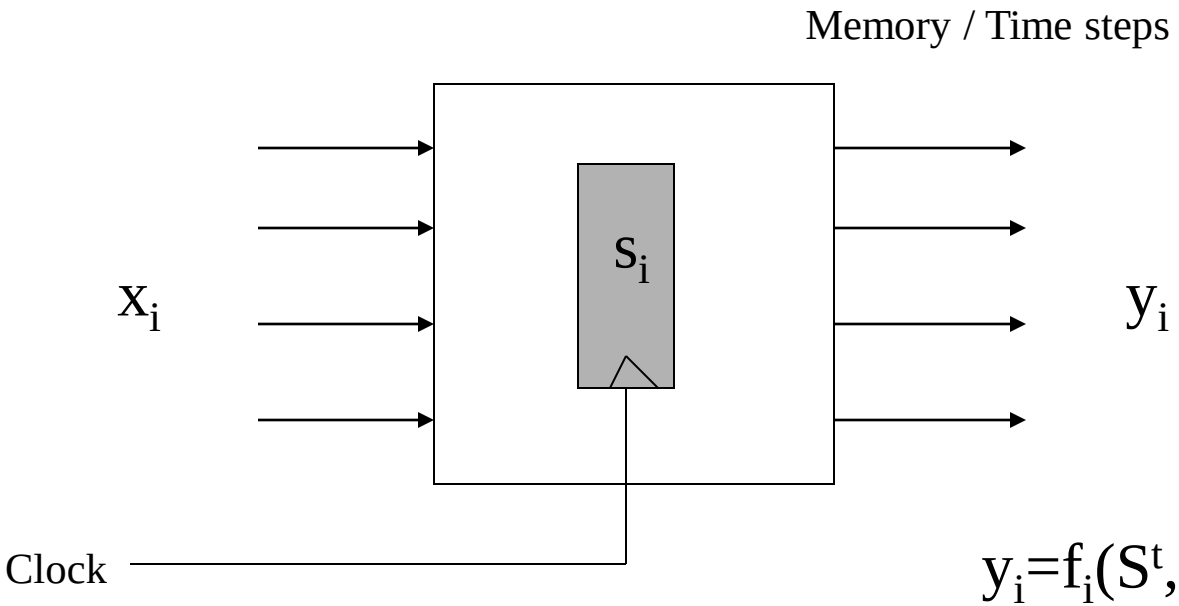
Sequential Logic

Subhasis Bhattacharjee

Sequential Networks



$y_i = f_i(S^t, X)$

$s_i^{t+1} = g_i(S^t, X)$

Memory: Flip flops

Specification: Finite State Machines

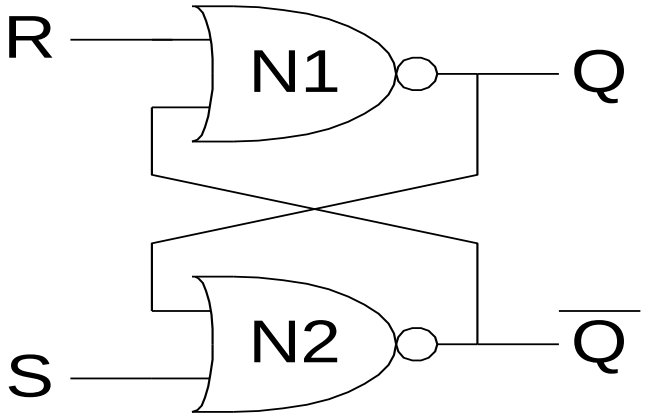
Implementation: Excitation Tables

Memory Devices

- Memory Storage
- Latches
- Flip-Flops
  - SR, D, T

SR (Set/Reset) Latch

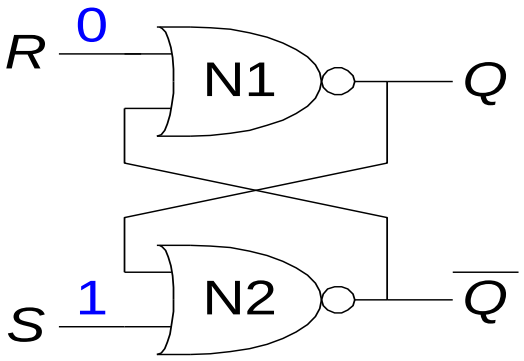
- SR Latch



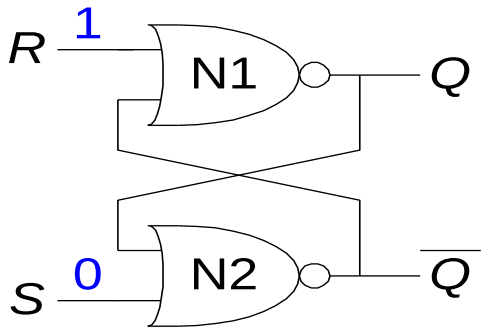
- Consider the four possible cases:
  - $S = 1, R = 0$
  - $S = 0, R = 1$
  - $S = 0, R = 0$
  - $S = 1, R = 1$

