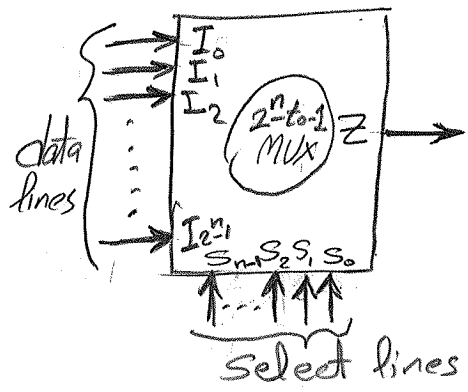
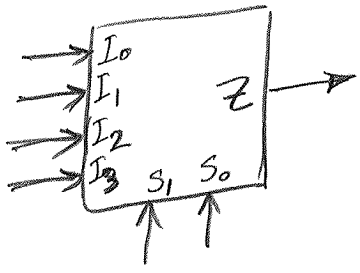
Multiplexers: (Data Selectors)

* Commonly called **2^n-to-1 MUX**

* purpose: Receives **2^n bits of data** and an **n-bit binary code** and selects the data bit whose position is determined by the code (on the select lines), and passes it along to the output Z.



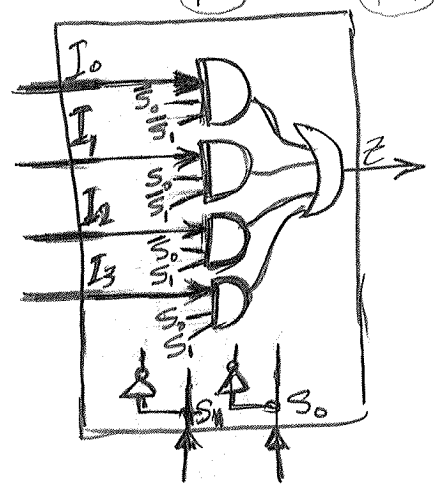
* Example: 4-to-1 Mux



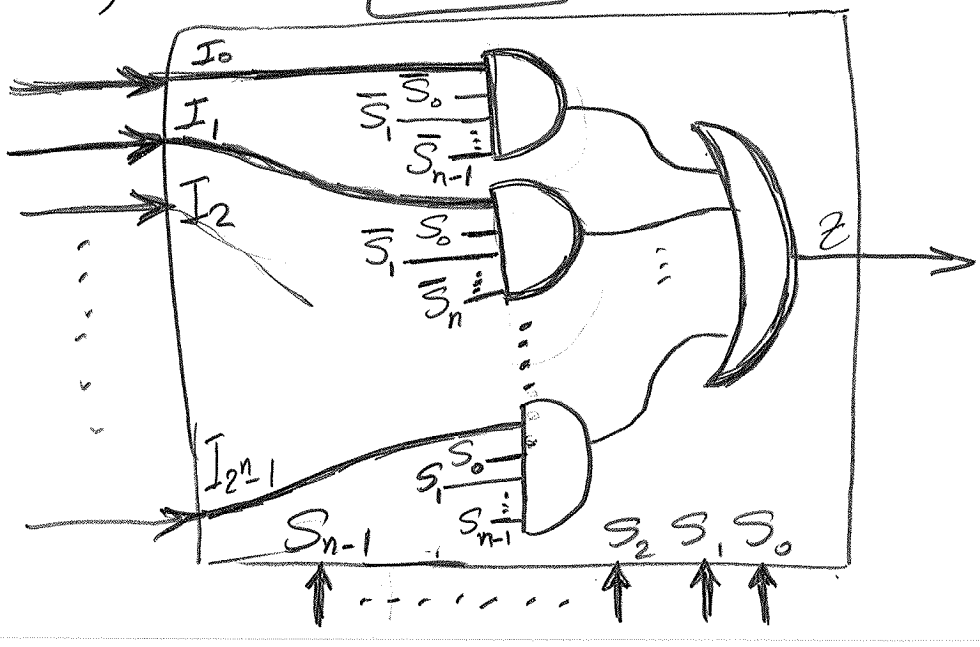
I_3	I_2	I_1	I_0	S_1	S_0	Z	
				0	0	1	$\rightarrow I_0 \bar{S}_1 \bar{S}_0$
				0	1	1	$\rightarrow I_1 \bar{S}_1 S_0$
				1	0	1	$\rightarrow I_2 S_1 \bar{S}_0$
				1	1	1	$\rightarrow I_3 S_1 S_0$

$$Z = I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0$$

(00) (01) (10) (11)

