Laboratory 5:

AN ENHANCED PROCESSOR

OBJECTIVES

- The purpose of this lab is to learn how to connect simple input (switches) and output devices (LEDs and 7-segment) to an FPGA chip and implement a circuit that uses these devices.
- Design an enhanced processor based on the simple processor in lab 4.

PREPARATION FOR LAB 5

- Finish Pre Lab 5 at home.
- Students have to simulate all the exercises in Pre Lab 5 at home. All results (codes, waveform, RTL viewer, …) have to be captured and submitted to instructors prior to the lab session. *If not, students will not participate in the lab and be considered absent this session.*

REFERENCE

1. Intel FPGA training

Laboratory 5:

AN ENHANCED PROCESSOR

EXPERIMENT 1

Objective: Design and implement a simple processor.

Requirement: Design and implement the processor which is shown in Figure 1. Then connect the enhanced processors to a memory and output register (Figure 2).

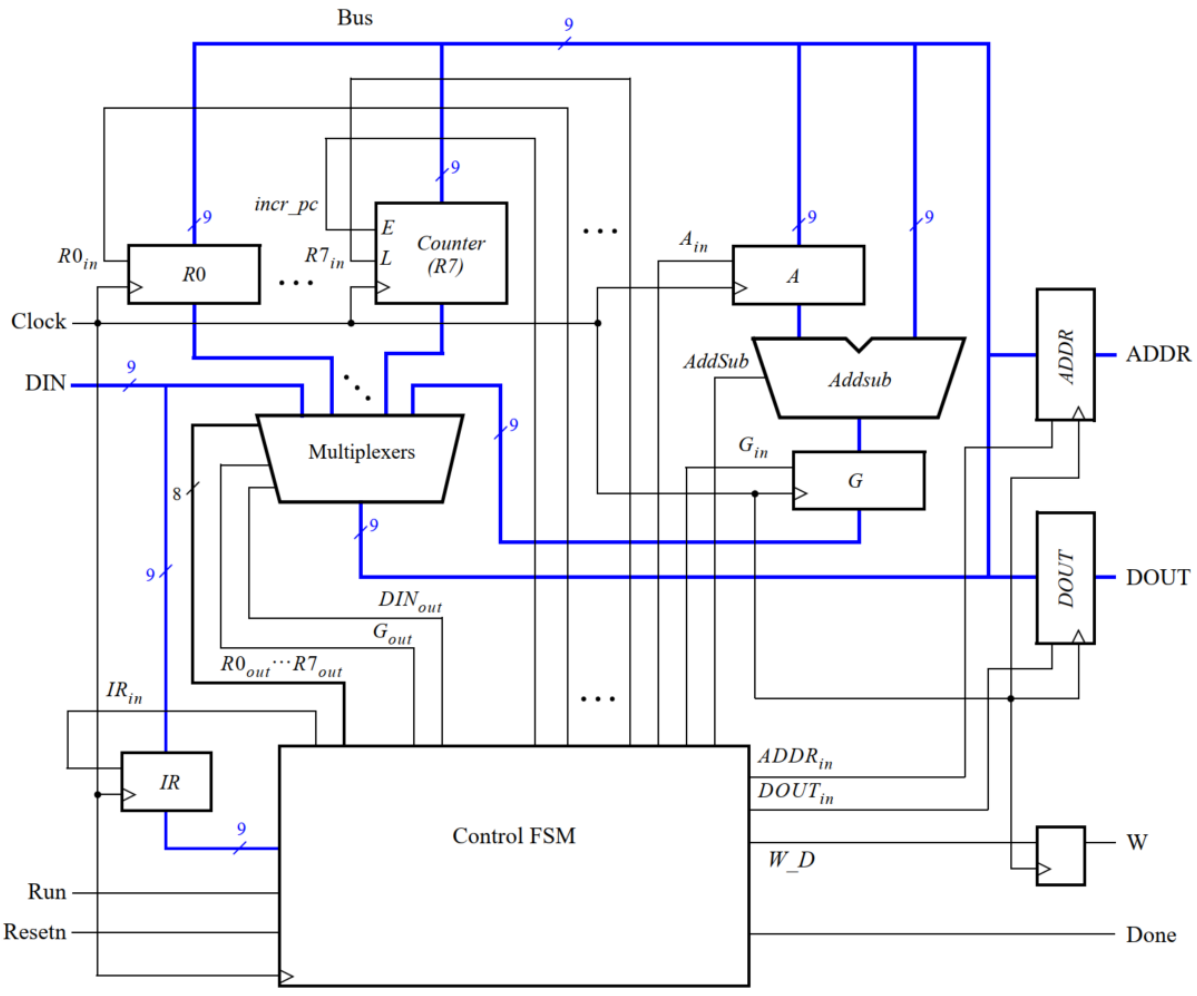

Figure 1: A reference schematic of the enhanced processor.

