DADA Multiplier

Design a Dadda multiplier for unsigned 16x16 bit multiplication with a Brent Kung adder for the final addition in synthesizable VHDL.

1. Assuming that library contains components with delays as given below:
    Inverter 100 ps
    NAND gate 150 ps
    NOR gate 150 ps
    A+B.C 200 ps
    Tiny XOR 200 ps
    Half Adder (carry) 250 ps
    Half Adder (sum) 200 ps
    Full Adder (carry) 400 ps
    Full Adder (sum) 400 ps

Designing a Dadda multiplier for unsigned 16x16 bit multiplication with a Brent Kung adder for the final addition.

Dadda multipliers uses the three stages:

1. Generate all bits of the partial products in parallel.
2. Collect all partial products bits with the same place value in bunches of wires and reduce these in several layers of adders till each weight has no more than two wires.
3. For all bit positions which have two wires, take one wire at corresponding place values to form one number, and the other wire to form another number. Add these two numbers using a fast adder of appropriate size.

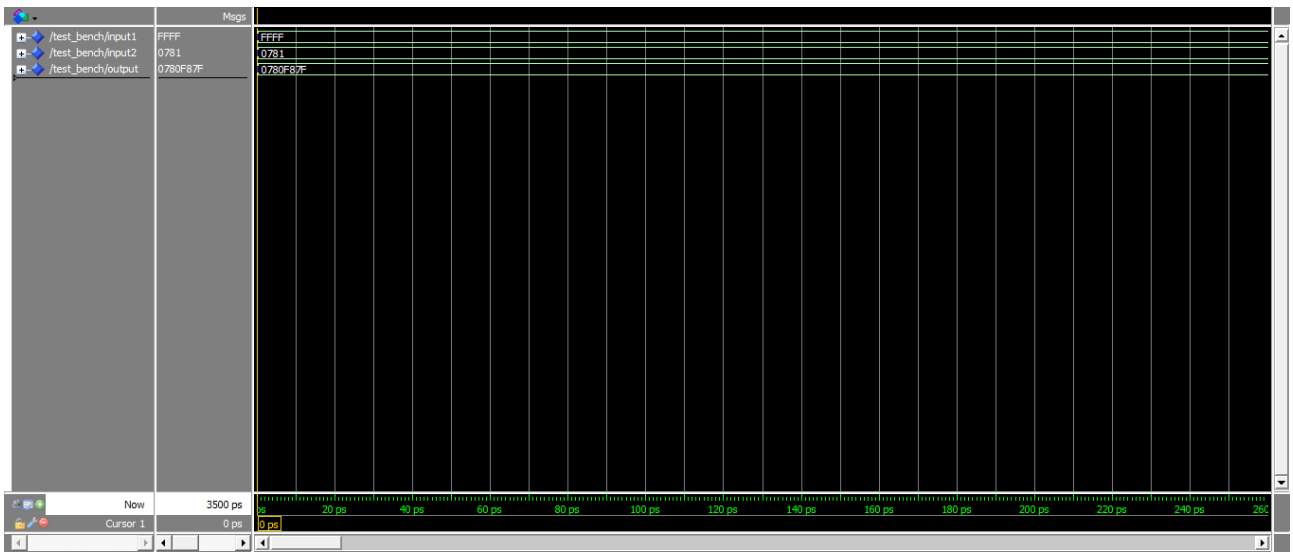
Below shown is the scheme for reduction. Here, F represents full adder and H represents, Half Adder.
F-2 : Represents two full adders to reduce six wires from present stage to 2 (Sum) in present stage and 2 (carry) in next stage.
H-1: Represents one half adders to reduce 2 wires from present stage to 1 (sum) in present stage and 1 (carry) in next stage.

