Digital System Design with FPGA Assignment 2 : 8 bit Radix 4 Booth Recoded Array Multiplier

Guhan Rajasekar, Mtech ESE , 22410 , guhanr@iisc.ac.in, DESE, IISc

I Introduction

- This report is a summary of the design of 8 bit radix 4 booth recoded array multiplier.

II RTL Block

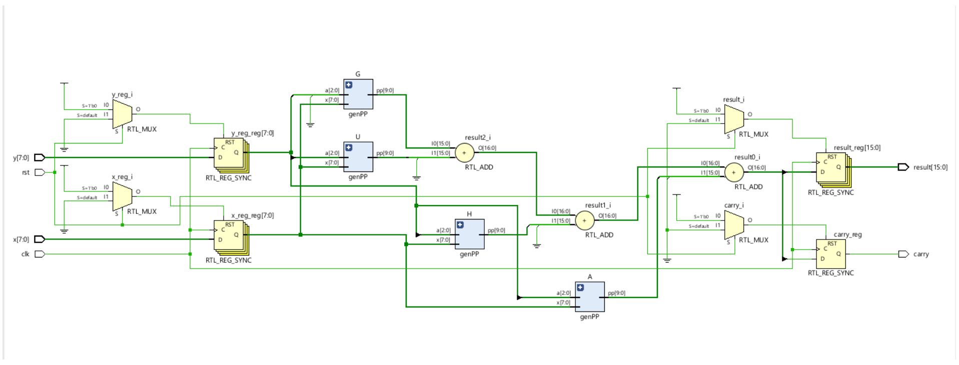
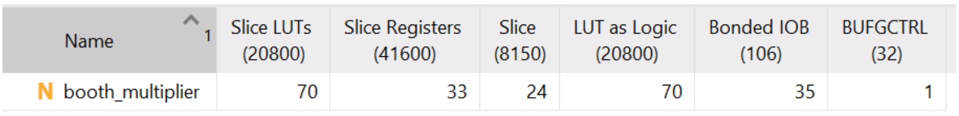


Figure 1: RTL Block of Booth Multiplier

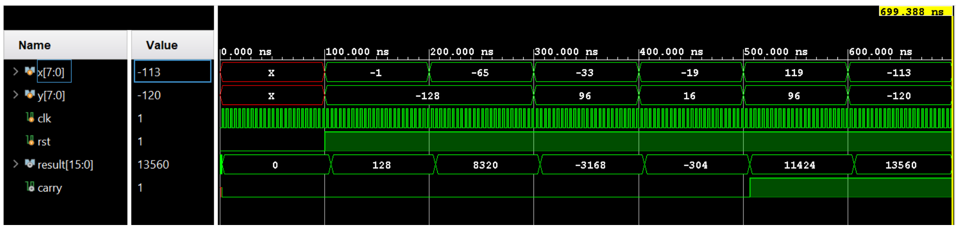
III Resource Utilization



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)	BUFGCTRL (32)
booth_multiplier	70	33	24	70	35	1

Figure 2: Resource Utilization of Booth Multiplier

IV Functional Simulation



Name	Value
x[7:0]	-113
y[7:0]	-120
clk	1
rst	1
result[15:0]	13560
carry	1

Figure 3: Functional Simulation of Booth Multiplier

- The above waveform shows the simulation results that were obtained before implementation.

