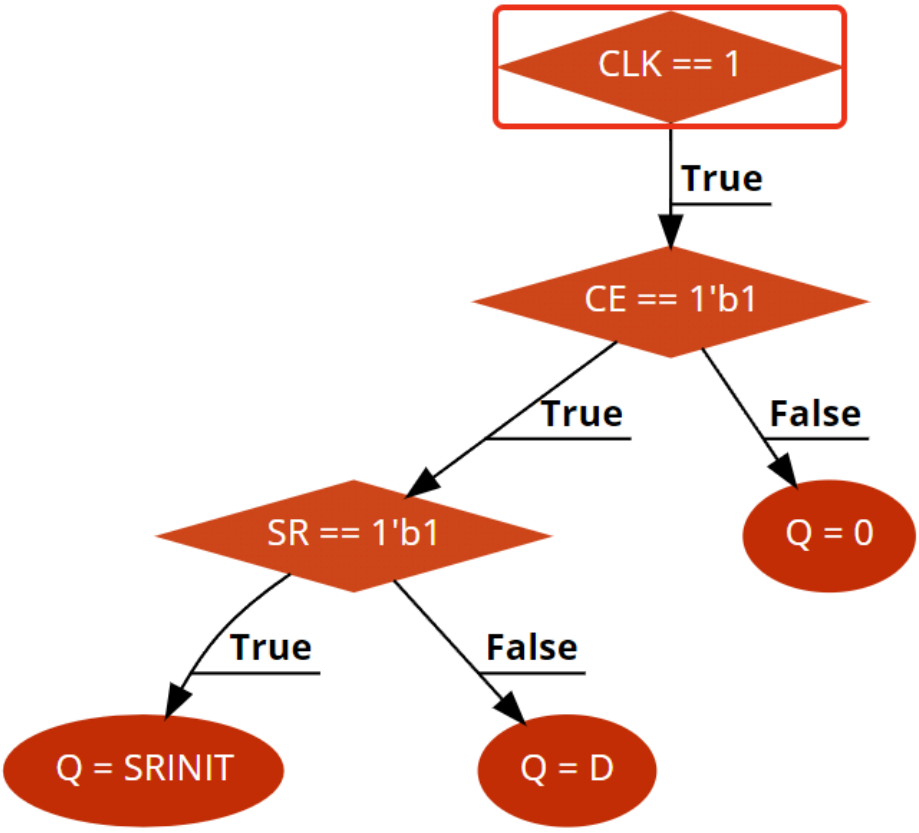
Grant Lance

Spring 2024 - 4/15/2024

ECE 474 - Assignment 1

1. Control Flow Graphs and Whitebox/Blackbox Testing
    a. DFF_SSR



i. From the chart above, it can be seen that there are three possible paths: 1 - (CE == 1’b0, Q = 0), 2 - (CE == 1’b1, SR == 1’b0, Q = D), and 3 - (CE == 1’b1, SR == 1’b1, Q = SRINIT). From these three paths, three test cases can be examined.

ii.

