ECSE 323 Digital System Design

Lab #1 – Using the Altera Quartus Software

Group 27

Lulan Shen (260449509)
Loren Lugosch (260404057)

Design of a 14-bit Barrel-Shifter

1. Overview

A **barrel shifter** is a combinatorial circuit that takes as **input** a (in this case 14-bit) number on a bus line and gives as **output** the input number shifted by a certain number of bits. That **shift amount** is determined by the values on a third bus line.

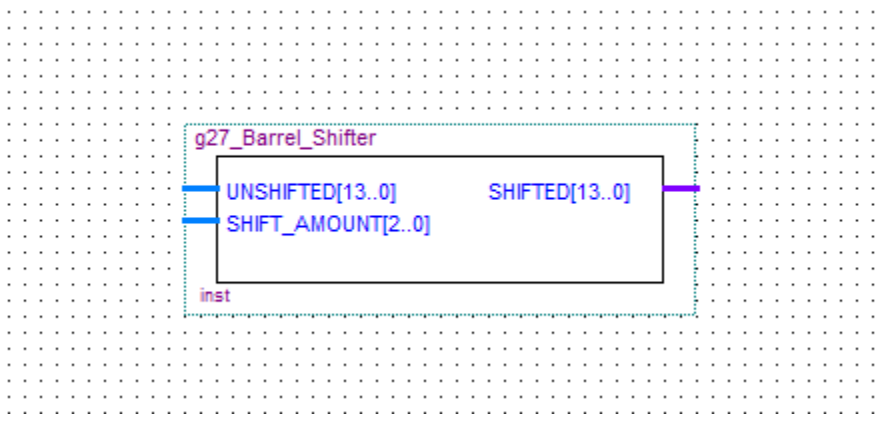


Figure 1 - Barrel Shifter Symbol

The circuit is built on bus multiplexers. In the following sections we describe the designing and testing of the circuit from the single multiplexer layer up.

2. Design of a 2-1 Multiplexer

We were asked to make a bus multiplexer, i.e. a circuit that can select one of two buses as output. A bus multiplexer can be thought of as a logical grouping of several 2-to-1 multiplexers, so we started by designing a 2-to-1 multiplexer.

A 2-to-1 multiplexer selects one input if select = 0 and the other input if select = 1. We wrote out the truth table:

S	A	B	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figure 2 - Truth Table for 2-to-1 Multiplexer

The table generates the function F (in minterms) = S'AB' + S'AB + SA'B + SAB, which simplifies to F = S'A + SB. F can be synthesized in AND, OR, and NOT gates as ((NOT S) AND A) OR (S AND B). We built this circuit in Quartus II using each of the pairs of lines of the input buses as single inputs to F.

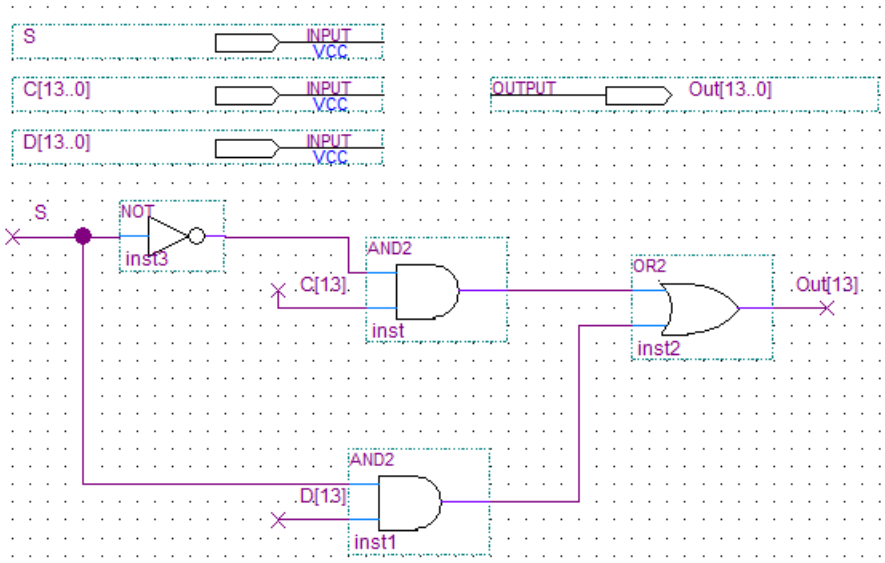


Figure 3 - g27_ 2to1_mux

3. Design of a 14-bit Bus Multiplexer

Once we had made a 2-to-1 multiplexer, we made 13 more instances of the circuit. We completed the bus multiplexer by wiring each pair of bits from the input to the inputs A and B of the multiplexers, the select line S to each of the multiplexer selects, and the outputs to the output bus.

