NCDC

NUST CHIP DESIGN CENTRE

Digital Logic Design

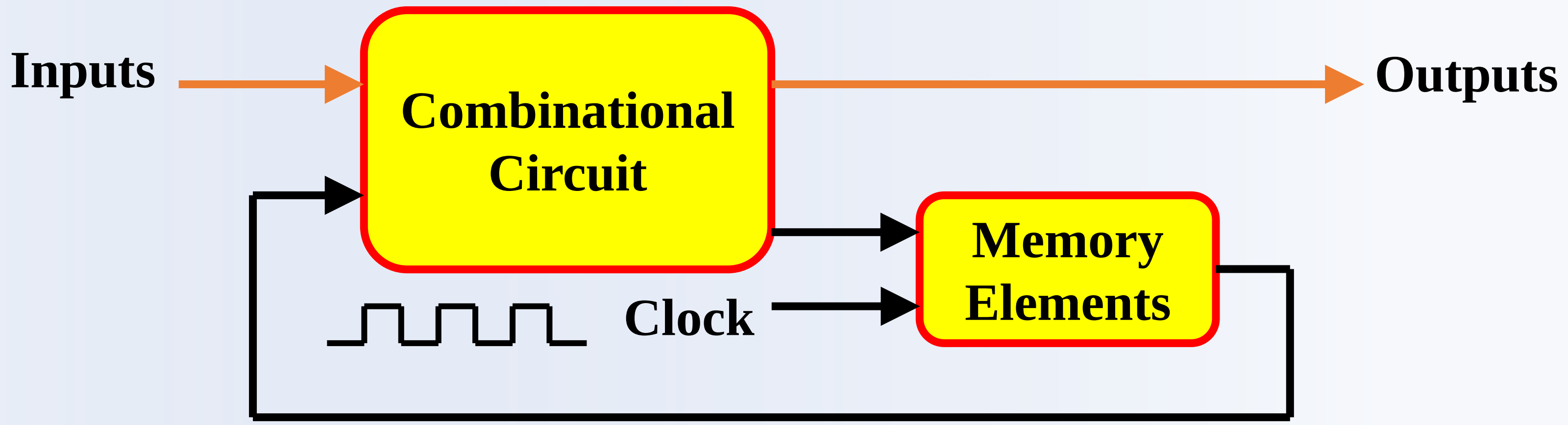
Sequential Logic

What is a sequential circuit?

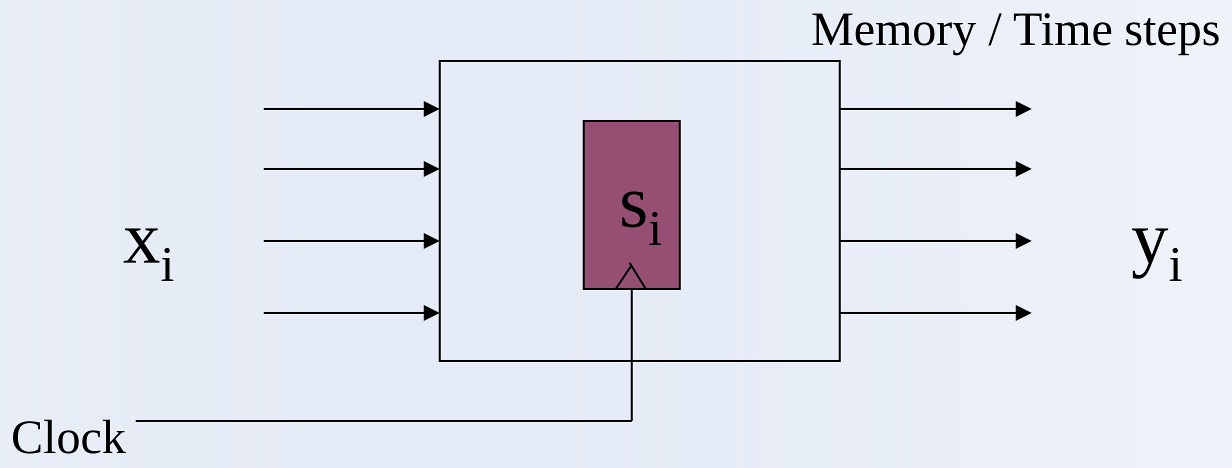
“A circuit whose output depends on current inputs and past outputs”

“A circuit with memory”

Memory => Time



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

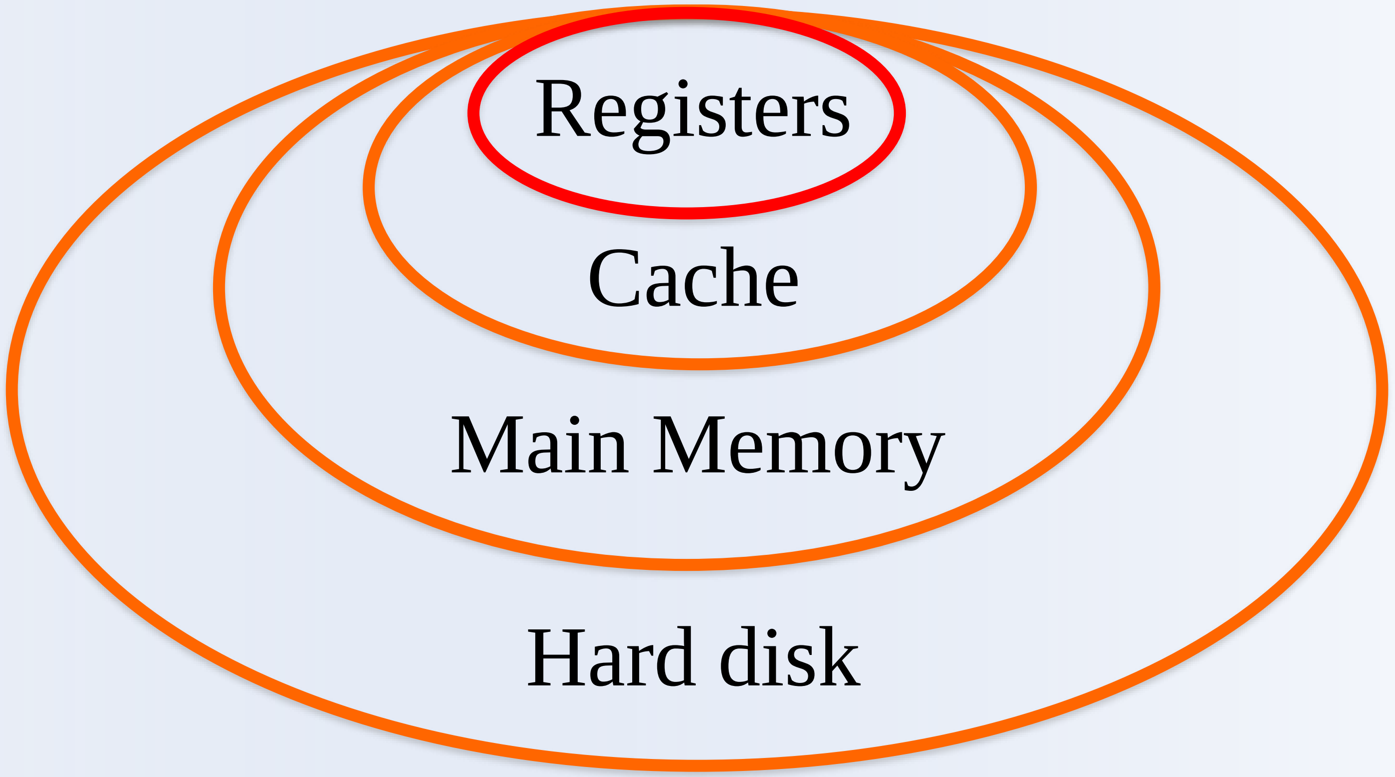
Main Theme: Timing

Present time = t and next time = t+1

Timing constraints to separate the present and next times.

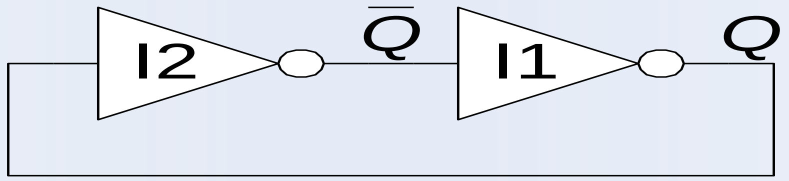
Memory Hierarchy

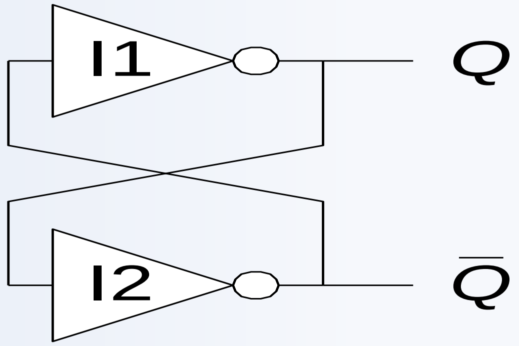
What are registers made of ? Flip-Flops, Latches



Fundamental Memory Mechanism

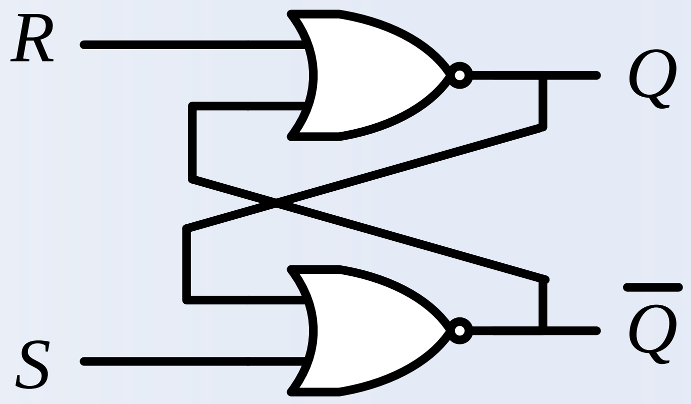
- Fundamental building block of sequential circuits
- Two outputs: *Q'*, *Q*
- There is a feedback loop!
  - Recall: Combinational logics have no feedback loop.



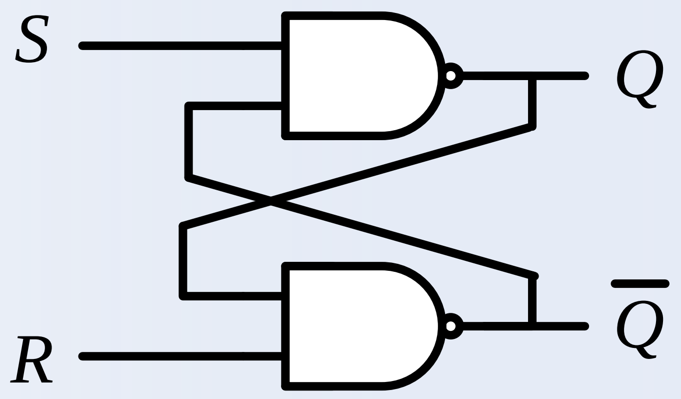


Latches

- *SR* Latch



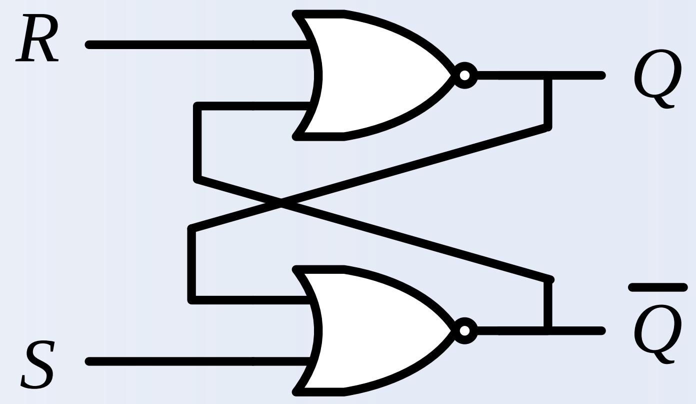
S R	*Q*	
0 0	Q_0	No change
0 1	0	Reset
1 0	1	Set
1 1	*Q*=*Q'*=0	Invalid



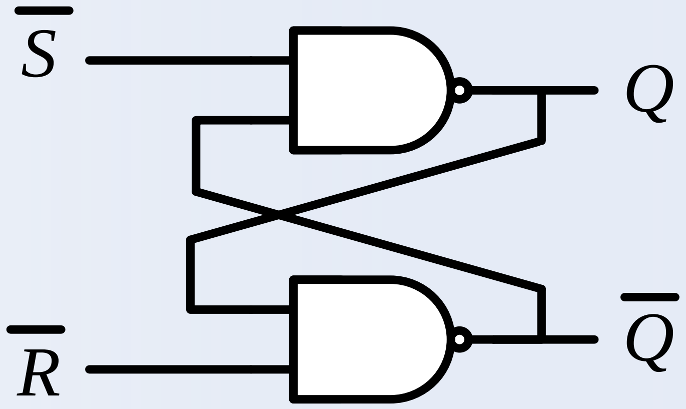
S R	*Q*	
0 0	*Q*=*Q'*=1	Invalid
0 1	1	Set
1 0	0	Reset
1 1	Q_0	No change

Latches

- *SR* Latch



S *R*	*Q*	
0 0	Q_0	No change
0 1	0	Reset
1 0	1	Set
1 1	$Q=Q'=0$	Invalid

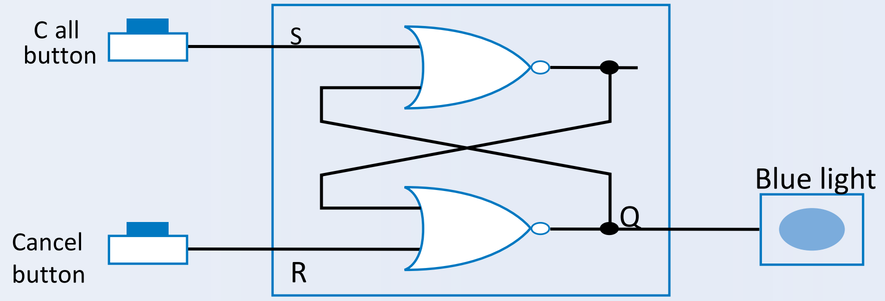


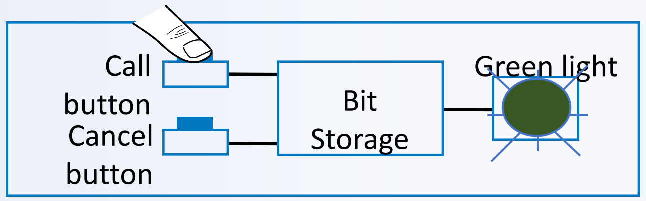
S *R*	*S'* *R'*	*Q*	
1 1	0 0	$Q=Q'=1$	Invalid
1 0	0 1	1	Set
0 1	1 0	0	Reset
0 0	1 1	Q_0	No change

