

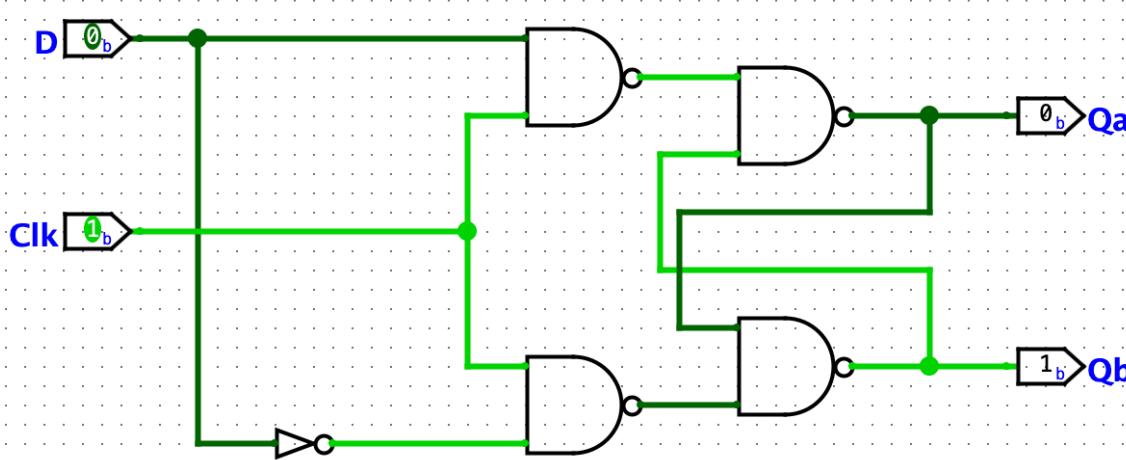
Lab 4. Latches, Flip-flops, and Registers

Student Name: Zewen Ma.

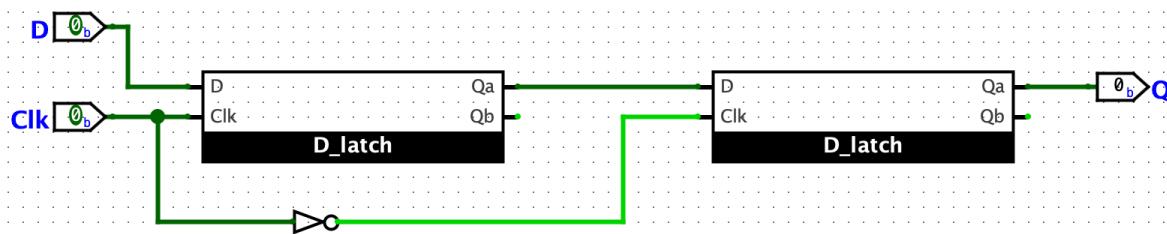
Student Number: 1005968375

Part 1:

1. Schematic of gated D latch:



Schematic of master slave flip-flop:



3. We know that, from the SR latch, the outputs of the SR latch are different when S and R are different (i.e., they are complement to each other).

As we can tell from the D latch, it directly makes 2 complement inputs. Therefore, when Clk is