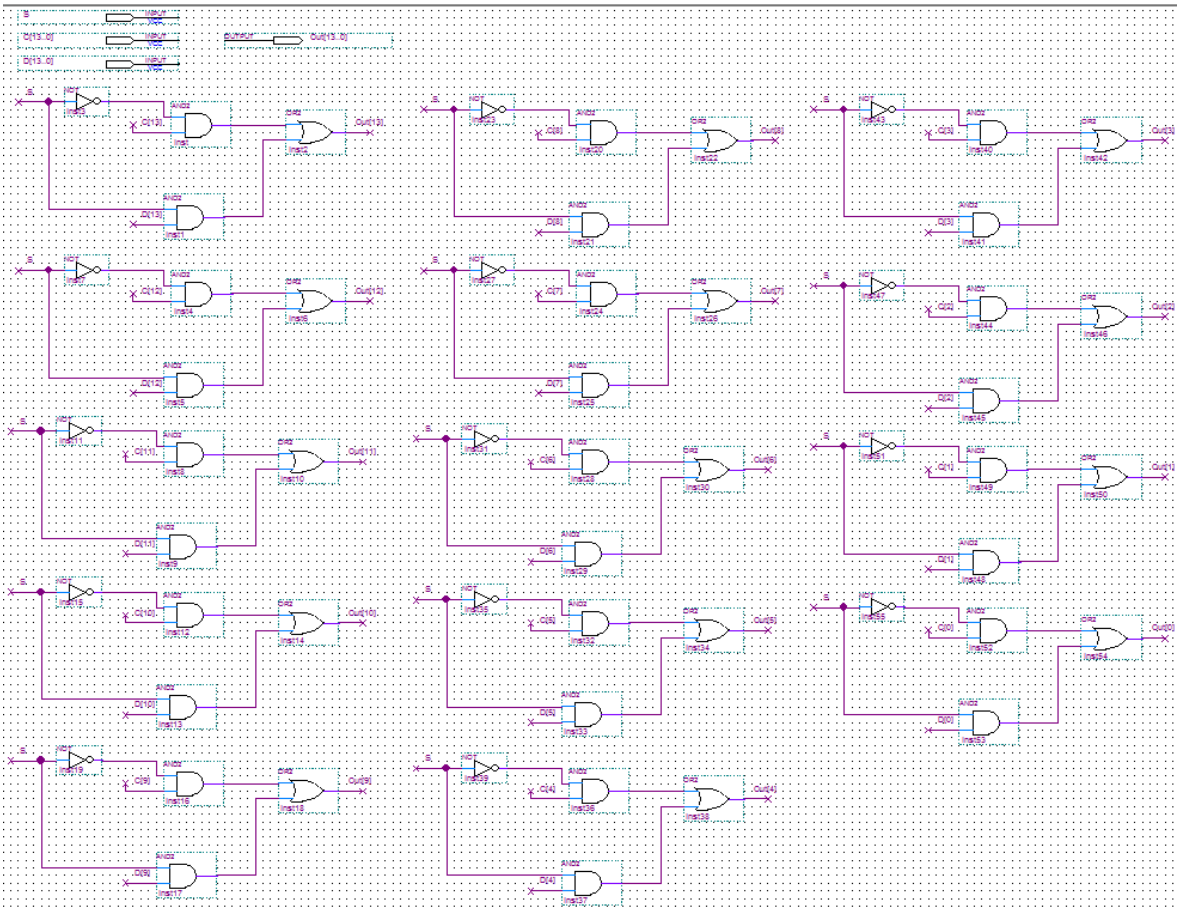


Figure 4 - Circuit g27_busmux

To make this large circuit easily replicable and usable, we made a symbol out of it.

