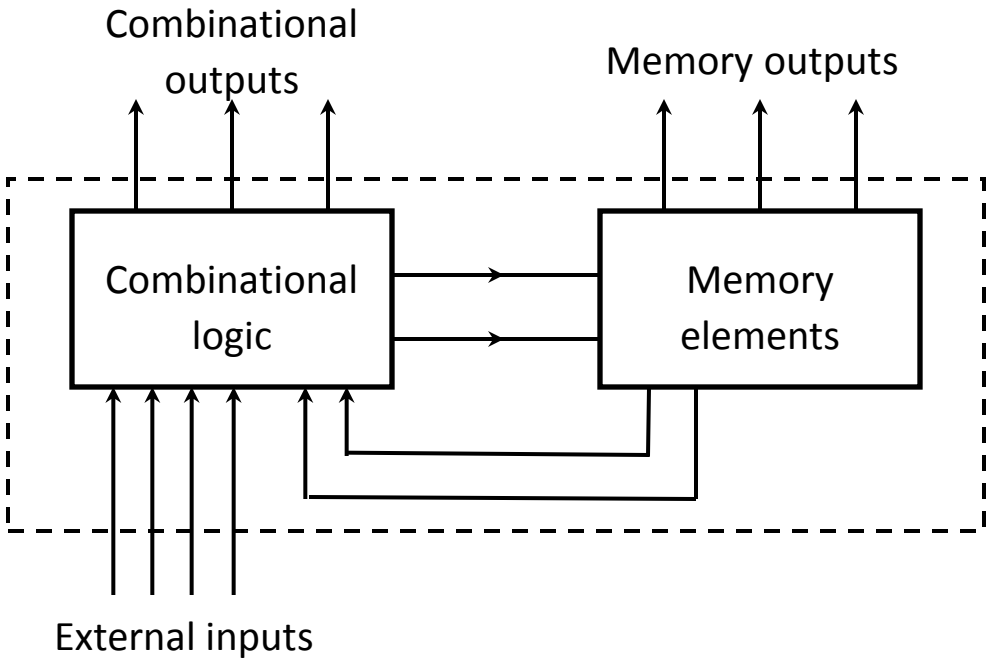
Unit 3
Combinational MOS Logic Circuits

Introduction

- A **sequential circuit** consists of a *feedback path*, and employs some *memory elements*.



Sequential circuit = Combinational logic + Memory Elements

Sequential Circuits

- Definitions
- Latches
  - Set-Reset – SR
  - The D Latch

- Material from section 5-1 and 5-2 of text

Combination Logic

- Logic that performs some transformation operation on the inputs to produce outputs which are simple logic functions of the input.
- The outputs reflect a function of the current values on the inputs.

- There is not the capability to hold the value of the input.

Sequential Logic

- Logic elements capable of storing a logic value.
- Sequential circuits are those circuits that employ these elements.

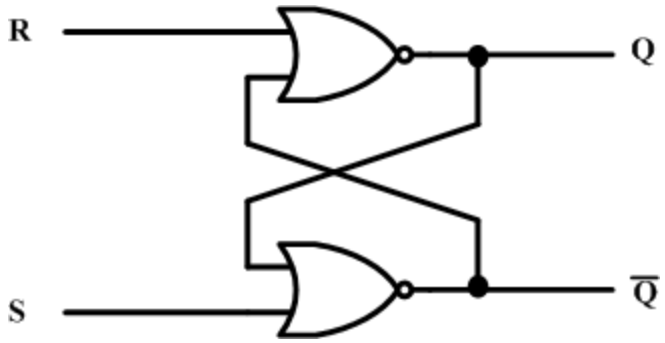
- Will be looking at the methodologies for sequential circuit design.

Definitions

- **Inputs** – All the outside logic signal inputs to the circuit. Typically, the clock is not consider part of the signal inputs of the circuit.
- **Outputs** – The logic signal outputs.
- **Present State** – the logic value of all the state variables of the system. These are stored in the state memory.
- **Next State** – Given the present state and the current values on the inputs, the next state represents the next logic state the circuit will transition to on the next clock.