SR Latch Analysis

– $S = 1, R = 0$: then $Q = 1$ and $\overline{Q} = 0$

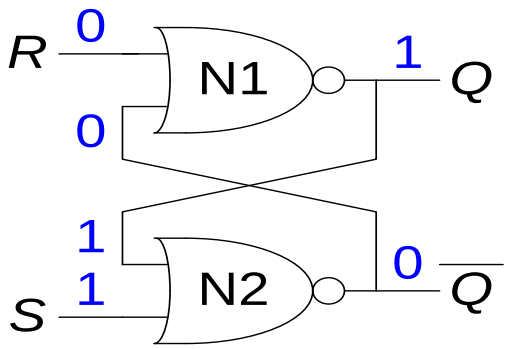


– $S = 0, R = 1$: then $Q = 0$ and $\overline{Q} = 1$

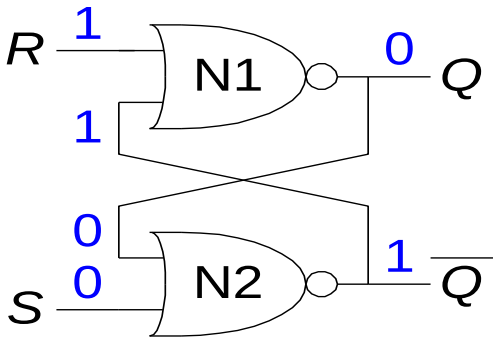


SR Latch Analysis

– $S = 1, R = 0$: then $Q = 1$ and $\overline{Q} = 0$



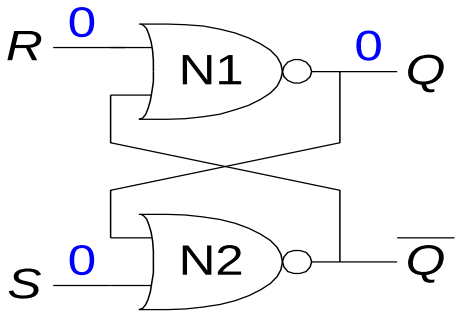
– $S = 0, R = 1$: then $Q = 0$ and $\overline{Q} = 1$



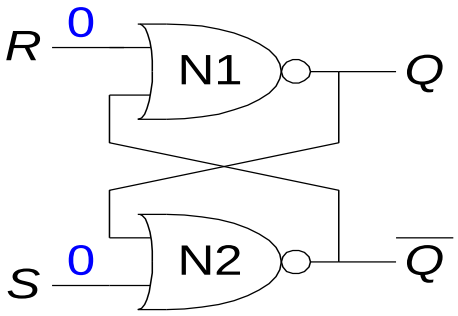
SR Latch Analysis

– $S = 0, R = 0$: then $Q = Q_{prev}$

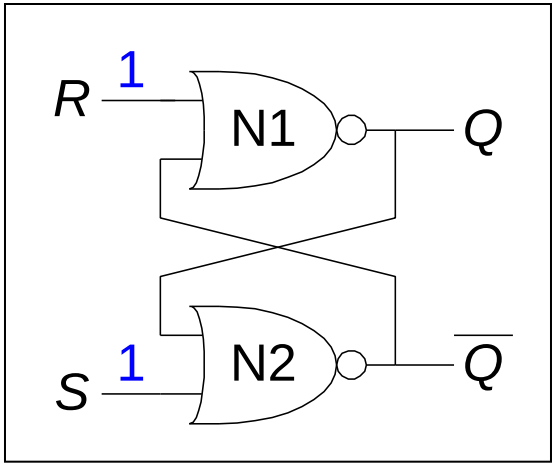
$Q_{prev} = 0$

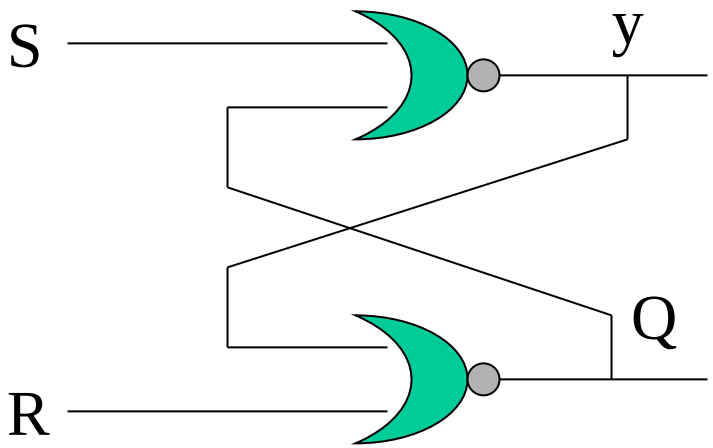


$Q_{prev} = 1$



– $S = 1, R = 1$: then $Q = 0$ and $\overline{Q} = 0$





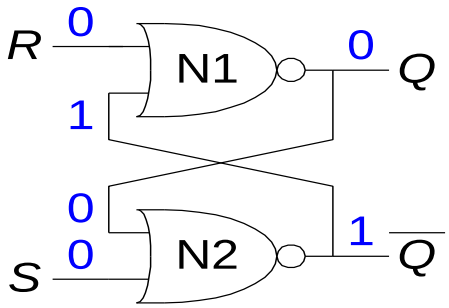
$y = (S+Q)'$

$Q = (R+y)'$

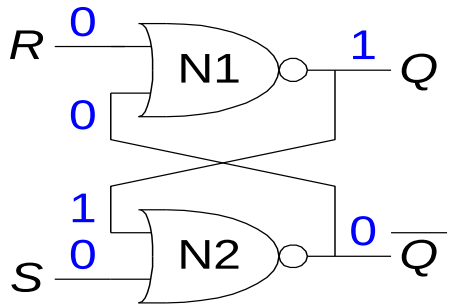
SR Latch Analysis

– $S = 0, R = 0$: then $Q = Q_{prev}$ and $Q = \overline{Q_{prev}}$ (memory!)