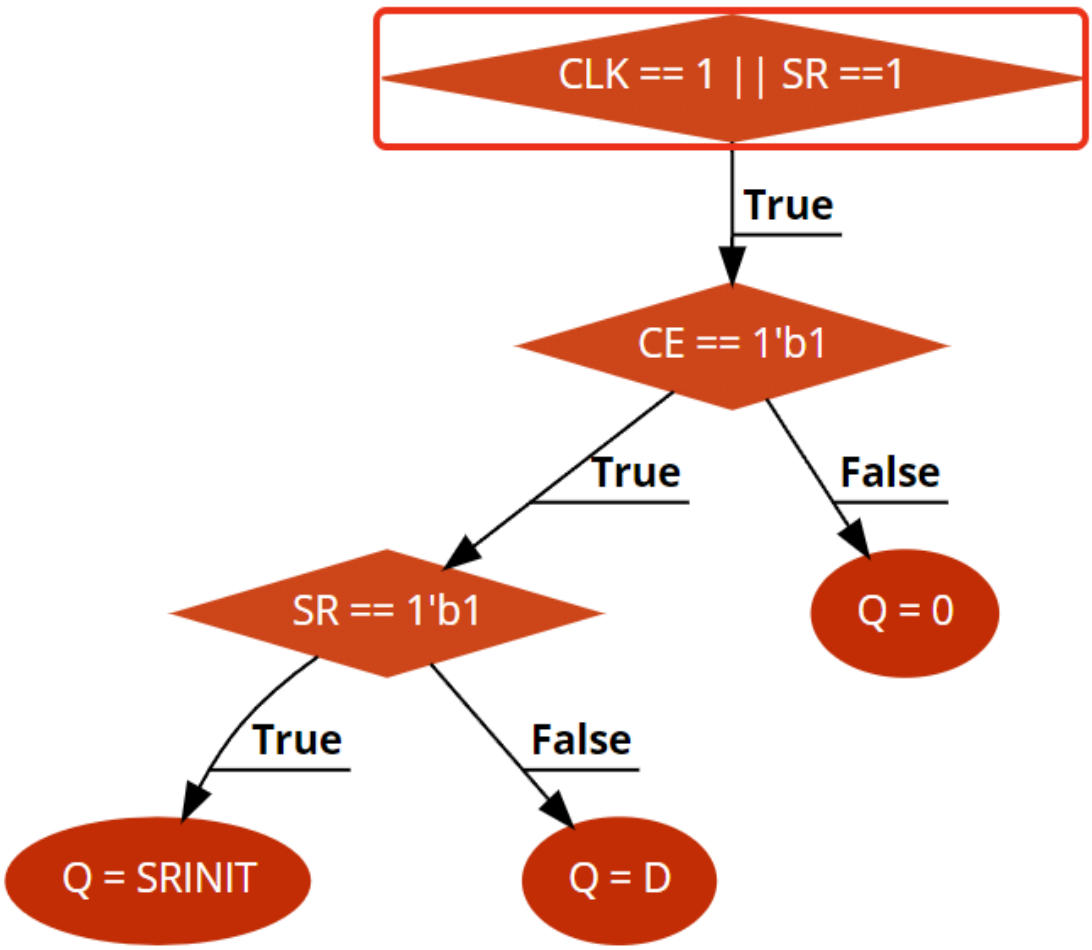
Test Case/Path	CLK	CE	SR	Q
Default	0	n/a	n/a	n/a
1	1	0 (not 1)	n/a	1

2	1	1	0	D
3	1	1	1	SRINIT

iii.

iv.

b. DFF_ASR



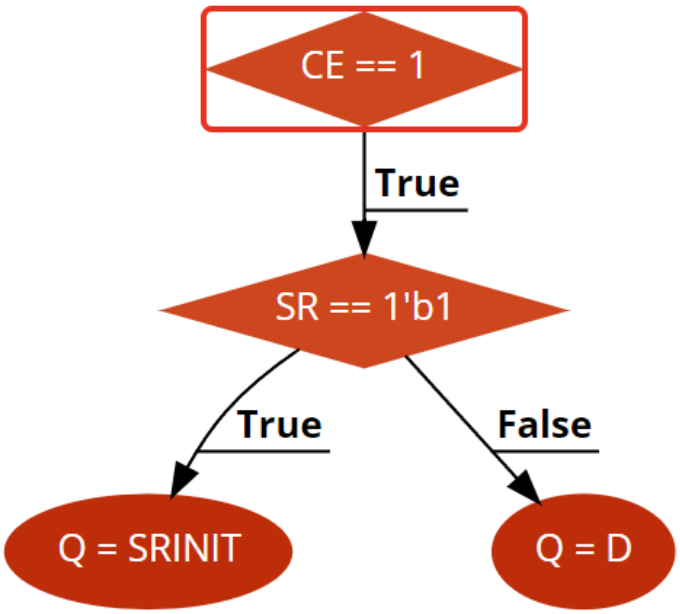
i. Also here, it can be seen that there are three possible paths from CLK == 1 || SR == 1 (just like DFF_SSR): 1 - (CE == 1'b1, CE == 1'b0, Q = 0), 2 - (CE == 1'b1, SR == 1'b0, Q = D), and 3 - (CE == 1'b1, SR == 1'b1, Q = SRINIT). From these three paths, three test cases can be examined.

ii.

