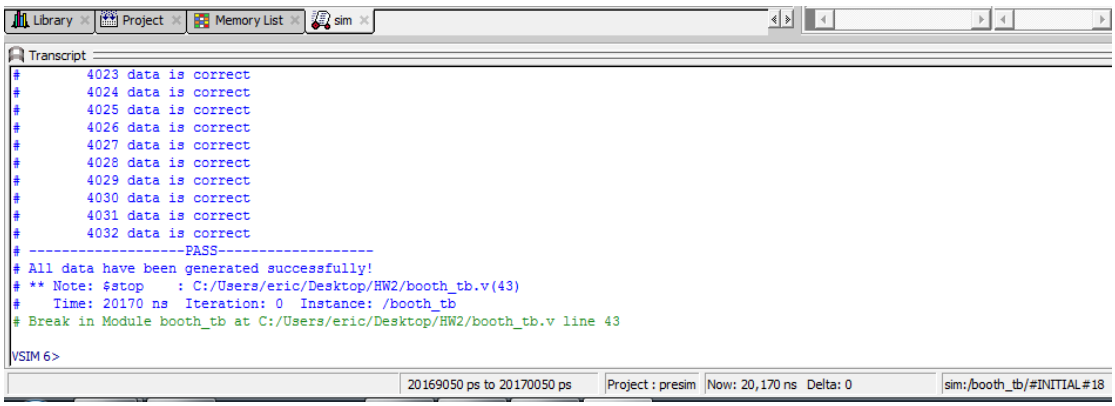
Functional Simulation:

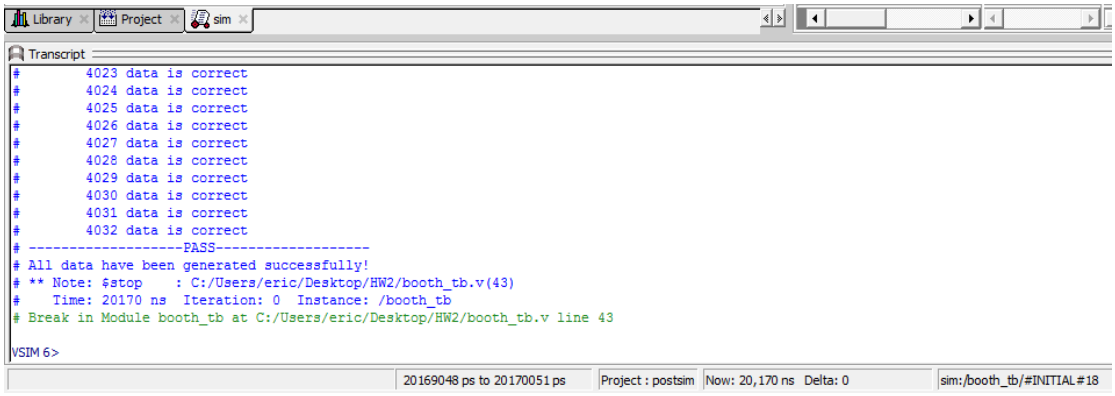


Gate-Level Simulation:

Synthesis

Flow Summary	
Flow Status	Successful - Sun Nov 05 14:07:19 2017
Quartus Prime Version	16.0.2 Build 222 07/20/2016 SJ Standard Edition
Revision Name	booth
Top-level Entity Name	booth
Family	Cyclone V
Device	5CGTFD9E5F35C7
Timing Models	Final
Logic utilization (in ALMs)	60 / 113,560 (< 1 %)
Total registers	0
Total pins	24 / 616 (4 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

Simulation



宣告 input in1(multiplicand)、int2(multiplier)及 output (最終 product)，再宣告一個 reg [12:0] P[0:12]的陣列，陣列大小為 13 格，每格存放 13bit P[0]存著一開始初始化的 product 值，後面用來儲存每一次運算的結果，初始值為 0(6 bits)_in2(6 bits)_0(1 bit)

用 for 迴圈執行每次 P 末兩 bit 的 case 判斷，並做出相對應的運算，將存在 P[0]的值依 csae 判斷做運算完後把結果存在 P[1]，向右位移一格再存到 P[2]，即算完成一輪，以此類推重複上述過程 6 次，最後算完的結果會存在 P[12]，再把 P[12]的值只截取前面 12 個 bit 給 output 即完成 Booth Algorithm。

```
1  `timescale 1ns / 10ps
2  module booth(out, in1, in2);
3
4  parameter width = 6;
5
6  input   [width-1:0] in1;   //multiplicand
7  input   [width-1:0] in2;   //multiplier
8  output  [2*width-1:0] out; //product
9
10 reg [2*width:0]P[0:2*width];
11 reg [2*width-1:0] out; //product
12
13 integer i ;
14
15 always@ (in1 or in2)
16 begin
17
18 P[0] = {6'b0,in2,1'b0};
19
20     for (i= 0; i<width; i=i+1)
21     begin
22         case({P[2*i][1:0]})
23             2'b01: P[2*i+1] = {P[2*i][12:7]+in1,P[2*i][6:0]};
24             2'b10: P[2*i+1] = {P[2*i][12:7]-in1,P[2*i][6:0]};
25             default: P[2*i+1] = P[2*i];
26         endcase
27         P[2*i+2] ={P[2*i+1][2*width], P[2*i+1][2*width:1]};
28     end
29
30 out = P[2*width][2*width:1];
31 end
32
33 endmodule
34
```