Lab 1- Barrel Shifter

Cody Joy

November 27, 2017

1 Introduction

The purpose of this lab is to create an 8-bit and 16-bit left or right barrel shifter using two seperate methods. Method 1 is to use a case statement and method 2 is to use a multistage system. These two methods will be compared as to the run time as well as the building difficulty.

A barrel shifter is a digital circuit that can shift a number of bits by a certain number of bits either left or right. A barrel shifter can be thought of as similar to a barrel, because no new bits are added to the string of bits, the bits are just rotated. For example: if the following string of bits (00000001) is shifted to the left by 1, the following string of bits will be output (00000010). If that same starting string of bits is shifted to the right, the following string will be output (10000000). Notice that in the first case, the 1 shifted to the left by 1 bit, or the first 0 bit in the original string is shifted to be the last bit, Similarily with the second example, the 1 is shifted to the right by 1 bit, or the 1 is shifted to be the first bit. With a barrel shifter, the bits can be shifted any number of times to the left or right (after n bits shifted, the original string of bits is formed).

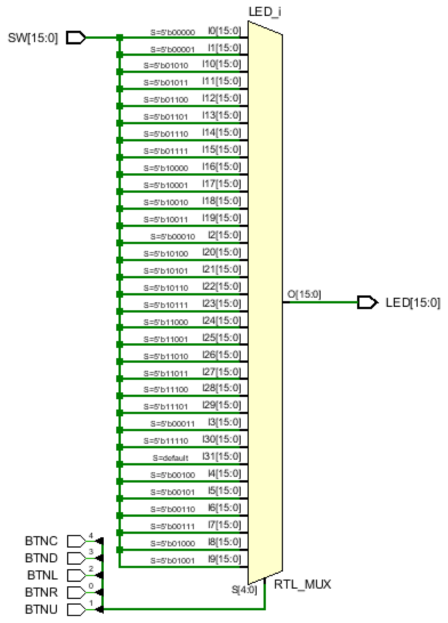
2 Interface

To implement the case and multistage barrel shifters, Verilog programs were created that were to be run on Nexys 4 DDR boards. The initial string of 8 or 16 bits are created using the switches on the board. The corresponding LEDs will show the output after the desired shifting has occured. Five buttons are used to switch between shifting left and shifting right as well as the amount of shifting for the string of bits. The default position is shift right, to shift left, press the center button (BTNC). To shift the bit by 1, press the right button(BTNR). To shift by 2, press the upper button (BTNU). To shift by 4, press the left button (BTNL). Finally, to shift by 8, press the down button (BTND). In order to shift by any number not 1,2,4, or 8, a combination of buttons need to be pressed. For example, if the desired bit shift is 5, press both the right button and the left button.

3 Design

The main difference between the case barrel shifter and the stage barrel shifter is the number and size of muxes used. For the case barrel shifter each of the different combination of shifts are inputed into a mux. For an 8 bit barrel shifter, an 8 input mux is required. For the 16 bit barrel shifter, a 16 bit mux is required. The main advantage of the case barrel shifter is the speed at which the action occurs. Since each of the signals go though only one mux, meaning the signal only goes through 2 logic gates. As the number of bits increases, the number of mux stages stays the same. The main disadvantage of the case barrel shifter is the number of inputs. With the number of inputs required for the case barrel shifter, fan in problems can occur. Fan in problems occur when the number of inputs is high enough, causing a high current to come in, which could potentially lead to overheating of the mux. Figure 1 shows the design of the case barrel shifter.