Test Case/Path	CLK || SR	CE	SR	Q

Default	0	n/a	n/a	n/a
1	1	0 (not 1)	n/a	1
2	1	1	0	D
3	1	1	1	SRINIT

c. DLATCH_SSR



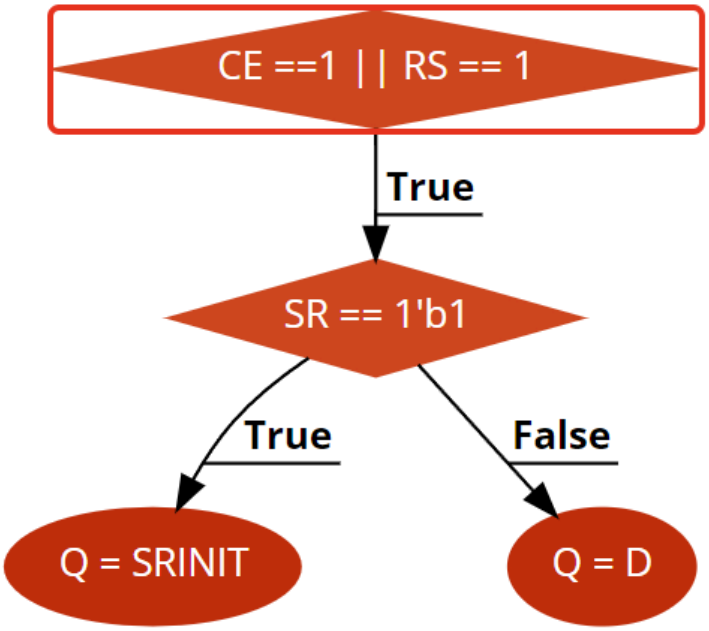
i. The latch- being simpler than the DFF only has two paths (and a technical third that all modules have- CE/CLK==0 and nothing happens). The two paths here are: 1 - (SR == 1'b0 to Q = D) and 2 - (SR == 1'b1 to Q = SRINIT). The DLATCH will either be reset or relay whatever is in the input. From these two paths, two test cases are formed.

ii.

Test Case/Path	CE	SR	Q
Default	0	n/a	n/a
1	1	0	D
2	0	1	SRINIT

iii.

