ESc201, Lecture 33: (Digital) Examples of DMUX

An input in BCD is to be displayed decimally over 7-segment displays. There would be 4-bit inputs (w, x, y, z) and from the DMUX there should be 10 outputs (4x10 DMUX). The Truth Table has to be written first.

w x y z	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
0 0 0 0	1	0	0	0	0	0	0	0	0	0
0 0 0 1	0	1	0	0	0	0	0	0	0	0
0 0 1 0	0	0	1	0	0	0	0	0	0	0
0 0 1 1	0	0	0	1	0	0	0	0	0	0
0 1 0 0	0	0	0	0	1	0	0	0	0	0
0 1 0 1	0	0	0	0	0	1	0	0	0	0
0 1 1 0	0	0	0	0	0	0	1	0	0	0
0 1 1 1	0	0	0	0	0	0	0	1	0	0
1 0 0 0	0	0	0	0	0	0	0	0	1	0
1 0 0 1	0	0	0	0	0	0	0	0	0	1
1 0 1 0	x	x	x	x	x	x	x	x	x	x
1 0 1 1	x	x	x	x	x	x	x	x	x	x
1 1 0 0	x	x	x	x	x	x	x	x	x	x
1 1 0 1	x	x	x	x	x	x	x	x	x	x
1 1 1 0	x	x	x	x	x	x	x	x	x	x
1 1 1 1	x	x	x	x	x	x	x	x	x	x

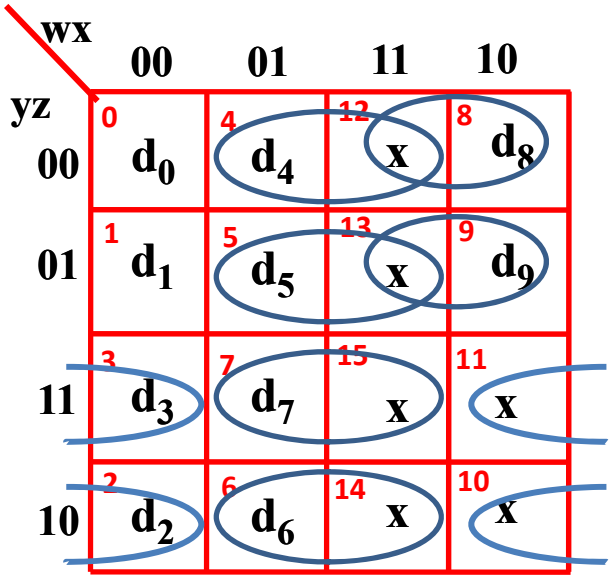
This has to be minimized now by writing 10 K-maps (for d_0 to d_9, but one can do it by just one K-map as this is to be implemented with a DMUX with a few don't care states.

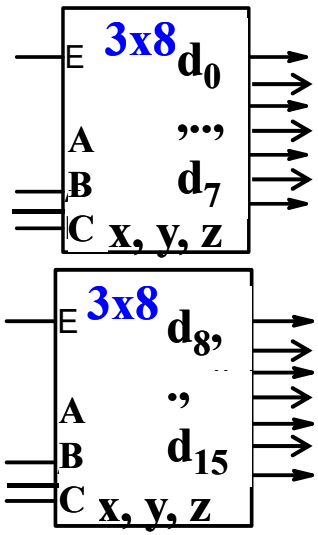
$$d_0 = \bar{w}.\bar{x}.\bar{y}.\bar{z}$$