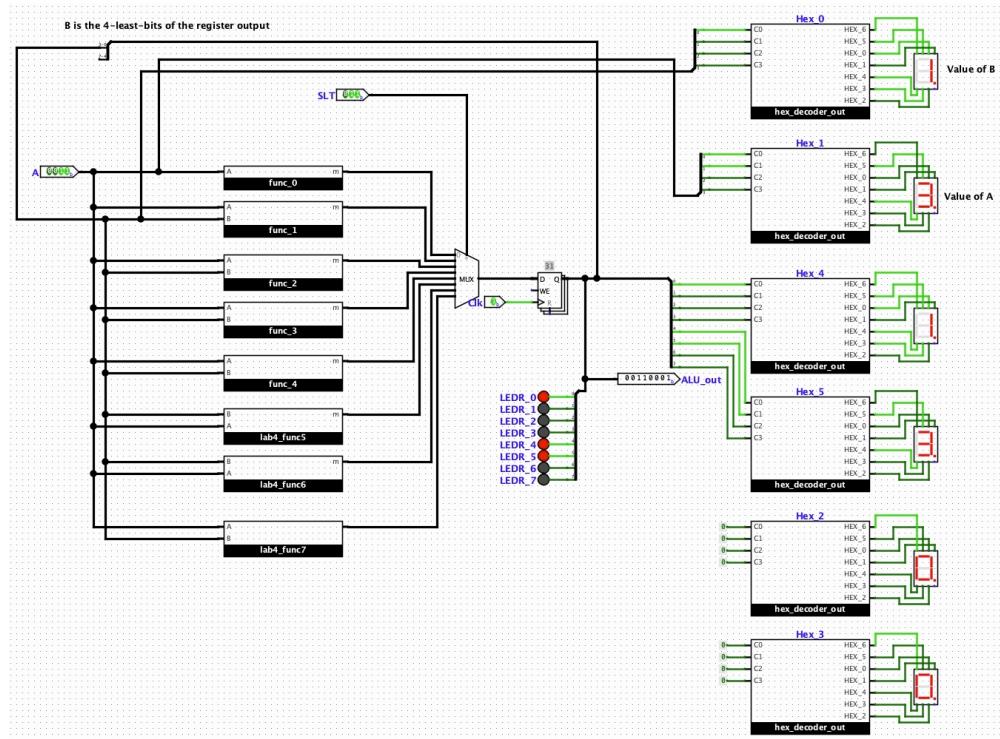
first set to 0, then both \bar{S} and \bar{R} are 1, then after both output value enter the 2nd NAND Gate, there would be an error. (Because NAND outputs 1 as long as there exists a '0' input, which means that if we have a 0 input for the NAND gate, we know the output would be 1, but if we only have a 1 input, we cannot anticipate the output values.)

Therefore $Ck=0$ cannot be first tested.

When $Ck=1$, regardless of what D value is, the outputs will automatically generate a complement (i.e. $[Qa=1 \text{ and } Qb=0]$ or $[Qa=0 \text{ and } Qb=1]$) Then, when Ck changes back to 0, the output values stores in Qa and Qb , respectively.

Part 2.

1.



2. (a). If there isn't a register, i.e B continuously is the four least-significant digits of the ALU output. As A changes, the ALU output changes immediately, where B changes immediately, which results in an infinite loop because B keeps changing.

(b). Assume A and B are both n-bit binary number, then the maximum value of them would be $2^n - 1$.

Therefore the largest value of their multiplication would be $(2^n - 1)^2$. which is:

$$\begin{aligned}(2^n - 1)^2 &= (2^n)^2 - 2 \cdot 2^n + 1 \\ &= 2^{2n} - 2^{n+1} + 1 \\ &< 2^{2n}\end{aligned}$$

Thus the number of bits we need to store a value would be $\log_2 2^{2n} = \underline{\underline{2n}}$.

3. (a)	A	SLT	CK.	ALU_out	B
func-0	0000	000	1	00000001	0001
func-1	0001	001	0	00000001	0001
	0001	001	1	00000010	0010
func-2	0001	010	0	00000010	0010
	0001	010	1	00000011	0011
func-3	1110	011	0	00000011	0011
	1110	011	1	111101	1101

Func-4	{	1110	100	0	1111101	1101
		1110	100	1	00000001	0001
Func-5	{	0001	101	0	00000001	0001
		0001	101	1	00000010	0010
Func-6	{	0001	110	0	00000010	0010
		0001	110	1	00000001	0001
	{	1111	111	0	00000001	0001
Func-7	{	1111	111	1	00001111	1111
		1111	111	0	00001111	1111
		1111	111	1	11100001	0001

(b). Tests for Lab4 functions:

func-5:

Logisim: Test Vector lab4_func5 of lab_4				
Passed: 8 Failed: 0				
status	A	B	m	
pass	0001	1111	0001	1110
pass	0010	1111	0011	1100
pass	0011	1111	0111	1000
pass	0100	1111	1111	0000
pass	0101	1111	1110	0000
pass	0110	1111	1100	0000
pass	0111	1111	1000	0000
pass	1000	1111	0000	0000

func-6:

Logisim: Test Vector lab4_func6 of lab_4

Passed: 8 Failed: 0

status	A	B	m
pass	0001	1111	0000 0111
pass	0010	1111	0000 0011
pass	0011	1111	0000 0001
pass	0100	1111	0000 0000
pass	0101	1111	0000 0000
pass	0110	1111	0000 0000
pass	0111	1111	0000 0000
pass	1000	1111	0000 0000

Load Vector Run Stop Reset Close Window

func-7.