d. DLATCH_ASR

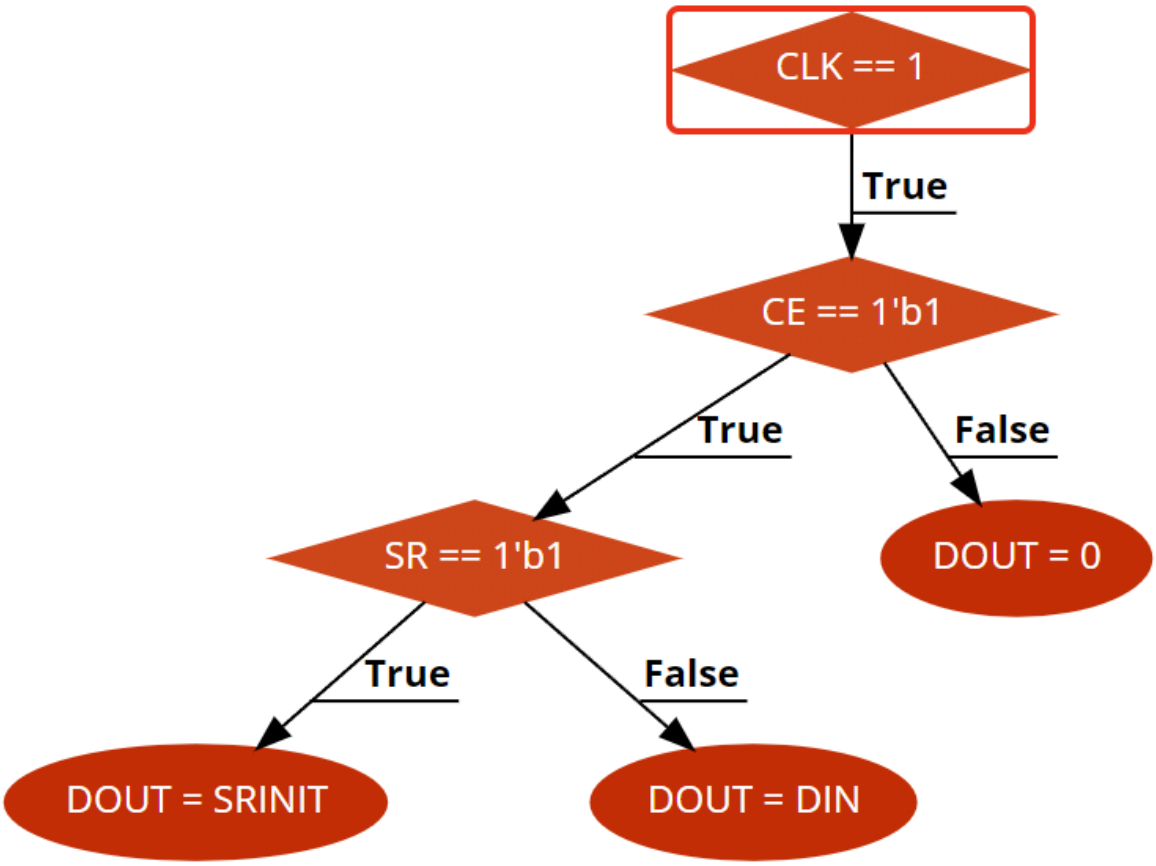


i. Again, being simpler than the DFF only has two paths (and a technical third that all modules have- CE/CLK==0 and nothing happens). The two paths here are: 1 - (SR == 1'b0 to Q = D) and 2 - (SR == 1'b1 to Q = SRINIT). The DLATCH will either be reset or relay whatever is in the input. Just like DLATCH_SSR however there are some timing differences that are easier to see from the simulation results. From these two paths, two test cases are formed.

ii.

Test Case/Path	CE	SR	Q
Default	0	n/a	n/a
1	1	0	D
2	0	1	SRINIT

e. nBitRegister_SSR



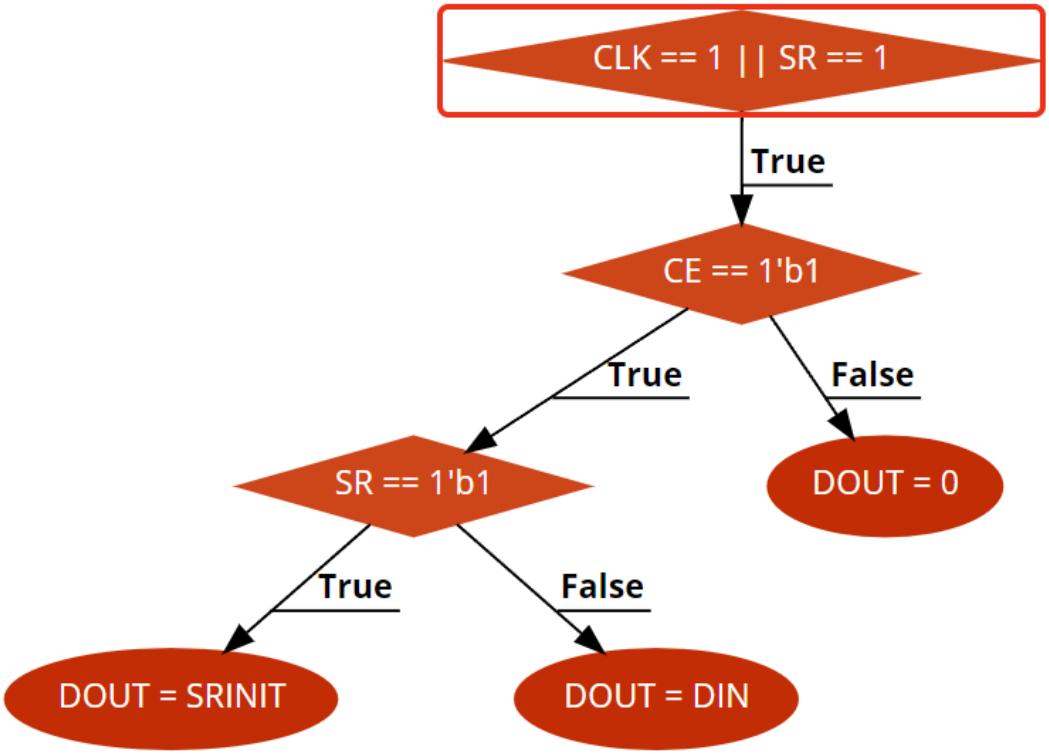
i. From the chart above, it can be seen that there are three possible paths (just like DFF_SSR and DFF_ASR): 1 - (CE == 1'b0, DOUT = 0), 2 - (CE == 1'b1, SR == 1'b0, DOUT = DIN), and 3 - (CE == 1'b1, SR == 1'b1, DOUT = SRINIT). As a register (of "N" bits) is essentially some array of DFF, the control flow charts should be the same. From these three paths, three test cases can be examined.

ii.

