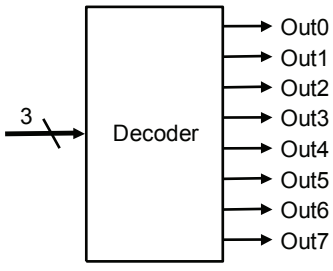
Decoders

• A full decoder with n inputs has 2^n outputs. Let inputs be labeled In_0, In_1, In_2,..., In_{n-1}, and let outputs be labeled Out_0, Out_1,..., Out_{2^n-1}.

• A full decoder functions as follows: Only one of outputs has value 1 (it is active) while all other outputs have value 0. The only output set to 1 is one labeled with the decimal value equal to the (binary) value on input lines.

• In general, a decoder with n inputs may have fewer than 2^n outputs. Sometime those are called partial decoders.

3-Input Full Decoder



Input			Output							
I2	I1	I0	O7	O6	O5	O4	O3	O2	O1	O0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Figure B.3.1

Multiplexers

• A basic multiplexer has only one output line z. There are two sets of input lines: data lines and select lines.

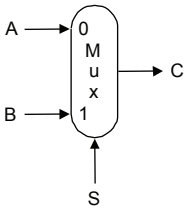
• Let a number of data lines be N, labeled d_0, d_1, d_2,... d_{N-1}. There are m select lines, labeled s_0, s_1,..., s_{m-1}. m is such that any of data lines can be referenced (selected) by a decimal value on select lines. Thus, m has to satisfy the following inequality: $2^{m-1} < N \leq 2^m$.

• A multiplexer functions as follows: Output z has the value of the data input line labeled by a decimal value equal to a (binary) value on select lines.

Simplest Basic Multiplexer

Simplest mux is one with 2 data input lines and 1 select line.

Symbol



Truth Table

A	B	S	C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Design

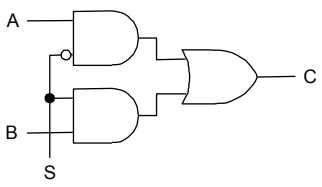


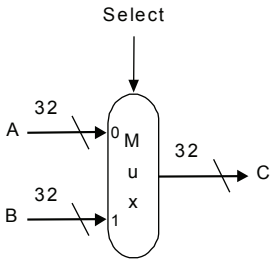
Figure B.3.2

3-Data Multiplexer Truth Table

do	d1	d2	s1	s0	z	
0	0	0	0	0	0	
0	0	0	0	1	0	
0	0	0	1	0	0	
0	0	0	1	1	d	This input not allowed
0	0	1	0	0	0	
0	0	1	0	1	0	
0	0	1	1	0	1	
0	0	1	1	1	d	This input not allowed
0	1	0	0	0	0	
0	1	0	0	1	1	
0	1	0	1	0	0	
0	1	0	1	1	d	This input not allowed
0	1	1	0	0	0	
-	-	-	-	-	-	
1	1	1	1	0	1	
1	1	1	1	1	d	

Complex Multiplexer

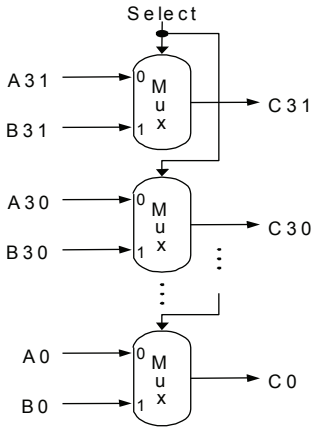
- Instead N single data lines and one output line as in a basic mux, a complex mux has N sets of data lines and one set of output lines and each set has K lines.
- No changes with select lines.