Flight attendant call button

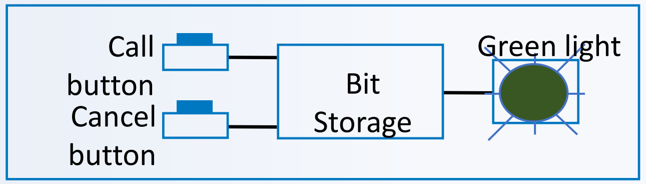
- Flight attendant call button
  - Press call: light turns on
    - ***Stays on*** after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

- SR latch implementation
  - Call=1 : sets Q to 1 and keeps it at 1
  - Cancel=1 : resets Q to 0

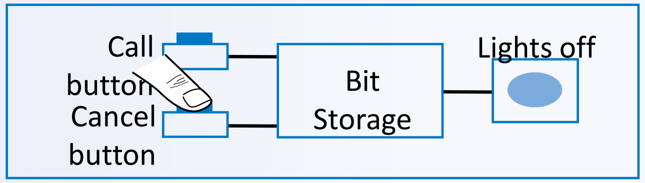




1. Call button pressed – light turns on



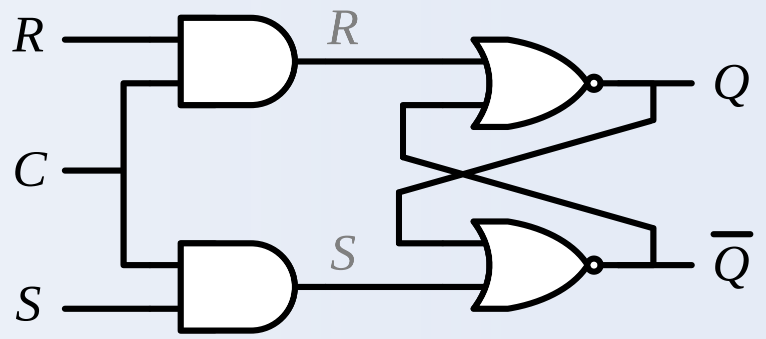
*2. Call button released – light **stays on***

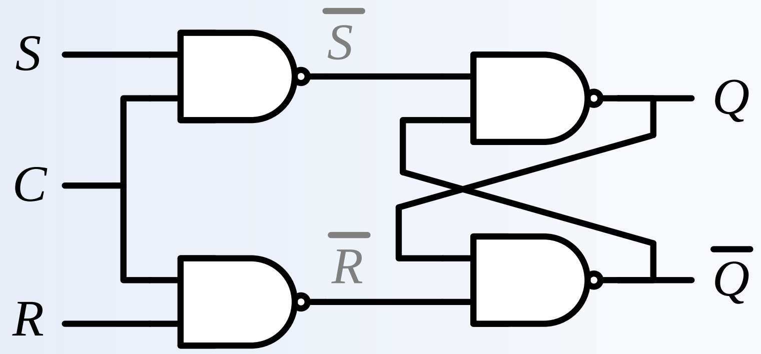


3. Cancel button pressed – light turns off

Controlled Latches

- *SR* Latch with Control Input

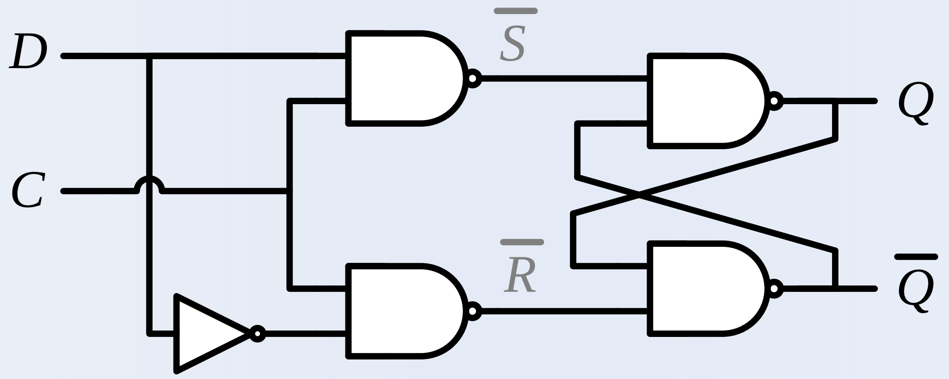




C S R	*Q*	
0 x x	Q_0	No change
1 0 0	Q_0	No change
1 0 1	0	Reset
1 1 0	1	Set
1 1 1	*Q=Q'*	Invalid

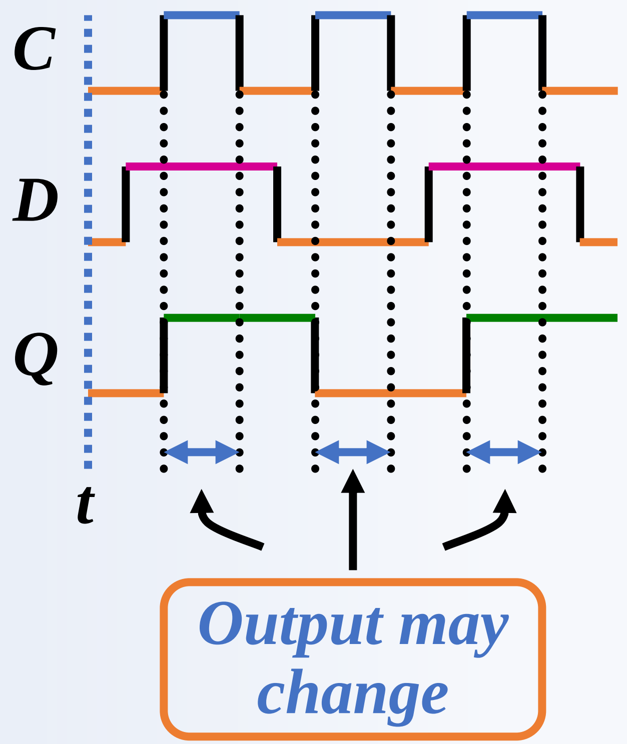
Controlled Latches

- *D* Latch (*D = Data*)



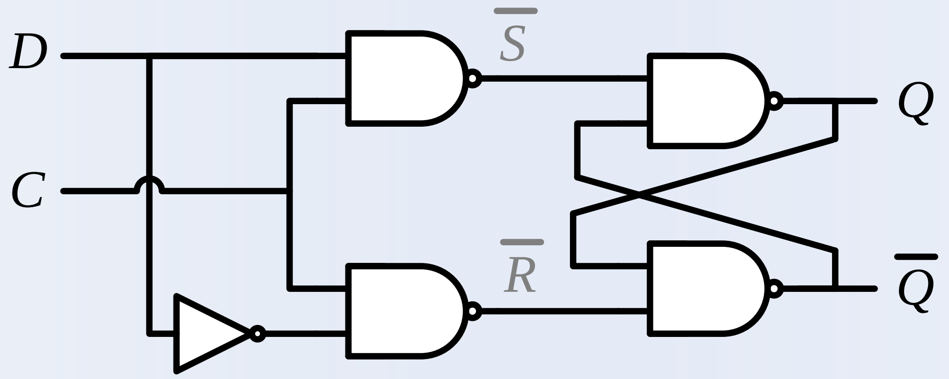
C D	Q	
0 x	Q_0	No change
1 0	0	Reset
1 1	1	Set

Timing Diagram



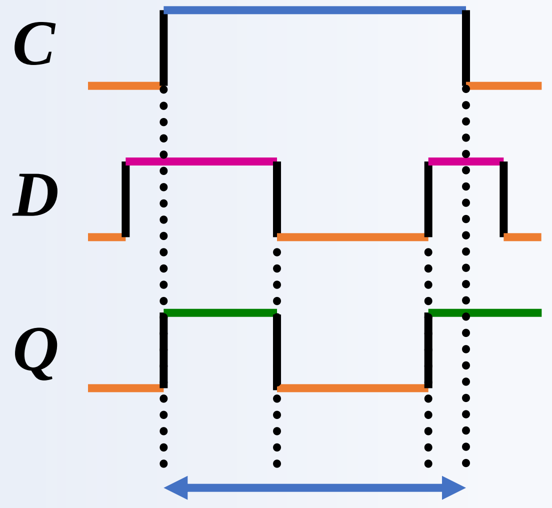
Controlled Latches

- *D* Latch (*D = Data*)