Logisim: Test Vector lab4_func7 of lab_4

Passed: 5 Failed: 0

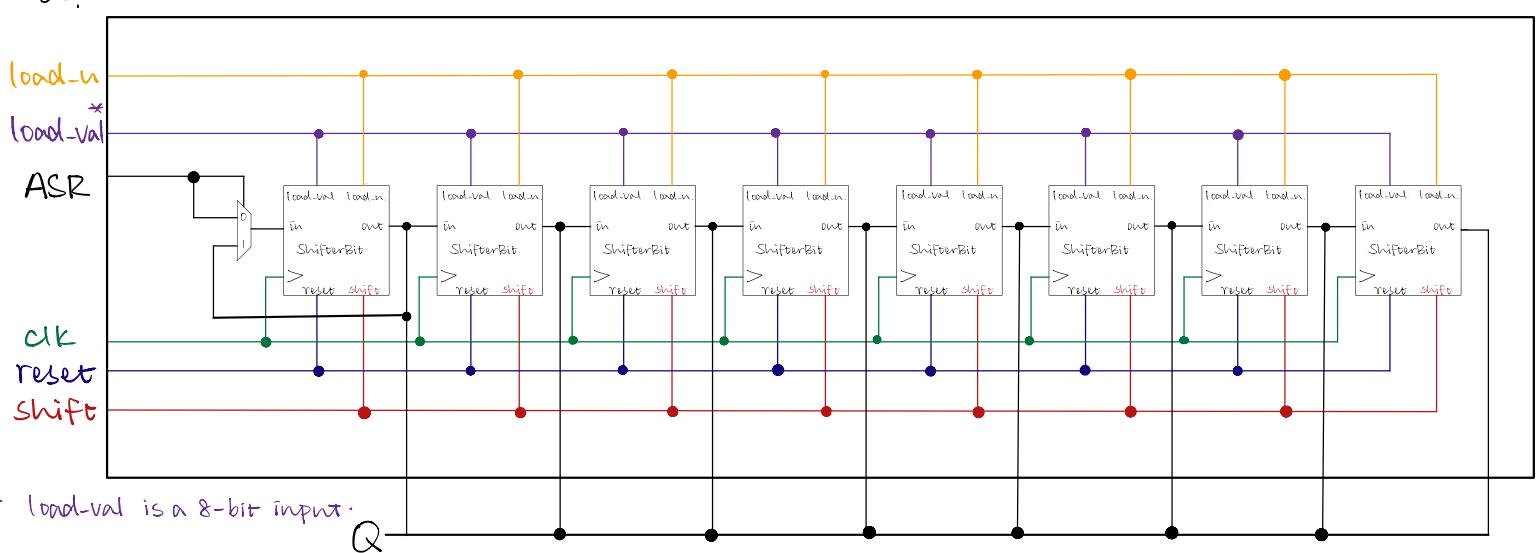
status	A	B	m
pass	0010	1111	0001 1110
pass	0100	0010	0000 1000
pass	1000	1110	0111 0000
pass	0001	1010	0000 1010
pass	1111	1111	1110 0001