N=2, K=32

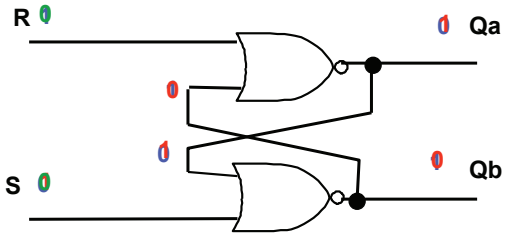
Figure B.3.6

Design of mux on left at basic mux level



R-S Latch: Simplest Sequential Circuit

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0



1. Let us start with: S = 0 & R = 1 → Qa = 0 & Qb = 1
2. Let us now change R to 0: S = 0 & R = 0 → Qa = 0 & Qb = 1 → no change
3. Let us now change S to 1: S = 1 & R = 0 → Qa = 1 & Qb = 0
4. Let us now change S to 0: S = 0 & R = 0 → Qa = 1 & Qb = 0 → no change

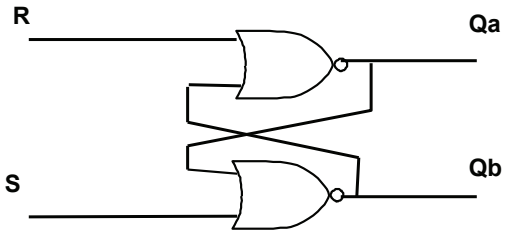
Thus, for steps 2 and 4 inputs are identical while outputs are different, i.e. we have a sequential circuit.

Study: B.7 & B.8

R-S Latch Characteristics

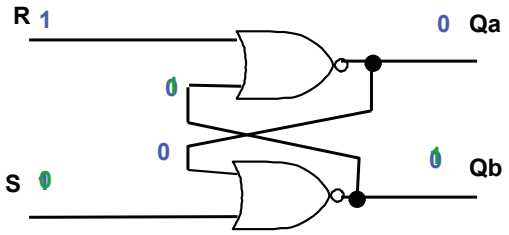
R-S latch is a memory element that "remembers" which of two inputs has most recently had value 1:

- Outputs Qa =1 & Qb = 0 indicate that S is currently or was 1 last
- Outputs Qa =0 & Qb = 1 indicate that R is currently or was 1 last



R-S Latch Characteristics (continued)

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0



5. Let us consider case: S = 1 & R = 1 → Qa = 0 & Qb = 0

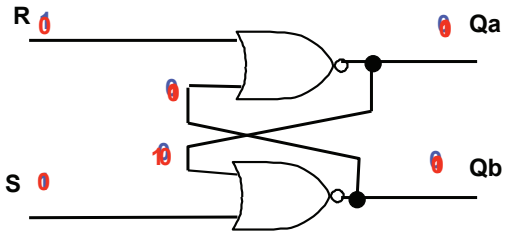
Qa = 0 & Qb = 0 indicate that currently S = 1 and R = 1

6a. Let us now change S to 0: S = 0 & R = 1 →Qa = 0 & Qb = 1,

→Identical to Step 1 on Slide 21

R-S Latch Characteristics (continued)

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0



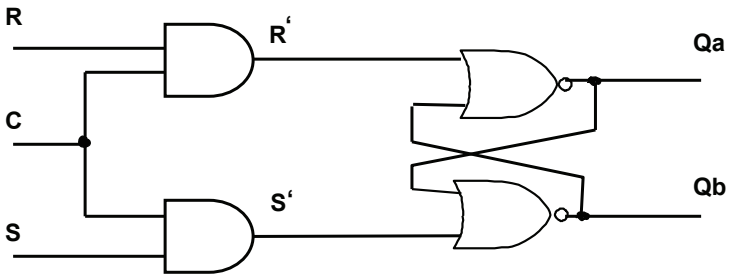
5. Let us again consider case: S = 1 & R = 1 → Qa = 0 & Qb = 0

6b. Let us now change S and R simultaneously to 0: S = 0 & R = 0

→ Unstable state

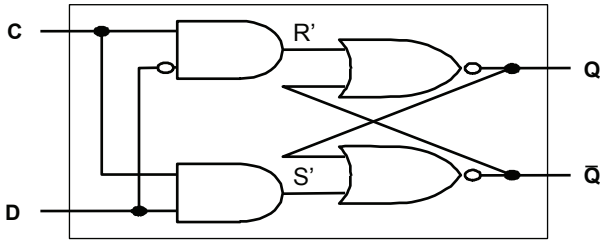
Note: Set=Reset=1 at the same time should be avoided

