EE577A Fall 2019 – VLSI System Design

Instructor: Prof. Massoud Pedram TA: Hongxiang Gao

Lab Assignment 1: 8-bit Signed Multiplier Design

Using Modified (Radix-4) Booth Coding

Notes:

- Due on 10/09/2019 12:00 pm (For student who didn't take Prof Pedram's EE477L, you can cooperate with someone who also has this issue or you can have a one-week extension (10/16/2019 12:00 pm).)
- This assignment is an individual work, no collaboration is allowed. (Unless you have the issue listed above.)
- The cells you designed in this lab are going to be used in your final project
- Your lab report must be a pdf file, which include all the materials of the corresponding parts. Other file formats will not be graded.
- Submit your final report on the Blackboard website.
- Ask questions only on blackboard discussion forum or in the office hours. DO NOT send email asking technical questions.

Why is the size of output 16 bit ?
$2^8 \times 2^8 = 2^{16}$

Introduction

2's complement (보수)

Design an 8-bit signed 2's complement multiplier (the inputs are two signed 8-bit 2's complement numbers and the output will be a 16-bit signed 2's complement number) in both schematics and layout. You need to apply Modified (Radix-4) Booth Coding Algorithm. Figure 1 shows the algorithm. You will need to apply this algorithm in hardware level and build an 8-bit signed 2's complement multiplier.

Radix-4 Booth Coding

- $Y = -y_7 2^7 + \sum_{i=0}^{6} y_i 2^i$
- $Y = -y_7 2^7 + y_6 2^6 + y_5 2^5 + y_4 2^4 + y_3 2^3 + y_2 2^2 + y_1 2^1 + y_0 2^0$
- $Y = (-2y_7 + y_6 + y_5)2^6 + (-2y_5 + y_4 + y_3)2^4 + (-2y_3 + y_2 + y_1)2^2 + (-2y_1 + y_0 + y_{-1})2^0$
- $Y = \sum_{i=0}^{3}(-2y_{2i+1} + y_{2i} + y_{2i-1})\, 4^i$
- Partial product:
- $P_i = (-2y_{2i+1} + y_{2i} + y_{2i-1})X$
- $Z = \sum_{i=0}^{3} P_i\, 4^i$
- Number of partial products halves!

Figure 1: Booth Multiplication Algorithm

A general realization of this algorithm is shown in Figure 2.

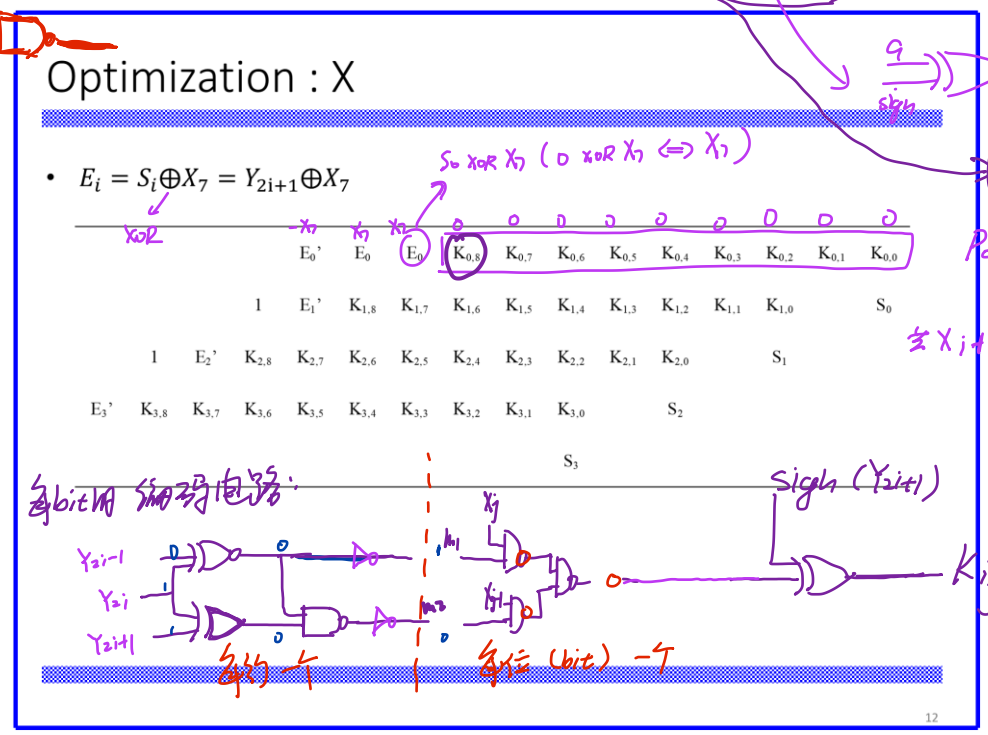
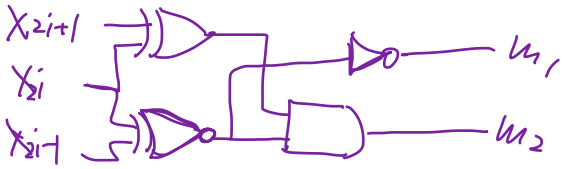


Figure 2: Booth Multiplication Realization

And all these K and S values can be derived based on the truth table below:

y_{2i+1}	y_{2i}	y_{2i-1}	Partial Product	$K_{i,j}$	S_i
0	0	0	0	0	0
0	0	1	1X	X_j	0
0	1	0	1X	X_j	0
0	1	1	2X	X_{j-1}	0
1	0	0	-2X	X'_{j-1}	1
1	0	1	-X	X'_j	1
1	1	0	-X	X'_j	1
1	1	1	-0	1	1



General Instructions

a) Use $\beta = \mu_n/\mu_p = 4$ for all gates in your design.

b) You can either use Full-Adder to function as a Half-Adder or you can re-design a Half-Adder.