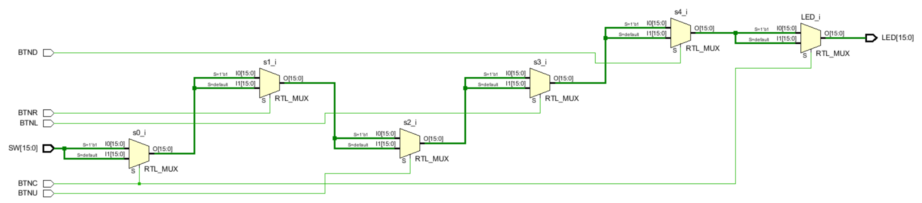
Figure 1: Case Barrel Shifter Schematic



The stage barrel shifter has the opposite advantages and disadvantages as compared with the case barrel shifter. In a stage barrel shifter, there are numerous stages of 2 input muxes. The advantages is the space required. Since each mux is only a two input mux, there are no problems with fan in. The main disadvantage of the stage barrel shifter is the speed at which the signal goes through all the muxes. For example, to implement a 16-bit stage barrel shifter, 4 stages are required, or 8 logic gates. It will take the signal 4 times as long to go through the entire system as compared with the case barrel shifter. For an

8-bit stage barrel shifter, 3 stages are required. As the number of bits increases, so does the required mux stages. Figure 2 shows the design of the stage barrel shifter.

Figure 2: Stage Barrel Shifter Schematic



4 Implementation

Appendix A shows the 16-bit case barrel shifter. This barrel shifter requires 16 switches (for the 16 bit input), 5 buttons (one for shifting to the left, one for shifting by 1 bit, one for shifting by 2 bits, one for shifting by 4 bits, and one for shifting by 8 bits), and 16 LEDs to show the output of the shifts. The case statement makes up the bulk of the code. The five buttons are combined together into one 5 bit number. If the button is pressed a 1 is recorded, if the button is not pressed, a zero. For no shift, non of the buttons will be pressed, leading to a binary value of 00000. The value 10000 produces the same result, but was add to ensure no glitches occur by just pressing the lift shift button. The next 15 cases are various amounts of shift, from 1 to 15. The binary values are thus the binary value of 1 to the binary value of 15. Normally, the bit position would be, from left to right, initial bit 15 to initial bit 0. With a right shift of 1, the initial bit 0 is moved to the far left, followed by the initial bit 15. This pattern continues, for a shift of 2, the farthest right bit is initial 1, for a shift of 3, the farthest right bit is initial 2, etc. The last 16 cases are all shifting to the left. To differential the left and right shift, the center button is used, resulting in a binary pattern starting with 1 for the left shift. It can be imagined that the first bit in this five bit sequence is similar to a sign bit. A left shift is negative, a right shift is positive. What can also be noticed is that the shifting pattern for the left shift is the opposite as that of the right shift. This is because a left shift of 1 is the same as a right shift of 15. Each of these shifted bit are set to the LED to display the shifted bit pattern.

Appendix B shows the 16-bit stage barrel shifter. The stage barrel shifter has the same input and outputs as compared with the case barrel shifter, with a few extra intermediate signals. The first stage, the assign stage, differentiates between the left and right shift if the shift is right (center button equals 0) the bit are left in the same order. If the center button is pressed, the order of the bits is inverted. This is due to the fact that a left shift is simply the opposite of a right shift. The bits of the left shift will be initially inverted, moved right, and then inverted again. The second stage shifts the bits either 1 (right button pressed) or 0 (right button not pressed). The third stage will shift the bits either 2 (upper button pressed) or 0 (upper button not pressed). The fourth stage will shift the bits either 4 (left button pressed) or 0 (left button not pressed). The fifth stage will shift the bits either 8 (down button pressed) or 0 (down button not pressed). The sixth and final stage will invert the bits if the center button is pressed and not invert the bits if the center button is not pressed. This final stage also will display the shifted result on the LEDs. For example, a bit shifted by 1 to the right will not be inverted, then shifted by 1, then not shifted, not shifted, not shifted, and then not inverted. A bit shift to the left by 15 will be inverted, shifted by 1, shifted by 2, shifted by 4, shifted by 8, and then inverted again.

Appendix C and D show the 8-bit versions the case and stage barrel shifters, respectively. The main differences with the shifters is the case has 8 fewer defined cases and the stage has one fewer stage.