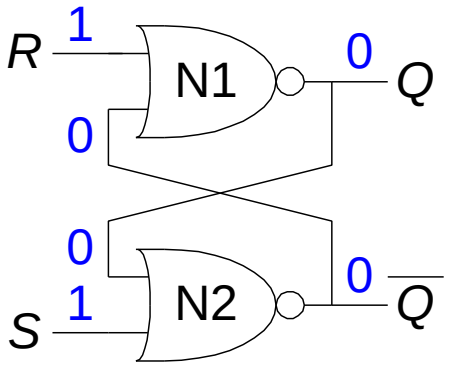
$Q_{prev} = 0$



$Q_{prev} = 1$



– $S = 1, R = 1$: then $Q = 0$ and $\overline{Q} = 0$ (invalid state: $Q \neq$ NOT $\overline{Q}$)

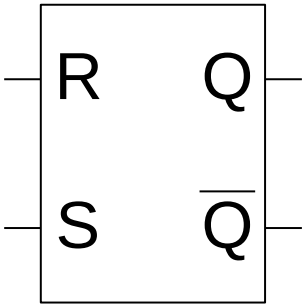


SR Latch Symbol

- SR stands for Set/Reset Latch
  - Stores one bit of state (Q)
- Control what value is being stored with S, R inputs
  - Set: Make the output 1 ($S = 1$, $R = 0$, $Q = \mathbf{1}$)
  - Reset: Make the output 0 ($S = 0$, $R = 1$, $Q = \mathbf{0}$)

- **Must do something to avoid invalid state (when $S = R = 1$)**

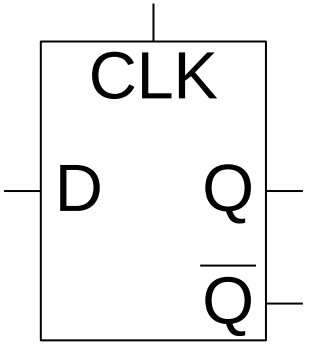
SR Latch Symbol



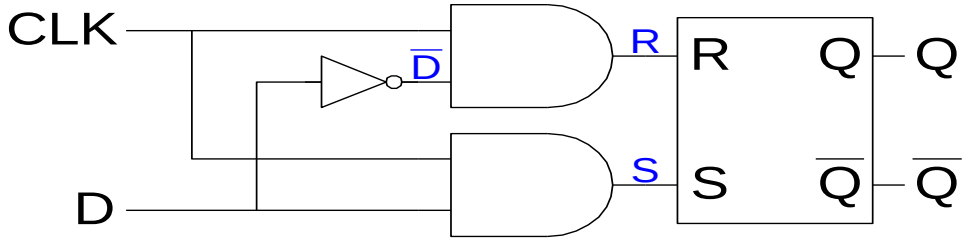
D Latch

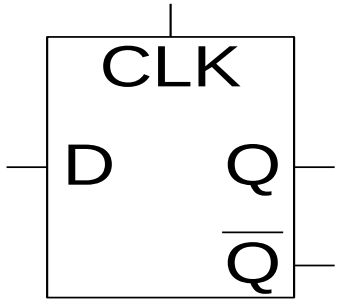
- Two inputs: *CLK*, *D*
  - *CLK*: controls *when* the output changes
  - *D* (the data input): controls *what* the output changes to
- Function
  - When *CLK* = 1, *D* passes through to *Q* (the latch is *transparent*)
  - When *CLK* = 0, *Q* holds its previous value (the latch is *opaque*)
- Avoids invalid case when $Q \neq \text{NOT } \overline{Q}$

D Latch Symbol



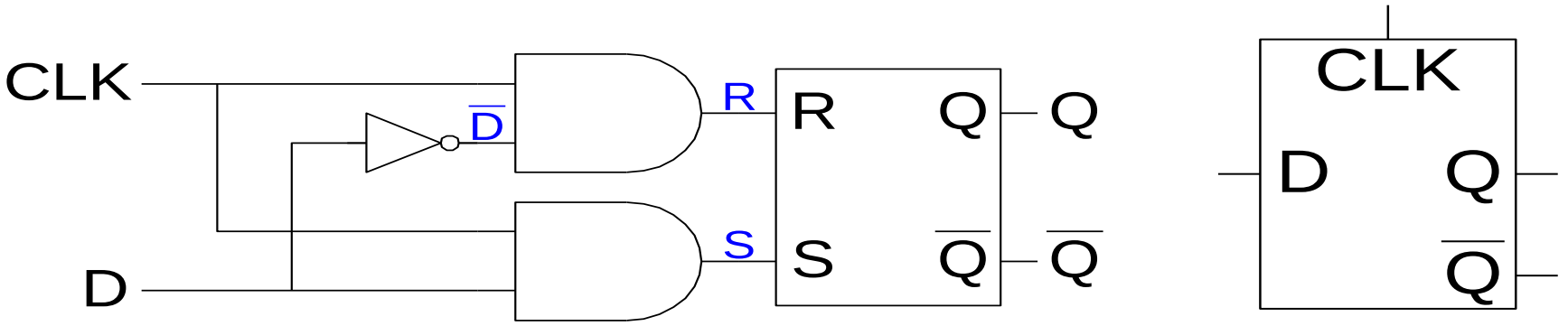
D Latch Internal Circuit





CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X					
1	0					
1	1					

D Latch Internal Circuit

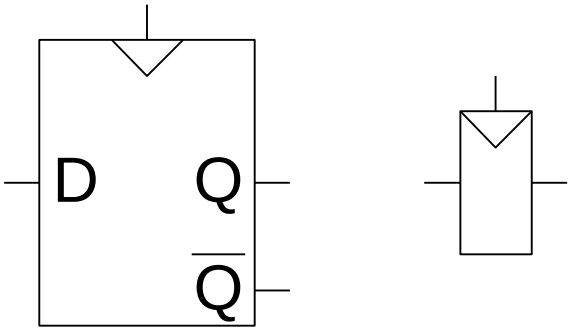


CLK	*D*	$\overline{D}$	*S*	*R*	*Q*	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D Flip-Flop