Table: Reduction Scheme

Bit Number	Number of Wires (16)			Reduction to 11			Reduction to 8			Redution to 6			Reduction to 4			Reduction to 3			Reduction to 2	
31	0			0			0			0			0			0			0	
30	1			1			1			1			1			1			2	
29	2			2			2			2			2			3	F-1		2	
28	3			3			3			3			4	F-1		3	F-1		2	
27	4			4			4			4	F-1		4	F-1		3	F-1		2	
26	5			5			5			6	F-2		4	F-1		3	F-1		2	
25	6			6			6	F-1		6	F-2		4	F-1		3	F-1		2	
24	7			7			8	F-2		6	F-2		4	F-1		3	F-1		2	
23	8			8	F-1		8	F-2		6	F-2		4	F-1		3	F-1		2	
22	9			9	F-2		8	F-2		6	F-2		4	F-1		3	F-1		2	
21	10			11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
20	11	F-1		11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
19	12	F-2		11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
18	13	F-3		11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
17	14	F-4		11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
16	15	F-4	H-1	11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
15	16	F-4	H-1	11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
14	15	F-3	H-1	11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
13	14	F-2	H-1	11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
12	13	F-1	H-1	11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
11	12		H-1	11	F-3		8	F-2		6	F-2		4	F-1		3	F-1		2	
10	11			11	F-2	H-1	8	F-2		6	F-2		4	F-1		3	F-1		2	
9	10			10	F-1	H-1	8	F-2		6	F-2		4	F-1		3	F-1		2	
8	9			9		H-1	8	F-2		6	F-2		4	F-1		3	F-1		2	
7	8			8			8	F-1	H-1	6	F-2		4	F-1		3	F-1		2	
6	7			7			7		H-1	6	F-2		4	F-1		3	F-1		2	
5	6			6			6			6	F-1	H-1	4	F-1		3	F-1		2	
4	5			5			5			5		H-1	4	F-1		3	F-1		2	
3	4			4			4			4			4		H-1	3	F-1		2	
2	3			3			3			3			3			3		H-1	2	
1	2			2			2			2			2			2			2	
0	1			1			1			1			1			1			1	

31 bit Adder

Test Case:



Here,
Multiplier is, m= 0xFFFF
Multiplicant is, n = 0x0781
So, the product is, 0x0780F87F

2. Considering the delay as mentioned above, back annotating the delays for costituent components and re-simulating the circuit.

Test Case:

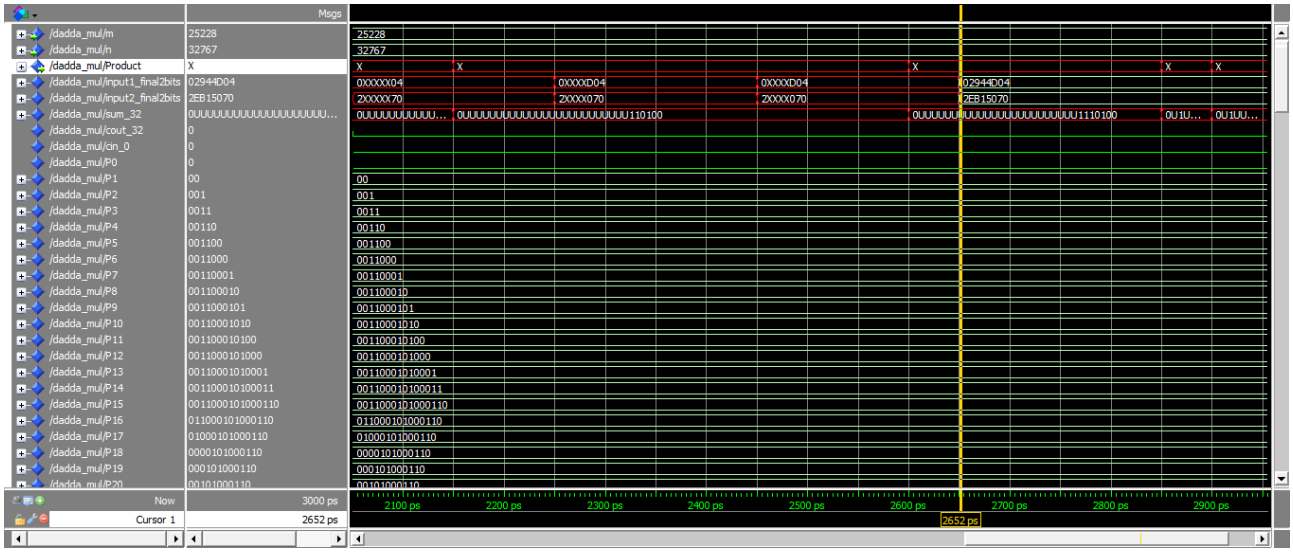
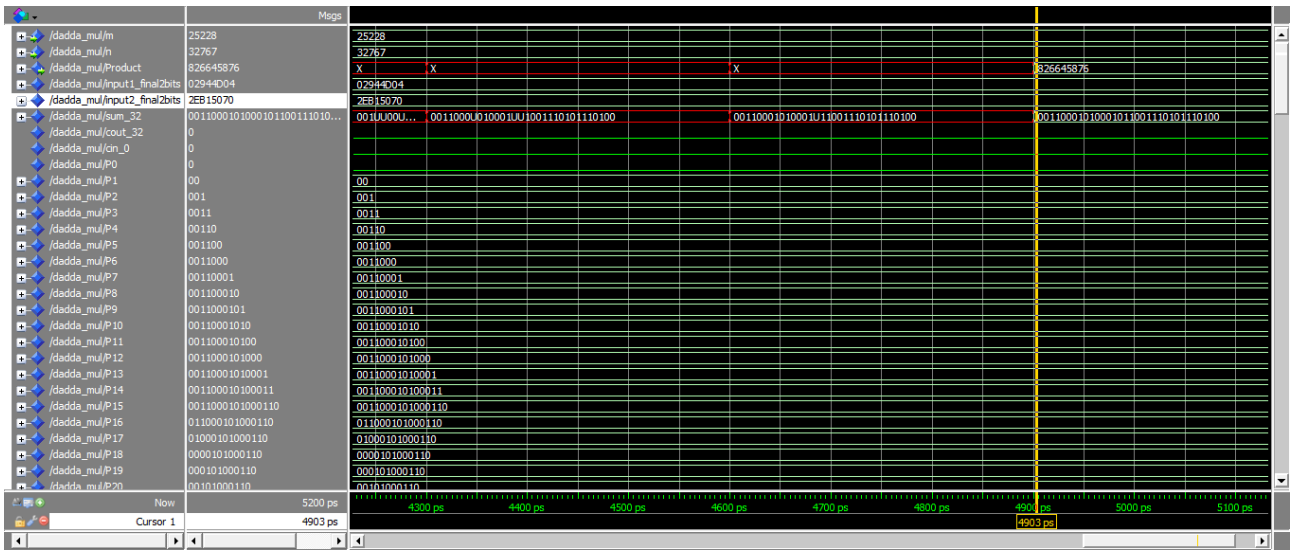