5 Test Bench Design

Three different test benches where used for the 16-bit to test for various different potential errors 0000000000000001, 1111111111111110, and 1111110101010101. The purpose of the first test bench was to ensure that each of the different shifts will have one bit go high. This is to ensure that none of the switches, LEDs, or shift amount were set to zero. Each of the left and right shifts were successfully completed, with the one bit lighting up only one LED for each shift. The purpose of the second test bench was to ensure that each of the different shifts will have one bit that goes low. Similar to the first bench test the second bench test was to ensure that none of the switches, LEDs , or shift amount were set to one. Each of the left and right shifts were successfully completed, with the one bit keeping one LED off for each shift. The final test bench was to ensure that a semi-random set of bits with multiple on and off would function properly. In the program there could have been an issue with having a similar number of LEDs on and off. Similar to the first two test benches, this test bench was able to successfully shift left and right the string of bits.

6 Simulation

Simulations were completed for the 16-bit versions of the case and stage barrel shifter to verify the above test benches. The simulation for the case barrel shifter is shown in figure 3. Figure 4 shows the simulation for the stage barrel shifter. For the first test bench, the bits were shifted by 1 to the right each 20 ns. The results for both the case and stage barrel shifters are as expected. The one high bit shifter by one to the right. For the second test bench, the bits were shifted left or right a random amount. All possible shifts of the bits were tested for this stage. Again, the results were as expected. For the third test bench, all the bits were shifted by left by one bit at a time each 20 ns. Again, the results were as expected. From what can be seen from the simulation, moving either left or right a random or set amount does not cause any errors to the solution.

Figure 3: Case Barrel Shifter Simulation

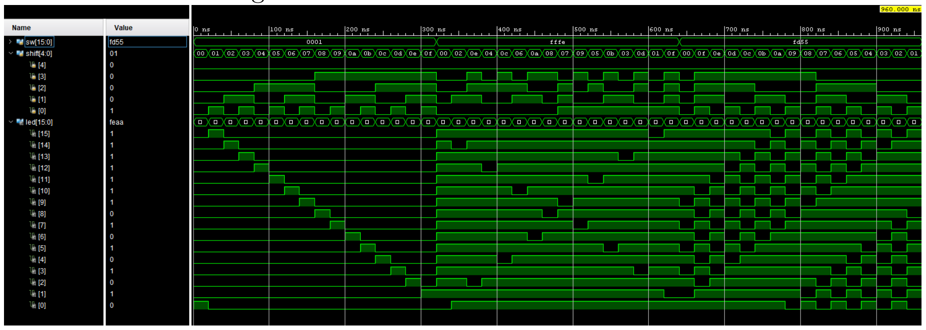
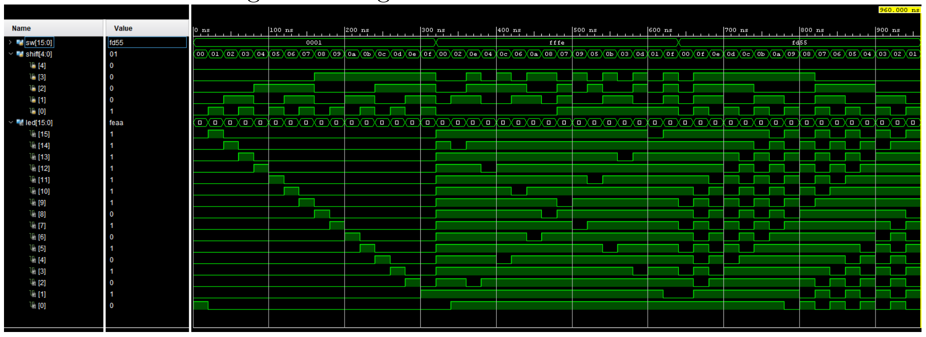


Figure 4: Stage Barrel Shifter Simulation



7 FPGA Realization and Final Verification

In order to finally realize the barrel shifter, the code was modified to allow for implementation on the Nexys 4 DDR board. The 16 switches were used as the input string of bits to be shifted, the 5 buttons were used to determine the number of bits the string of bits would shift and in what direction. Finally, the 16 LEDs were used to display the string of bits after the shifting had occured. The same test benches were tested on the board. The expected results were shown for each of the three test benches.