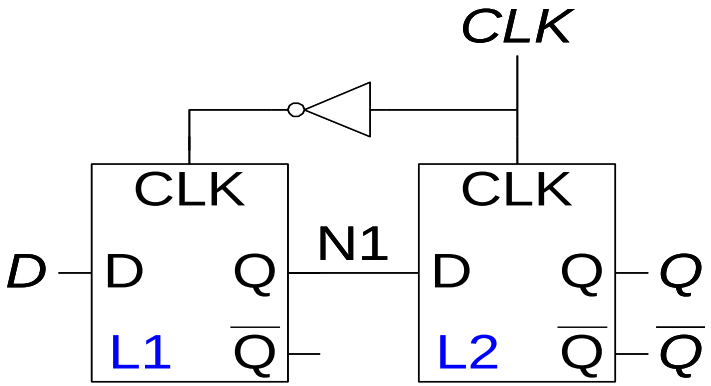
- Two inputs: *CLK*, *D*
- *Function*
  - The flip-flop "samples" *D* on the rising edge of *CLK*
    - When *CLK* rises from 0 to 1, *D* passes through to *Q*
    - Otherwise, *Q* holds its previous value
  - *Q* changes only on the rising edge of *CLK*
- A flip-flop is called an *edge-triggered* device because it is activated on the clock edge

D Flip-Flop Symbols

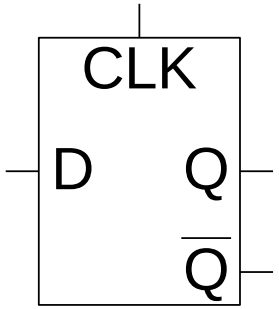


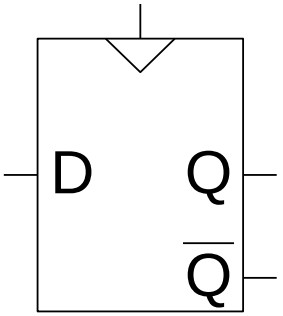
D Flip-Flop Internal Circuit

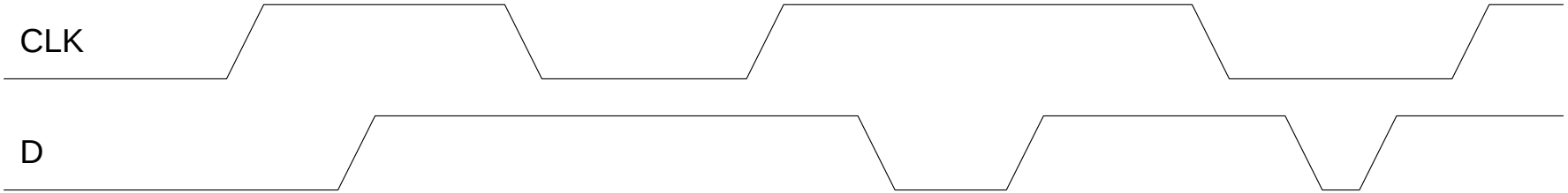
- Two back-to-back latches (L1 and L2) controlled by complementary clocks
- When *CLK* = 0
  - L1 is transparent, L2 is opaque
  - *D* passes through to N1
- When *CLK* = 1
  - L2 is transparent, L1 is opaque
  - N1 passes through to *Q*
- Thus, on the edge of the clock (when *CLK* rises from 0 →1)
  - *D* passes through to *Q*



D Flip-Flop vs. D Latch



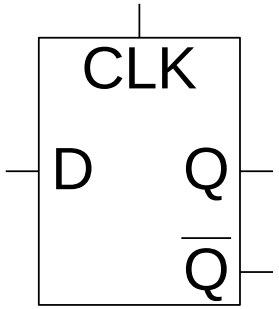


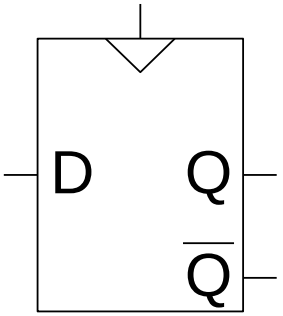


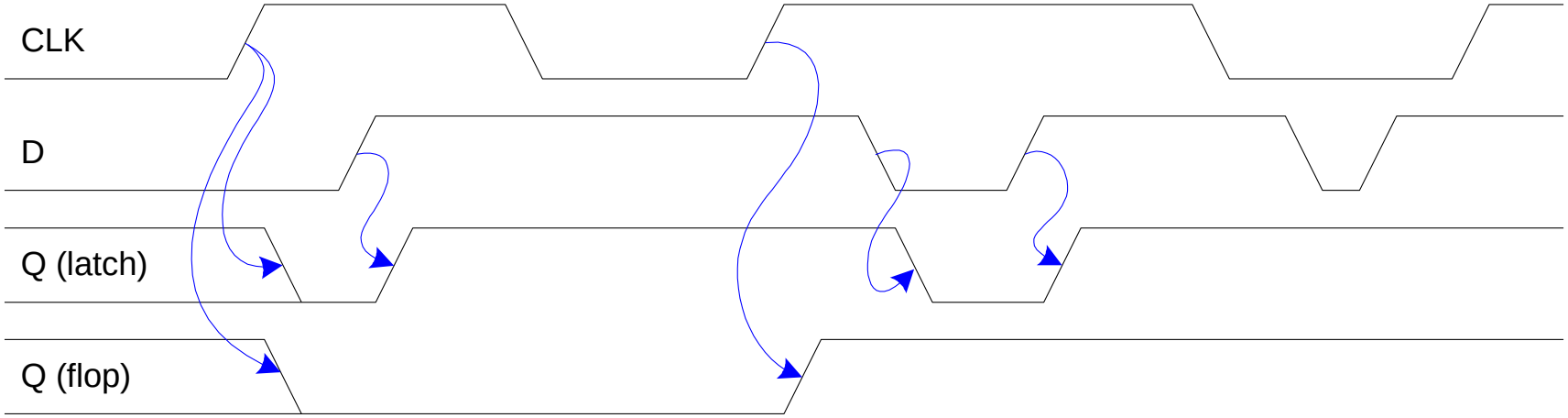
Q (latch)