$$d_1 = \bar{w}.\bar{x}.\bar{y}.z$$

$$d_5 = x.\bar{y}.z$$

$$d_9 = w.y.z$$

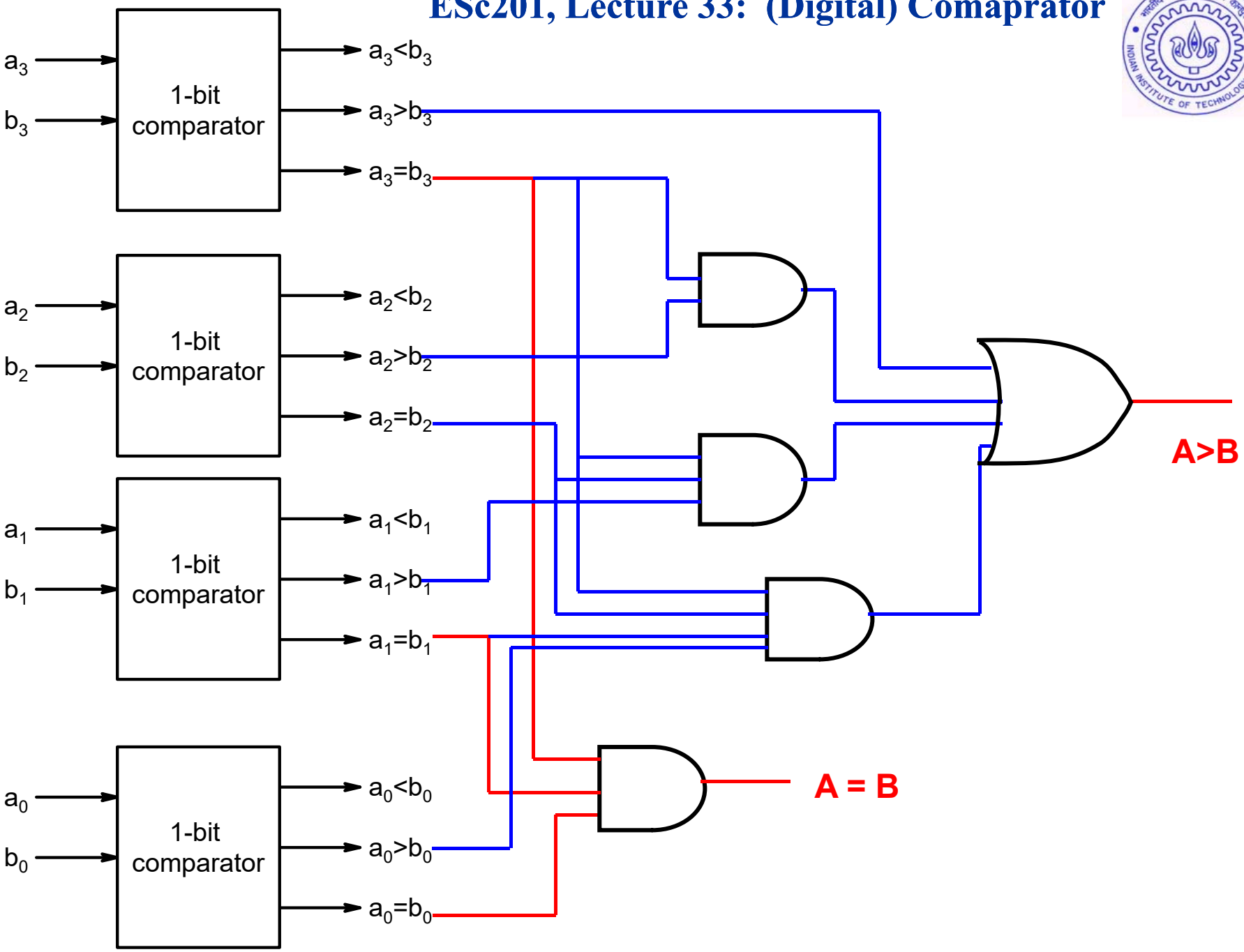




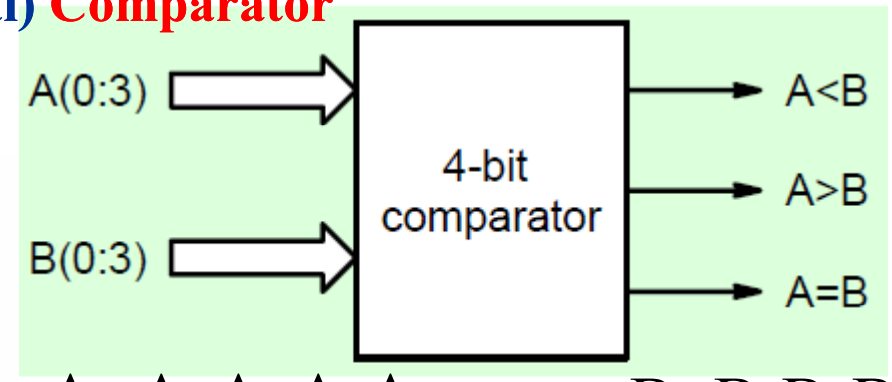
This can be implemented with two 3x8 DMUX. The input to the 1st is controlled by inputs x, y, z where w=0, and the other input as 1 with E=1. The second is also controlled by x, y, z where w=1, E=1 when w=1.

