Tony Hong
1005109482
Oct. 14, 2020

CSC258 Prelab (Lab 4)

Part 1: Gated D latch and Edge Triggered Flip Flop

3. For the D latch, if the clock input is 0 the output values will not change no matter what input D is given. To see if the D latch is working, one should turn the clock to 1 and check if when D is 1, Q is 1 and when D is 0, Q is 0. When the clock is turned off once again, the D latch will store whatever output was present right before the clock was turned off and the output will remain the same until the clock is once again turned to 1.

Part 2: Extended ALU with Register

2(a): If I did not include the register, the output would be unstable and oscillate.

2(b): When multiplying two n-bit binary numbers, we will need 2n bits to store the result.

3(a): Test sequence 1: To test if adding 1 to A works, test values of A=0, 1, 2, 4. Each time, in order to show the result, turn the clock from 0 to 1 (positive edge triggered).

Test sequence 2: To test if adding A and B (the previous value calculated) works, set A=1 and proceed to cycle the clock value to accumulate a final value of 15 after 15 clock cycles. Then test A=2 and proceed to cycle the clock value to accumulate to a final value of 14 after 7 cycles.

Test sequence 3: Same testing as test sequence 2

Test sequence 4: Set A to a randomly chosen value. When we cycle the clock, every cycle should yield the value from 2 cycles ago. This value of course will be seen again in another 2 clock cycles.

Test sequence 5: Using multiplication by 0, set the output to be 0. If A=0, output should be 0. Any other A and output should be 1.

Test sequence 6: Set the output to be stored as 00000001 (Therefore “B” is 1). Then set A to value 1. Cycle the clock 4 times and ensure the bit shifts left once each time.

Test sequence 7: Set the output to be stored as 00010000. Then set A to value 1. Cycle the clock 4 times and ensure the bit shifts right once each time.

Test sequence 8: Set the output as 00000001 and then set A=2. Cycle the clock to check if the bits are shifted once every clock cycle. Set the output as 15 and then set A=15. Make sure that the result after cycling the clock is 225.

3(b): Below are some tests ran to check the validity of my implementation of functions 5-7.

Function 5: Left Shift

Passed: 8 Failed: 0

status	A	B	Y
pass	0000	1111	0000 1111
pass	0000	0010	0000 0010
pass	0001	0001	0000 0010
pass	0001	1111	0001 1110
pass	0011	0001	0000 1000
pass	0011	1111	0111 1000
pass	0111	0001	1000 0000
pass	0111	1110	0000 0000

Function 6: Logical Right Shift

Passed: 8 Failed: 0

status	A	B	Y
pass	0000	1111	0000 1111
pass	0000	0010	0000 0010
pass	0001	0001	0000 0000
pass	0001	1111	0000 0111
pass	0011	1000	0000 0001
pass	0011	0111	0000 0000
pass	0111	0001	0000 0000
pass	0111	1110	0000 0000