Gated (Clocked) R-S Latch



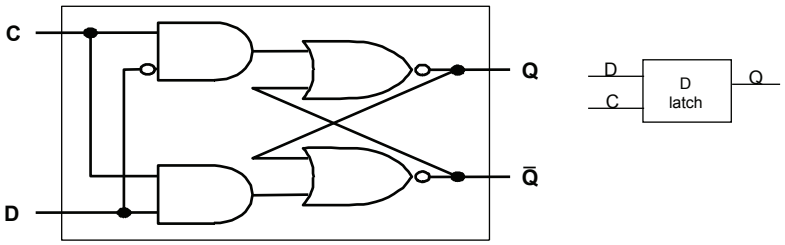
- C input is write enable
- When C = 1, a gated R-S latch behaves as an ordinary R-S latch, since R'=R and S'=S.
- When C = 0, changes in R and S do not influence outputs.
- Note that the case R'=1 & S'=1 is still possible, and then an unstable state can be reached easily. How?

(Gated) D-Latch



- Two inputs:
  - the data value to be stored (D)
  - the write enable signal (C) indicating when to read & store D
- Two outputs:
  - the value of the internal state (Q) and it's complement (often unused)
- Note: The case R'=1 & S'=1 is not possible.

D-Latch Functioning

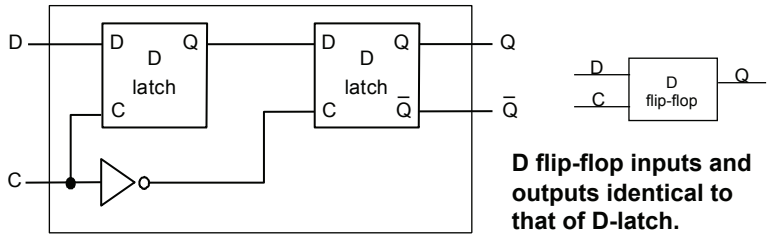


- *D-latch functions as follows:*

– when C=1, D-latch state (and Q-output) is identical to D-input, i.e. any change in the value of D-input is 'immediately 'followed by the change of Q-output.

– When C=0, D-latch state is unchanged and it keeps the value it had at the time when C input changed from 1 to 0.

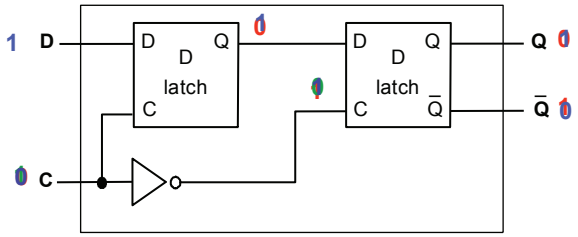
D Flip-Flop



D flip-flop inputs and outputs identical to that of D-latch.

- Two inputs:
  - the data value to be stored (D)
  - the write enable signal (C) indicating when to read & store D
- Two outputs:
  - the value of the internal state (Q) and it's complement (often unused)

D Flip-Flop Functioning



- *D-flip-flop functions as follows:*
  - When C changes its value from 1 to 0, i.e. on the falling edge, D-flip flop state (and Q output) gets the value D-input has at that moment,
  - During all other times, D-flip flop state is unchanged and it keeps the value it had at the time of the falling edge of C-input.

Set-up and Hold Time

- **There is critical period Tcr around the falling edge of C during which value on D should not change. Tcr is split into two parts, the setup time before the C edge, and the hold time after the C edge.**
- **The set-up time is the minimum time the D signal must be stable before the C signal falling edge .**
- **The hold time is the minimum time the D signal must be stable after C signal falling edge.**

Figure B.8.6

- **Failure to meet these minimum requirements can result in a situation where the output of the flip-flop may not be predicable.**
- **Note that hold times are usually either 0 or very small and thus not a major concern.**

Register File Design

- Register file = set of registers that can be read/written by supplying a register number
- D flip-flop - building block
- Typical register file has:
  - 2 read ports
  - 1 write port
- Read operation:
  - supply reg# at input
  - output on Read data line
- Write operation:
  - select reg#
  - set write data signal