The 0-9 outputs can each be used by to run a seven segment display by a MUX.

ESc201, Lecture 33: (Digital) Comaprator



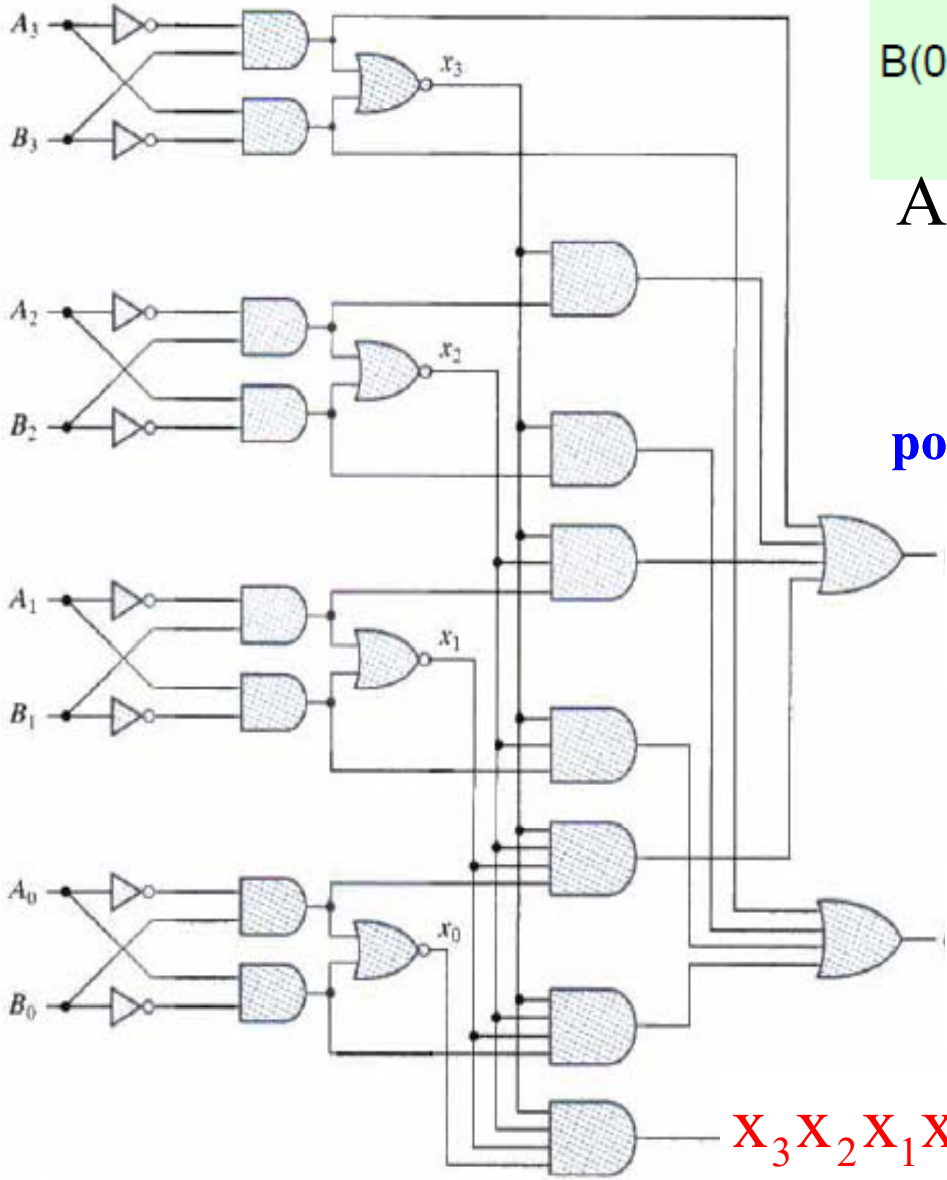
ESc201, Lecture 33: (Digital) Comparator



$A=A_3A_2A_1A_0$ $B=B_3B_2B_1B_0$

$x_i = A_i.B_i + \overline{A_i}.\overline{B_i}$ for i=0,1,2,3

Where x_i = 1 only if the pair of bits in position i are equal. (i.e., if both are 1 or 0).



