Santa Clara University

Lab #9: Registers

ELEN 21L 51306 Tuesday 2:15-5pm
June 6, 2017

Group 2
Matthew Placide
Yutong Li

Lab #9: Registers

I. Objectives:

In this lab, we will:

- Use flip-flops to build a register.
- Understand the operation of shift registers.
- Use multiplexers to design a multi-function shift register.

II. Introduction:

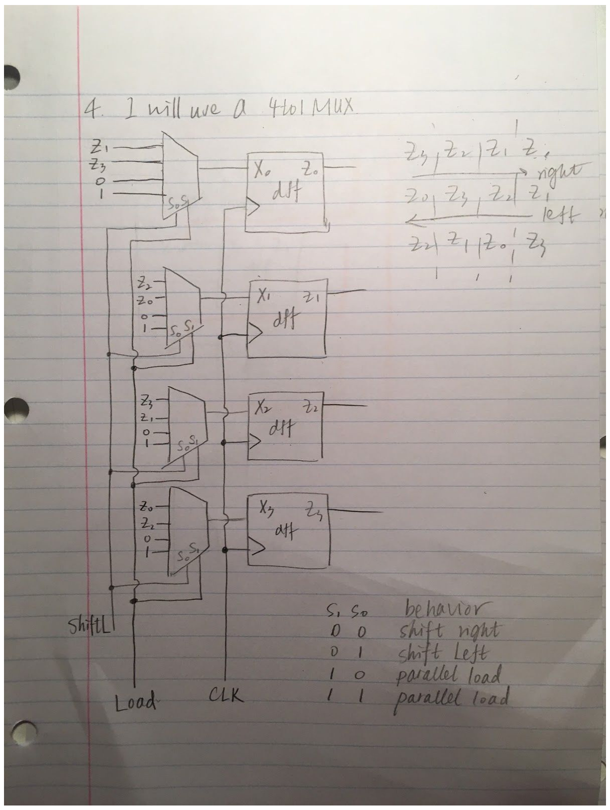
In this lab,we will build a register that can: 1) store bits, 2) shift right and left, and 3) parallel load. First we will build a simple register only containing D flip-flops to store the bit. Then we add a mux that can shift the bits to right or left depending on the switch. Finally, to implement storing, parallel load, shift right and shift right with one single circuit, we use a 4to1mux, and the output of the mux will be dependent on the combination of the switches.

III. Procedure:

Part 1: Data Storage Register

In order to build a 4-bit storage register, we use 4 D flip-flops, and let their inputs be four switches, and the outputs be both four LED, and a 7 segment display, and the clock be controlled by a push button.

Part 2: Shift Register



S1 S0	behavior
0 0	shift right
0 1	shift Left
1 0	parallel load
1 1	parallel load

Figure 1. Schematic for a 3-bit shift register

From the pre-lab, in order to implement shift left and right in the same circuit, we decide to use a 4to1mux. When the switches are 00, the bits will shift right, and when the switches are 01, the bits will shift left.
In order not to leave any vacant bit in the shift register, we decide to load 0(which is GND) when the switches are 10, and load 1(which is vcc) when the switches are 11.

Part 3: Universal Register
Universal shift register is a register which can be configured to load and retrieve the data in any mode (either serial or parallel) by shifting it either right or left. With the shift register that we had in part 2, we need to add led to the input of the D flip-flops. We have already had 2 switches in the 4to1 muxes.

Part 4: Arithmetic Operations using Shift

Opcode Input	Output Z	Which means
O_1O_0		
00	Z = Z	No change
01	Z = X	Parallel load
10	Z = floor(Z/2)	Shift right
11	Z = 2Z	Shift left

The arithmetic operation wants us to implement corresponding operation in the table above with specific combination of switches. As what we indicate what the arithmetic operation really want us to do, we know that all of the operations can be implemented with either shift or parallel load.
When the switches are 00, the inputs of the muxes should be the outputs of the D flip-flops.
When the switches are 01, the inputs should be the switches.
When the switches are 10, the inputs, except that of the leftmost bit, should be the inputs for shifting right, as shown in Figure 2, while that of the leftmost bit should be ground, because we want half of the bit for both even and odd number, when feeding the rightmost bit to the leftmost bit will give us three times of the odd number.
When the switches are 11, the inputs, except that of the rightmost bit, should be the inputs for shifting left, as shown in Figure 2, while that of the rightmost bit should be ground.