Q (flop)

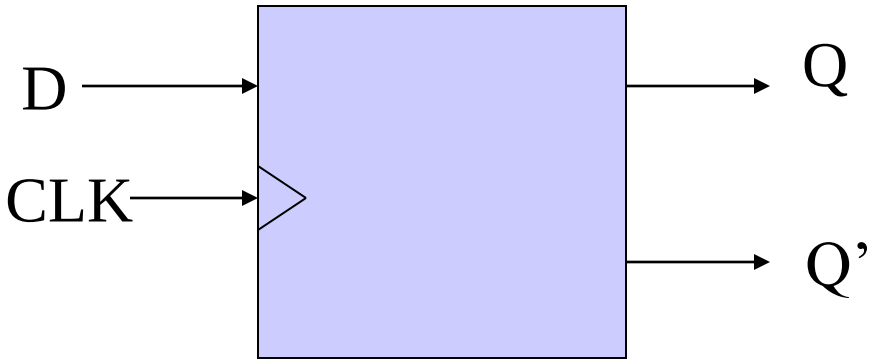
D Flip-Flop vs. D Latch







D Flip-Flop (Delay)



Id	D	Q(t)	Q(t+1)
0	0	0	0
1	0	1	0
2	1	0	1
3	1	1	1

State table

PS \ D	0	1
0	0	1
1	0	1

NS= Q(t+1)

Characteristic Expression

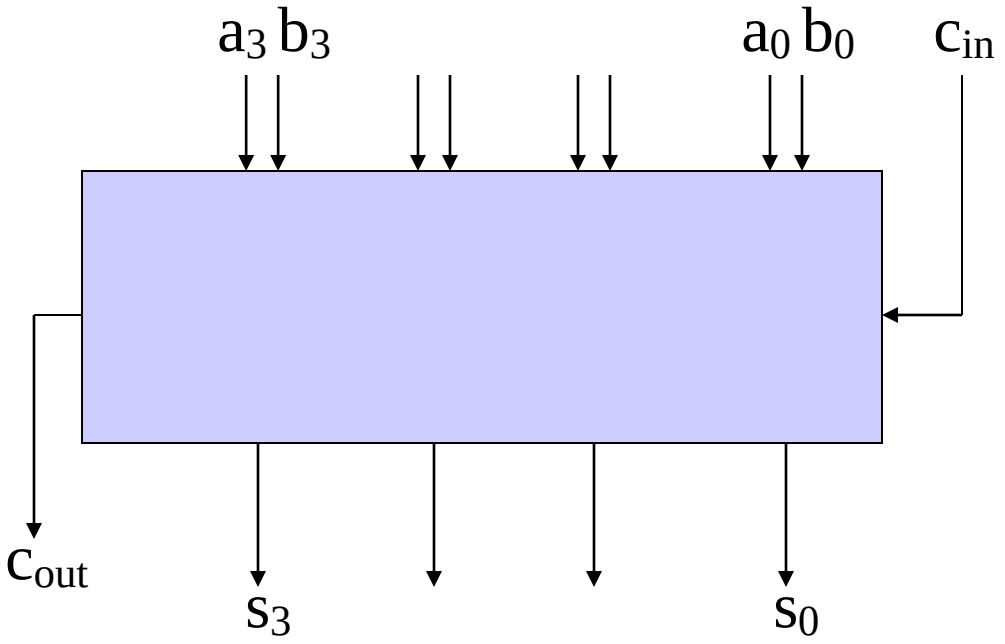
$Q(t+1) = D(t)$

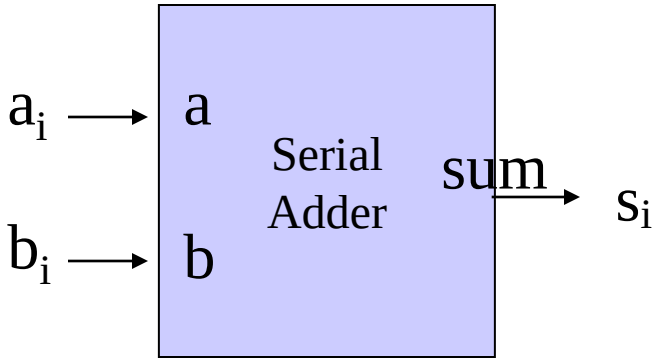
Sequential Modules: Introduction

- Slice operation bitwise
- Perform process in a series of time
- Advantage:
  - Cheaper hardware
  - Fit for FPGA architecture
  - Pipelining for excellent throughput
- Disadvantage: Longer latency