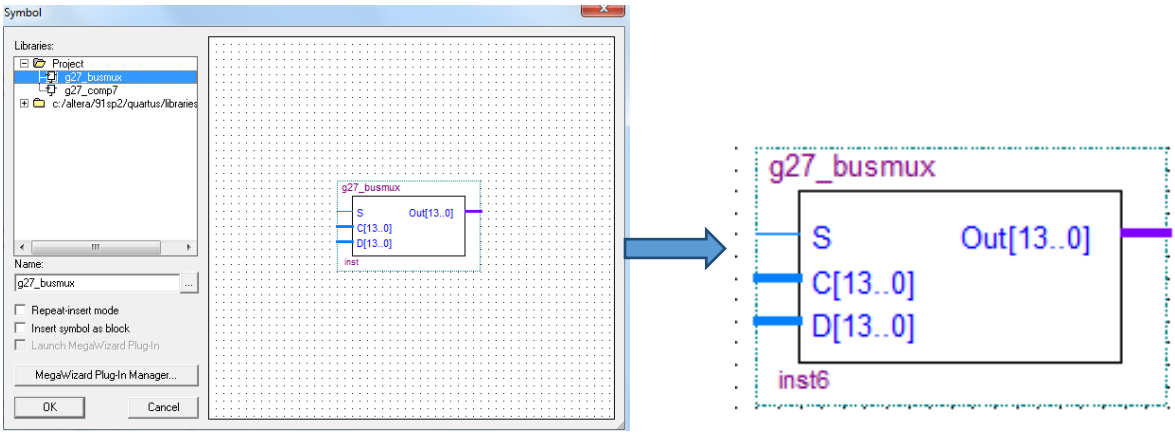


Figure 5 – Creating a Symbol

4. Design of a 14-bit Barrel-Shifter

The barrel-shifter has one 14-bit input to be shifted and one 3-bit input representing the number of bits that the other input should shift to give the output. The leftmost bits after shifting are zero. The only components we were allowed to use were three bus multiplexers. We only had one input to the circuit, and the multiplexers have two inputs, so we wired the same input, UNSHIFTED, to both inputs of the first multiplexer, except one of those inputs was "shifted."

We made this shift happen by setting the most significant bit of one of the inputs to zero and the other 13 bits to the 13 most significant bits of UNSHIFTED. (Setting the bit to zero meant creating a wire to ground called "zero" and wiring zero to the appropriate bit.) This makes a 1-bit barrel-shifter, where the select bit of the multiplexer determines whether the output of the shifter is shifted by 0 or by 1.

We repeated that trick for shifters that can shift the input by either 0 bits or 2 bits or 0 bits or 4 bits using the remaining two multiplexers, and wired the output of one multiplexer to the next, so that the final output was the output, SHIFTED, of the circuit, and the ith-shifter contributed 2^SHIFT_AMOUNT[i] to the number of shifts.