C	*D*	*Q*	
0	x	Q_0	No change
1	0	0	Reset
1	1	1	Set

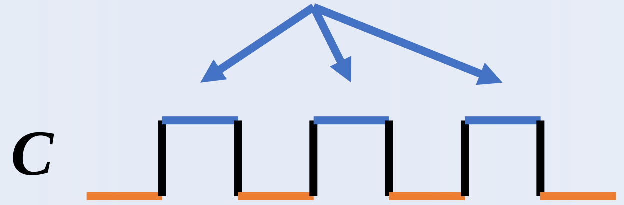
Timing Diagram



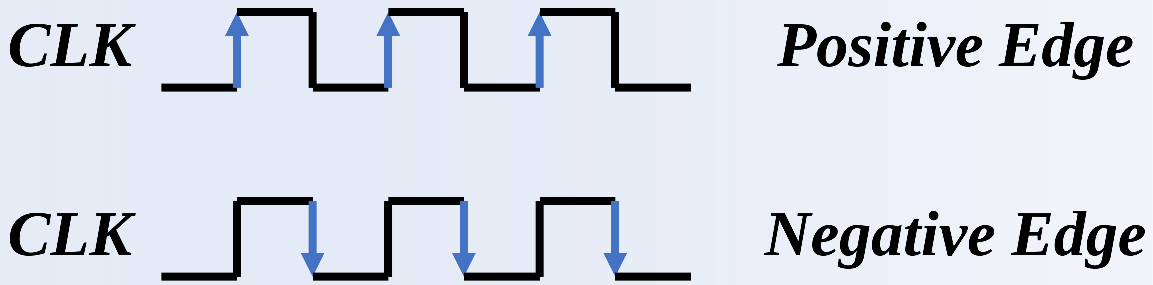
Output may change

Flip-Flops

- Controlled latches are level-triggered

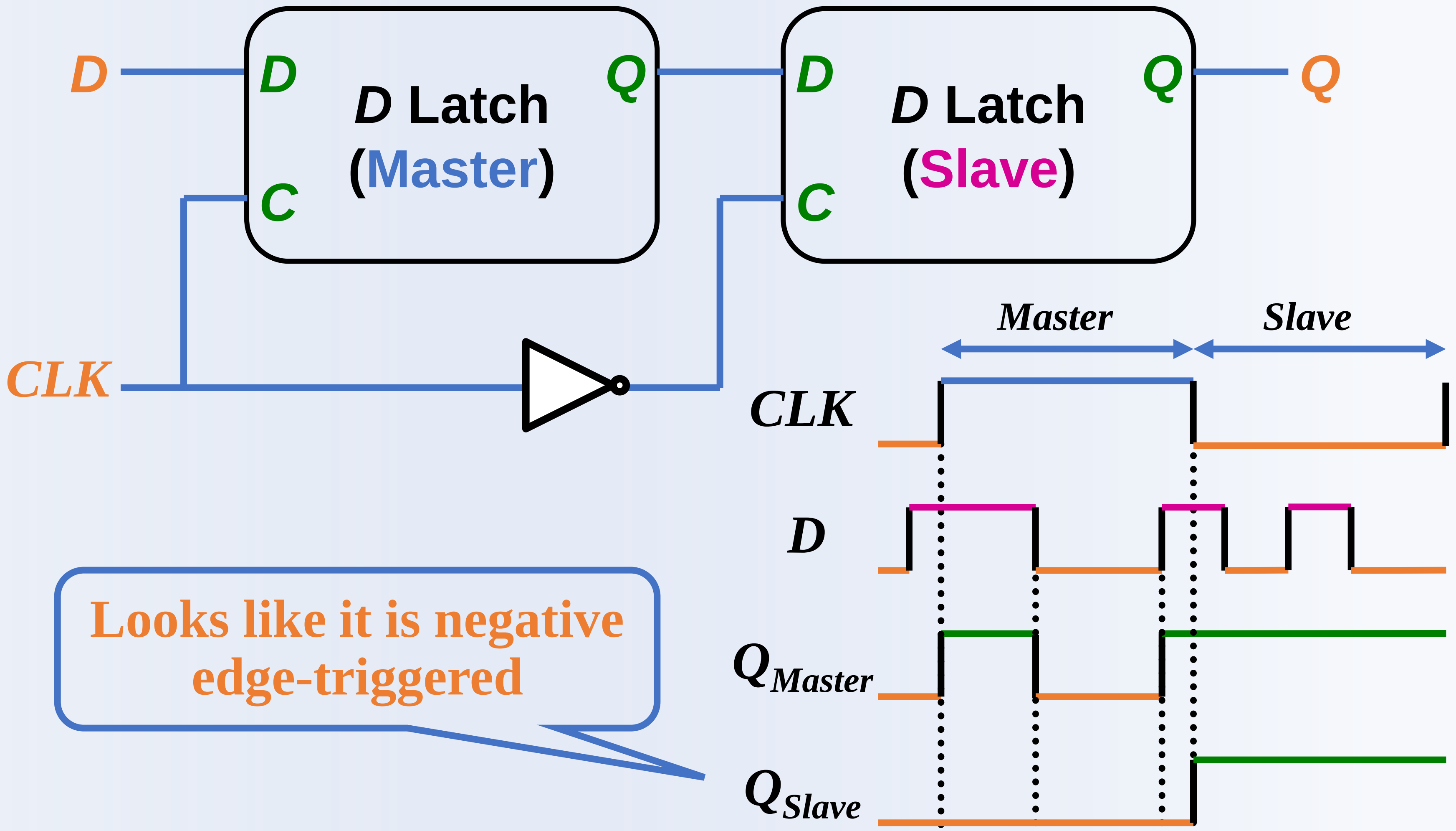


- Flip-Flops are edge-triggered



Flip-Flops

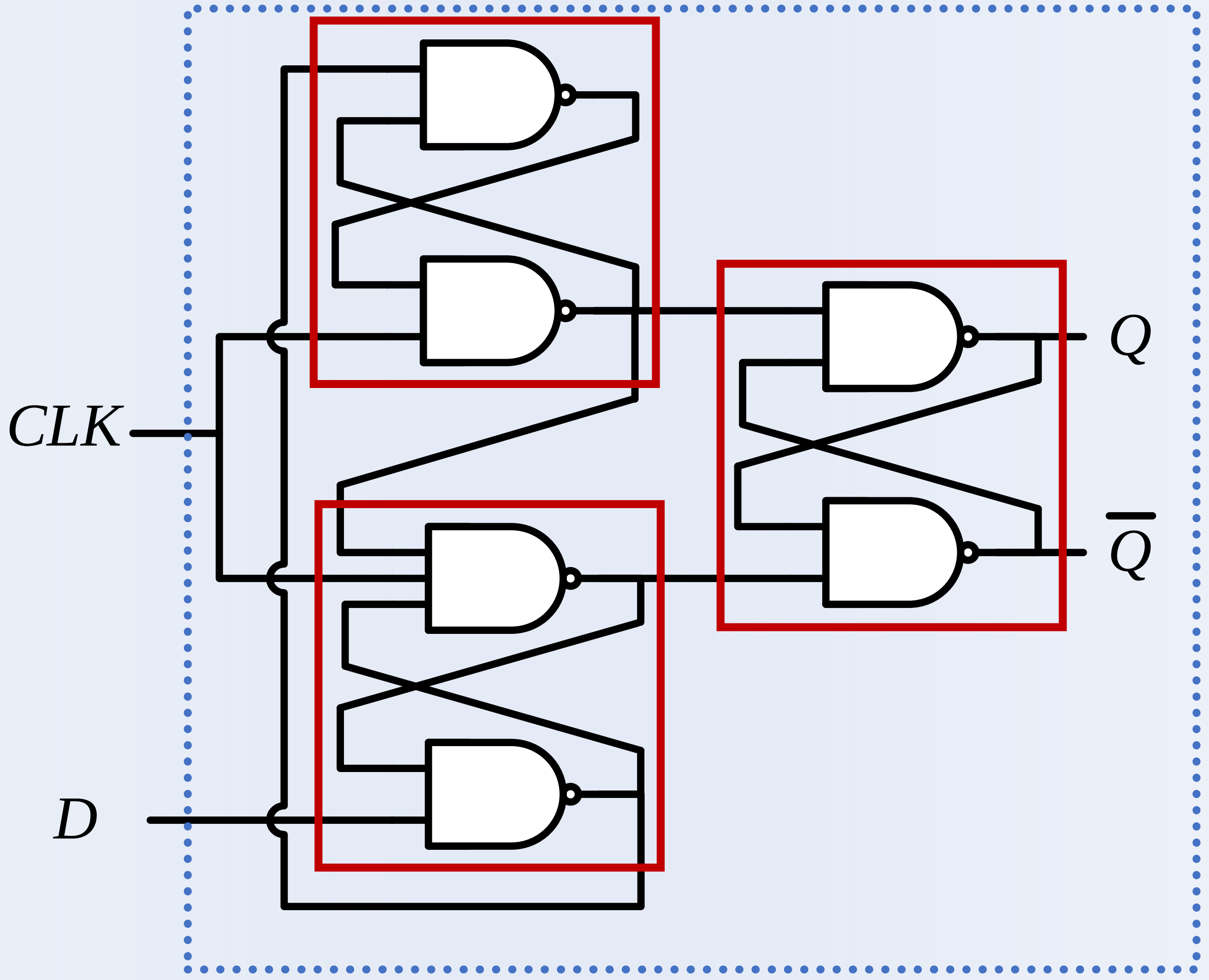
- Master-Slave *D* Flip-Flop

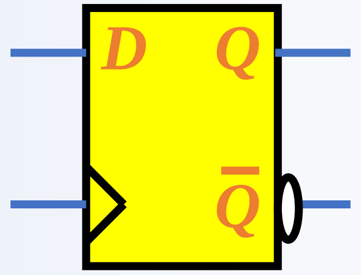


Looks like it is negative edge-triggered

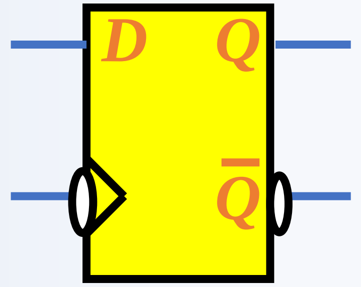
Flip-Flops

- Edge-Triggered *D* Flip-Flop





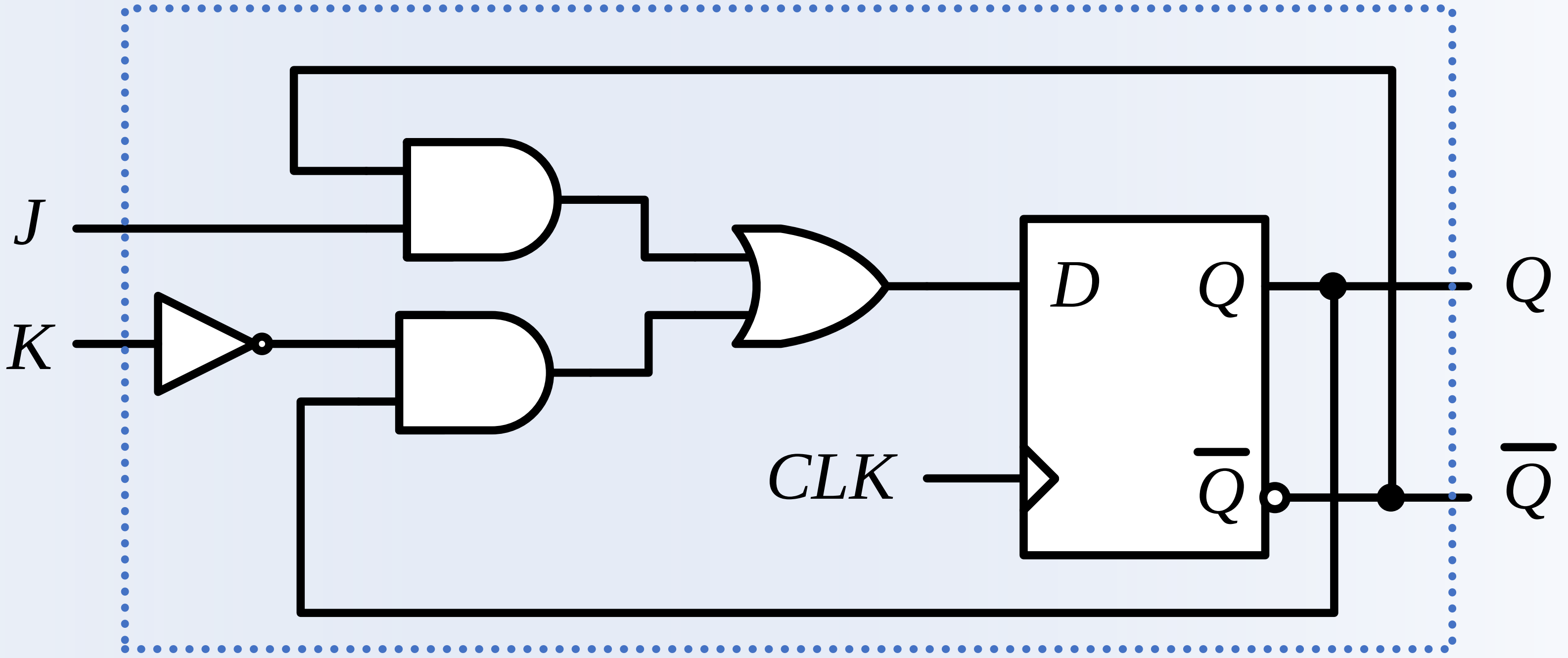
Positive Edge



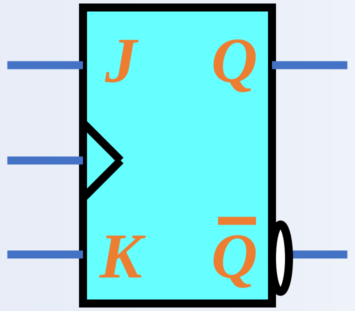
Negative Edge

Flip-Flops

- *JK* Flip-Flop

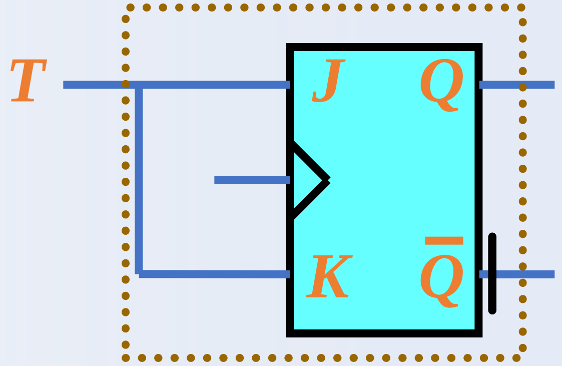


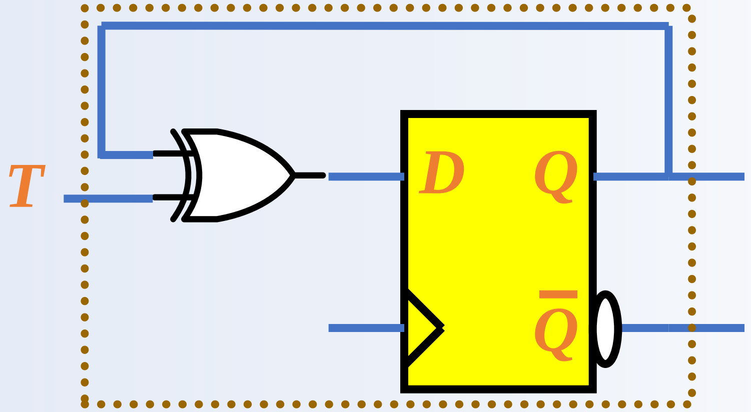
$D = JQ' + K'Q$



Flip-Flops

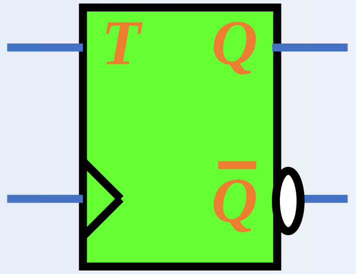
- *T* Flip-Flop





$$D = JQ' + K'Q$$

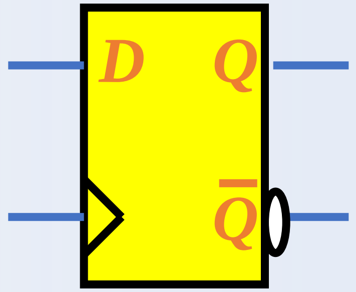
$$D = TQ' + T'Q = T \oplus Q$$



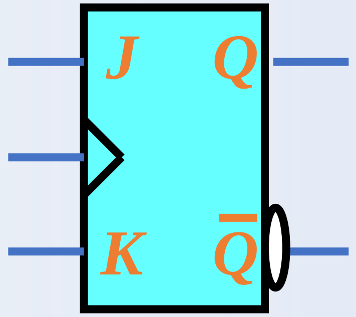
Characteristic Table and Equation

- A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form. It represents the next state of flip-flop in terms of flip-flop input and current state.
- A characteristic equation is the algebraic representation of characteristic table.

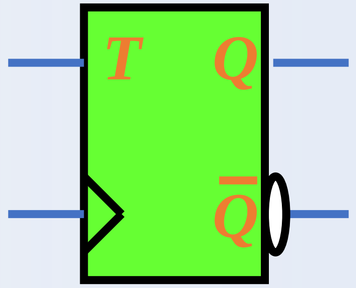
Flip-Flop Characteristic Tables



D	$Q(t+1)$	
0	0	Reset
1	1	Set

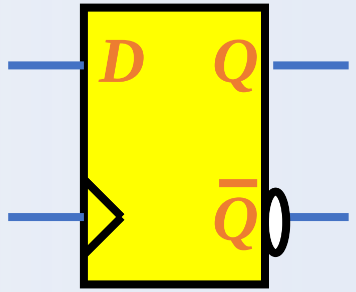


J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Toggle



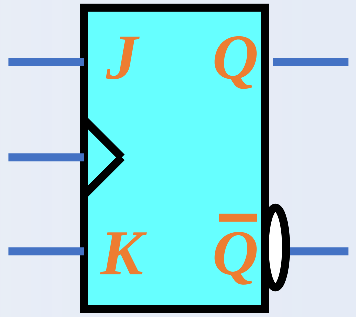
T	$Q(t+1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Toggle

Flip-Flop Characteristic Equations



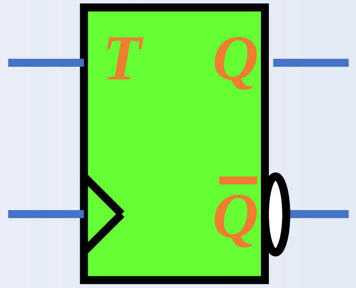
D	*Q*(*t*+1)
0	0
1	1

$Q(t+1) = D$



J	*K*	*Q*(*t*+1)
0	0	*Q*(*t*)
0	1	0
1	0	1
1	1	*Q'*(*t*)

$Q(t+1) = JQ' + K'Q$

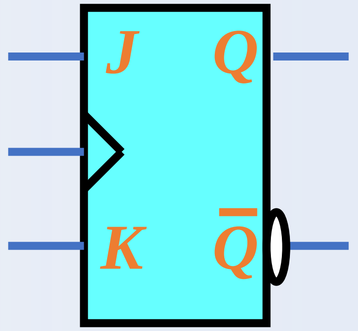


T	*Q*(*t*+1)
0	*Q*(*t*)
1	*Q'*(*t*)