Figure: Simulation shows that the reduction to 2 wires from 16 wires is done in constant time of 2650 pico-seconds.

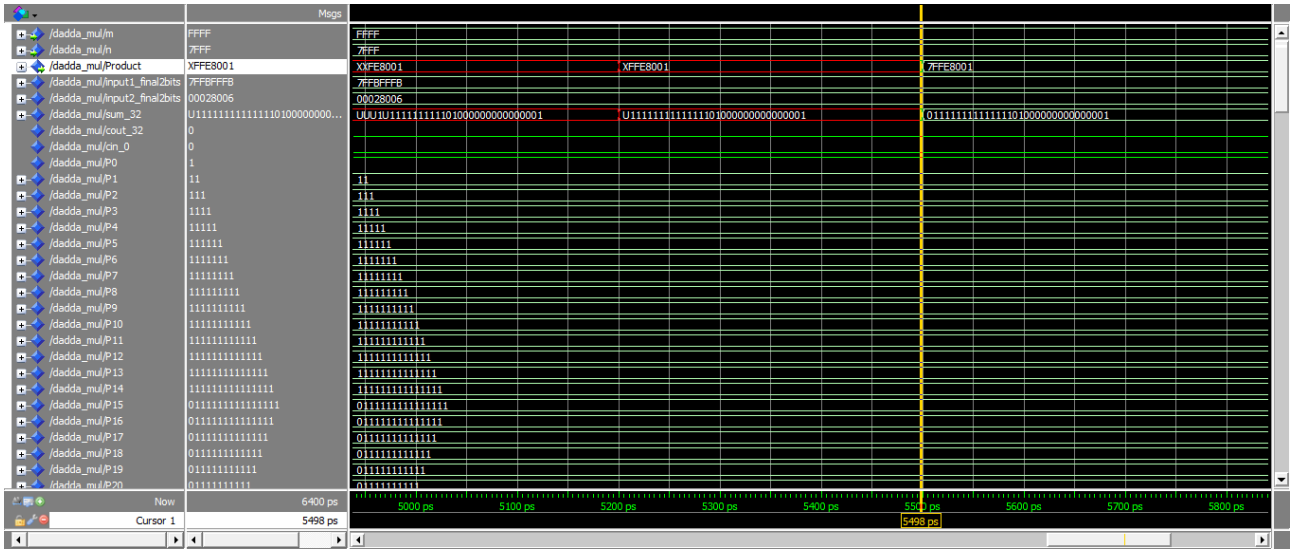


After the reduction, the Brent Kung adder takes additional time (2250 pico-sec, for this test case) to complete the multiplication operation in 4900 pico-sec.

Worst Case Test:

2850pico seconds is the worst case for the 31 bit Brent kung Adder (Actually using 32 bits),
So, the total delay would be, 2650 pico-seconds for reduction to 2 wire stage and 2850pico seconds for Brent Kung.

So, one of the possible test case would be:



Delay: 5500 pico seconds. That is by this much time one can guarantee that the multiplication will be completed.

Theoretical Critical Path:
Considering a bit travelling in yellow boxes shown in table above,

Delay = T_And_gate_(NAND+NOT) + 6 *(T_Full_Adder_Carry) + T_Brent_Kung_Worst_Case
= (150+100) + 6* (400) + 2850
= 2650+ 2850
= 5500 pico- seconds