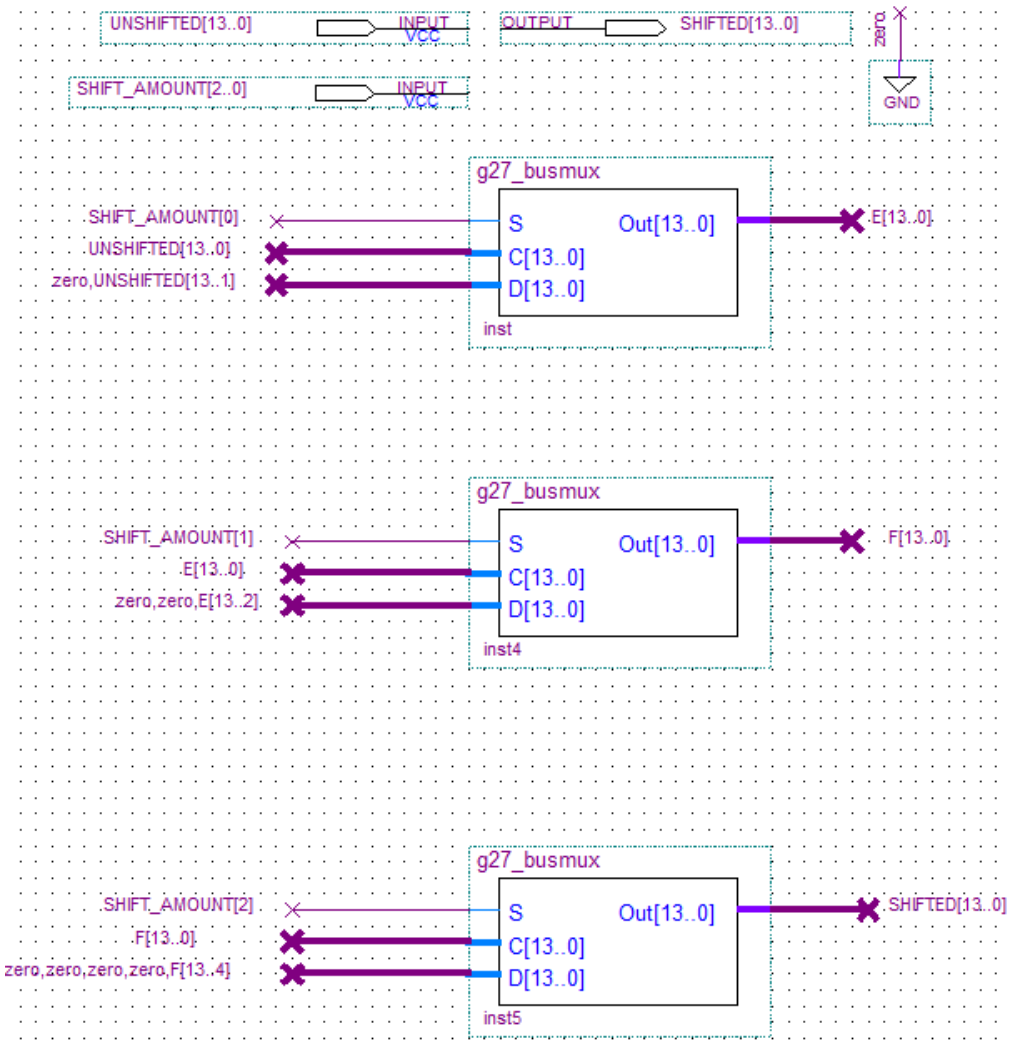


Figure 6 - g27_Barrel_Shifter

5. Functional simulation of the 14-bit Barrel-Shifter circuit

The lab description asked finally for a functional simulation of the circuit. As the lab description says, circuit simulation gives us two pieces of information:

1. does the circuit implement the function we want?
2. does the circuit meet timing requirements?

For this lab we only wanted the first of those pieces of information: confirmation that the Barrel Shifter shifts the input correctly. Simulation waveforms give us an easy way to verify the circuit; in this case, we can simply look at the number in the SHIFTED waveform, compare it to the number in the UNSHIFTED waveform, and decide whether the input has really been shifted by SHIFT_AMOUNT bits.
When we

The best input waveforms have 1 in some of the more significant bits (bits 13-6) because those 1s help us to check two of the requirements of the circuit:

1. Bits shifted in from the left are zero.
2. The circuit works for all 8 possible shift amounts.

Having some 1s in those bits helps us to check **1.** because it is more visually obvious where the shifted bits start (i.e. if we shift by two bits and see a 1 in either of those two bits, it is easier to see that something has gone wrong if the third bit should be a 1 than if the first several bits of the unshifted input are 0). That condition also helps us to check **2.** because if the input waveform has 1s in only bits 5-0, we can't check whether the shifter works for shifting by 7 bits because the output will be all 0s after a certain point. But we don't want just want 1s because we also want to make sure that the bit-pattern is preserved. The best inputs have 1s in the more significant bits, but not just 1s.

We chose an end time for our simulation to match the number of combinations to test: (8 possible shift values) x (4 numbers to test) x (40 ns per combination, long enough to make them visible on the waveform) = 1280 ns.