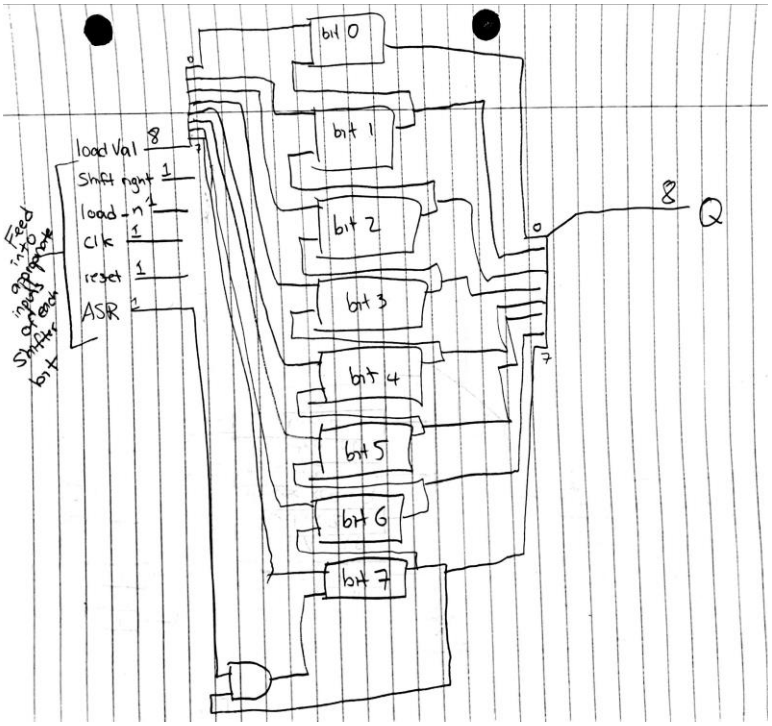
Function 7: Multiplication

Passed: 8 Failed: 0

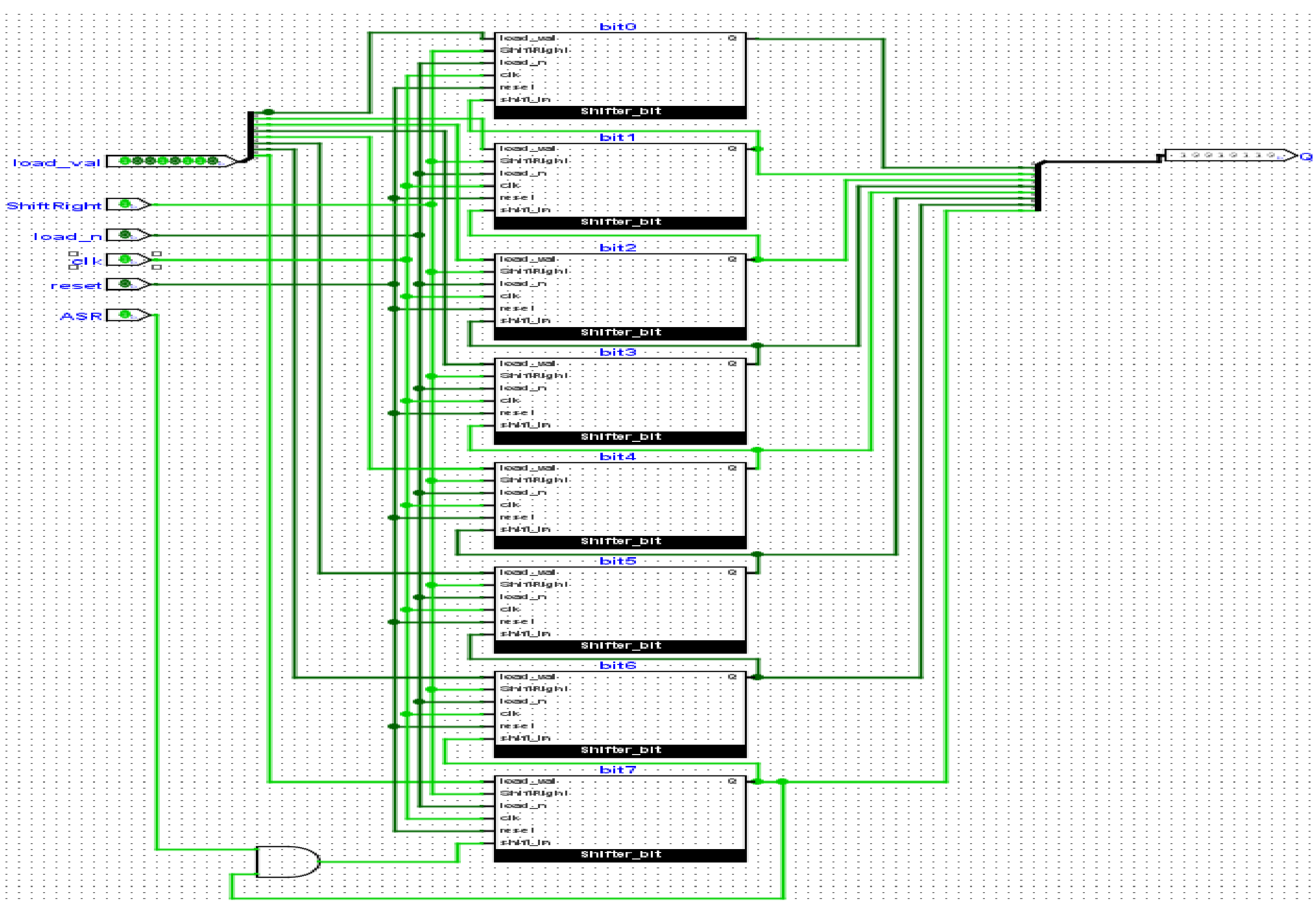
status	A	B	Y
pass	0000	1111	0000 0000
pass	0000	0000	0000 0000
pass	0001	0001	0000 0001
pass	0001	1111	0000 1111
pass	0011	1000	0001 1000
pass	0011	0111	0001 0101
pass	1111	1111	1110 0001
pass	1000	1000	0100 0000