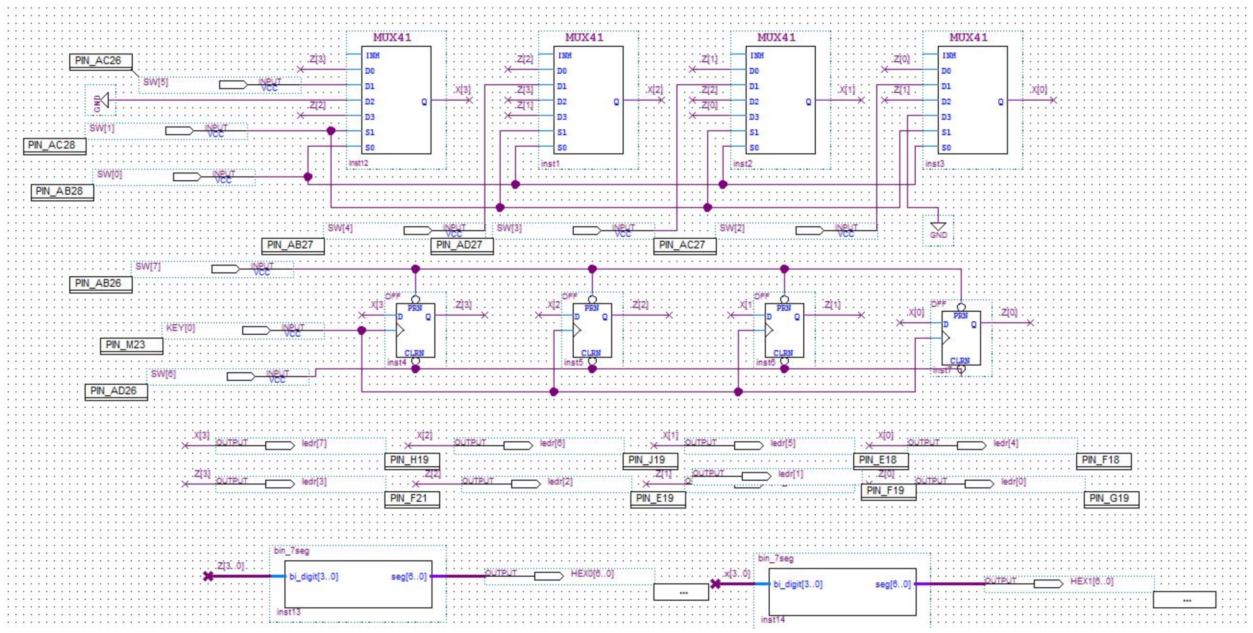


Figure 2: Schematic of universal shift register

IV. <u>Results:</u>

Part 1. Data Storage Register

When downloading to the board, the register is loading the data correctly.
The 7 segment display, the switches and the LED of the output match each other.

Part 2. Shift Register

When downloading to the board, the register is able to shift left and right, and parallel load 0 and 1.
For example, when the input is 0100(4), when the switches are 00, the register will show 0010(2), and when the switches are 01, the register will show 1000(8).

Part 3. Universal Register

We notice that when we input an even number, for example, 2, the number will change from 2 to 1 when we shift right, which is a half of 2, and the number will change from 2 to 4 when we shift left, which is a double of 2.
When we input an odd number, for example 3, the number will change from 3 to 9 when we shift right, which is three times of 3, and the number will change from 3 to 6 when we shift left, which is a double of 3. The reason that we get three times of 3 instead of a half of 3 is that we feed 1 to the left most bit instead of 0, which means if we want a half of 3, we will need to feed 0 to the left most bit.

input	Shift left	Shift right
2(0010)	4(0100, double)	1(0001, half)
3(0011)	6(0110, double)	9(1001, three times if we feed the leftmost to the rightmost) 1(0001, half if we feed 0 to the rightmost)

Part 4. Arithmetic Operations using Shift

When the switches are 00, and the input of the 4to1 muxes is 3(0011), the register will have 3(0011) when we press the push button.
When the switches are 01, and we change the input of the 4to1 muxes to 5(0101), the register will have 5(0101) when we press the push button.
When the switches are 11, and the input is 1(0001), whenever we press the push button, the register will be 2(0010), 4(0100), 8(1000), 0(0000).
When the switches are 10, and the input is 10(1010), whenever we press the push button, the register will be 5(0101), 2(0010), 1(0001), 0(0000).

V. **Final Questions**

1. *How would you modify the circuit to allow opcoes for both logical and circular shifts to the left and right?*
   In order to implement logical shifts, we need to feed zeroes from the right if we want to shift left, and we need to feed zeroes from the left if we want to shift right. In order to implement circular shift, we connect the last bit to the input on the other side.

2. *How would you use your circuit to load a 4-bit data word (parallel load) and create a serial bit sequence that could be the input to a serial communication circuit such as USB?*
   The parallel date would have to be loaded into the register simultaneously and shifted out of the register serially one bit at a time under the control of the clock.

3. *How would you add logic to allow for getting Z=3X?*
   We know that with one left shift, we can get the double of the original bit. So we think that we can add the bits after shifting left once and the original bits, which means we are adding X and 2X, which will give us 3X. The way we modify is to add an OR2 gate whose inputs are the original bits and the bits after shifting left once. Also, we decide to add an led to indicate the overflow.

VI. **Conclusion:**

We did pretty well in the first three parts, however, we still encountered a few difficulties especially in part 4. In the first three parts, we built a 4-bit register which can store and shift the bits from D flip-flop and through muxes. First, we didn’t give the right name to the verilog file, so we kept having the error when we were trying to compile the verilog file. We learned that we needed to give the same name to the file and the title in the file. Then, one line on one of the 7 segment display was on even though we didn’t assign any output to that spot. We checked the circuit and found out that we skipped a switch accidentally, and after we reassigned switches, the display disappeared. So we thought that the unexpected display might due to skipping a switch. We met main difficulties in the last part. We found that we can’t shift the bits once we pressed the button, instead we could only shift once. After working with the logic equations for both shifting left and shift right, we found that we need to ground the 2nd (0, 1, 2, 3) input of the mux in the most significant bit, because to shift right, we have to feed in zeroes from the left. And same for the 3rd input of the mux in the least significant bit, because we have to feed in zeroes from the right. After grounding these two bits, we were able to implement shifting. In this lab, we were implementing more difficult

VII. **Reference lists:**

- Dr. Sally Wood, Dr. Samiha Mourad, Dr. Radhika Grover. *Laboratory #9: Registers.* Spring 2017. Print
- Bin_7seg_temp.txt. Spring 2017. Print
- Dr. Sally Wood, Dr. Shoba Krishnan. *7-Segment Display Tutorial.* Spring 2017. Print
- Altera Corporation - University Program. *Quartus II Introduction Using Schematic Designs.* Spring 2017. Print
- Altera Corporation - University Program. *Quartus II Introduction to Simulation of Verilog Designs.* Spring 2017. Print
- Terasic Technologies Inc. *DE2-115 User Manual.* Spring 2017. Print.