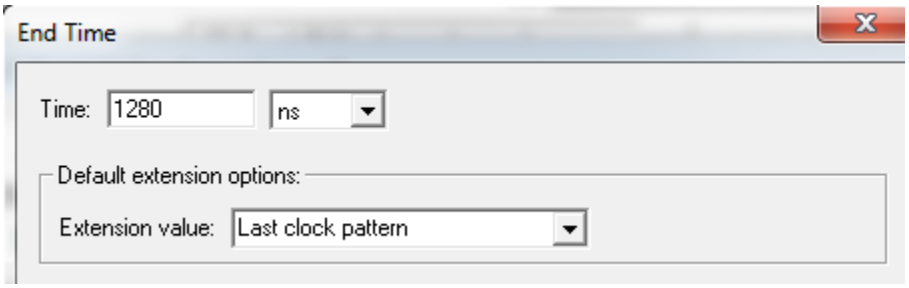


Figure 7 - End Time Selection

To get four values that met the criteria for “good input” listed above, we started the inputs at 10110010000000 and incremented by 1010101 to get 10110011010101, 10110100101010, and 10110101111111. We multiplied the 40 ns resting time by 8 since each number will be tested with 8 shift values. We did the same thing for the shift value, only we shifted by 1 (to get 000, 001, 010, up to 111) and didn’t multiply the resting time- that aligns the 8 shift values with each input number.