8 Conclusions

In conclusion, two different designs were created for an 8 and 16 bit barrel shifter: case and stage. The case barrel shifter had the advantage of being fast, but was not size efficient. One the other hand, the stage barrel shifter was size efficient, but was slow in executing. Both types were created using Verilog and testing on a Nexys 4 DDR board using 8 or 16 switches as the input string of bits, 5 buttons to control the direction of the shift, and the amount of shift. The reuslts were displayed using 8 or 16 LEDs. Three different test benches were used to verify that the code would produce the desired results. The results came out exactly as expected. If in told the program to shift to the left by 5, both shifters would shift to the left by five. If I told the program to shift to the right by 11, both shifters would shift to the right by 11.

In order to maximize the benefits of the two systems, a hybrid system would be preferred. The shifting can be completed like the stage barrel shifter and the rest will be completed by the case barrel shifter. This design will cut in half the time of the stage barrel shifter and cut in half the size of the case barrel shifter. Overall, I learned the importance of creating good test case for testing the implementation of the code. I also learned practically the tradoff between speed and size and that sometimes a compromise is the best option.

A 16-bit case barrel shifter

```
// Listing 3.18
module barrel_shifter_case
   (
     input [15:0] SW,
     input BTNL,
     input BTND,
     input BTNR,
     input BTNU,
     input BTNC,
     output reg [15:0] LED
    );

   // body
   always @* begin
      case({BTNC,BTND,BTNL,BTNU,BTNR})
         5'b00000: LED = SW;
         5'b00001: LED = {SW[0], SW[15:1]};
         5'b00010: LED = {SW[1:0], SW[15:2]};
         5'b00011: LED = {SW[2:0], SW[15:3]};
         5'b00100: LED = {SW[3:0], SW[15:4]};
         5'b00101: LED = {SW[4:0], SW[15:5]};
         5'b00110: LED = {SW[5:0], SW[15:6]};
         5'b00111: LED = {SW[6:0], SW[15:7]};
         5'b01000: LED = {SW[7:0], SW[15:8]};
         5'b01001: LED = {SW[8:0], SW[15:9]};
         5'b01010: LED = {SW[9:0], SW[15:10]};
         5'b01011: LED = {SW[10:0], SW[15:11]};
         5'b01100: LED = {SW[11:0], SW[15:12]};
         5'b01101: LED = {SW[12:0], SW[15:13]};
         5'b01110: LED = {SW[13:0], SW[15:14]};
         5'b01111: LED = {SW[14:0], SW[15]};
         5'b10000: LED = SW;
         5'b10001:LED = {SW[14:0], SW[15]};
         5'b10010:LED = {SW[13:0], SW[15:14]};
         5'b10011:LED = {SW[12:0], SW[15:13]};
         5'b10100:LED = {SW[11:0], SW[15:12]};
         5'b10101:LED = {SW[10:0], SW[15:11]};
         5'b10110:LED = {SW[9:0], SW[15:10]};
         5'b10111:LED = {SW[8:0], SW[15:9]};
         5'b11000:LED = {SW[7:0], SW[15:8]};
         5'b11001:LED = {SW[6:0], SW[15:7]};
         5'b11010:LED = {SW[5:0], SW[15:6]};
         5'b11011:LED = {SW[4:0], SW[15:5]};
```

```
            5'b11100:LED = {SW[3:0], SW[15:4]};
            5'b11101:LED = {SW[2:0], SW[15:3]};
            5'b11110:LED = {SW[1:0], SW[15:2]};
            default: LED = {SW[0], SW[15:1]};
        endcase
end

endmodule
```