$\overline{A_3}B_3 + x_3\overline{A_2}B_2 + x_3x_2\overline{A_1}B_1 + x_3x_2x_1\overline{A_0}B_0$

For (A<B)

$A_3\overline{B_3} + x_3A_2\overline{B_2} + x_3x_2A_1\overline{B_1} + x_3x_2x_1A_0\overline{B_0}$

For (A>B)

$x_3x_2x_1x_0$ For (A=B)

All x_i variable must be equal to 1

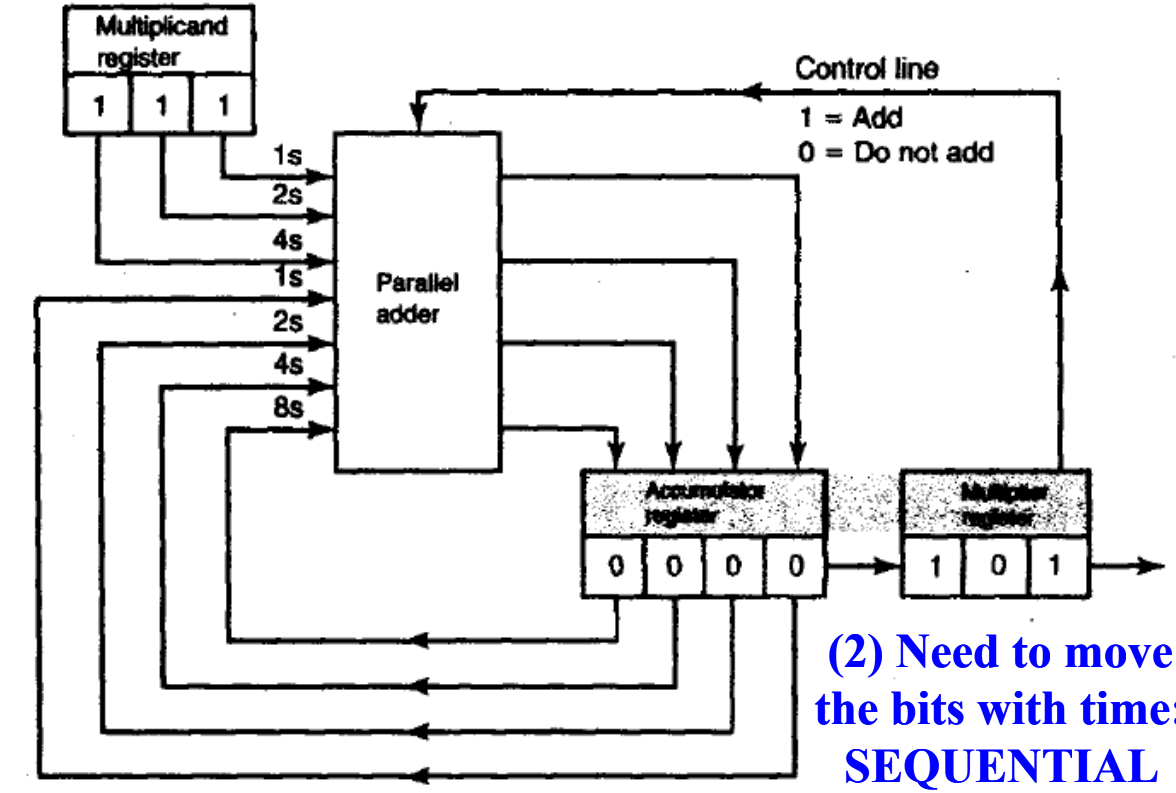
ESc201, Lecture 33: (Digital)