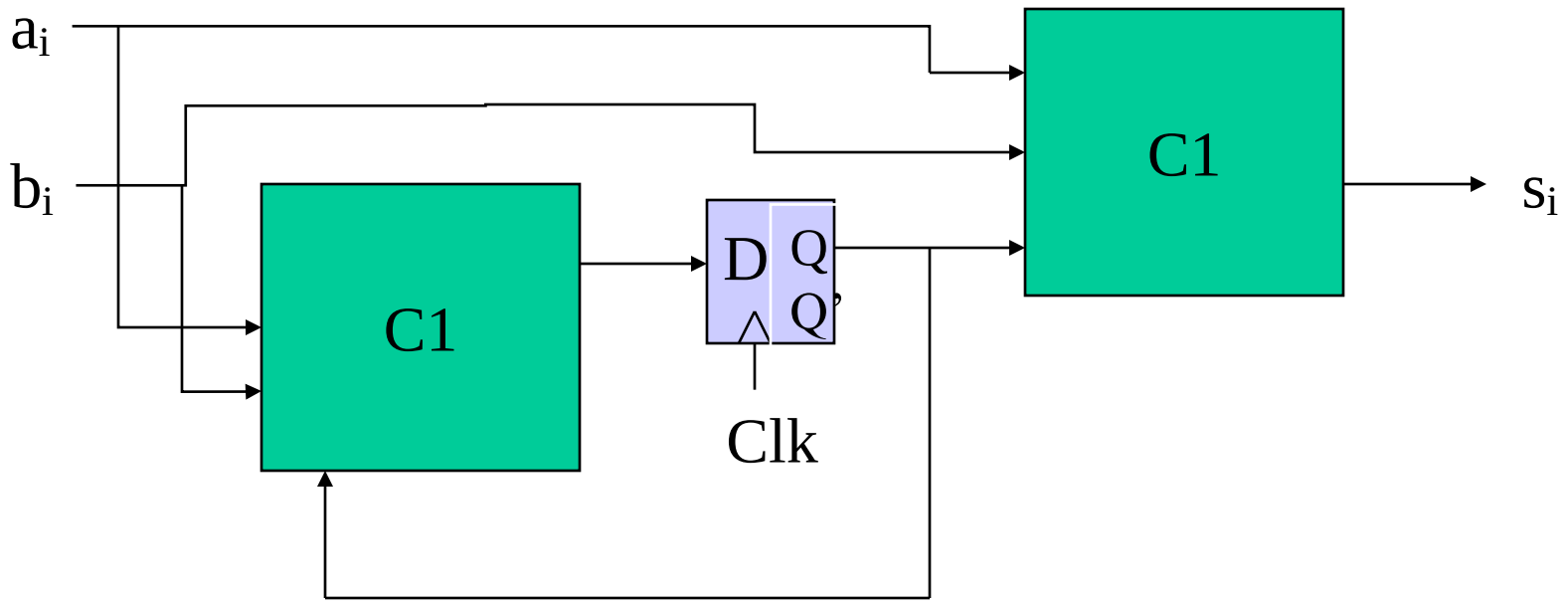
Serial Adder: Perform serial bit-addition

At time i, read a_i and b_i. Produce s_i and c_{i+1}
Internal state stores c_i. Carry bit c_0 is set as c_{in}





Serial Adder using D F-F



Feed a_i and b_i and generate s_i at time i.
Where is c_i and c_{i+1}?

Serial Adder using a D Flip-Flop

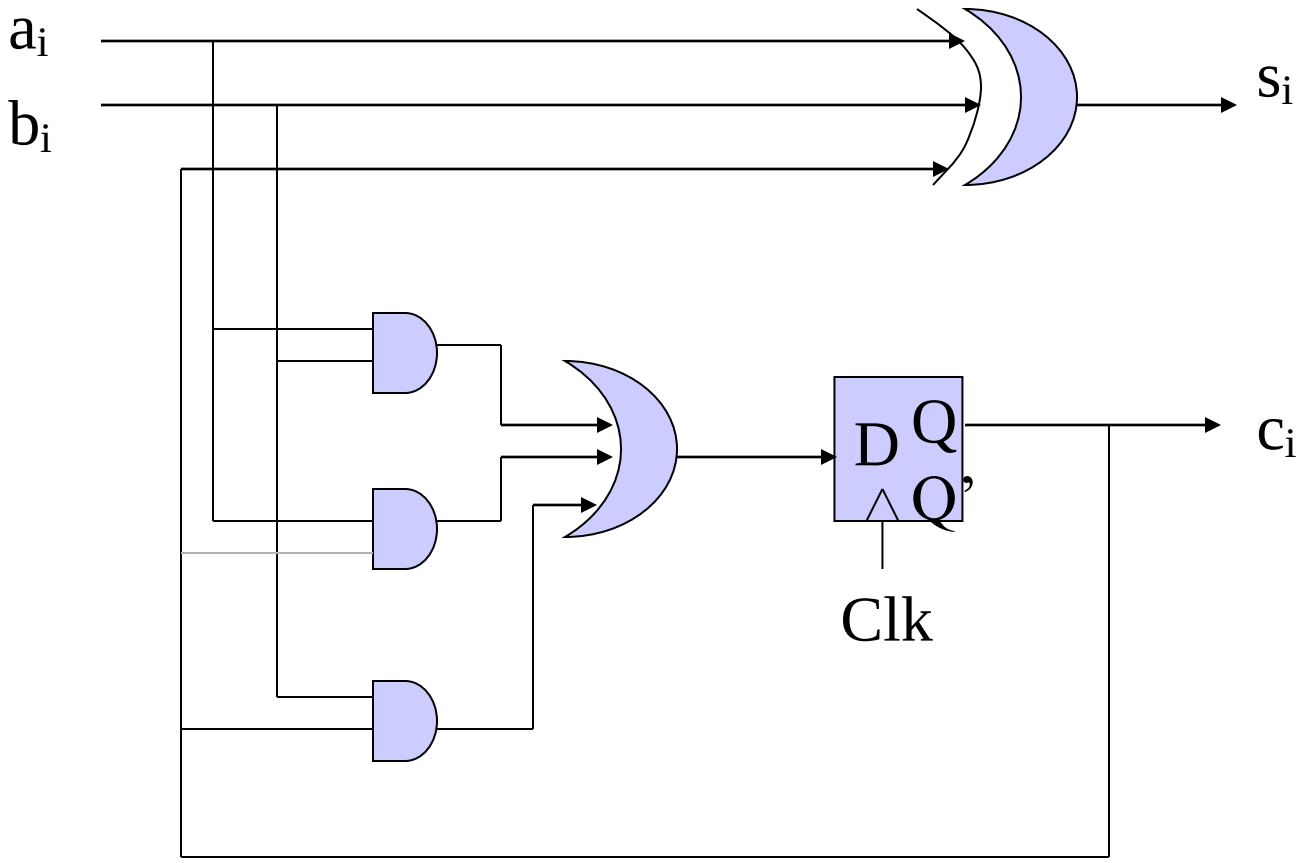
id	a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$D=c_{i+1}$