$Q(t+1) = T \oplus Q$

Flip-Flop Characteristic Equations

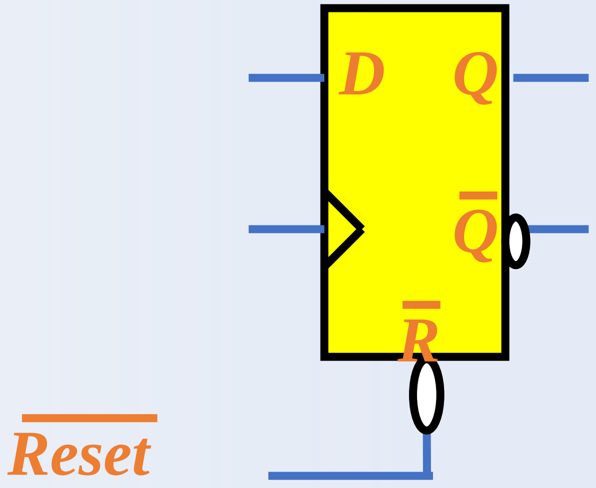
- Analysis / Derivation



J	K	$Q(t)$	$Q(t+1)$	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

Flip-Flops with Direct Inputs

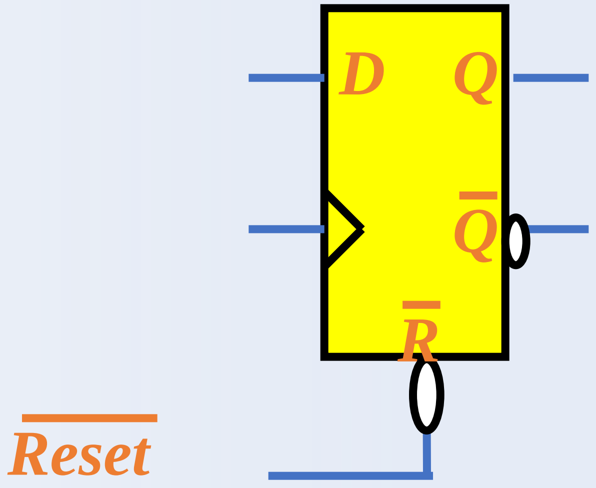
- Asynchronous Reset



R'	D	CLK	Q(t+1)
0	x	x	0

Flip-Flops with Direct Inputs

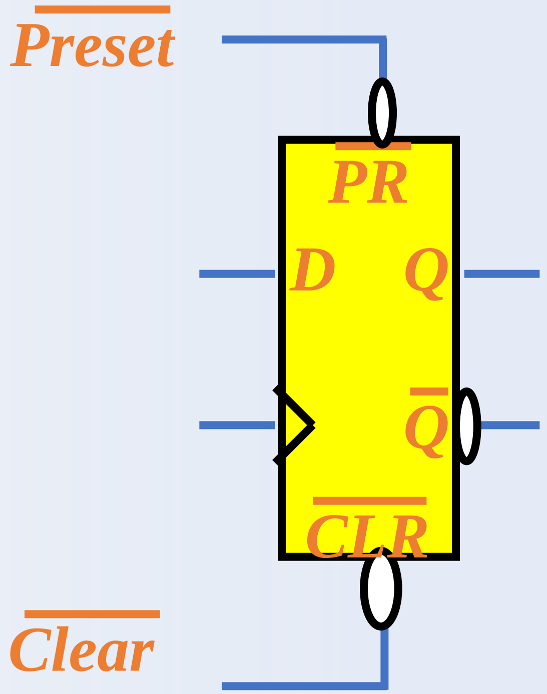
- Asynchronous Reset



R’	*D*	*CLK*	*Q(t+1)*
0	x	x	0
1	0	↑	0
1	1	↑	1

Flip-Flops with Direct Inputs

- Asynchronous Preset and Clear

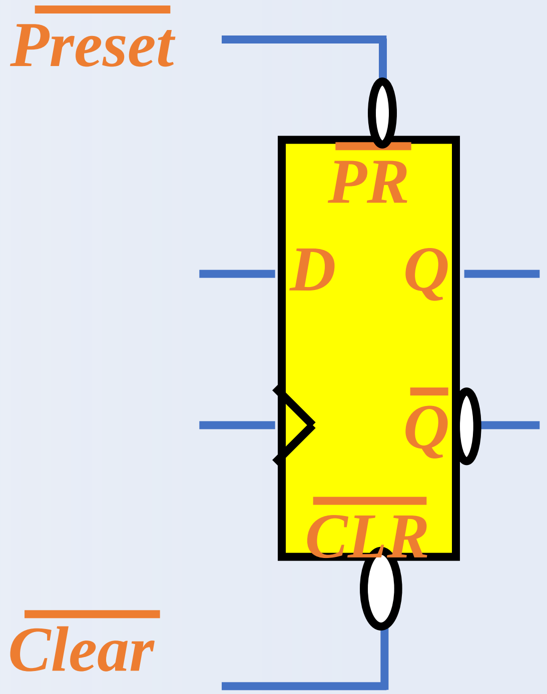


PR’	CLR’	D	CLK	Q(t+1)
1	0	x	x	0

CLR’ : Q is forced to 0

Flip-Flops with Direct Inputs

- Asynchronous Preset and Clear

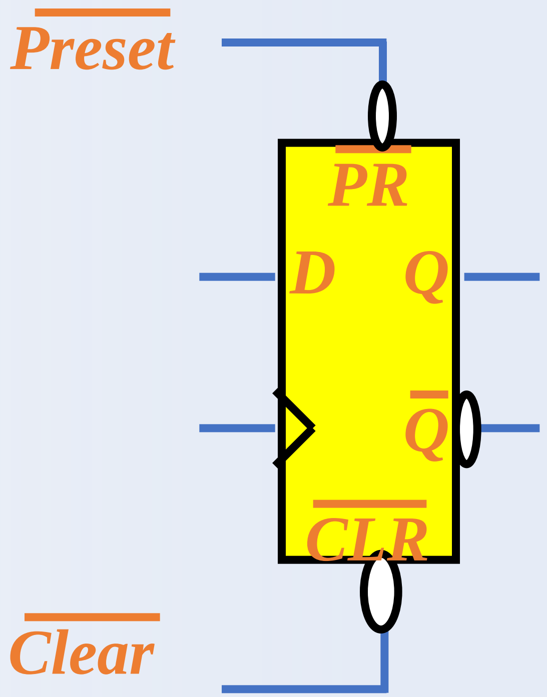


PR'	CLR'	D	CLK	Q(t+1)
1	0	x	x	0
0	1	x	x	1

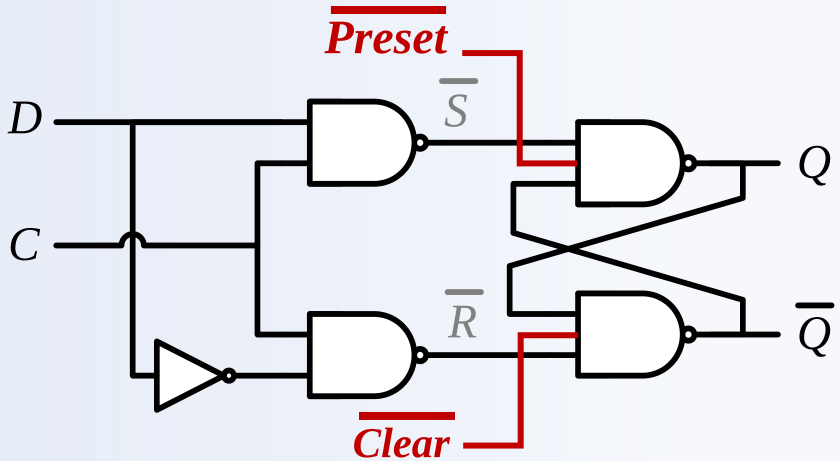
PR' : Q is forced to 1

Flip-Flops with Direct Inputs

- Asynchronous Preset and Clear



PR'	CLR'	D	CLK	Q(t+1)
1	0	x	x	0
0	1	x	x	1
1	1	0	↑	0
1	1	1	↑	1

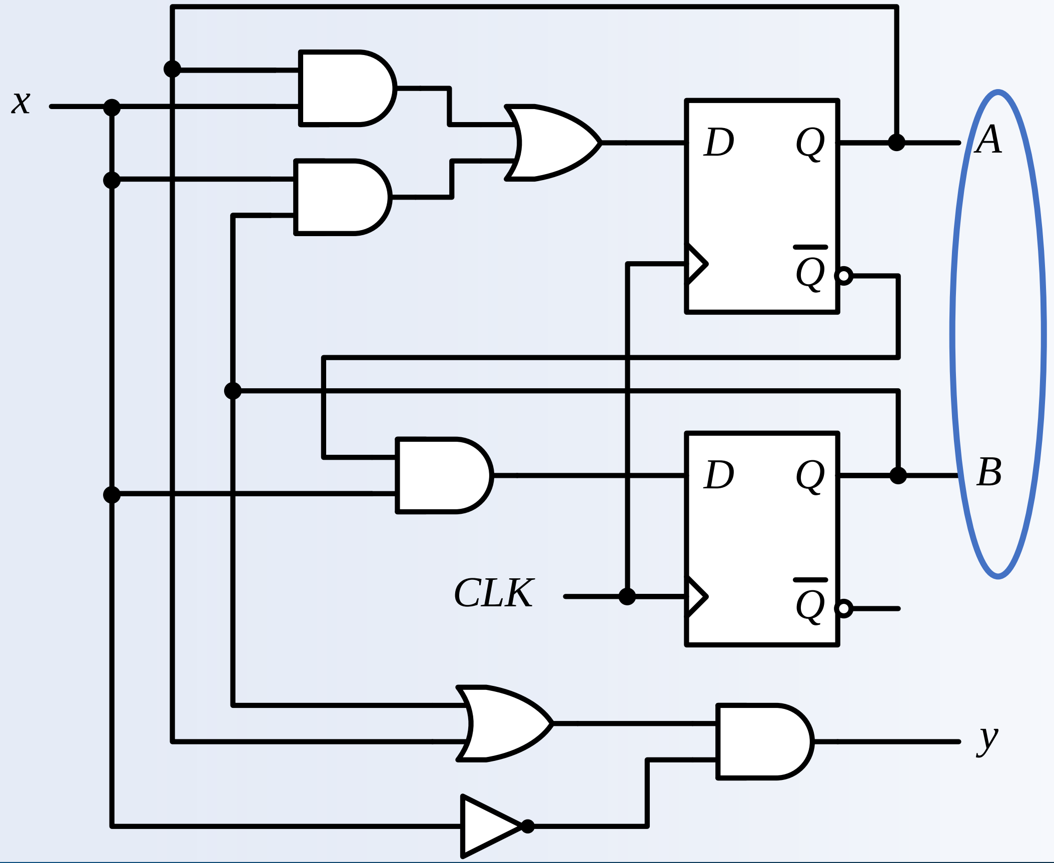


Analysis of Clocked Sequential Circuits

- The State
  - State = Values of all Flip-Flops

Example

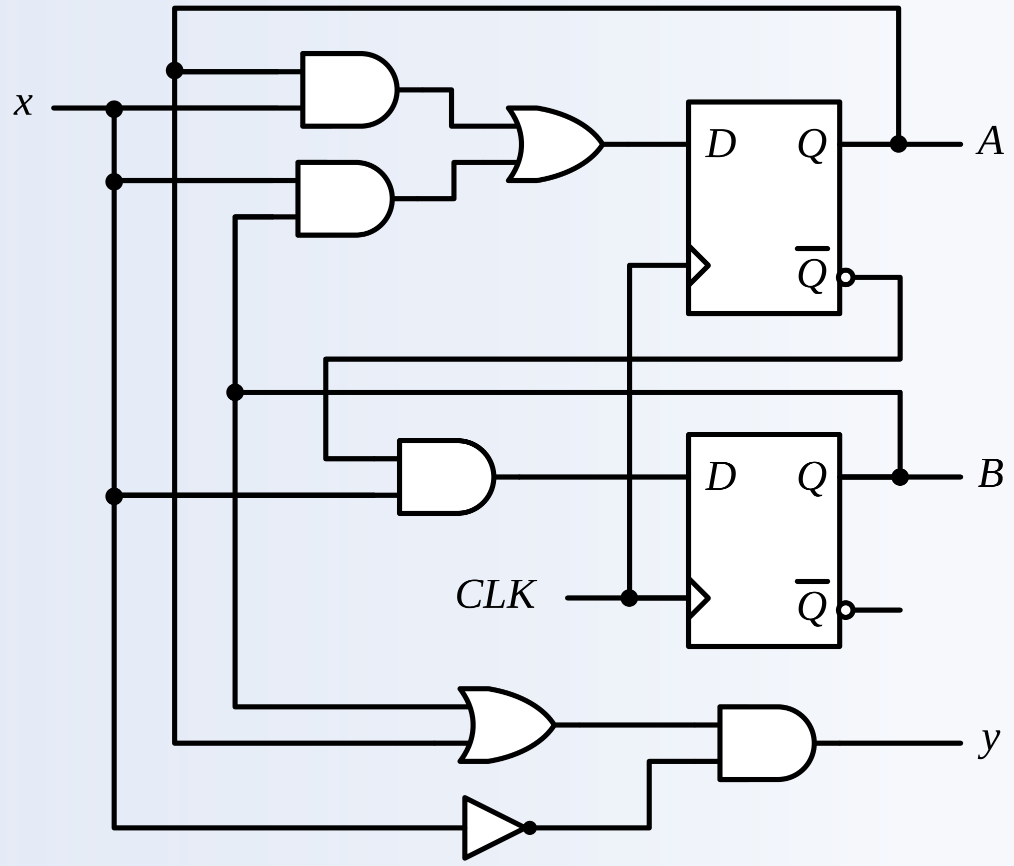
$A\ B = 0\ 0$



Analysis of Clocked Sequential Circuits

- State Equations

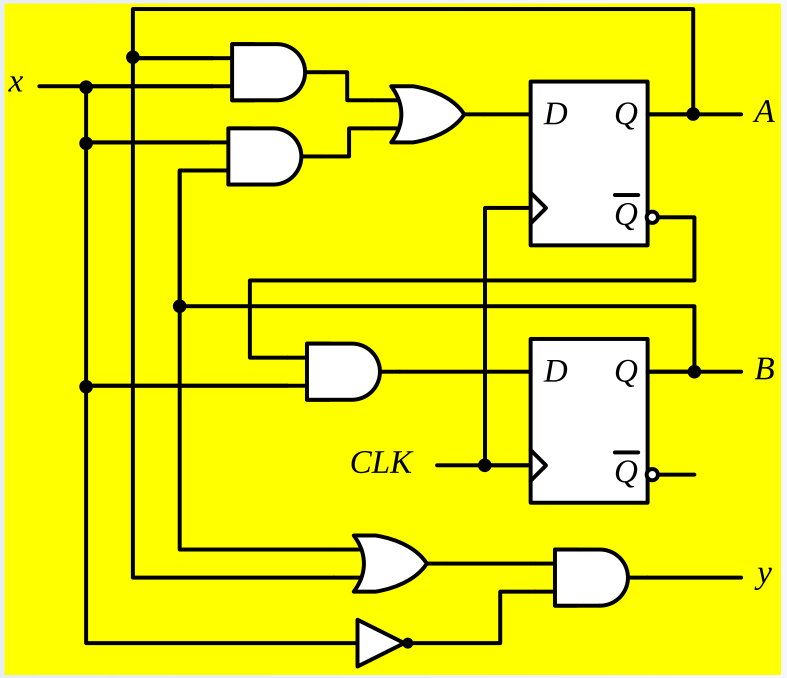
$$A(t+1) = D_A$$
$$= A(t)\,x(t) + B(t)\,x(t)$$
$$= A\,x + B\,x$$
$$B(t+1) = D_B$$
$$= A'(t)\,x(t)$$
$$= A'\,x$$
$$y(t) = [A(t) + B(t)]\,x'(t)$$
$$= (A + B)\,x'$$



