~reg0	clk	clk	1.000	-0.127	17.9
2	-17.103	M[1]	P_out[15]~reg0	clk	clk	1.000	-0.127	17.9
3	-16.926	Q[1]	P_out[14]~reg0	clk	clk	1.000	-0.127	17.8
4	-16.925	Q[1]	P_out[15]~reg0	clk	clk	1.000	-0.127	17.8
5	-16.779	Q[0]	P_out[14]~reg0	clk	clk	1.000	-0.127	17.6
6	-16.778	Q[0]	P_out[15]~reg0	clk	clk	1.000	-0.127	17.6
7	-16.476	M[0]	P_out[14]~reg0	clk	clk	1.000	-0.127	17.3
8	-16.475	M[0]	P_out[15]~reg0	clk	clk	1.000	-0.127	17.3
9	-16.364	Q[2]	P_out[14]~reg0	clk	clk	1.000	-0.127	17.2
10	-16.363	Q[2]	P_out[15]~reg0	clk	clk	1.000	-0.127	17.2
11	-16.239	M[1]	P_out[13]~reg0	clk	clk	1.000	-0.127	17.1
12	-16.215	M[2]	P_out[14]~reg0	clk	clk	1.000	-0.127	17.1
13	-16.214	M[2]	P_out[15]~reg0	clk	clk	1.000	-0.127	17.1
14	-16.061	Q[1]	P_out[13]~reg0	clk	clk	1.000	-0.127	16.9
15	-15.914	Q[0]	P_out[13]~reg0	clk	clk	1.000	-0.127	16.8
16	-15.813	M[3]	P_out[14]~reg0	clk	clk	1.000	-0.127	16.7
17	-15.812	M[3]	P_out[15]~reg0	clk	clk	1.000	-0.127	16.7
18	-15.783	M[1]	P_out[12]~reg0	clk	clk	1.000	-0.126	16.6
19	-15.611	M[0]	P_out[13]~reg0	clk	clk	1.000	-0.127	16.5
20	-15.605	Q[1]	P_out[12]~reg0	clk	clk	1.000	-0.126	16.5

Timing Analysis for Array Multiplier

Conclusion

Design for Booth multiplication algorithm with Carry Save Adders is implemented in VHDL. Various test cases are given to check the proper functioning of implemented algorithm. The speed of operation of Booth multiplier is shown to be greater than array multiplier.

References

Digital VLSI lectures by Prof. Dinesh Sharma, IIT-bombay