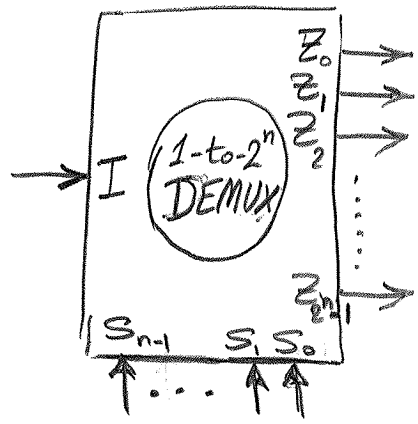


* To build a 2^n-to-1 Mux, we need **2^n AND** gates (each one has n+1 inputs) and **1 OR** gate with 2^n inputs

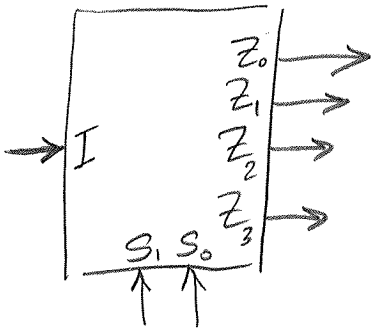


Demultiplexers:

* Commonly called **1-to-2^n DEMUX**
* purpose: Receives **1 bit of data** and an **n-bit binary code**, and directs the data to the output whose position equals the code (located on the select lines)



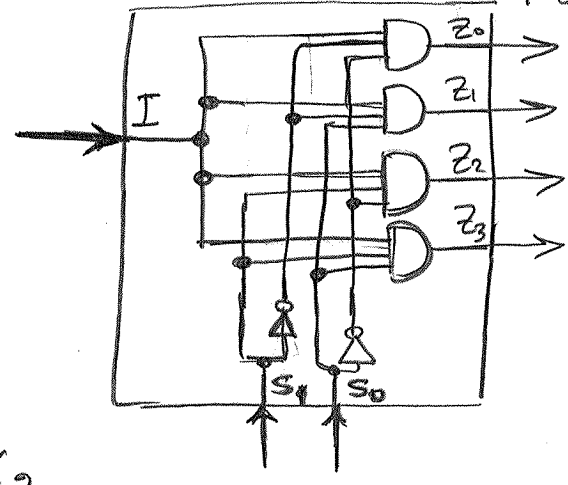
* Example: 1-to-4 DEMUX



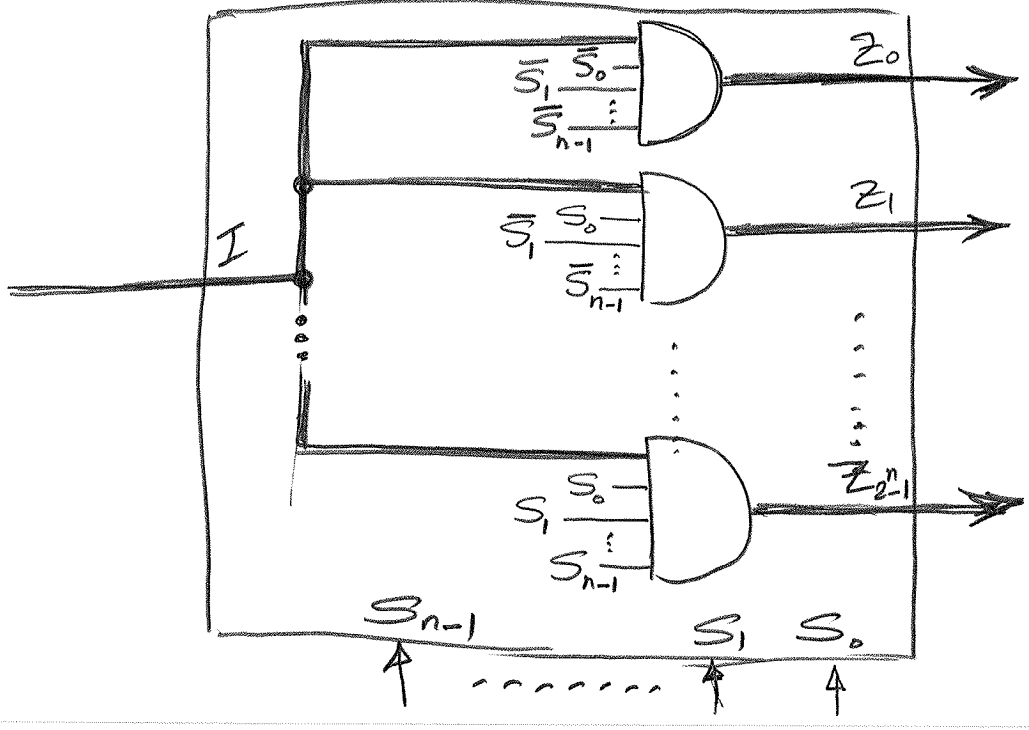
I	S_1	S_0	Z_0	Z_1	Z_2	Z_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$Z_0 = I\bar{S}_1\bar{S}_0$, $Z_1 = I\bar{S}_1 S_0$