Analysis of Clocked Sequential Circuits

- State Table (Transition Table)

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0
t			$t+1$		t



$$A(t+1) = A\,x + B\,x$$

$$B(t+1) = A'\,x$$

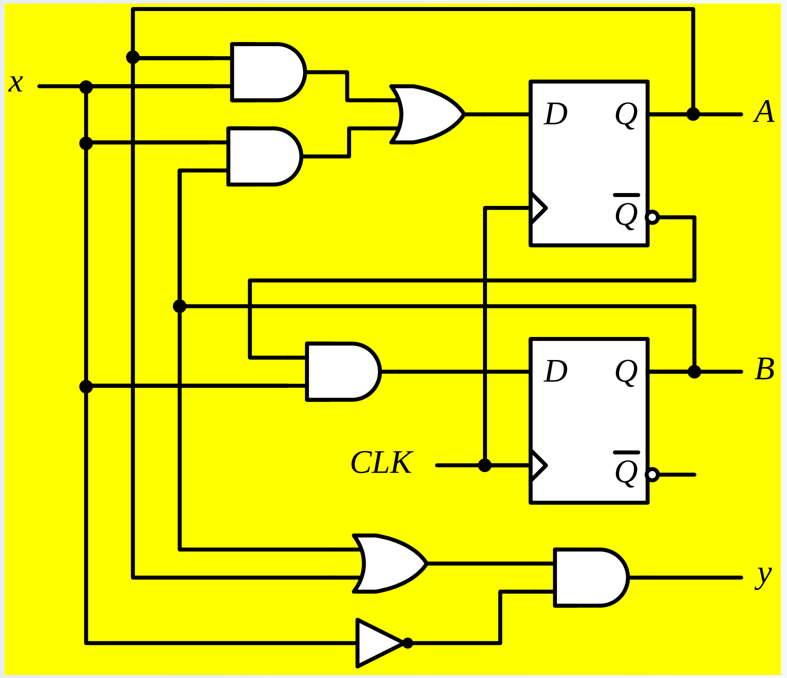
$$y(t) = (A + B)\,x'$$

Analysis of Clocked Sequential Circuits

- State Table (Transition Table)

Present State	Next State				Output	
	$x = 0$		$x = 1$		$x = 0$	$x = 1$
$A\ B$	A	B	A	B	y	y
0 0	0	0	0	1	0	0
0 1	0	0	1	1	1	0
1 0	0	0	1	0	1	0
1 1	0	0	1	0	1	0

t $t+1$ t



$$A(t+1) = A\,x + B\,x$$

$$B(t+1) = A'\,x$$

$$y(t) = (A + B)\,x'$$

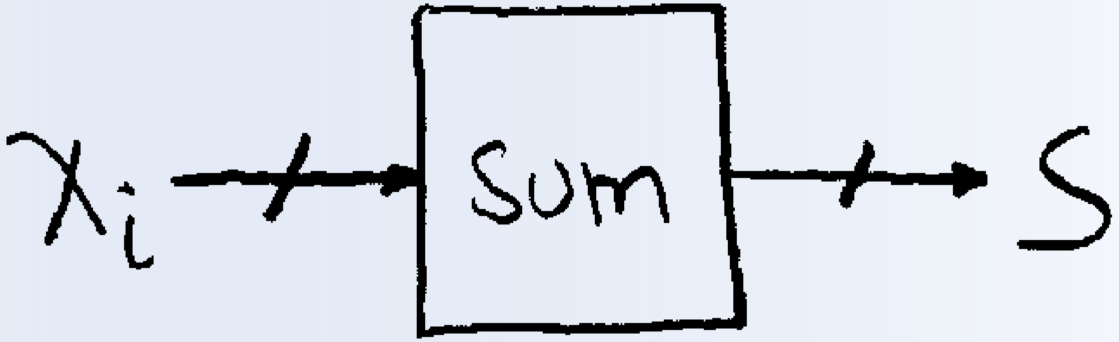
Accumulator

Uses for State Elements

- As a place to store values for some indeterminate amount of time:
  - Register files (like **x0-x31** on the RISC-V)
  - Memory (caches, and main memory)
- Help control the flow of information between combinational logic blocks.
  - State elements are used to hold up the movement of information at the inputs to combinational logic blocks and allow for orderly passage.

Accumulator Example

- Why do we need to control the flow of information?

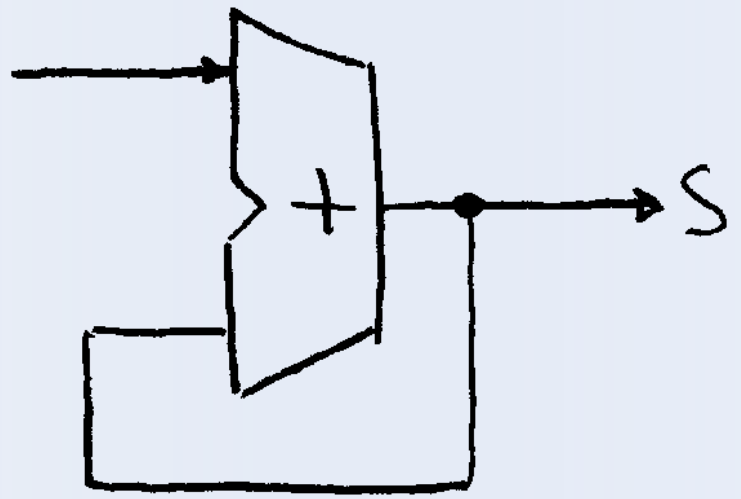


- Want:

```
S=0;
for (i=0;i<n;i++)
  S = S + X_i
```

- Assume
  - Each **X** value is applied in succession, one per cycle.
  - After **n** cycles the sum is present on **S**.

First try…Does this work?

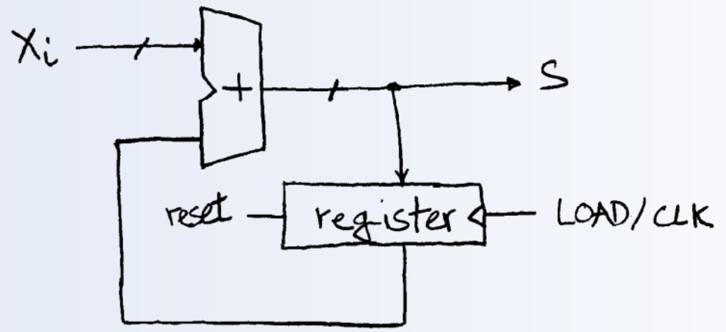


Feedback

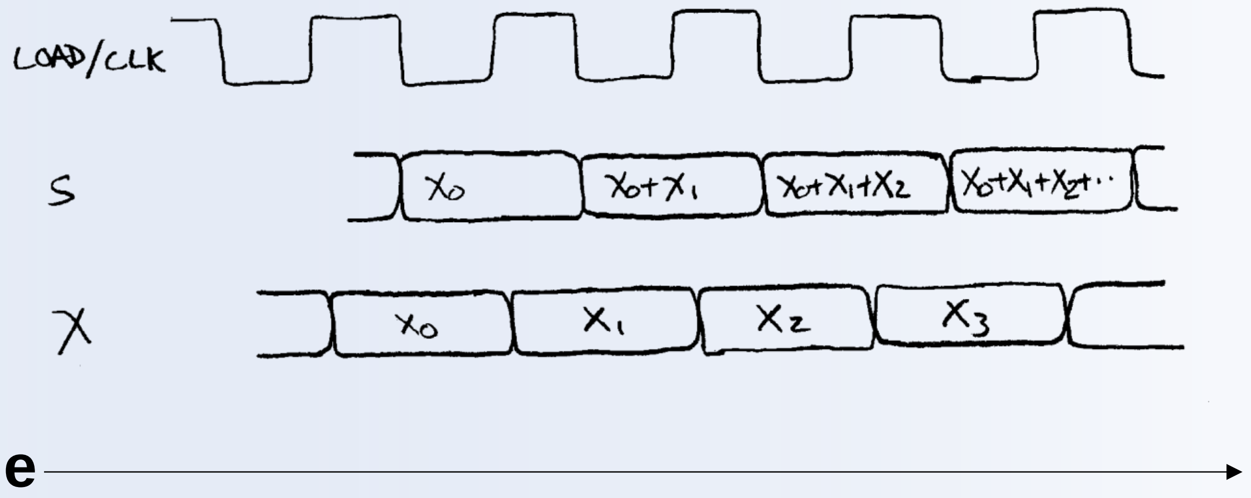
- Nope!
  - Reason #1… What is there to control the next iteration of the '**for**' loop?
  - Reason #2… How do we say: '**S=0**'?

Second try…How about this?

- Second try…How about this?

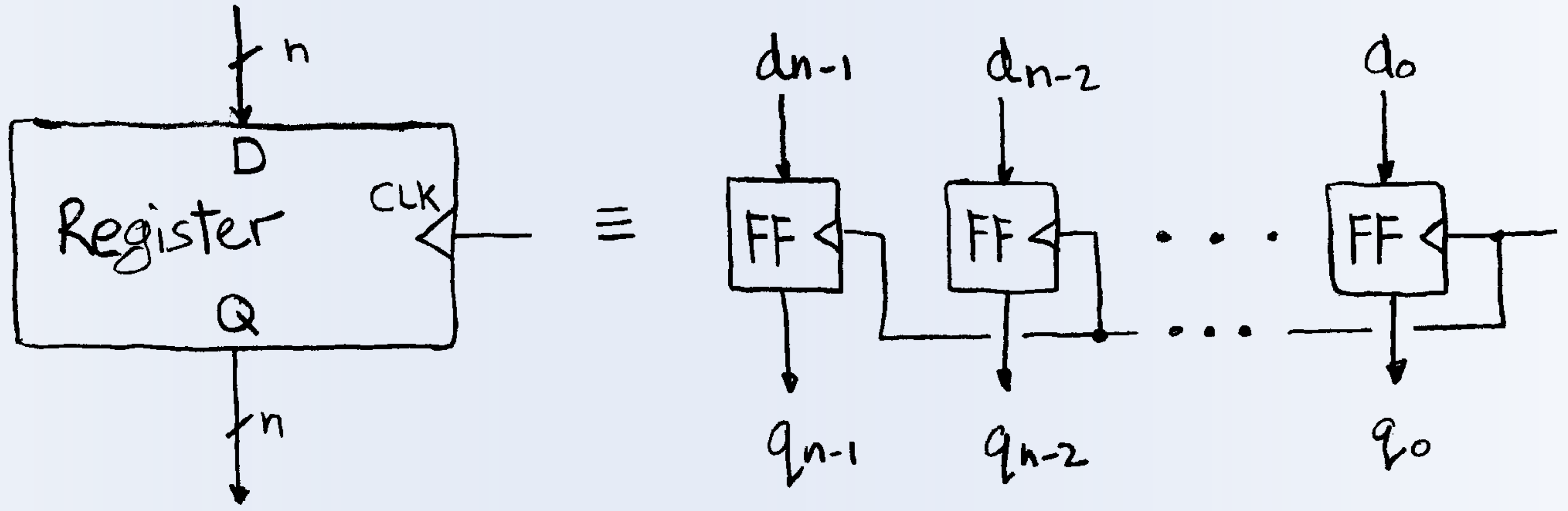


Rough timing…



Register Details
Flip-flops

Register Details…What’s inside?



n instances of a “Flip-Flop”

Flip-flop name because the output flips and flops between and 0,1

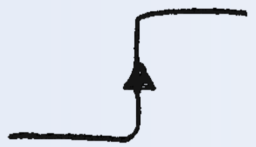
D is “data”, Q is “output”

Also called “D-type Flip-Flop”

- There used to be other types of flip-flops

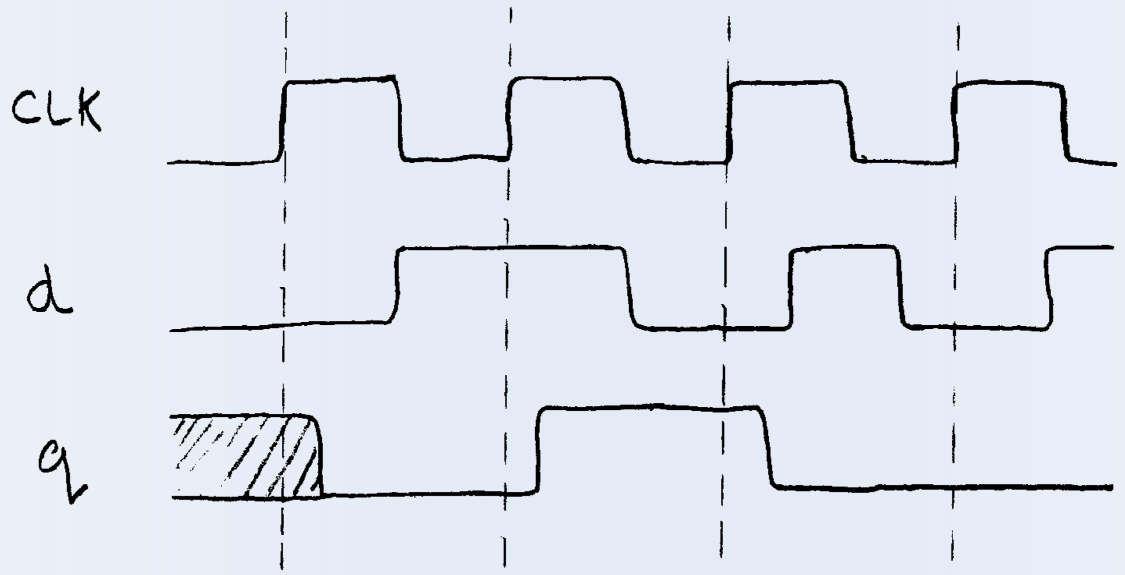
What's the timing of a Flip-flop? (1/2)

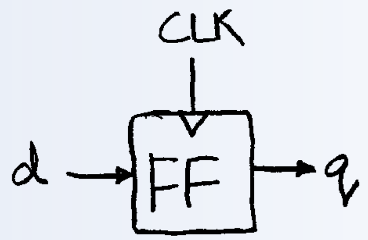
- Edge-triggered d-type flip-flop
  - This one is "rising edge-triggered"
  - Also called "positive edge"