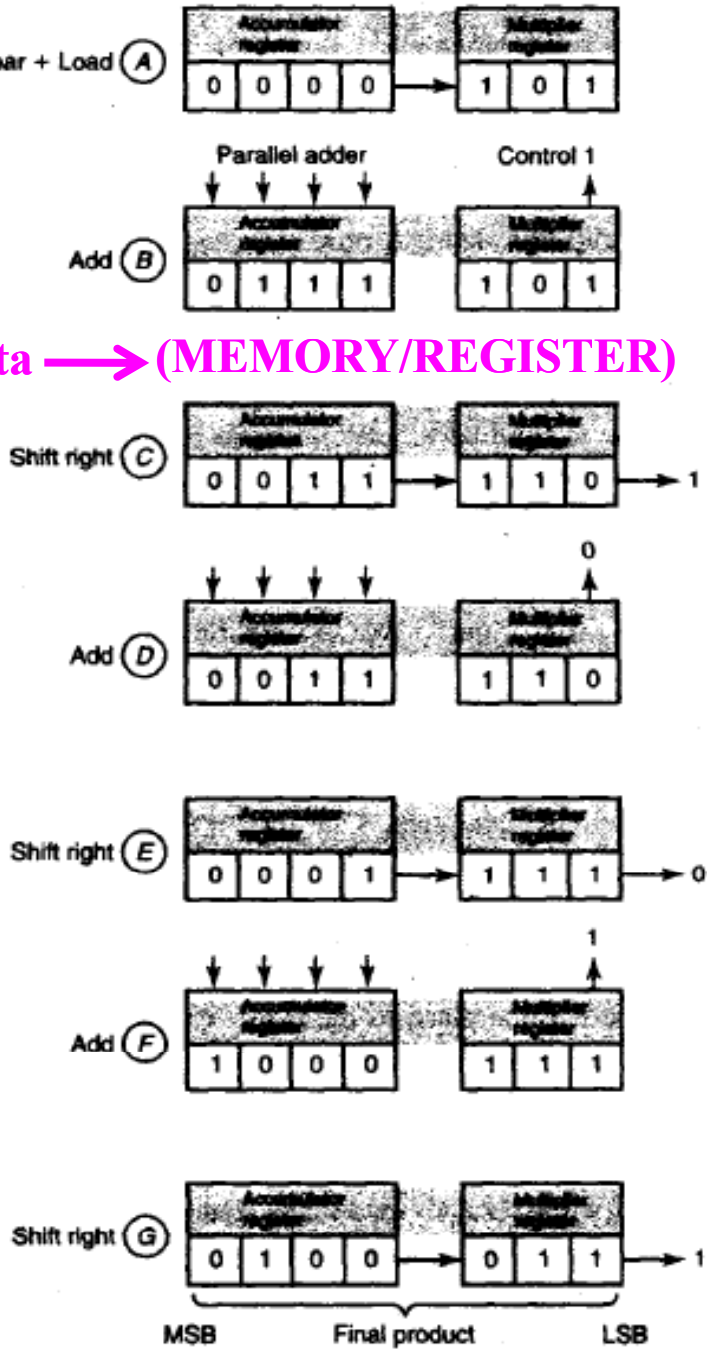
For 4-bit add 4-Stages and 12 FAs required.

Multiplication using register

			a_3	a_2	a_1	a_0	Mulitiplicand
			b_3	b_2	b_1	b_0	Multiplier
		p_{30}	p_{20}	p_{10}	p_{00}	
	p_{31}	p_{21}	p_{11}	p_{01}		Partial product
p_{32}	p_{22}	p_{12}	p_{02}			
p_{33}	p_{23}	p_{13}	p_{03}			
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	Product

(1) Need to store the data → (MEMORY/REGISTER) as they are generated.





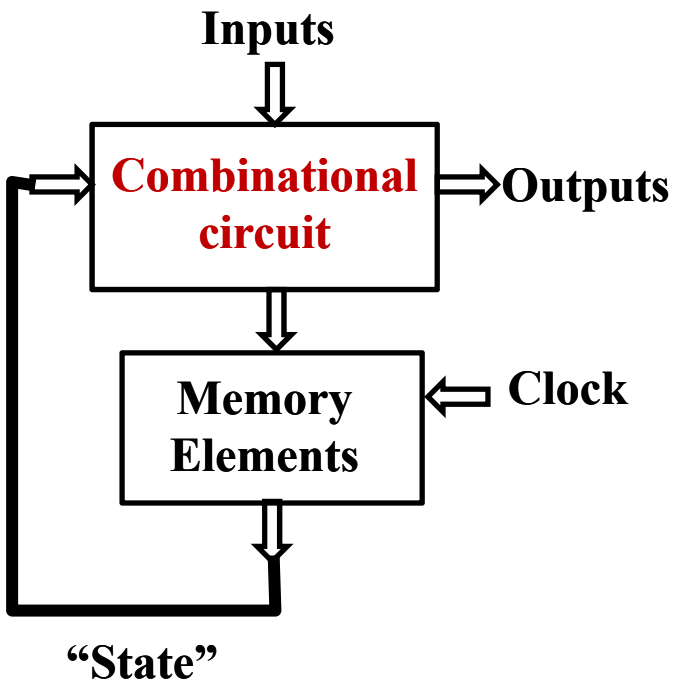
(2) Need to move the bits with time: SEQUENTIAL Processing