Test Case/Path	CLK	CE	SR	DOUT
Default	0	n/a	n/a	n/a
1	1	0 (not 1)	n/a	0
2	1	1	0 (not 1)	DIN

3	1	1	1	SRINIT

f. nBitRegister_ASR

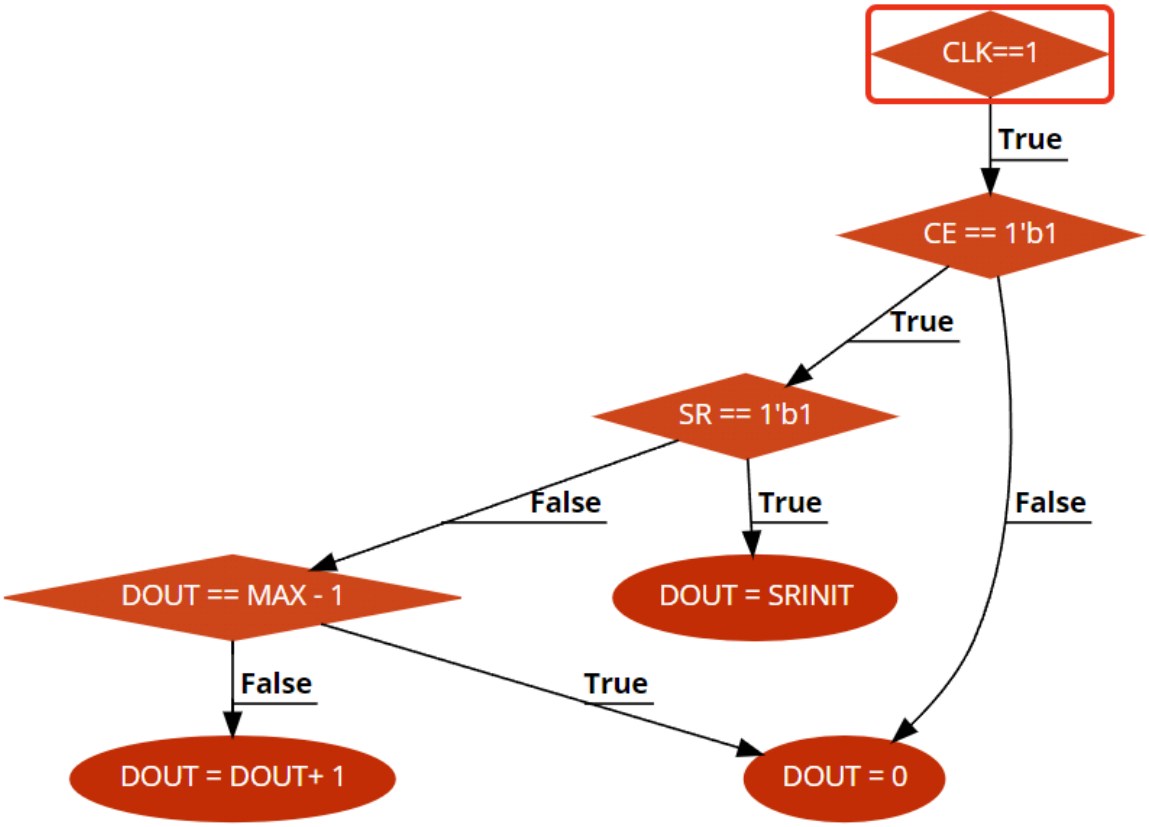


i. It can be seen that there are three possible paths: 1 - (CE == 1’b1, CE == 1’b0, DOUT = 0), 2 - (CE == 1’b1, SR == 1’b0, DOUT = DIN), and 3 - (CE == 1’b1, SR == 1’b1, DOUT = SRINIT). From these three paths, three test cases can be examined.

ii.

Test Case/Path	CLK || SR	CE	SR	DOUT
Default	0	n/a	n/a	n/a
1	1	0 (not 1)	n/a	0
2	1	1	0 (not 1)	DIN
3	1	1	1	SRINIT