There also exist "falling edge" FFs

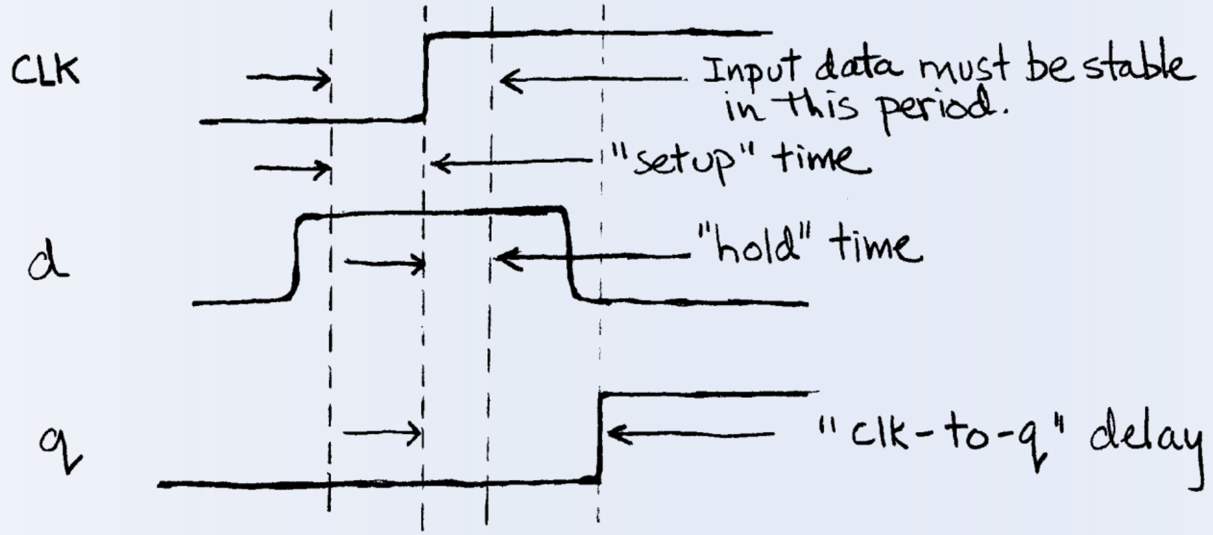
- "On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."
- Example waveforms:

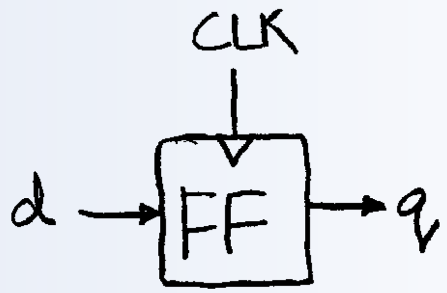




What's the timing of a Flip-flop? (2/2)

- Edge-triggered d-type flip-flop
  - This one is "rising edge-triggered"
- "On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."
- Example waveforms (more detail):





Registers

- Sequential circuits are classified based in their function, e.g., registers.
- Register: A group of flip-flops each storing binary information.
- Registers include flip-flops and gates: flip-flops hold the information, gates control how the information is transferred to the register.
- Counter A sequential circuit that counts clock pulses and stores the count in binary form. It consists of flip-flops arranged in a sequence to track the number of occurrences of an event, typically driven by a clock signal.

4-bit Register

Loads in parallel

Clear: Cleans the output to all 0’s.

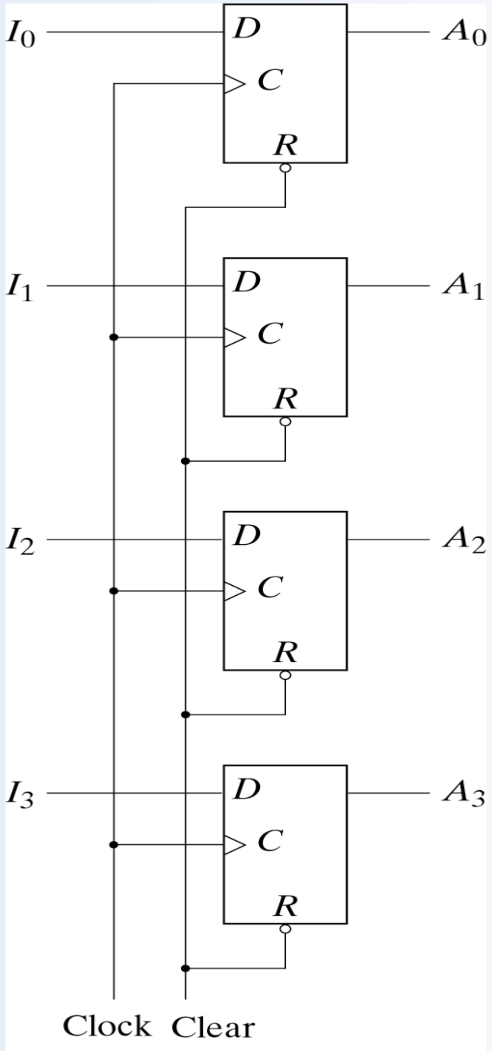


Fig. 6-1 4-Bit Register

Shift Registers

A register capable of shifting its binary information in one or both directions is called the shift register.

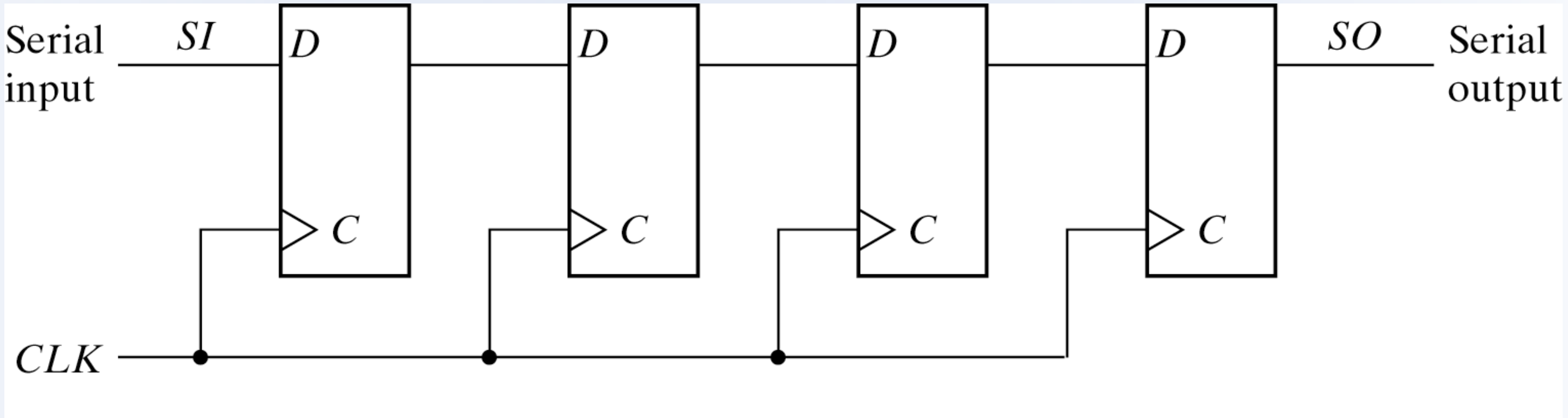


Fig. 6-3 4-Bit Shift Register

Serial Transfer

A digital system is in the serial mode when information is processed one bit at a time.

Serial transfer of information from A to B:

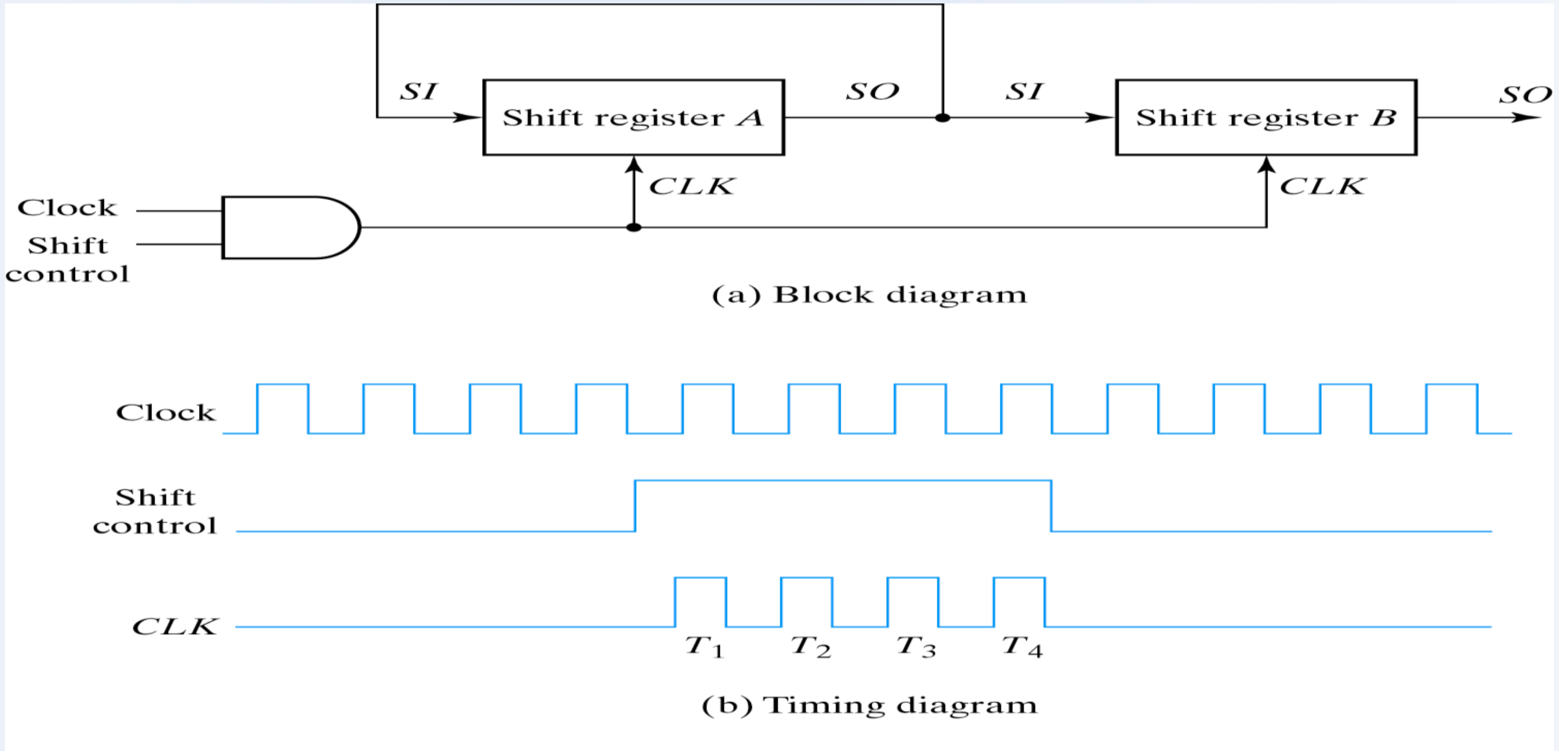


Fig. 6-4 Serial Transfer from Register *A* to register *B*

Remember 4-bit Parallel Adder Circuit?

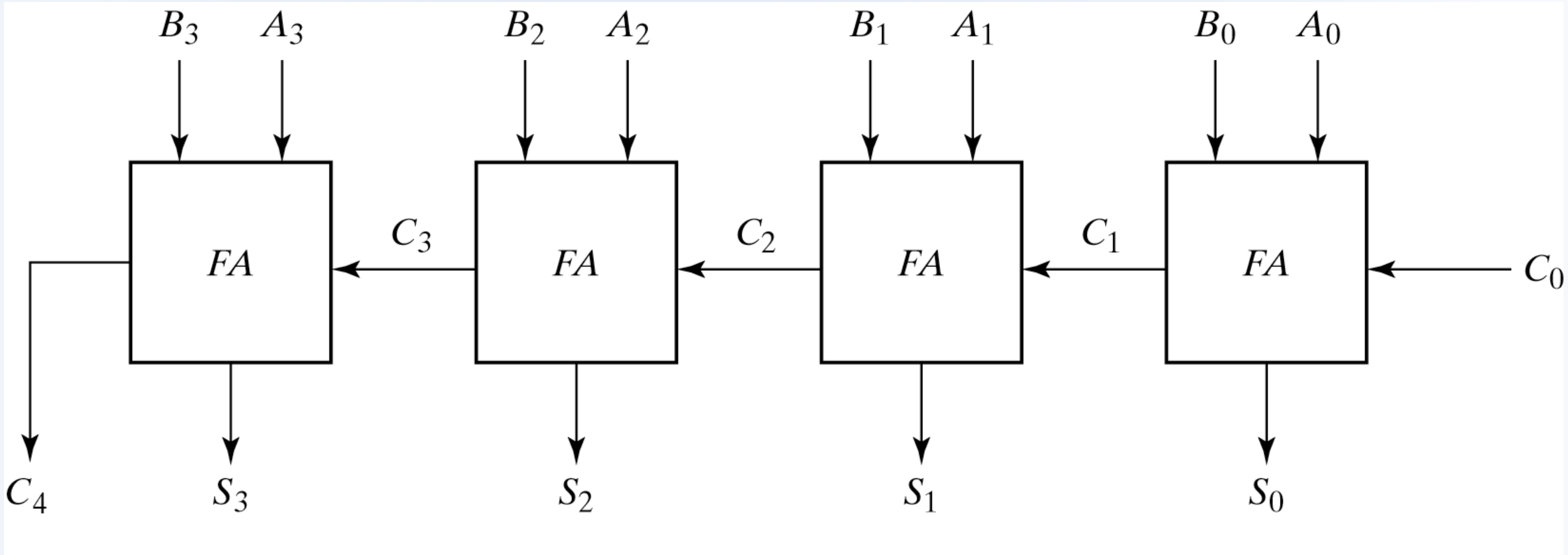
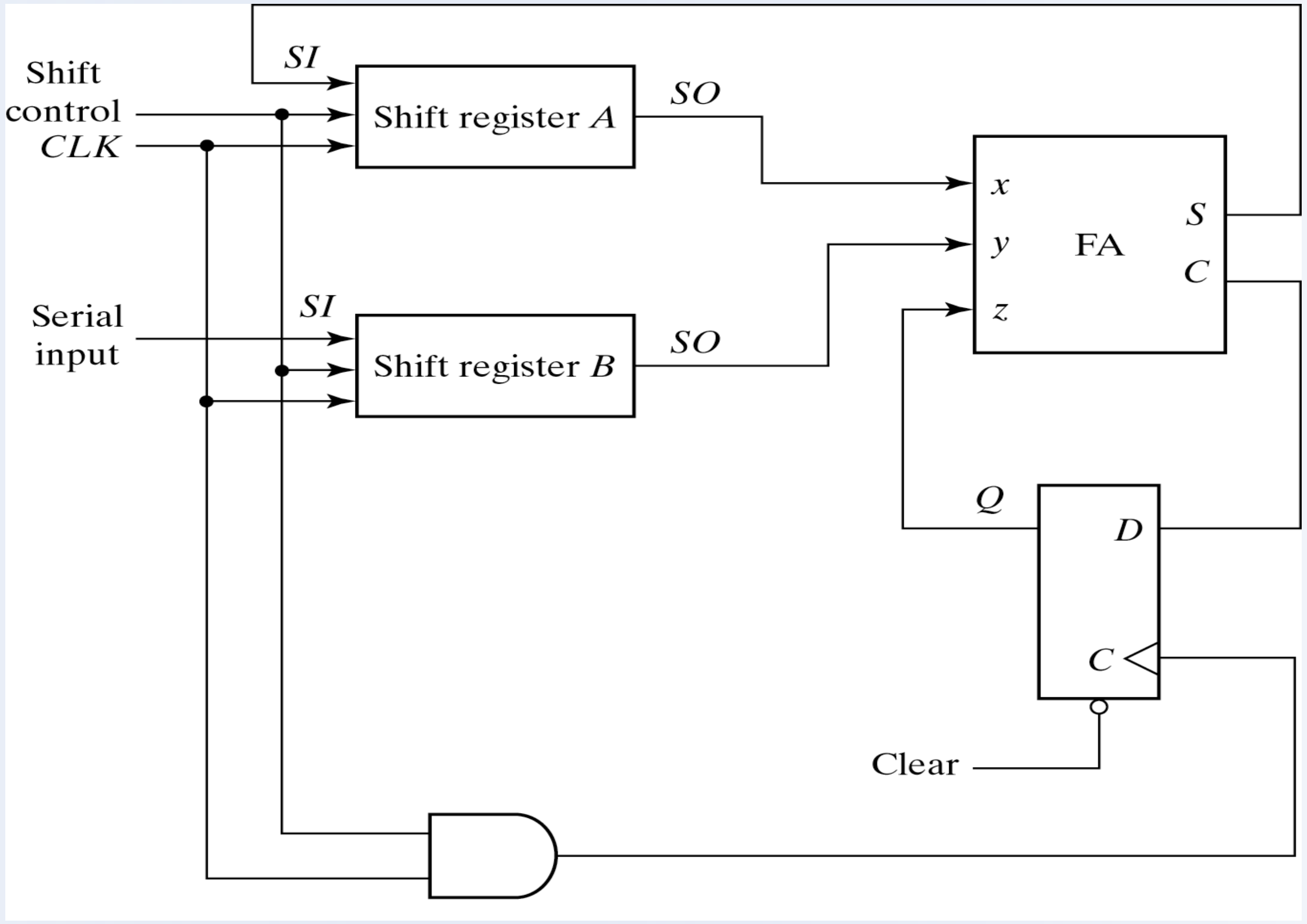