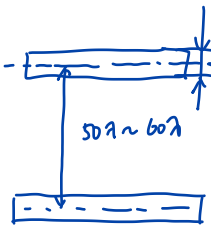
c) You are not required to sizing the transistors at this time. For simplicity, you can size all NMOS as 300 nm and all PMOS as 1.2 µm.

d) Use V_{DD} = 1.8V.

e) All inputs and outputs of the multipliers need to be registered with positive-edge triggered master-slave flip-flops.

f) Use rise/fall time = 10 ps.

g) You are not required to measure the worst cases delay, but the clock you set of FFs should be large enough to avoid the setup-time violation.



Layout Guidelines

a) The height of all your designs of one level should be 50-60 lambdas which is 5-6 µm in NCSU_TechLib_tsmc02 technology. The height of one level is measure from the middle of power rail to the middle of ground rail.

b) VDD and GND should be routed in metal 1 at top and bottom of the cell. The metal width of VDD and GND should be 10 lambdas (1 µm).

c) For the internal routing you can use poly (usually for short routing) or metal layer. (you may use any metal layer up to metal 4.) We recommend you use metal 1 for horizontal and metal 2 for vertical connections or vice versa. Try to use as few metal layers as possible therefore routing would be more convenient for your future assignments which may use this part.

d) You will have to try building your layout looks like a square shape. All the input signals should be given from the left-hand side and all the output signal should be connected at the right-hand side. For Figure 3, all input given from vector file are connected to the flip-flops and all outputs generated from the logic combination should also be connected to flip-flops.

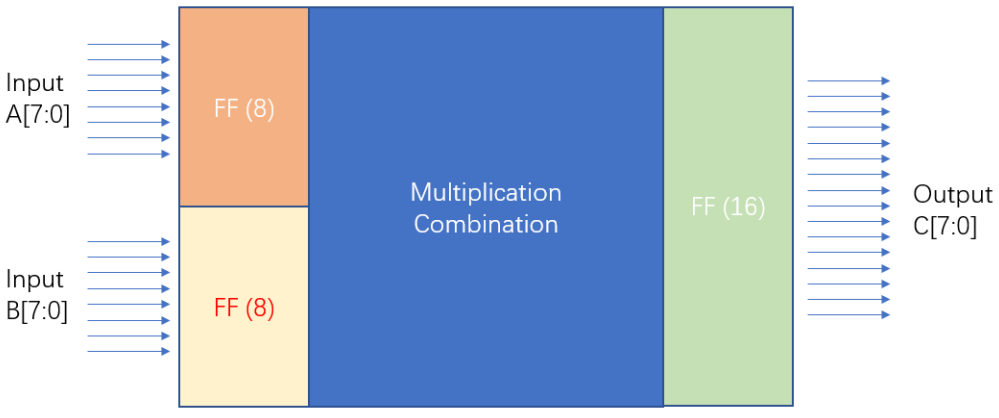


Figure 3: Block diagram of layout

Functionality Test

You will have to perform the functionality test using the input patterns generated by Python:

a) Design the Python script which can randomly generates input pattern and its corresponding result in both decimal and binary format.

b) Fill in the table given below (a sample is given) after running your script. (You will have to generate other four different test cases)

Op1 (dec)	Op2 (dec)	Result (dec)	Op1 (bin)	Op2 (bin)	Result (bin)
-20	6	-120	11101100	00000110	11111110001000

c) Apply the patterns in Cadence and compare the waveforms with the result generated above. You can either use vector file or directly set the input patterns in Cadence while vector file is preferred since in future lab assignments the input patterns will be much more complicated.

Report Checklist

a) Schematics and layout of your 8-bit signed 2's complement multiplier.

b) LVS match report of your 8-bit signed 2's complement multiplier.

c) Completed Python script and table of test patterns generated by your script. (It would be appreciated if you can add some comments in your script.)

d) Functionality test waveforms of both schematics and layout of the 8-bit 2's complement multiplier.

e) A summary of key parameters of your design, including the minimum clock, height, width and area of your layout. Also, you will have to measure the power consumption of your design. For example, as you run five test patterns in a roll, you can get the average power consumption of it.

Submission Guidelines

a) Briefly explain your work.

b) Name your report file in following format:
"firstname_lastname_studentID_Lab1_EE577A_Fall19.pdf", for example:
"Hongxiang_Gao_1111111111_Lab1_EE577A_Fall19.pdf"

c) For students who need cooperation permission or deadline extension, you will have to inform the grader or TA before you submit the report. You can either send an email or walk into the office hour.