g. nBitCounter_SSR

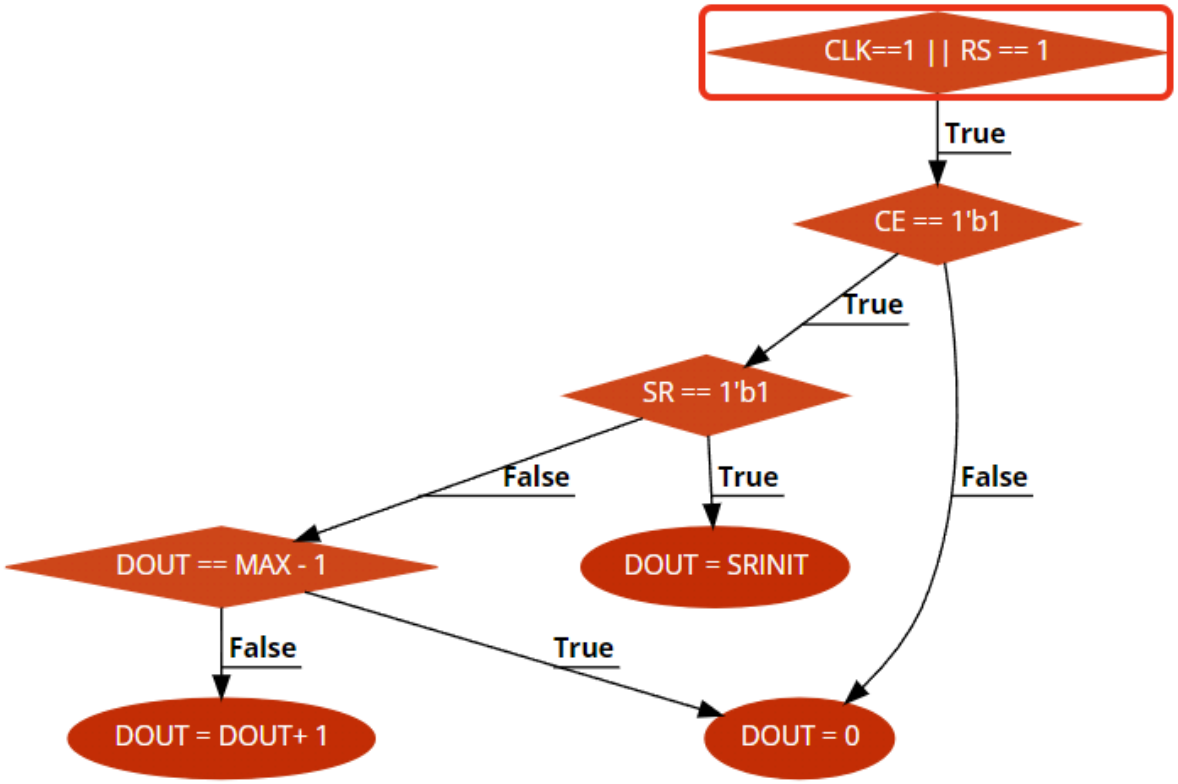


i. Through the counter there are four paths: 1 - (CE == 1’b0, DOUT = 0), 2 - (CE == 1’b1, SR == 1’b1, DOUT = SRINIT), 3 - (CE == 1’b1, SR == 1’b1, DOUT == MAX-1, DOUT = 0), and 4 - (CE == 1’b1, SR == 1’b1, DOUT != MAX-1, DOUT = DOUT + 1). From these four paths, four test cases can be examined.

ii.

Test Case/Path	CLK	CE	SR	DOUT == MAX-1	DOUT
Default	0	n/a	n/a	n/a	n/a
1	1	0	n/a	n/a	0
2	1	1	1	n/a	SRINIT
3	1	1	0	1	0
4	1	1	0	0	DOUT+1

h. nBitCounter_ASR



i. Again, through the counter there are four paths: 1 - (CE == 1'b1, CE == 1'b0, DOUT = 0), 2 - (CE == 1'b1, SR == 1'b1, DOUT = SRINIT), 3 - (CE == 1'b1, SR == 1'b1, DOUT == MAX-1, DOUT = 0), and 4 - (CE == 1'b1, SR == 1'b1, DOUT == MAX-1, DOUT = DOUT + 1). From these four paths, four test cases can be examined.

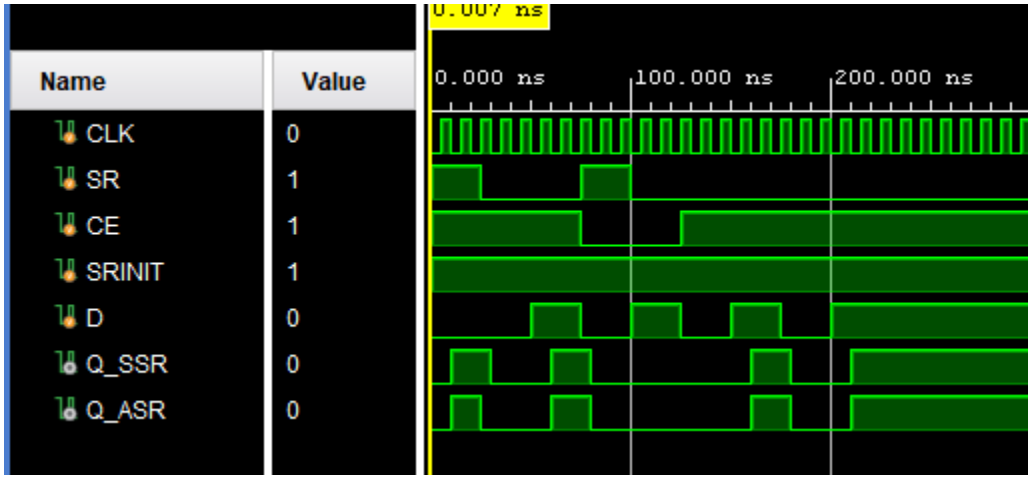
ii.