B 16-bit stage barrel shifter

```
// Listing 3.19
module barrel_shifter_stage
   (
    input  [15:0] SW,
    input BTNL,
    input BTNU,
    input BTNR,
    input BTND,
    input BTNC,
    output  [15:0] LED
   );

   // signal declaration
   wire [15:0] s0, s1, s2, s3, s4;

   // body
   // stage 0, invert for left
   assign s0 = BTNC ? {SW[0],SW[1],SW[2],SW[3],SW[4],SW
       [5],SW[6],SW[7],SW[8],SW[9],SW[10],SW[11],SW[12],SW
       [13],SW[14],SW[15]}  : SW;
   // stage 1, shift 0 or 1 bit
   assign s1 = BTNR ? {s0[0], s0[15:1]} : s0;
   // stage 2, shift 0 or 2 bits
   assign s2 = BTNU ? {s1[1:0]  , s1[15:2]} : s1;
   // stage 3, shift 0 or 4 bits
   assign s3 = BTNL ? {s2[3:0], s2[15:4]} : s2;
   // stage 4, shift 0 or 4 bits
   assign s4 = BTND ? {s3[7:0], s3[15:8]} : s3;
   // stage 5, invert back for left
   assign LED = BTNC ? {s4[0],s4[1],s4[2],s4[3],s4[4],s4
       [5],s4[6],s4[7],s4[8],s4[9],s4[10],s4[11],s4[12],s4
       [13],s4[14],s4[15]} : s4;

endmodule
```

C 8-bit case barrel shifter

```
// Listing 3.18
module barrel_shifter_case
   (
     input  [7:0] SW,
     input BTNL,
     input BTNR,
     input BTNU,
     input BTNC,
     output reg [7:0] LED
    );

   // body
   always @* begin
      case({BTNC,BTNL,BTNU,BTNR})
         4'b0000: LED = SW;
         4'b0001: LED = {SW[0], SW[7:1]};
         4'b0010: LED = {SW[1:0], SW[7:2]};
         4'b0011: LED = {SW[2:0], SW[7:3]};
         4'b0100: LED = {SW[3:0], SW[7:4]};
         4'b0101: LED = {SW[4:0], SW[7:5]};
         4'b0110: LED = {SW[5:0], SW[7:6]};
         4'b0111: LED = {SW[6:0], SW[7]};
         4'b1000: LED = SW;
         4'b1001:LED = {SW[6:0], SW[7]};
         4'b1010:LED = {SW[5:0], SW[7:6]};
         4'b1011:LED = {SW[4:0], SW[7:5]};
         4'b1100:LED = {SW[3:0], SW[7:4]};
         4'b1101:LED = {SW[2:0], SW[7:3]};
         4'b1110:LED = {SW[1:0], SW[7:2]};
         default: LED = {SW[0], SW[7:1]};
      endcase
end

endmodule
```

D 8-bit stage barrel shifter

```
// Listing 3.19
module barrel_shifter_stage
   (
    input  [7:0] SW,
    input BTNL,
    input BTNU,
    input BTNR,
    input BTNC,
    output  [7:0] LED
   );

   // signal declaration
   wire [7:0] s0, s1, s2, s3;

   // body
   // stage 0, invert for left
   assign s0 = BTNC ? {SW[0],SW[1],SW[2],SW[3],SW[4],SW
       [5],SW[6],SW[7]}  : SW;
   // stage 1, shift 0 or 1 bit
   assign s1 = BTNR ? {s0[0], s0[7:1]} : s0;
   // stage 2, shift 0 or 2 bits
   assign s2 = BTNU ? {s1[1:0]  , s1[7:2]} : s1;
   // stage 3, shift 0 or 4 bits
   assign s3 = BTNL ? {s2[3:0], s2[7:4]} : s2;
   // stage 5, invert back for left
   assign LED = BTNC ? {s3[0],s3[1],s3[2],s3[3],s3[4],s3
       [5],s3[6],s3[7]} : s3;

endmodule
```