(Need someone to maintain the sequence: → Clock)

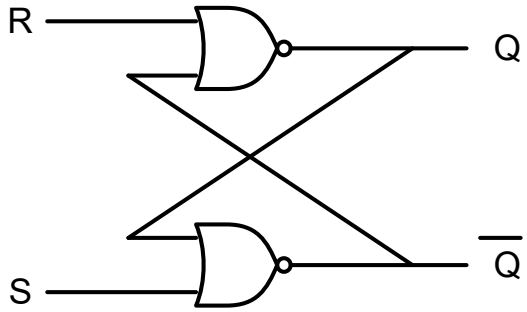
ESc201, Lecture 33: (Sequential logic)

Sequential Circuits Need Memory to Store the Previous State Information.

1. Sequential circuit consists of combinational circuit to which memory elements are connected to form a feedback path.
1. The binary information stored in the memory elements at any given time defines the 'state' of the sequential circuit.
2. The external outputs in a sequential circuit are a function of external inputs and the present state of the memory element
3. Next state of the memory elements is also a function of external inputs and the present state
4. There are two main types of sequential circuits.
5. A 'Synchronous' sequential circuit is a system whose behavior can be defined from the knowledge of its signals at discrete instance of time.
6. The behavior of an 'Asynchronous' sequential circuit depends upon the order in which its input signals change and can be affected at ant instant of time.
7. In synchronous sequential logic systems synchronization is achieved by a timing device called a master-clock generator, Which generates a periodic train of 'clock pulses'.
8. The memory elements change state in synchronization with the clock pulses.



Flip Flop is one kind of a memory element.



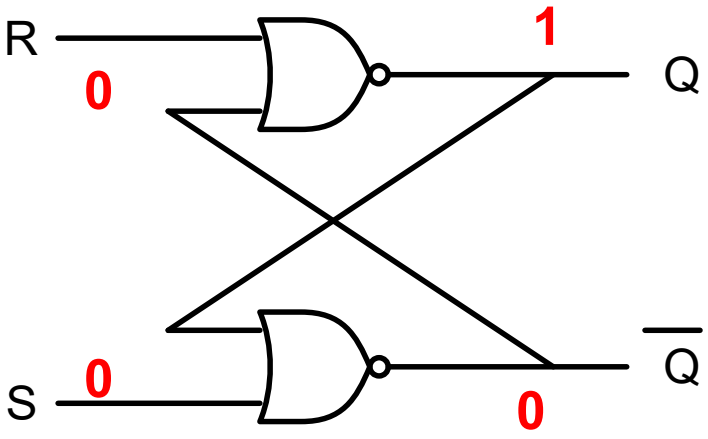
NOR SR Latch/Flip-Flop

$Q=1; \overline{Q}=0$ Set State

$Q=0; \overline{Q}=1$ Reset State

ESc201, Lecture 33: Digital (Memory using Combinational Logic)

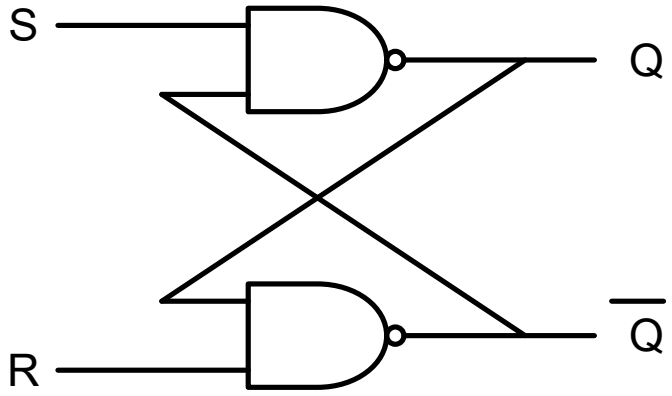
S-R Flip-Flop



S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	0	1	0	HOLD
0	1	0	1	RESET
0	0	0	1	HOLD

