University of Tehran
College of Engineering
School of Electrical & Computer Engineering

Experiment 2
Sessions 4,5

FPGA Realization of Radix-4 Multiplier

Digital Logic Laboratory
ECE 045
Laboratory Manual

Spring 1401

Contents

Figure 1: Block Diagram of ASSUMER4 multiplier unit

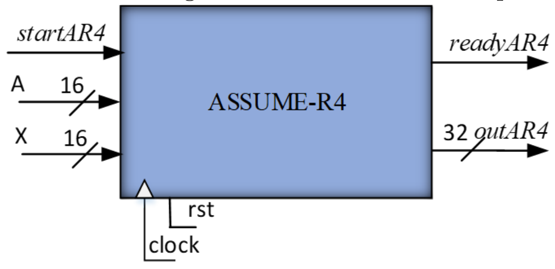


Table 1: Radix-4 Booth Encoding Table

Block	Partial Product
000	0
001	1*multiplicand
010	1*multiplicand
011	2*multiplicand
100	-2*multiplicand
101	-1*multiplicand
110	-1*multiplicand
111	0

Introduction

Multiplication is arguably the most important primitive for machine learning and digital signal processing applications.The majority of hardware optimisations for machine learning are focused on reducing the cost of the multiply and accumulate (MAC) operations. Hence, careful construction of the compute unit, with a focus on multiplication, leads to the largest performance impact.

The goal of this experiment is to design a multiplier faster than conventional Booth multipliers that can be used in Neural Networks or DSP applications.

By the end of this experiment, you should have learned:

- Hardware design
- Synthesis
- FPGA Implementation

In this experiment you are to design an n-bit sequential Radix-4 signed and unsigned multiplier. We refer to this engine as ASSUME-R4 (A Sizable Signed or Unsigned MultipliEr – Radix4), which is pronounced as "Assume R4". Figure 1 shows the block diagram of this multiplier module. The ASSUME-R4 circuit starts a new multiplication calculation when a positive pulse is issued on its startAR4 input. At this time, ASSUME-R4 will start the calculation, de-asserts its readyAR4 output, and when completed, it will assert its readyAR4 output. While readyAR4 is active, outAR4 holds the calculation result. Table 2 shows an example of Radix-4 multiplication. The algorithm of this multiplication is as described below:

Table 2: Example for Radix-4

A			01	00	01			17
X	*		11	01	11			-9
Y			$0\bar{1}$	10	$0\bar{1}$			recoded multiplier
			-A	+2A	-A			operation
Add -A	+		10	11	11			
2-bit shift		1	11	10	11	11		
Add 2A	+	0	10	00	10			
			01	11	01	11		
2-bit shift			00	01	11	01	11	
Add -A	+		10	11	11			
			11	01	10	01	11	-153

- Append a 0 to the right of the LSB of the multiplier.
- According to the value of each vector, each Partial Product will be 0, +y, -y, +2y or -2y.
- Radix-4 booth encoder performs the process of encoding the multiplicand based on multiplier bits. It will compare 3 bits at a time. Grouping starts from the LSB, and the first block only uses two bits of the multiplier and assumes a zero for the third bit. The functional operation of Radix-4 booth encoder is shown in the Table 1.

1 RTL Design and Simulation

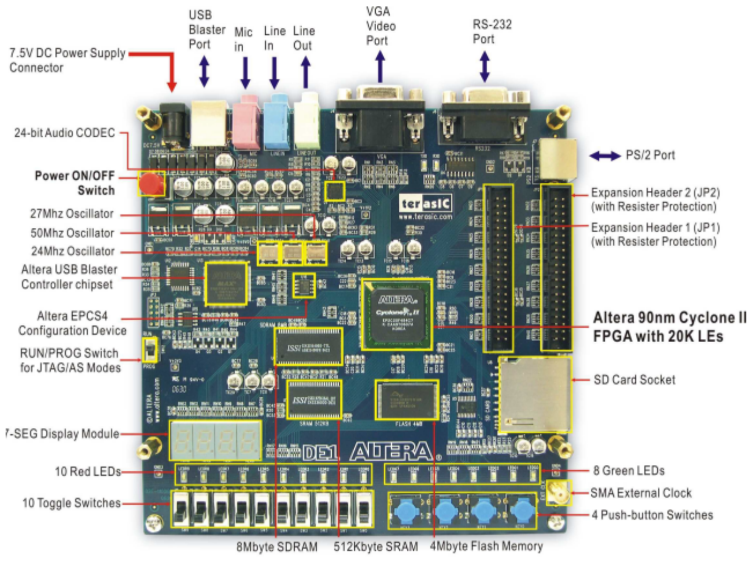
In this part you will start with RTL design of the Radix-4 multiplier. The RTL design includes a datapath and controller.

1. Show the datapath of the design.
2. Show the state diagram of the controller. Use the Huffman style coding for the controller.
3. Write the Verilog description of the design including both datapath and controller.
4. Write a testbench for the calculation unit in Verilog that tests the correctness of your circuit and show the results in Modelsim. Test your design for both signed and unsigned values of A and X.
5. In your report include and compare the simulation and theoretical results.

2 FPGA Implementation

In this part, you will be familiar with FPGA implementation. In this part you will use Intel Quatus Synthesis tool for mapping your designs on Altera CycloneII FPGA devices. The FPGA device that you will use in this and all other experiments until the end of the semester is *Cyclone II* device. Figure 2 shows the layout of the board.

Figure 2: Altera cyclone II board



In this section you will implement the Radix-4 multiplier on the FPGA board. For this purpose you need to provide the design with input data. There are 10 switches and 4 push buttons that can be used as inputs for your design. Since the bit width of multiplier and the multiplicand are 16 bits and there are only 10 switches, there must be an interfacing for receiving and transmitting data.

1. You will use eight switches for A and X inputs.For each of the inputs, you need two store the least and the most 8-bits separately.For this purpose, use another input named GetA and assign it to a push button. The eight least and most significant bits will be recieved On every complete pulse of GetA. Do the same for input X, using GetX signal.

2. Modify the controller of previous section to include the interface for receiving the data.

3. Use one switch for the startAR4 signal.

4. Make a Quartus II project and synthesize the design.

5. Go to the RTL Viewer in Quartus II Tools Menu and include the synthesized RTL view in your report.

6. Complete the pin assignment and program the device.

7. For displaying the results you will use seven segments.For this purpose you need a Hex to BCD converter. This code will be provided to you.show the results on the four seven segments.

Acknowledgment

This lab manual was prepared and developed by Katayoon Basharkhah, PHD student of Digital Systems at University of Tehran, under the supervision of professor Zain Navabi.