Fig. 4-9 4-Bit Adder

Serial Addition

Slower compared to parallel addition but uses less resources.



Universal Shift Register

- A register capable of shifting in both directions and loading in parallel.

Stores Information

Multiplexer Inputs:
0: No Change
1:Shift Right
2:Shift Left
3:Parallel load

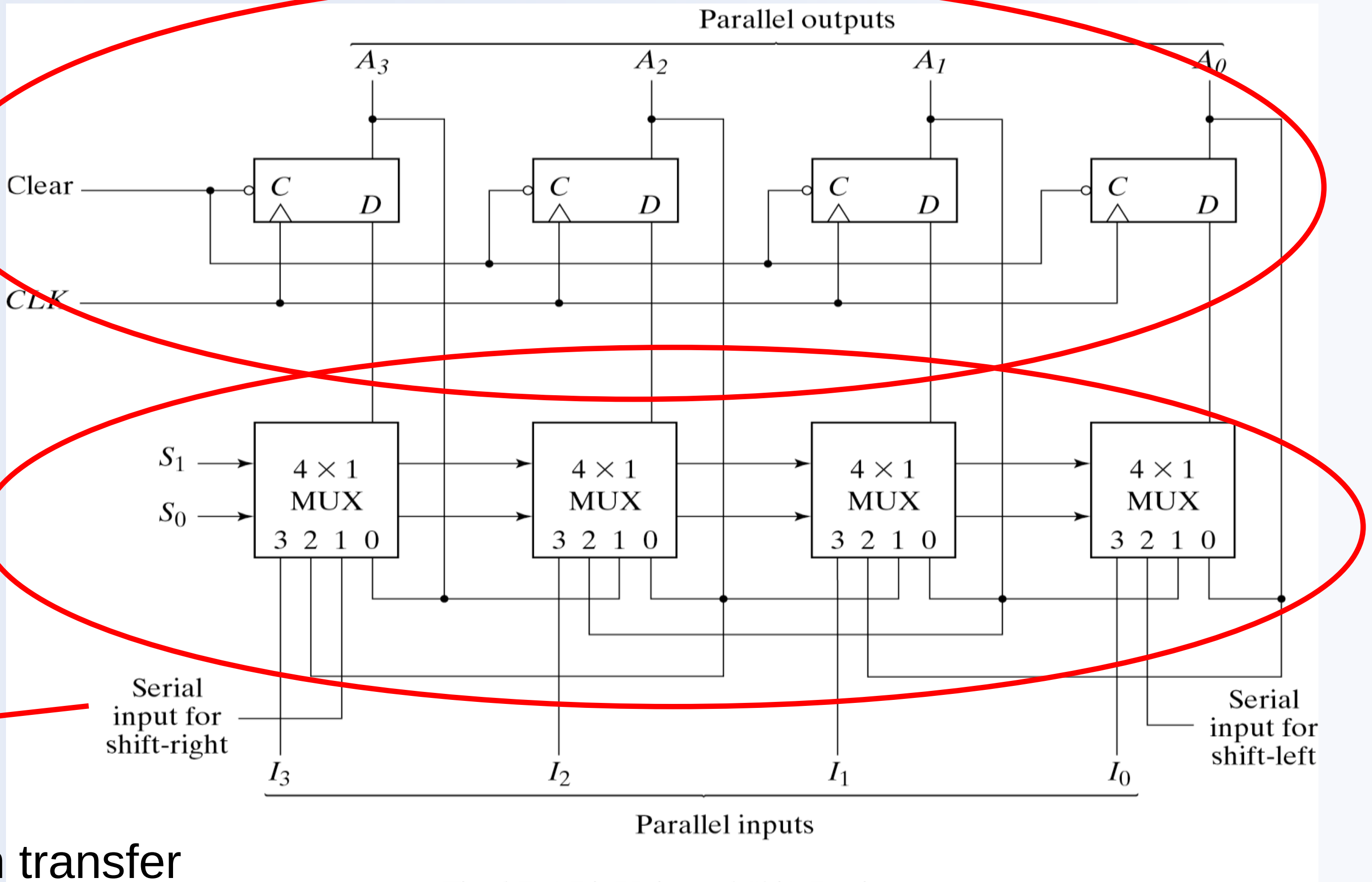


Fig. 6-7 4-Bit Universal Shift Register

Controls information transfer

Counters

Ripple Counters

- A register that goes through a prescribed sequence of states is called a counter.
- There are two groups of counters: Ripple counters and Synchronous counters.
- Ripple counters: The flip-flop output triggers other flip-flops.
- Synchronous counters count the clock.

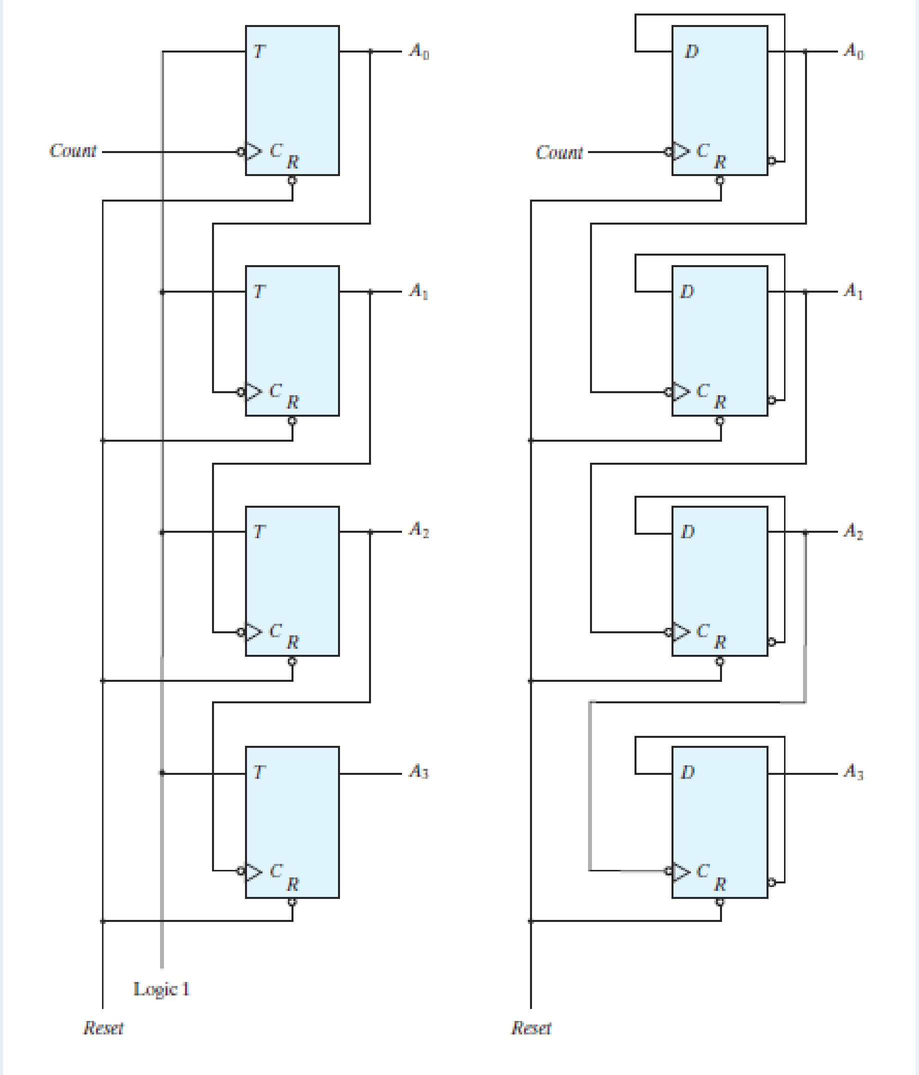
Binary Ripple Counter

- A binary ripple counter consists of a series of complementing flip-flops, with the output of each flip-flop connected to the next higher order.

- Examples of complementing flip-flops are T and D (with the output complement connected to the input) flip-flop.
- Binary Count Sequence

A3	A2	A1	A0	
0	0	0	0	A0 is complemented with each count pulse
0	0	0	1	A1 is complemented when A0 goes from 1 to 0
0	0	1	0	A2 is complemented when A1 goes from 1 to 0
0	0	1	1	A3 is complemented when A2 goes from 1 to 0
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	

Examples of Binary Ripple Counters



(a) With *T* flip-flops (b) With *D* flip-flops

Binary Ripple Counter

- Count-down counter: A binary counter with reverse count: Starts from 15 goes down.

- In a count-down counter the least significant bit is complemented with every count pulse. Any other bit is complemented if the previous bit goes from 0 to 1.

- Try to design with above description of the count down counter

BCD Ripple Counter

A BCD counter starts from 0 ends at 9.