Laboratory 5:

AN ENHANCED PROCESSOR

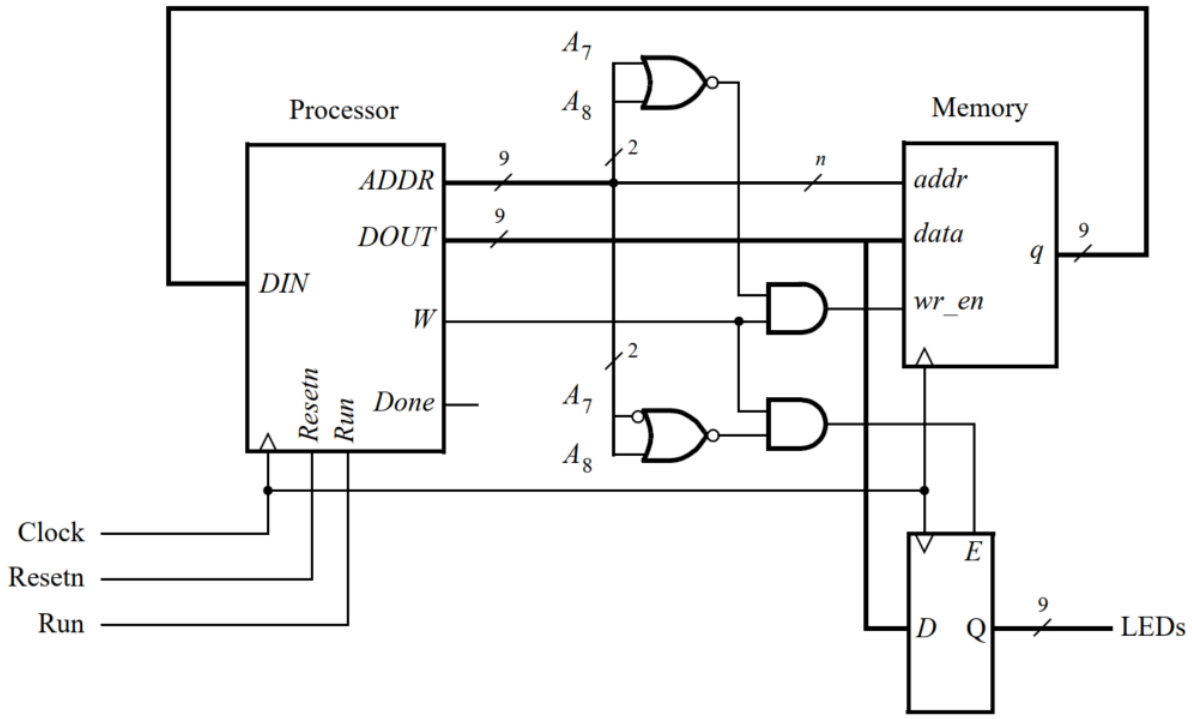

Figure 2: Connecting the enhanced processor to a memory and output register.

Instruction:

- Create a new Quartus project for the enhanced version of the processor.
- Write System Verilog code for the processor and test your circuit by using functional simulation: apply instructions to the DIN port and observe the internal processor signals as the instructions are executed. Pay careful attention to the timing of signals between your processor and external memory; account for the fact that the memory has registered input ports, as we discussed for Figure 2.
- Create another Quartus project that instantiates the processor, memory module, and register shown in Figure 2. Use the Quartus IP Catalog to create the RAM: 1-PORT memory module. Follow the instructions provided by the wizard to create a memory that has one 9-bit wide read/write data port and is 128 words deep. Ensure that the output is not registered. Use a MIF file to store instructions in the memory that are to be executed by your processor. An example program in the form of a MIF file is shown in Figure 3. This program display an 8-bit counter value on the

LEDs output port. Loops are used in the program to create delays so that the counter values are not changed too quickly to be observed. Comments are included in the MIF file in Figure 3 to describe the program's code.

```
DEPTH = 128;
WIDTH = 9;
ADDRESS_RADIX = HEX;
DATA_RADIX = BIN;
CONTENT
BEGIN

% This code displays a count (in register R2) on the red LEDs.                    %
00 : 001001000; %        mvi   R1,#1            // initialize R                  %
01 : 000000001;
02 : 001010000; %        mvi   R2,#0            // counter to display on LEDs    %
03 : 000000000;
04 : 001011000; % Loop   mvi   R3,#010000000    // R3 = address of LEDs register %
05 : 010000000;
06 : 101010011; %        st    R2,R3            // write to LEDs                 %
07 : 010010001; %        add   R2,R1            // increment counter for LEDs    %
08 : 001011000; %        mvi   R3,#111111111    // delay value                   %
09 : 111111111;
0A: 000101111; %         mv    R5,R7            // save address of next inst.    %
0B: 001100000; % Outer   mvi   R4,#111111111    // nested delay loop             %
0C: 111111111;
0D: 000000111; %         mv    R0,R7            // save address of next inst.    %
0E: 011100001; % Inner   sub   R4,R1            // decrement loop delay variable %
0F: 110111000; %         mvnz  R7,R0            // continue Inner loop if R4 !=0 %
10 : 011011001; %        sub   R3,R1            // decrement outer loop delay    %
11 : 110111101; %        mvnz  R7,R5            // continue Outer loop if R3 !=0 %
12 : 001111000; %        mvi   R7,#Loop         // execute again                 %
13 : 000000100;

END;
```

➢ *Figure 3:* An example program in a memory initialization file (MIF).

➢ Use functional simulation to test the circuit. Ensure that data is read properly from the memory and executed by the processor

➢ Include in your project the necessary pin assignments to implement your circuit on your DE-series board. Use switch SW9 to drive the processor's Run input, use KEY0 for Resetn, and

use the board's 50 MHz clock signal as the Clock input. Since the circuit needs to run properly at 50 MHz, make sure that a timing constraint is set in Quartus to constrain the circuit's clock to this frequency. Read the Report produced by the Quartus Timing Analyzer to ensure that your circuit operates at this speed; if not, use the Quartus tools to analyze your circuit and modify your Verilog code to make a more efficient design that meets the 50-MHz speed requirement. Also note that the Run input is asynchronous to the clock signal, so make sure to synchronize this input using flip-flops. Connect the LEDs register in Figure 2 to LEDR8−0 so that you can observe the output produced by the processor.

- Compile the circuit, download it into the FPGA chip, and ensure that your program runs properly.

Check: Your report has to show two results:

- The waveform to prove the circuit works correctly.
- The result of RTL viewer.