Load Vector Run Stop Reset Close Window

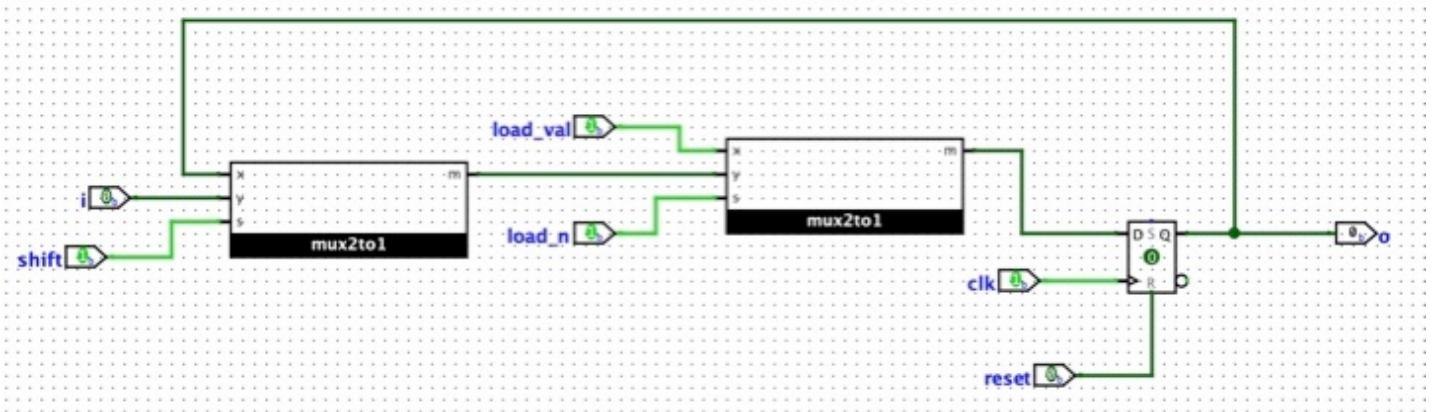
Part 3.

1. When Shift = 0. The output remains (no shifting).

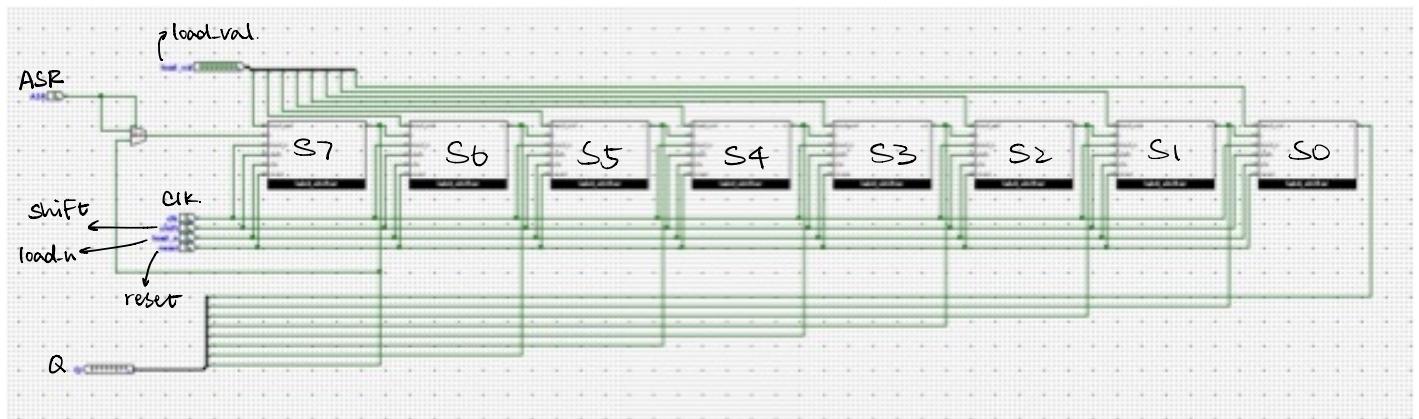
2.