V Post Route Timing Simulation

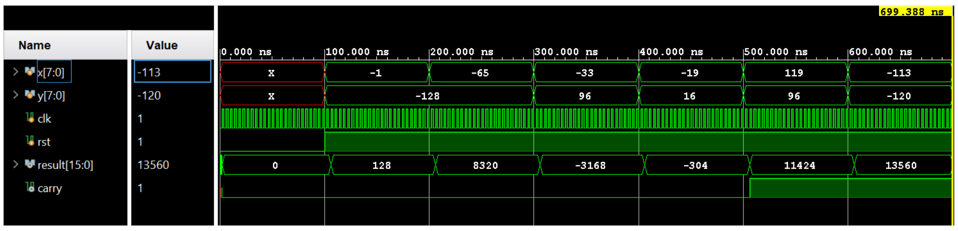
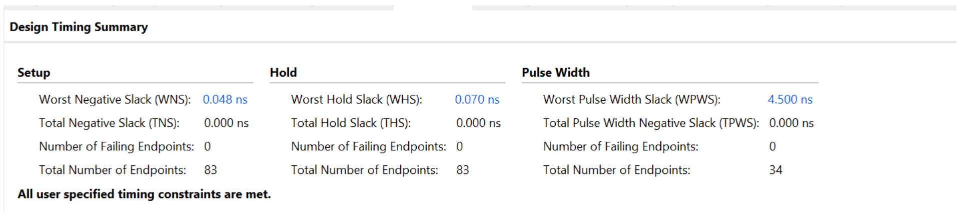


Figure 4: Post Route Timing Simulation of Booth Multiplier

- The above waveform shows the simulation results that were obtained after implementation.

VI Timing Report



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.048 ns	Worst Hold Slack (WHS):	0.070 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	83	Total Number of Endpoints:	83	Total Number of Endpoints:	34

All user specified timing constraints are met.

Figure 5: Timing Report Of Booth Multiplier

- The above picture shows the timing report that was obtained post implementation.
- Here since no slack is negative, we see that all the timing constraints have been met.
- Least Time period that can be used for operation is given by:

$$T = Original\ time\ period - Worst\ Negative\ Setup\ Slack\ = 10\ ns - 0.048\ ns = 9.952\ ns. \tag{1}$$

- Max operating frequency is given by:

$$f_{max} = \frac{1}{T} = \frac{1}{9.952ns} = 100.482Mhz \tag{2}$$

VII DESIGN OF BOOTH MULTIPLIER

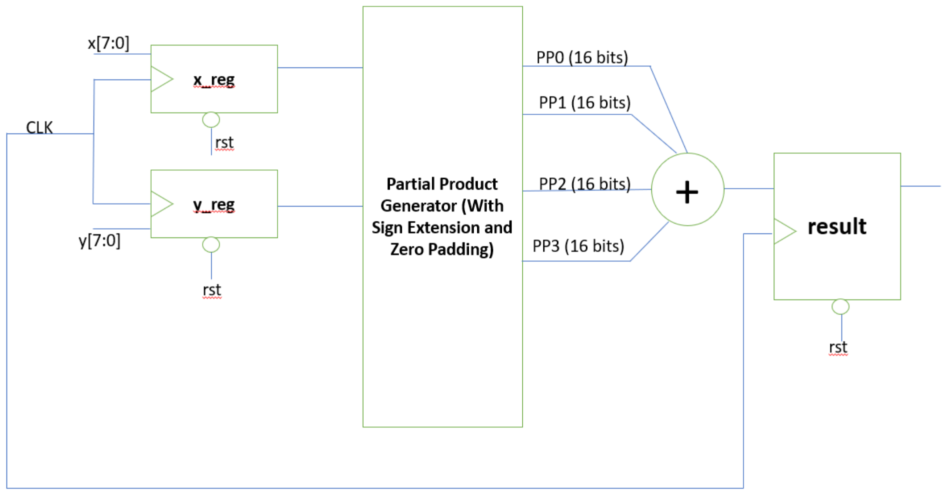


Figure 6: Booth Multiplier Block Diagram

- Here the inputs are wires that are continuously driven by the input data x and y.
- Signal from these wires are loaded in registers *x_reg* and *y_reg* at the positive edge of clock.
- Then inside the partial product generator, we first construct 10 bit partial products for which sign extension and zero padding is done later on.
- The reason for choosing 10 bits is to reliably produce **2x** and **-2x** when multiplicand **-128**.
- The module **genPP** present in the code is used to generate 10 bit partial products. This module is instantiated 4 times to generate 4 partial products.
- Once the 10 bit partial products are generated, we then perform sign extension at MSB and zero padding at LSB as required to create 16 bit partial products.
- Once the 16 bit partial products are created, we add them using the **+** operator, which implements a **Carry Propagate Adder** in verilog.
- Final result is loaded in an output register at the positive edge of clock.