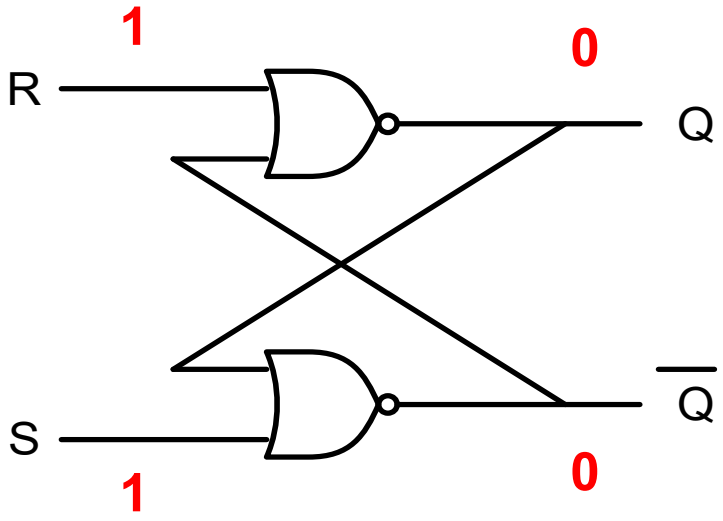
S	R	Q	$\overline{Q}$	State
1	0	1	0	SET
0	1	0	1	RESET
0	0	Q	$\overline{Q}$	HOLD
1	1	0	0	INVALID

NAND SR Flip-Flop/Latch



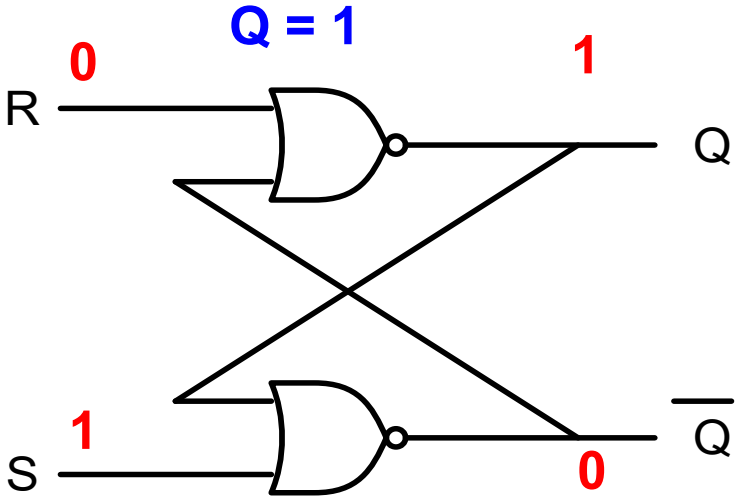
S	R	Q	$\overline{Q}$	State
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q	$\overline{Q}$	HOLD
0	0	1	1	INVALID

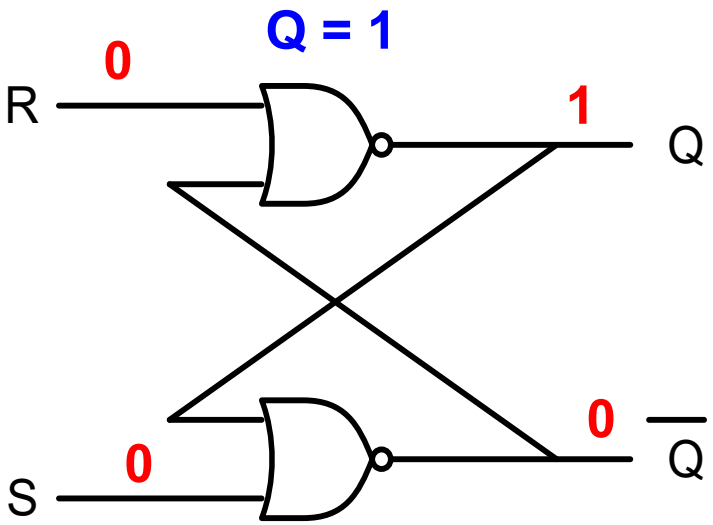
ESc201, Lecture 33: Digital : SR Flip-Flop/Latch—Invalid inputs



Both the outputs are well defined and 0. The first problem is that the output is not complementary.

A more serious problem occurs when S/R switches the latch to the hold state by changing RS from 11 → 00 . Suppose the inputs do not change simultaneously and the situation 11 → 01* → 00 arises, because the R-input has changed faster than S. So the 00 input which should have held 00 at the output is now set at 10.





To avoid this situation, a clock (Ck) can be used at the input, such that the inputs are not allowed to the NOR/NAND gates till it is ensured by the Ck that the inputs are stable.