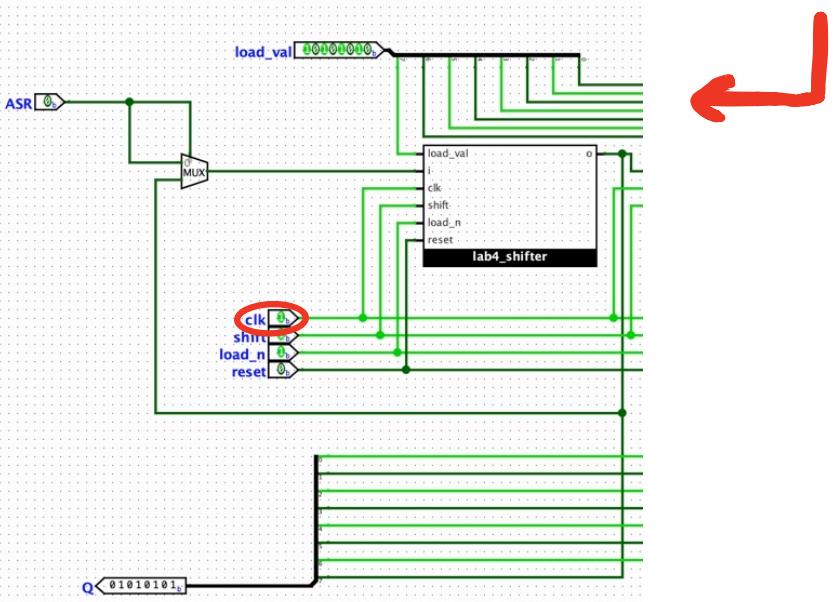
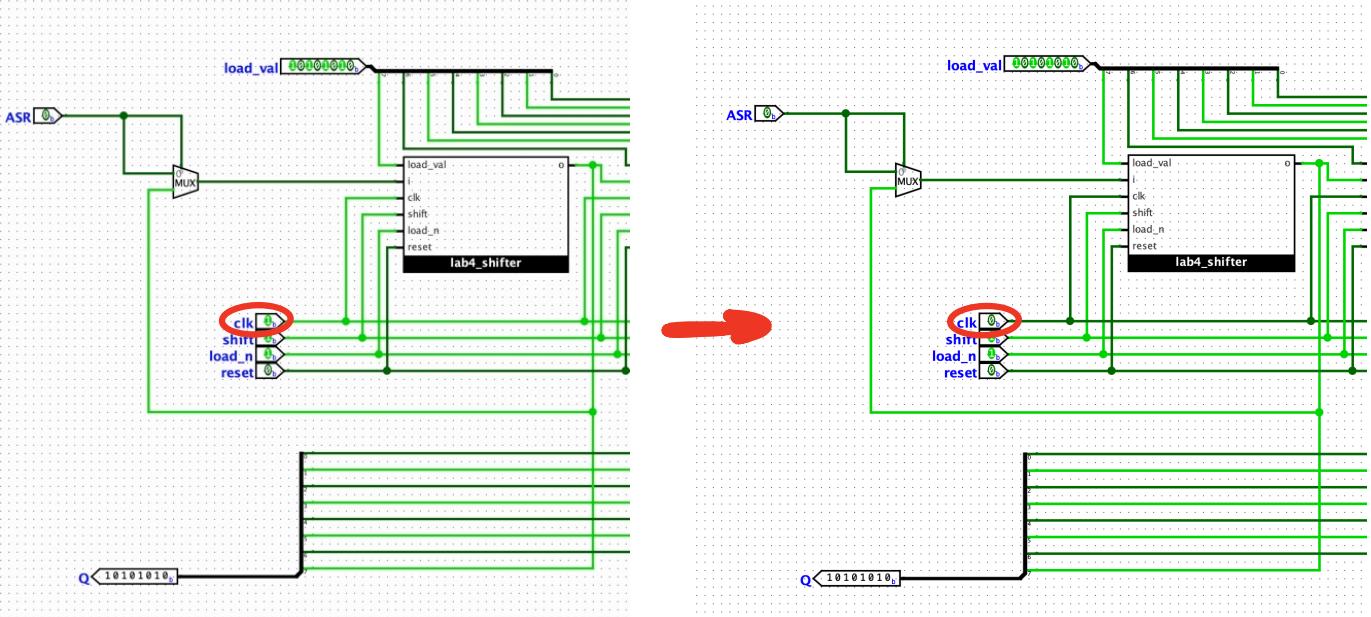
3. Single-bit shifter:



4.



5. Test where ASR=0.



After reset. Test where ASR=1.