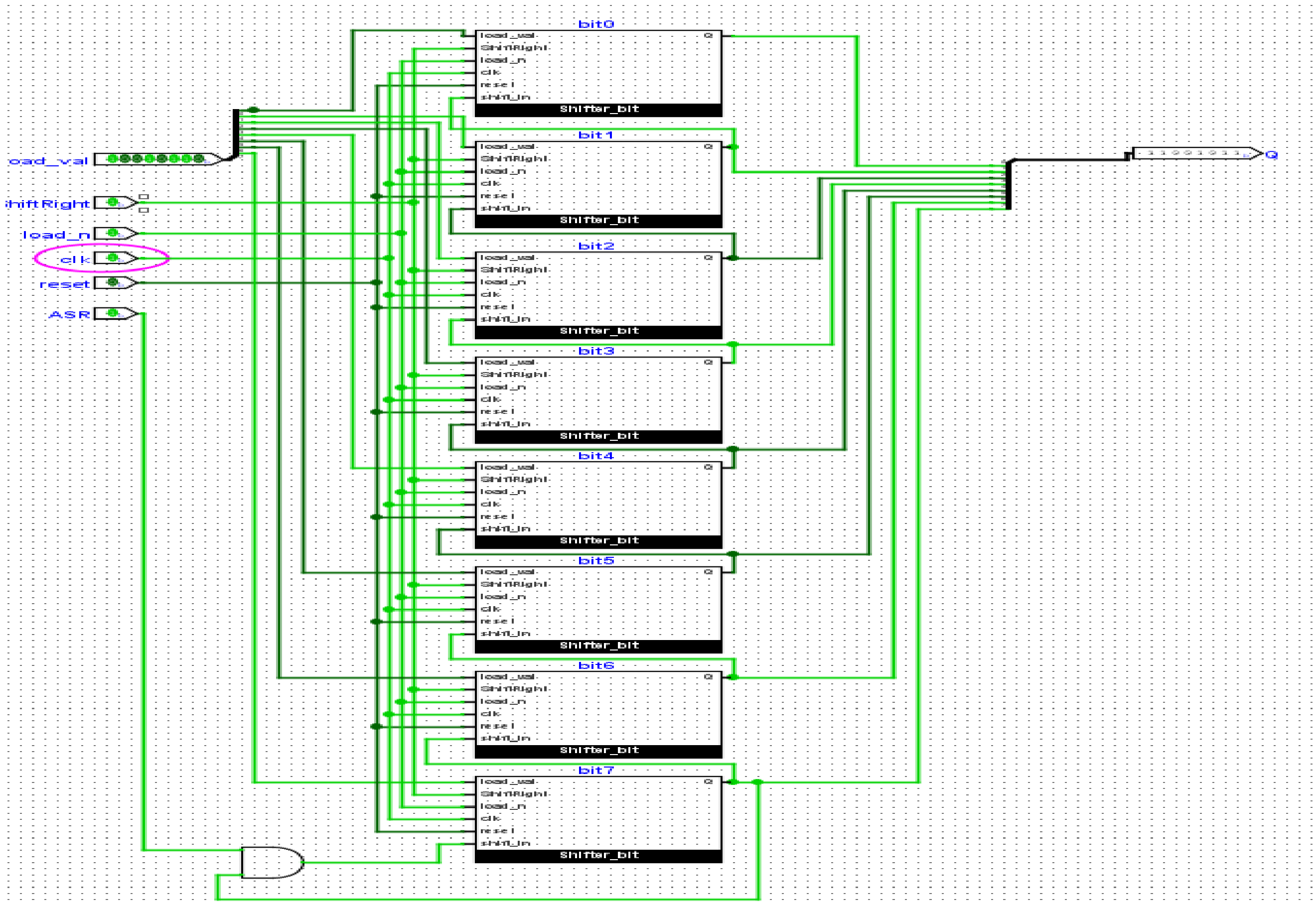
Part 3: Shift Register

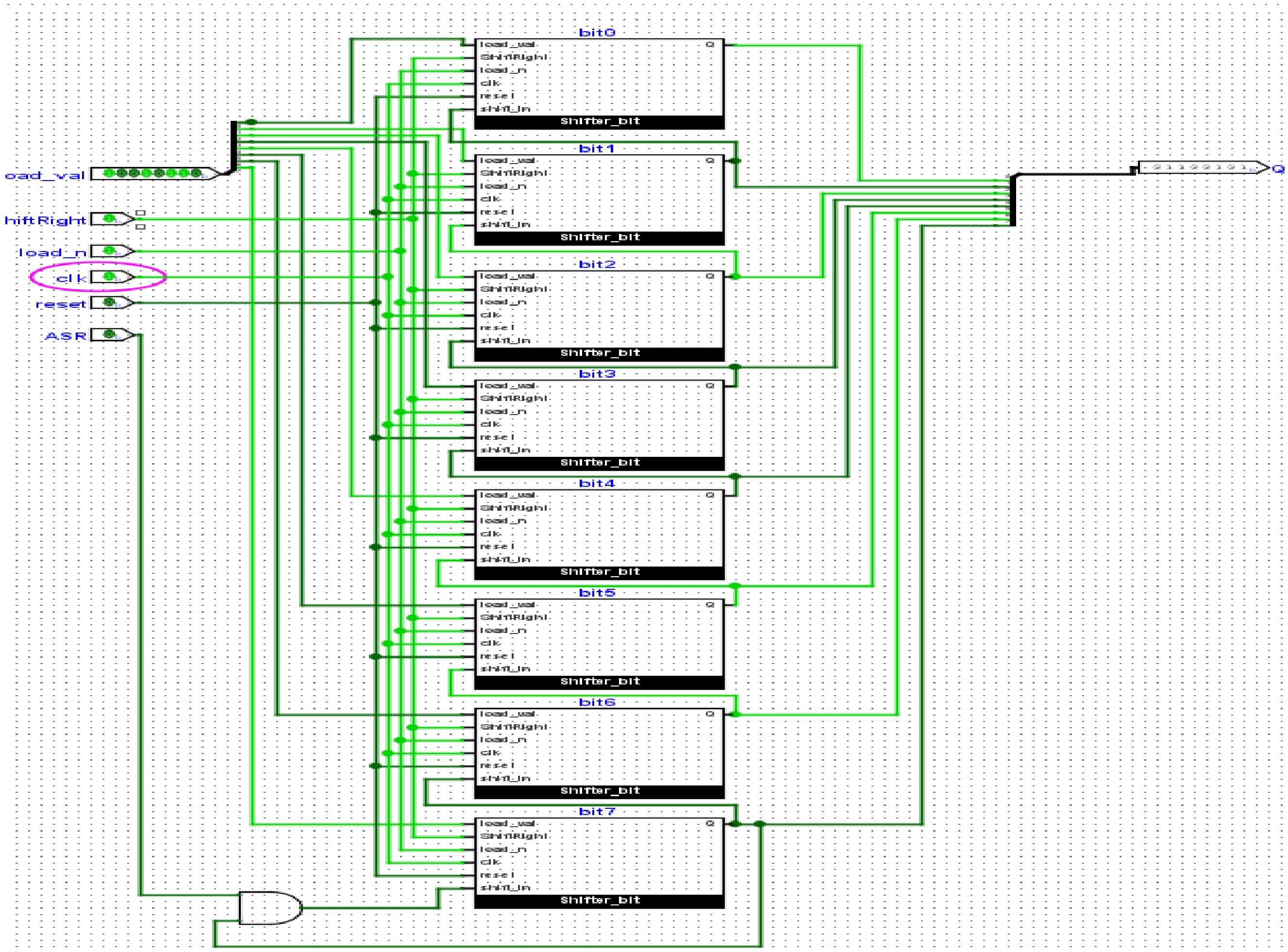
1. The behaviour of the 8-bit register when Load_n = 1 and ShiftRight = 0 is that each output in the bit register will be fed back as input effectively preserving the state of the entire shift register.
2. Below is a schematic for the 8-bit shift register that I will implement in Logisim.



5. Below is a simulation of loading values and then followed by an arithmetic right shift, and then a logical right shift.




bit0
bit1
bit2
bit3
bit4
bit5
bit6
bit7
load_val
ShiftRight
load_n
clk
reset
shift_in
Shifter_bit
Q
oad_val
hiftRight
load_n
clk
reset
ASR


bit0
bit1
bit2
bit3
bit4
bit5
bit6
bit7
load_val
ShiftRight
load_n
clk
reset
shift_in
Shifter_bit
Q
oad_val
hiftRight
load_n
clk
reset
ASR