2017 Digital IC Design
Homework 2: Booth Algorithm

1. Introduction

Booth algorithm is a multiplication operation that multiplies two numbers in two's complement notation. The detail algorithm is described as below:

(1) Assume that the multiplicand m is *x*-bit and multiplier *r* is *y*-bit.
Initialize a register *P* for the final result, and the length is *x*+*y*+1 bits.
The initial value of P is 0(*x* bits)_*r*(*y* bits)_0(1 bit).

(2) The rightmost 2 bits used for the selection of different executions.

LSB	Execution
00	No execution
01	Add *m* to the left part of *P*
10	Subtract *m* from the left part of *P*
11	No execution

* all of the overflow can be ignored during the execution

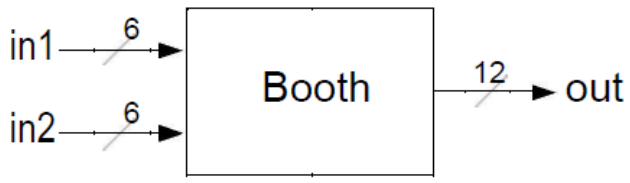
(3) **Arithmetically** shift 1 bit on P.

(4) Repeat the step (2) and (3) for *y* times.

(5) The final answer is obtained by dropping the LSB from P.

2. Design Specifications

2.1 Block Overview



2.2 I/O Interface

Signal Name	I/O	width	Description
in1	I	6	Multiplicand
in2	I	6	Multiplier
out	O	12	Product

2.3 File Description

File Name	Description
booth.v	The top module of the design.
booth_tb.v	The testbench file. You are **not allowed** to modify the content in this file.

3. Scoring

3.1 Functional Simulation (pre-sim) **[70%]**

All of the result should be generated correctly, and you will get the following message in ModleSim simulation.

```
#        4020 data is correct
#        4021 data is correct
#        4022 data is correct
#        4023 data is correct
#        4024 data is correct
#        4025 data is correct
#        4026 data is correct
#        4027 data is correct
#        4028 data is correct
#        4029 data is correct
#        4030 data is correct
#        4031 data is correct
#        4032 data is correct
# --------------------PASS--------------------
# All data have been generated successfully!
# ** Note: $stop    : E:/2016DICHW/HW2/booth_tb.v(43)
#    Time: 201700 ns  Iteration: 0  Instance: /booth_tb
# Break in Module booth_tb at E:/2016DICHW/HW2/booth_tb.v line 43
```

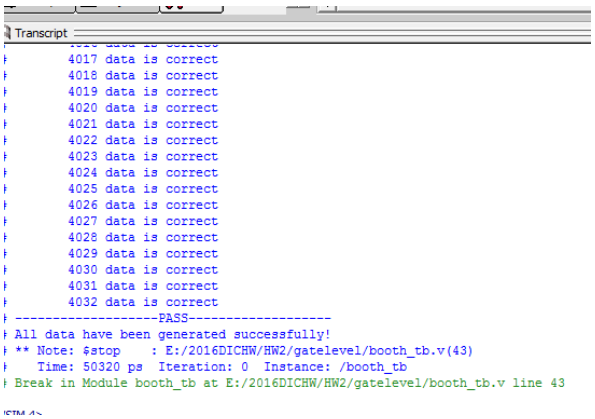
3.2 Gate-Level Simulation (post-sim) **[30%]**

3.2.1 Synthesis

Your code should be synthesizable. After synthesizing in Quartus, a file named *booth.vo* will be obtained.

3.2.2 Simulation

All of the result should be generated correctly using *booth.vo*, and you will get the following message in ModleSim simulation.



4. Submission

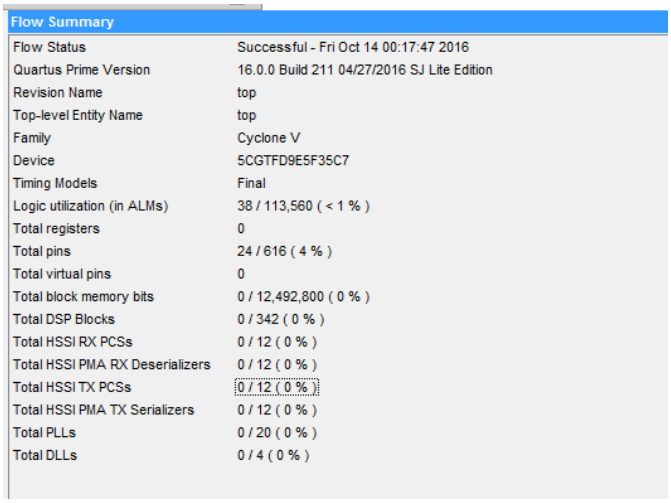
4.1 Submitted files

You should classified your files into three directories and compressed to .zip format. The naming rule is **HW2_*studentID_name_version*.zip**. The *vision* is v1 for the first submission, and v2, v3… for the revisions.

RTL category	
*.v	All of your verilog RTL code
Gate-Level category	
*.vo	Gate-Level netlist generated by Quartus
Documentary category	
*.pdf	The report file of your design (in pdf).

4.2 Report file

You have to describe how the circuit is designed as detailed as possible, and the flow summary result after synthesis is necessary.



Flow Summary	
Flow Status	Successful - Fri Oct 14 00:17:47 2016
Quartus Prime Version	16.0.0 Build 211 04/27/2016 SJ Lite Edition
Revision Name	top
Top-level Entity Name	top
Family	Cyclone V
Device	5CGTFD9E5F35C7
Timing Models	Final
Logic utilization (in ALMs)	38 / 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 %)

4.3 Please submit your **.zip** file to folder HW2 in the ftp site.

Deadline: 2017-11-06 23:59

ftp : 140.116.245.92

Usermame : ic_design

Password : icdesign

5. If you have any problem, please contact the TA by email : p78031175@mail.ncku.edu.tw