The SR Latch

- The SR (Set-Reset) Latch

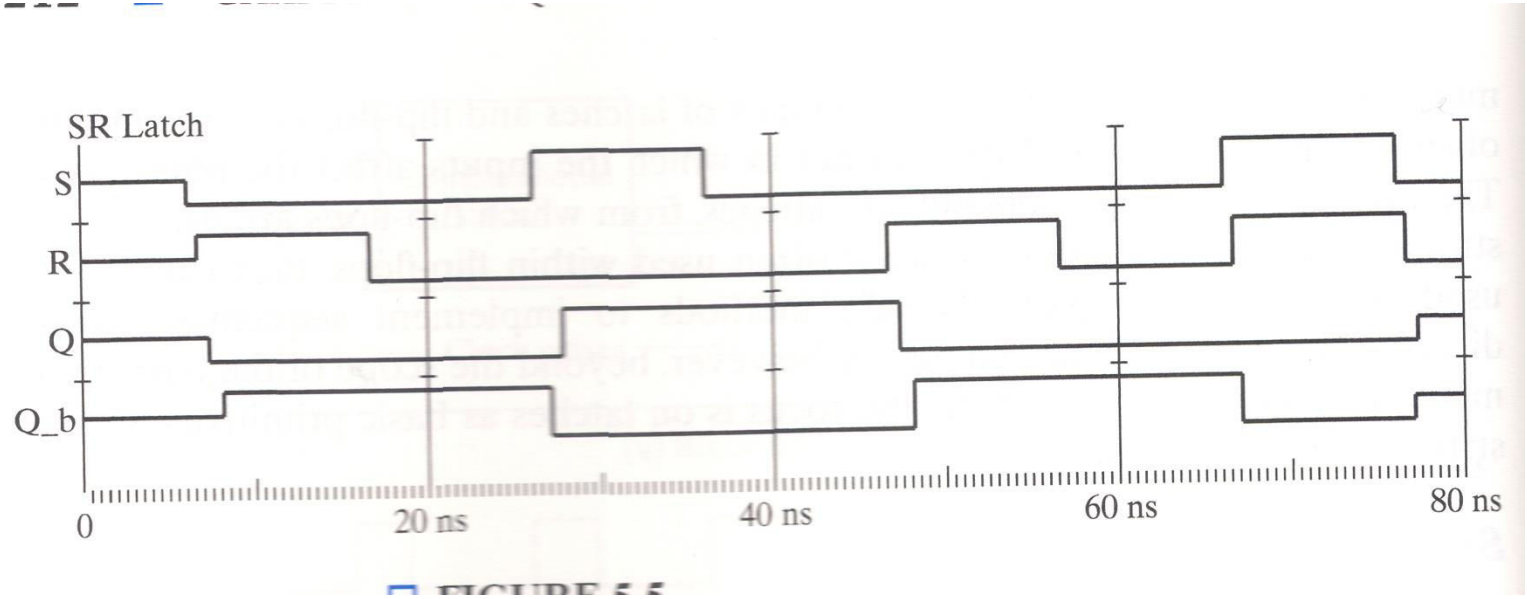


- And its operation

S	R	Q	Q′	
1	0	1	0	Set Q
0	0	1	0	Hold
0	1	0	1	Clear
0	0	0	1	Hold
1	1	0	0	Undefined

The waveform

- A simulation waveform would look something like this



S-R Latch

- *Complementary* outputs: *Q* and *Q'*.
- When *Q* is HIGH, the latch is in *SET* state.
- When *Q* is LOW, the latch is in *RESET* state.
- For *active-HIGH input* S-R latch (also known as NOR gate latch),

  R=HIGH (and *S*=LOW) ⇨ RESET state
  S=HIGH (and *R*=LOW) ⇨ SET state
  both inputs LOW ⇨ no change
  both inputs HIGH ⇨ *Q* and *Q'* both LOW (invalid)!

S-R Latch

- For *active-LOW input* S'-R' latch (also known as NAND gate latch),

  R′=LOW (and *S′*=HIGH) ⇨ RESET state

  S′=LOW (and *R′*=HIGH) ⇨ SET state

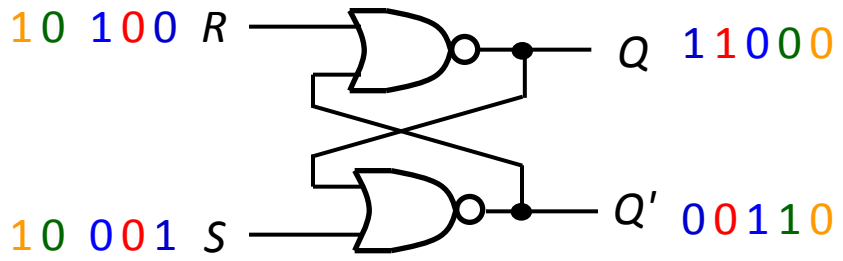
  both inputs HIGH ⇨ no change

  both inputs LOW ⇨ *Q* and *Q′* both HIGH (invalid)!

- Drawback of S-R latch: invalid condition exists and must be avoided.