$Q=c_i$

Serial Adder using a D Flip-Flop

Logic Diagram



Serial Adder using an SR Flip-Flop

Excitation Table

id	a_i	b_i	c_i (Q)	c_{i+1}	S	R
0	0	0	0	0	0	-
1	0	0	1	0	0	1
2	0	1	0	0	0	-
3	0	1	1	1	-	0
4	1	0	0	0	0	-
5	1	0	1	1	-	0
6	1	1	0	1	1	0
7	1	1	1	1	-	0

SR generate c_{i+1}

$Q=c_i$

Excitation Table of SR Flip-Flop

S

c_i \ b_i	0	1
0	0	-
1	-	-

(a_i)

State table

PS \ inputs SR	00	01	10	11
0	0	0	1	-
1	1	0	1	-

Q(t+1)

R

c_i \ b_i	0	1
0	-	-
1	1	0

(a_i)

Excitation table

Q(t) \ NS Q(t+1)	0	1
0	0 -	1 0
1	0 1	- 0

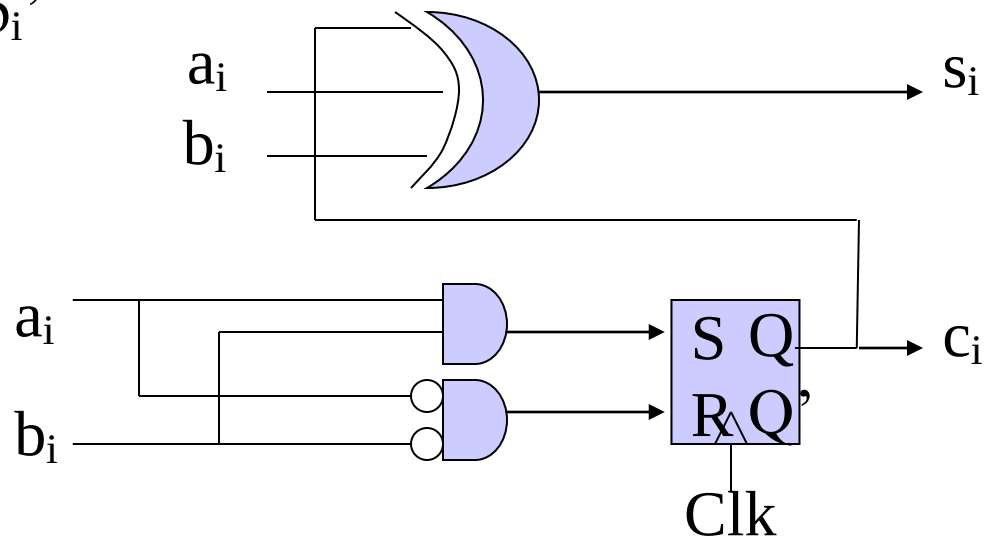
PS

Serial Adder using an SR Flip-Flop

Logic Diagram

$S = a_i b_i$

$R = a_i' b_i'$



Multiplication using Serial Addition

3 X 5 = 15

		0	1	1
	x	1	0	1
		0	1	1
	0	0	0	
+	0	1	1	
	0	1	1	1	1

		a_2	a_1	a_0
	x	b_2	b_1	b_0
		a_2b_0	a_1b_0	a_0b_0
	a_2b_1	a_1b_1	a_0b_1	
+	a_2b_1	a_1b_1	a_0b_1	
			m_1	m_0

For $m=AxB$, set $m^{(0)}=0$

At time i, perform $m^{(i+1)}=m^{(i)}+Ab_i2^i$

Standard Sequential Modules

1. Register
2. Shift Register
3. Counter

Register

We Will Study Rest of the Topics Later on

Shift Register

Counter

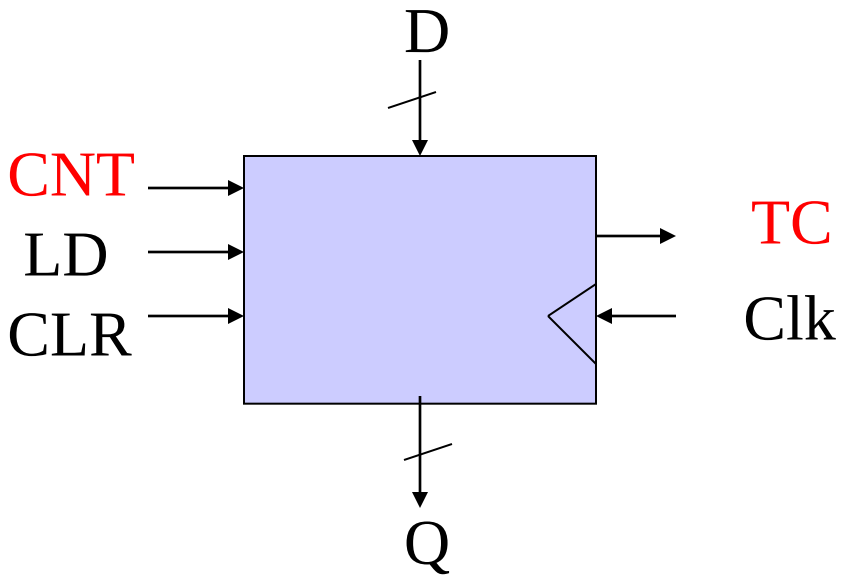
Counter: Applications

- Program Counter
- Address Keeper: FIFO, LIFO
- Clock Divider
- Sequential Machine