Test Case/Path	CLK || RS	CE	SR	DOUT == MAX-1	DOUT
Default	0	n/a	n/a	n/a	n/a
1	1	0	n/a	n/a	0
2	1	1	1	n/a	SRINIT
3	1	1	0	1	0

4	1	1	0	0	DOUT+1

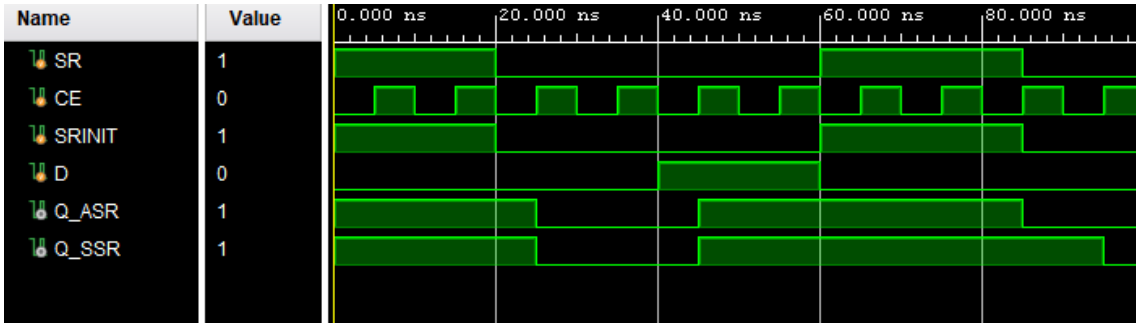
2. Testing

a. DFF



i. Above is a screen shot of DFF_tb. As can be seen, when the clock cycles from high to low, the input is relayed to the output. Note that the ASR variant updates its status on every falling edge of CLK and SR. Because of this, the ASR variant (output) responds to the change in SR faster than the SSR variant.

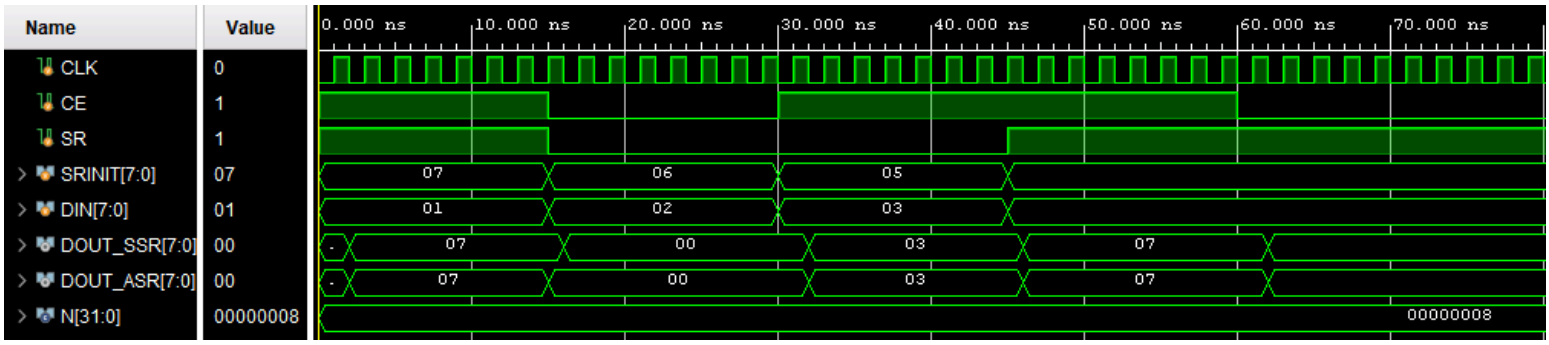
b. DLATCH



i. Above is a screenshot of the DLATCH_tb. The difference between the SSR/ASR block is that the SSR block only updates on “CE or D " whereas the ASR block updates on “CE or D or SR”. Because of this, the ASR block updates before the SSR one when SR goes from high to low. Note that at 85 nsec SR is going low while CE is going high. Because the SSR block is not dependent on SR, it will not update when it goes from high to