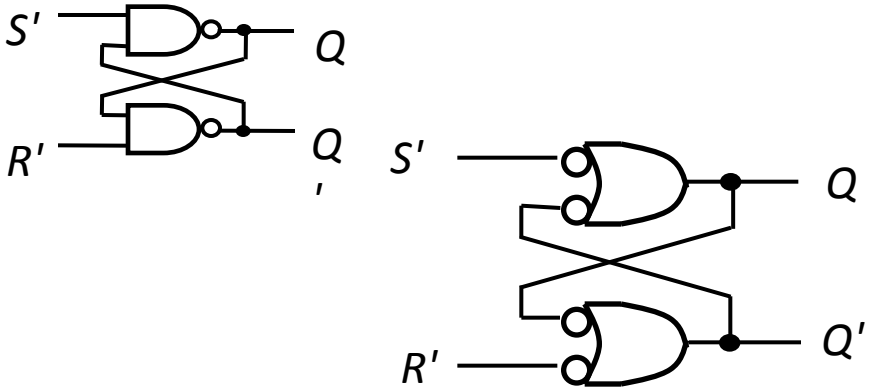
S-R Latch

- Active-HIGH input S-R latch



S	R	Q	Q′	
1	0	1	0	initial
0	0	1	0	(afer S=1, R=0)
0	1	0	1	
0	0	0	1	(after S=0, R=1)
1	1	0	0	invalid!

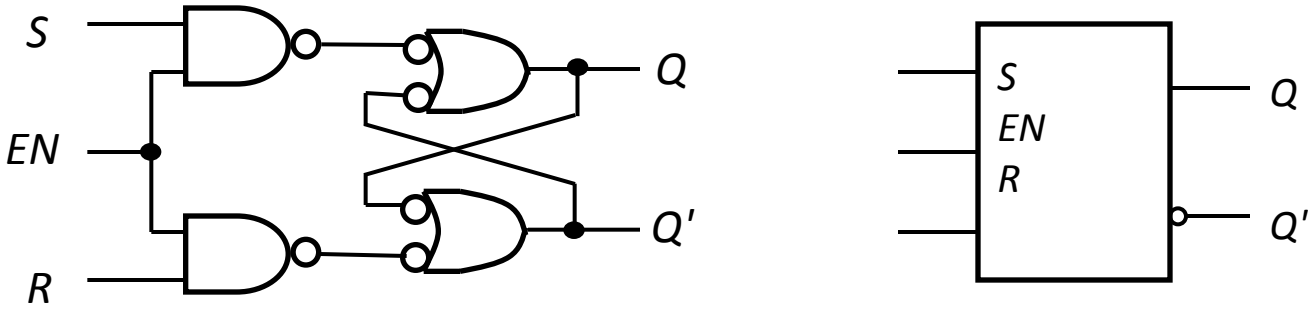
- Active-LOW input S'-R' latch



S′	R′	Q	Q′	
1	0	0	1	initial
1	1	0	1	(afer S′=1, R′=0)
0	1	1	0	
1	1	1	0	(after S′=0, R′=1)
0	0	1	1	invalid!

Gated S-R Latch

- S-R latch + *enable input* (*EN*) and 2 NAND gates → *gated S-R latch*.



Gated S-R Latch

- Outputs change (if necessary) only when *EN* is HIGH.
- Under what condition does the invalid state occur?
- Characteristic table:

EN=1

Q(t)	*S*	*R*	*Q(t+1)*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

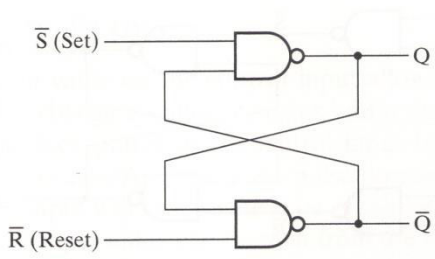
S	*R*	*Q(t+1)*	
0	0	*Q(t)*	No change
0	1	0	Reset
1	0	1	Set
1	1	indeterminate	

$Q(t+1) = S + R'.Q$

$S.R = 0$

Other implementations of the SR

- With NAND Gates

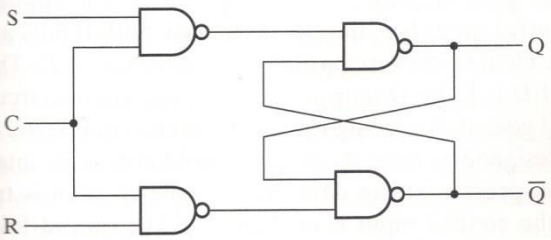


(a) Logic diagram

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

- And adding a control input



(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table