Counter

- Modulo-n Counter
- Modulo Counter (m<n)
- Counter (a-to-b)
- Counter of an Arbitrary Sequence
- Cascade Counter

Modulo-n Counter



Q (t+1) = (0, 0, .. , 0) if CLR = 1

= D if LD = 1 and CLR = 0

= (Q(t)+1)mod n if LD = 0, CNT = 1 and CLR = 0

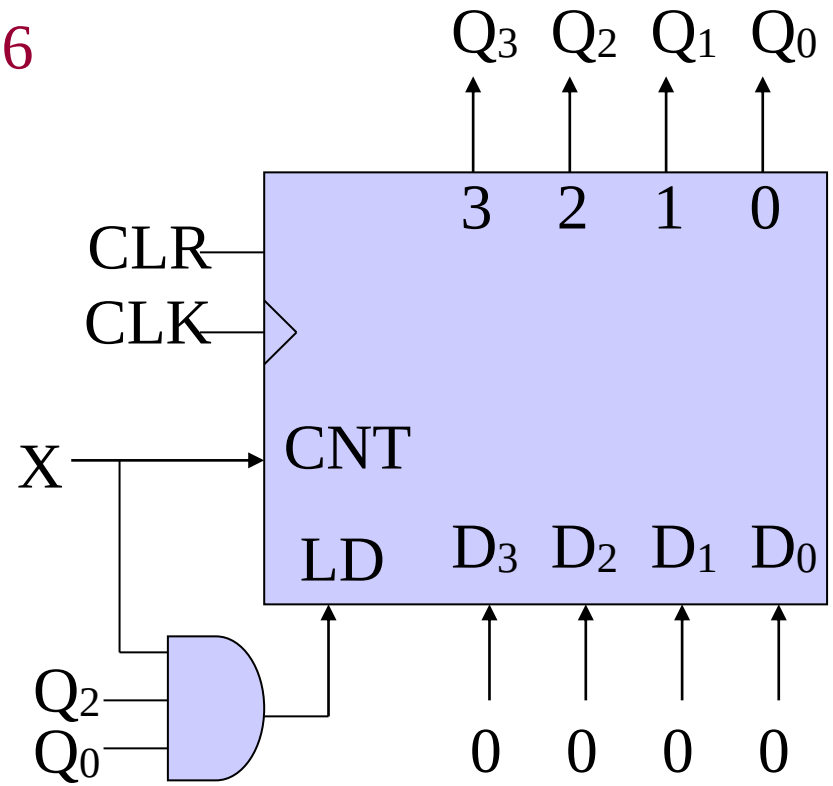
= Q (t) if LD = 0, CNT = 0 and CLR = 0

TC = 1 if Q (t) = n-1 and CNT = 1

= 0 otherwise

Modulo-m Counter (m< n)

Given a mod 16 counter, construct
a mod-m counter (0 < m < 16) with AND, OR, NOT gates

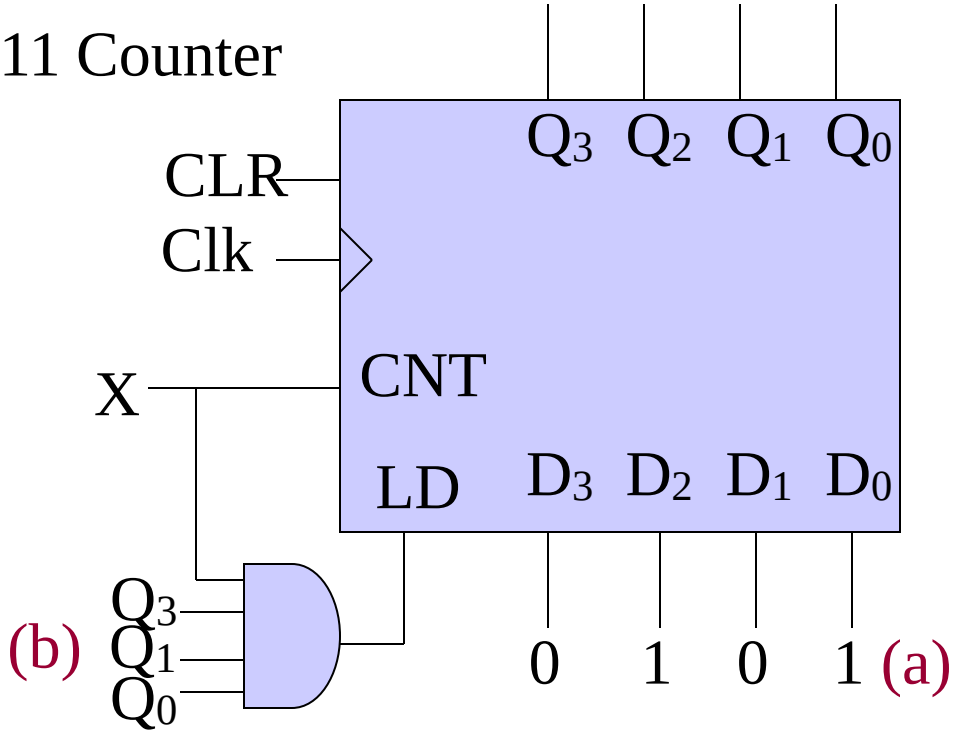


Set LD = 1 when X = 1 and ($Q_3Q_2Q_1Q_0$) = (0101), ie m-1

Counter (a-to-b)

Given a mod 16 counter, construct an a-to-b counter
(0 < a < b < 15)

A 5-to-11 Counter



Set LD = 1 when X = 1 and ($Q_3Q_2Q_1Q_0$) = b (in this case, 1011)

Counter of an Arbitrary Sequence

Given a mod 8 counter, construct
a counter with sequence 0 1 5 6 2 3 7

When Q = 1, load D = 5
When Q = 6, load D = 2
When Q = 3, load D = 7