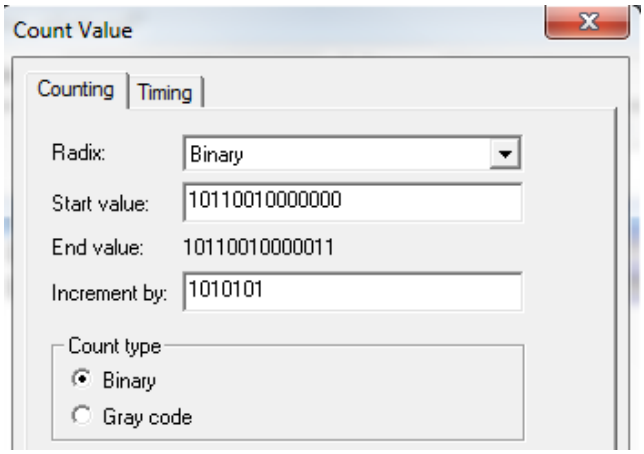


Figure 8 – Unshifted Counting

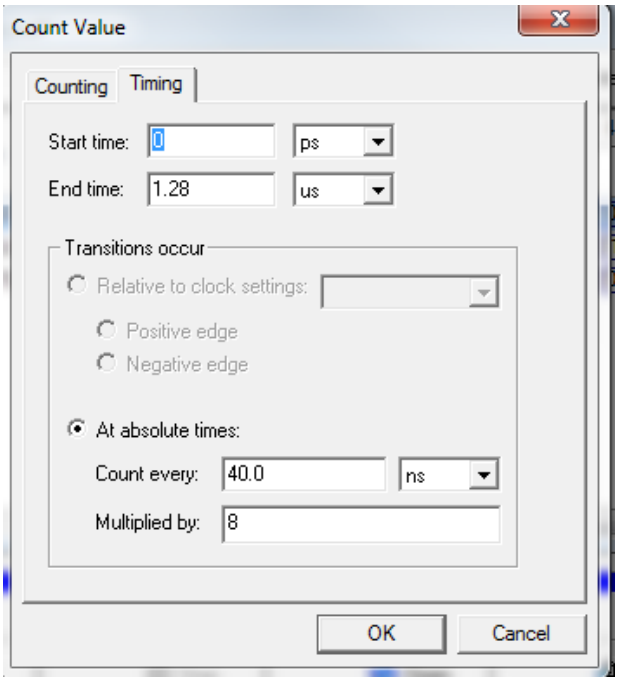


Figure 9 - Unshifted Timing

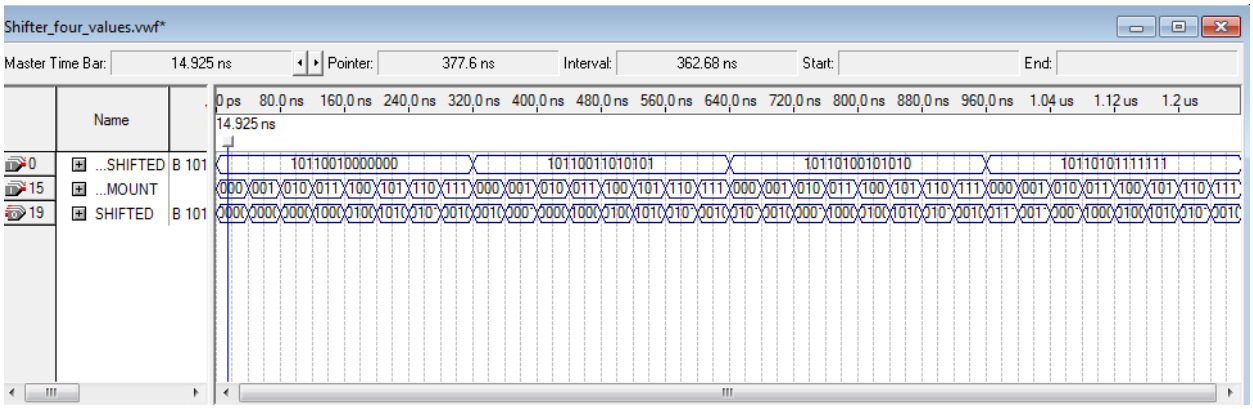
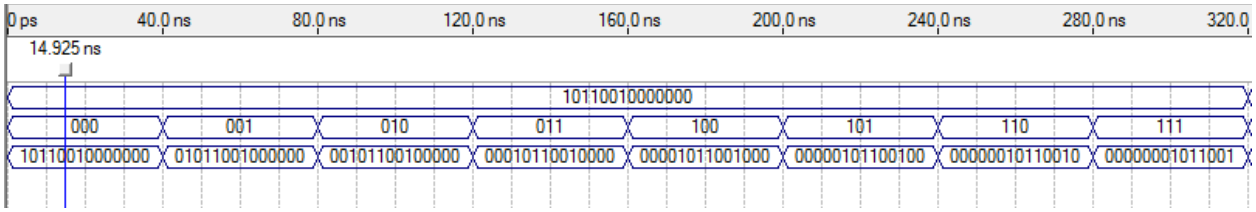


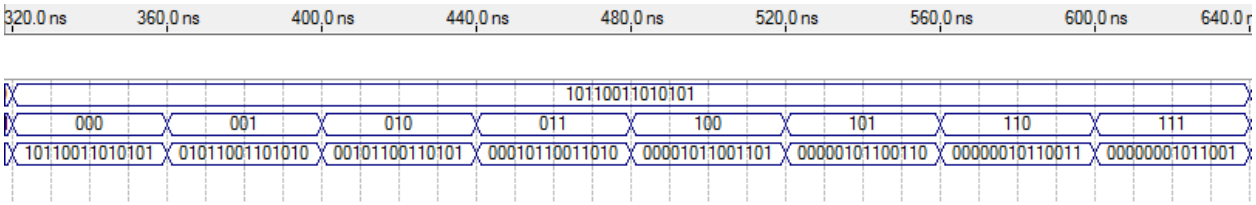
Figure 10 - Waveform With All Input Values Visible

The results of the simulation are displayed for each input number below:

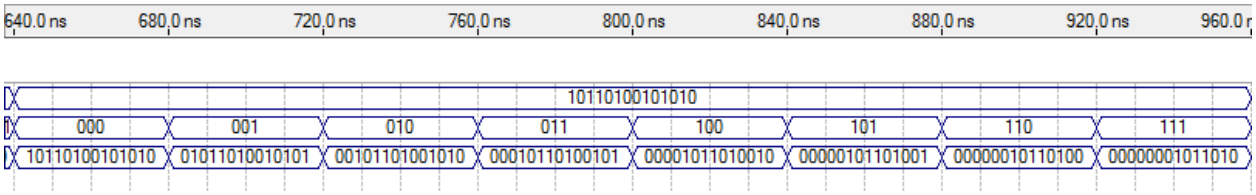
1. 10110010000000



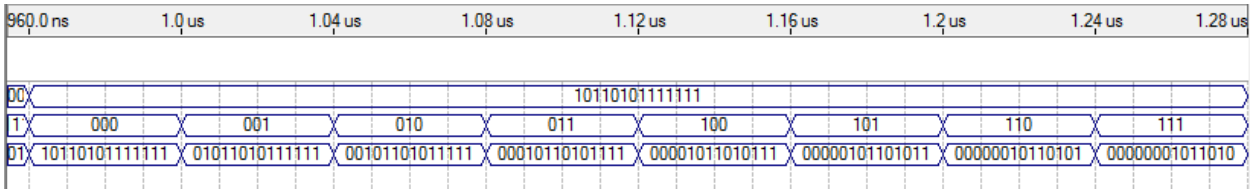
2. 10110011010101



3. 10110100101010



4. 10110101111111



Inspection of the simulation waveforms shows that shift does indeed happen.