$Z_2 = I S_1\bar{S}_0$, $Z_3 = I S_1 S_0$



* To build a 1-to-2^n DEMUX, we need 2^n AND gates (each one has n+1 inputs).



Sequential Logic: (Latches, Flip-flops, ...etc.)

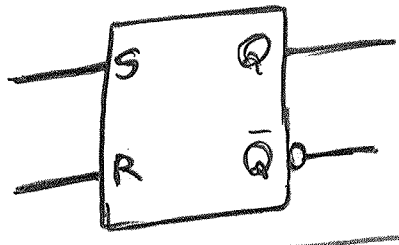
1 Latches:

* A latch is a temporary storage device that has two logic states → bistable
* Types: ① S-R (SET-RESET) Latch → Active-high / Active-low
  ② Gated S-R Latch
  ③ Gated D Latch

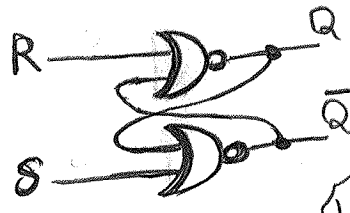
① S-R (SET-RESET) Latch

Active High

Logic Symbol



Logic Circuit

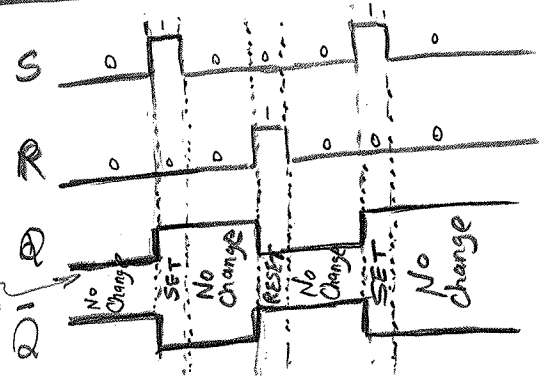


Truth Table

S	R	Q	$\bar{Q}$	
0	0	NC	NC	← No Change
0	1	0	1	← Latch RESET
1	0	1	0	← Latch SET
1	1	0	0	← Invalid Condition (Should be avoided!!)