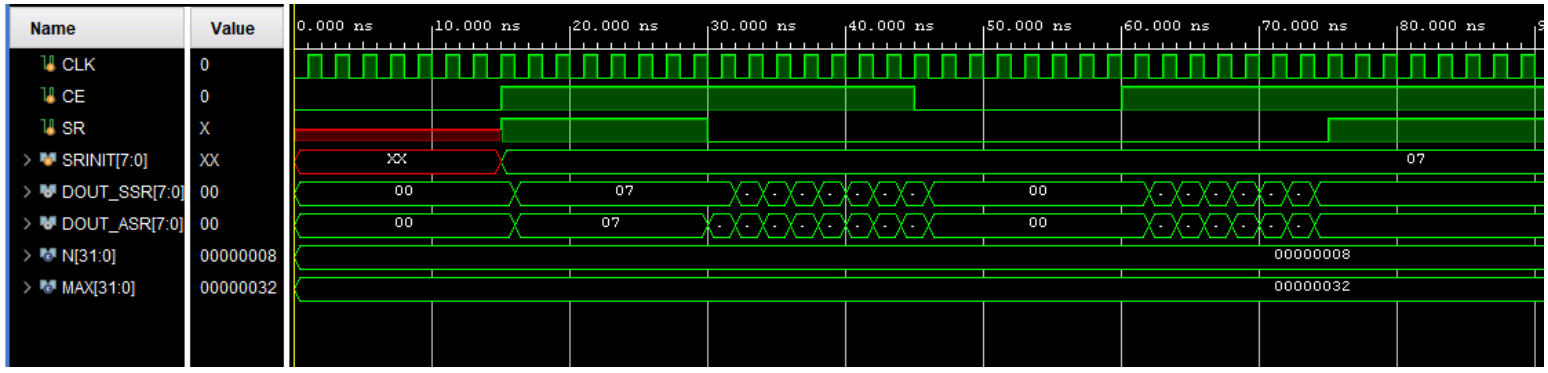
low. Instead the SSR block will “fall behind” the ASR one and the output will update on the next change in CE.

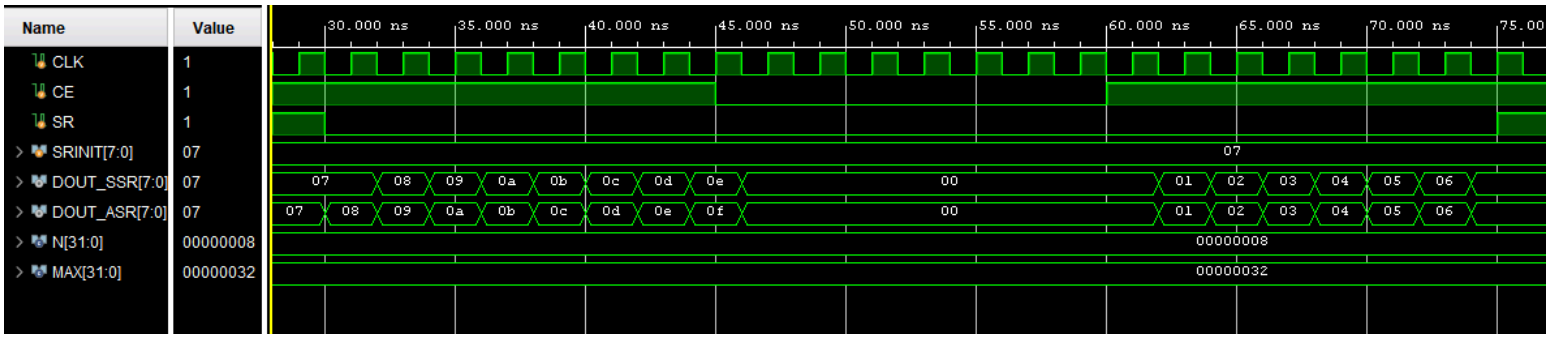
c. nBitRegister



i. Looking at a screenshot of the simulation output for nBitRegister, it can be seen that both function as intended. As with the single bit register/D Flip-Flop, the SSR variant of this block will update on the falling edge of SR causing it to update before the SSR variant when this happens.

d. nBitCounter





i. From the above screen shots of nBitCounter_tb, it can be seen that the blocks function as intended. Like the other blocks, the difference in SSR/ASR causes a timing difference between the two variants when SR is

used. Note that there are two screenshots. One shows the output of the testbench commands. The second shows the counter actually counting as the output is not displayed when viewing the waveforms at the scale of the first screen shot.