Latch remains in present state Q and $\bar{Q}$ don't change

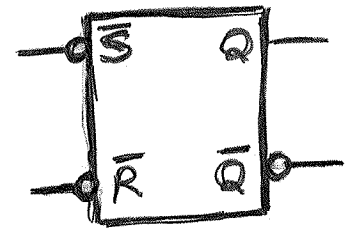
Timing Diagram



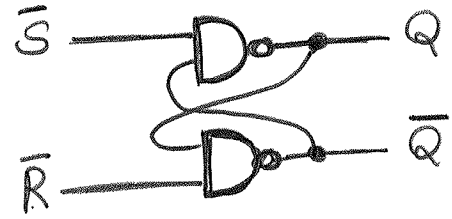
Assuming initially Q=0

Active Low

Logic Symbol



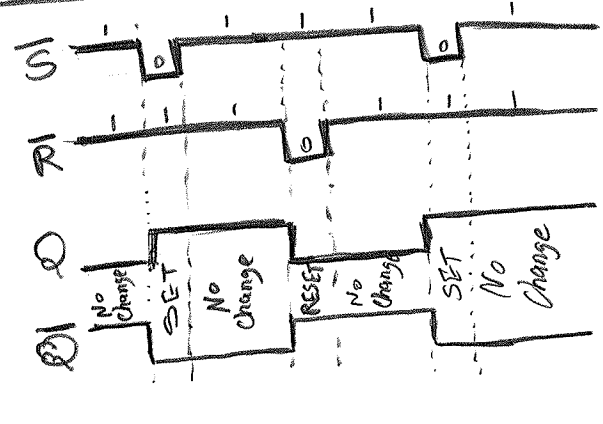
Logic Circuit



Truth Table

$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	
0	0	1	1	← Invalid Condition
0	1	1	0	← Latch SET
1	0	0	1	← Latch RESET
1	1	NC	NC	← No Change

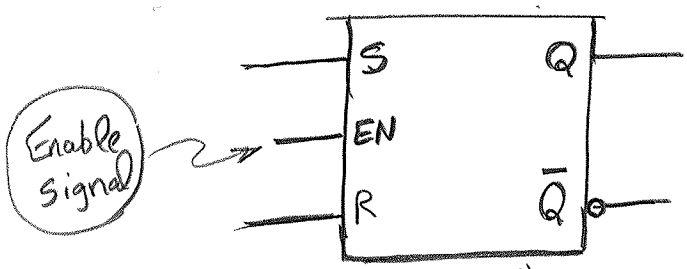
Timing Diagram



② Gated S-R Latch

We will consider only the active-high version which is built using an active-low S-R Latch

* Logic symbol *



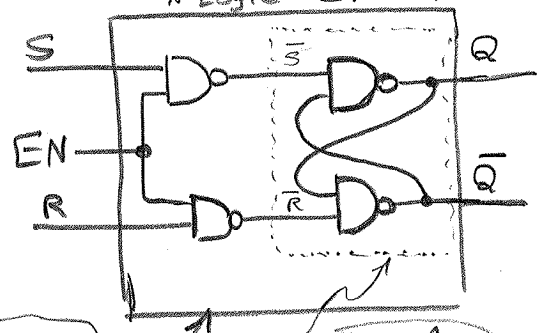
* Truth Table *

EN	S	R	Q	$\bar{Q}$	
0	0	0	NC	NC	No change
0	0	1	NC	NC	
0	1	0	NC	NC	
0	1	1	NC	NC	
1	0	0	NC	NC	← No change
1	0	1	0	1	← RESET
1	1	0	1	0	← SET
1	1	1	0	0	← Invalid

Latch is disabled

Latch is enabled

* Logic circuit *

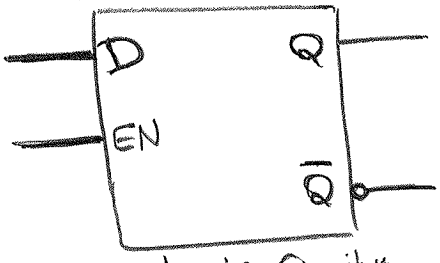


Active-high Gated S-R Latch

Active-low S-R Latch

③ Gated D Latch

* Logic Symbol *



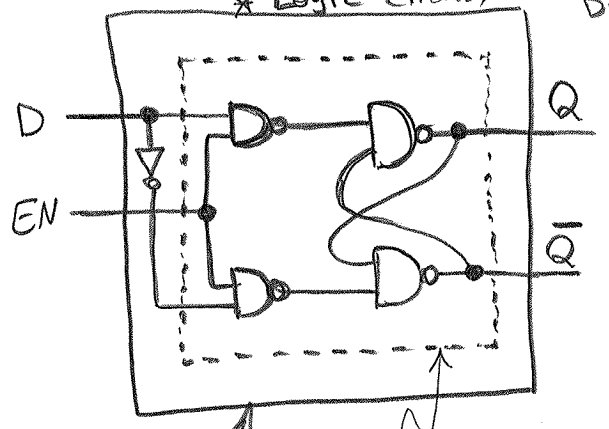
* Truth Table *

EN	D	Q	$\bar{Q}$	
0	0	NC	NC	No Change
0	1	NC	NC	
1	0	0	1	← RESET
1	1	1	0	← SET

Latch is Disabled

Latch is enabled

* Logic Circuit *



D-Latch

Gated S-R Latch

* Timing Diagram *