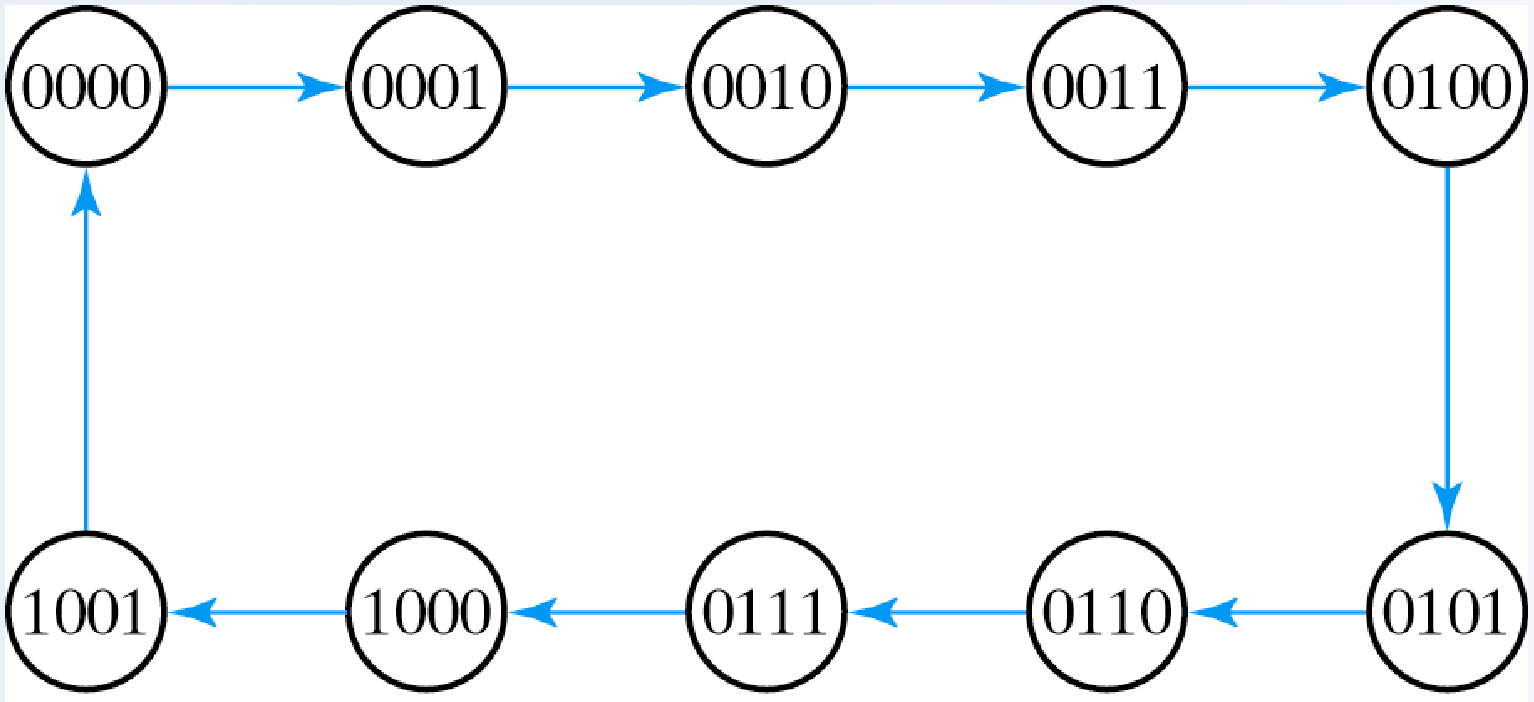


Fig. 6-9 State Diagram of a Decimal BCD-Counter

Logic Diagram of BCD Ripple Counter

Q1 is applied to the C inputs of Q2 and Q8

Q2 is applied to the C input of Q4

J and K are connected to either 1 or flip-flop outputs

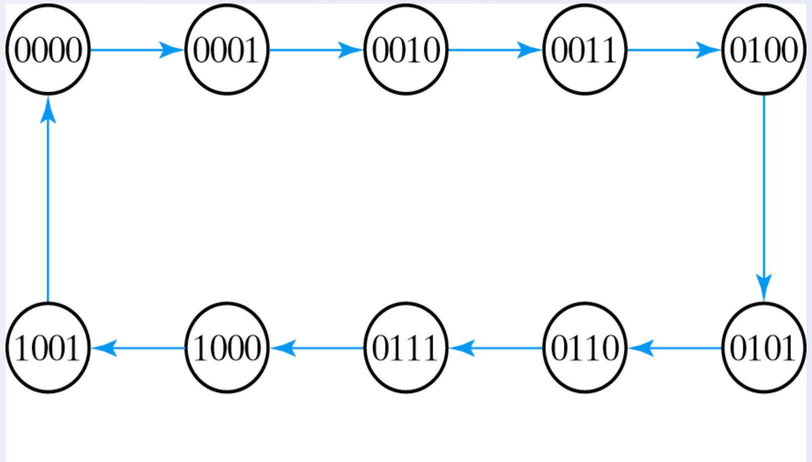
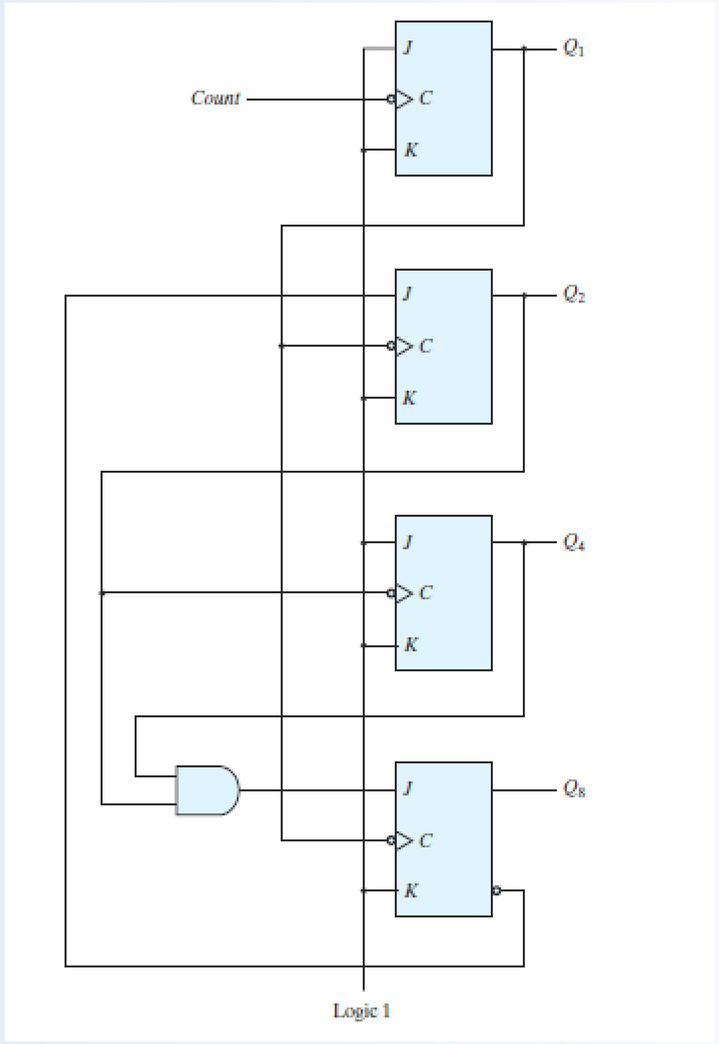


Fig. 6-9 State Diagram of a Decimal BCD-Counter



Logic Diagram of BCD Ripple Counter

Verification: Does the circuit follow the states?

Q1 is complemented with every count (J=K=1)

Q2 complements if Q1 goes from 1 to 0 and Q8 is 0

Q2 remains 0 if Q8 becomes 1

Q4 complements if Q2 goes from 1 to 0

Q8 remains 0 as long as Q2 or Q4 is 0

When Q2 and Q4 are 1, Q8 complements when Q1 goes from 1 to 0. Q8 clears at the next Q1 transition.

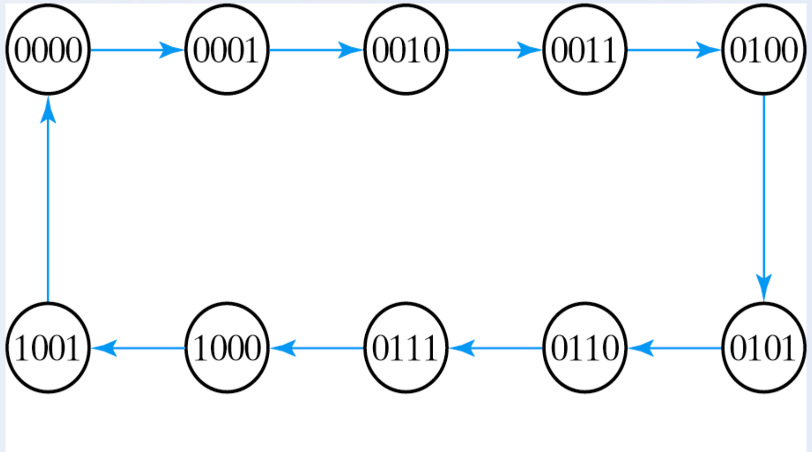


Fig. 6-9 State Diagram of a Decimal BCD-Counter

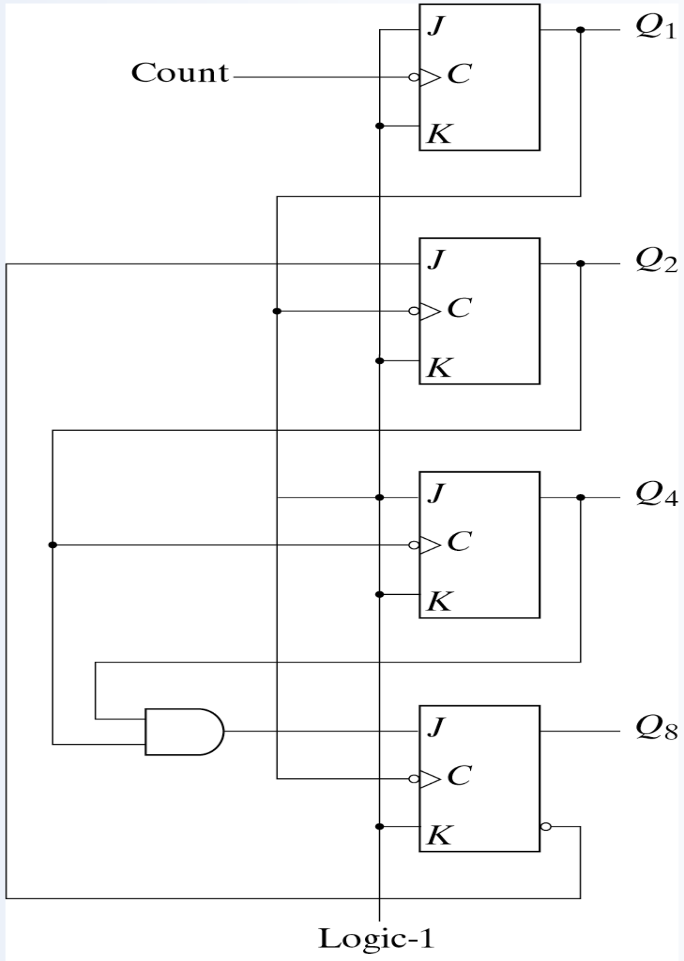


Fig. 6-10 BCD Ripple Counter

Three-Decade Decimal BCD Counter

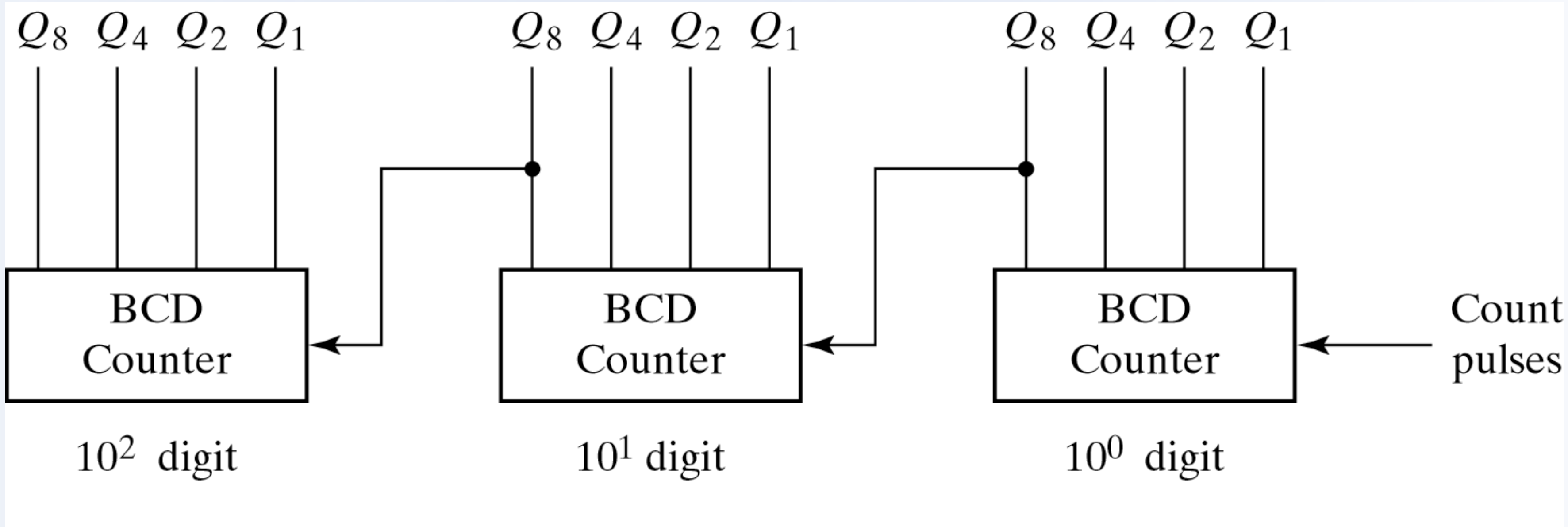


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

Counts from 0 to 999: When Q8 goes from 1 to 0 the next higher order decade is triggered

4-bit Synchronous Binary Counters

A flip-flop is complemented if all lower bits are 1.

A3	A2	A1	A0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

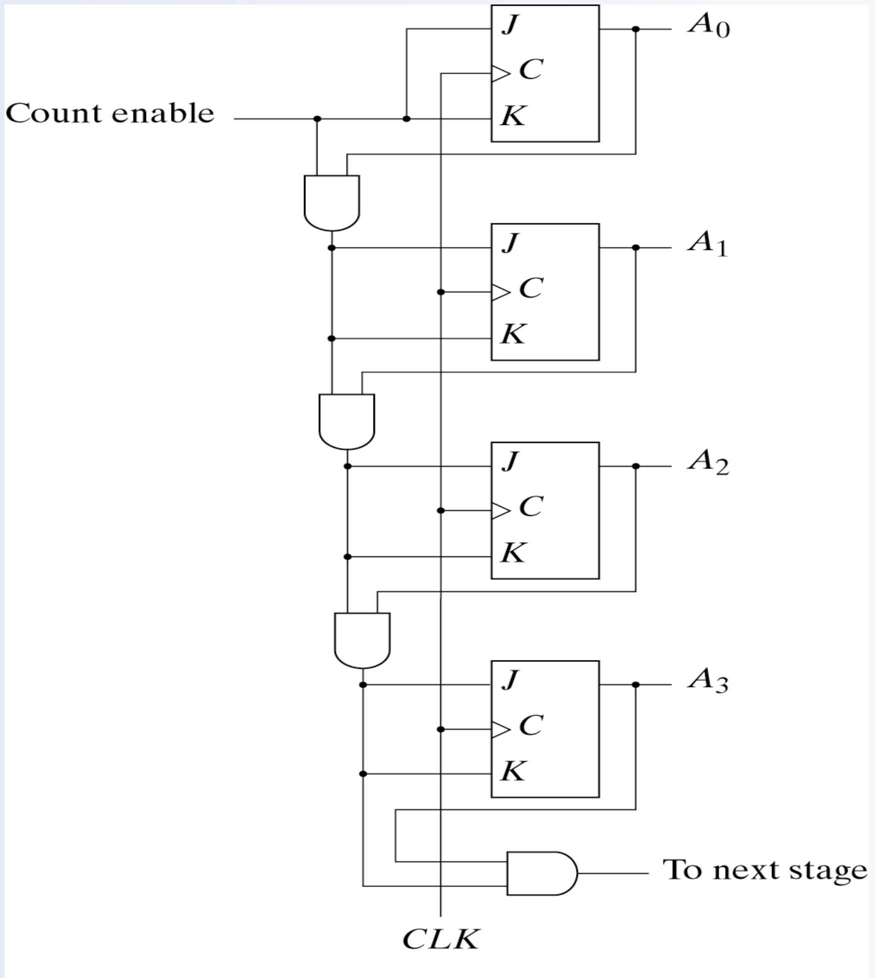


Fig. 6-12 4-Bit Synchronous Binary Counter

4-bit Up-Down Binary Counters

In a down binary counter
a) The least significant bit is always complemented
b) a bit is complemented if all lower bits are 0.

Change an up counter to a down counter:
The AND gates should come from the
complement outputs instead of the normal one

Up = 1, Down =0: Circuit counts up since input comes
from Normal output

Up = 0, Down =1: Circuit counts down since input comes
from Complemented output

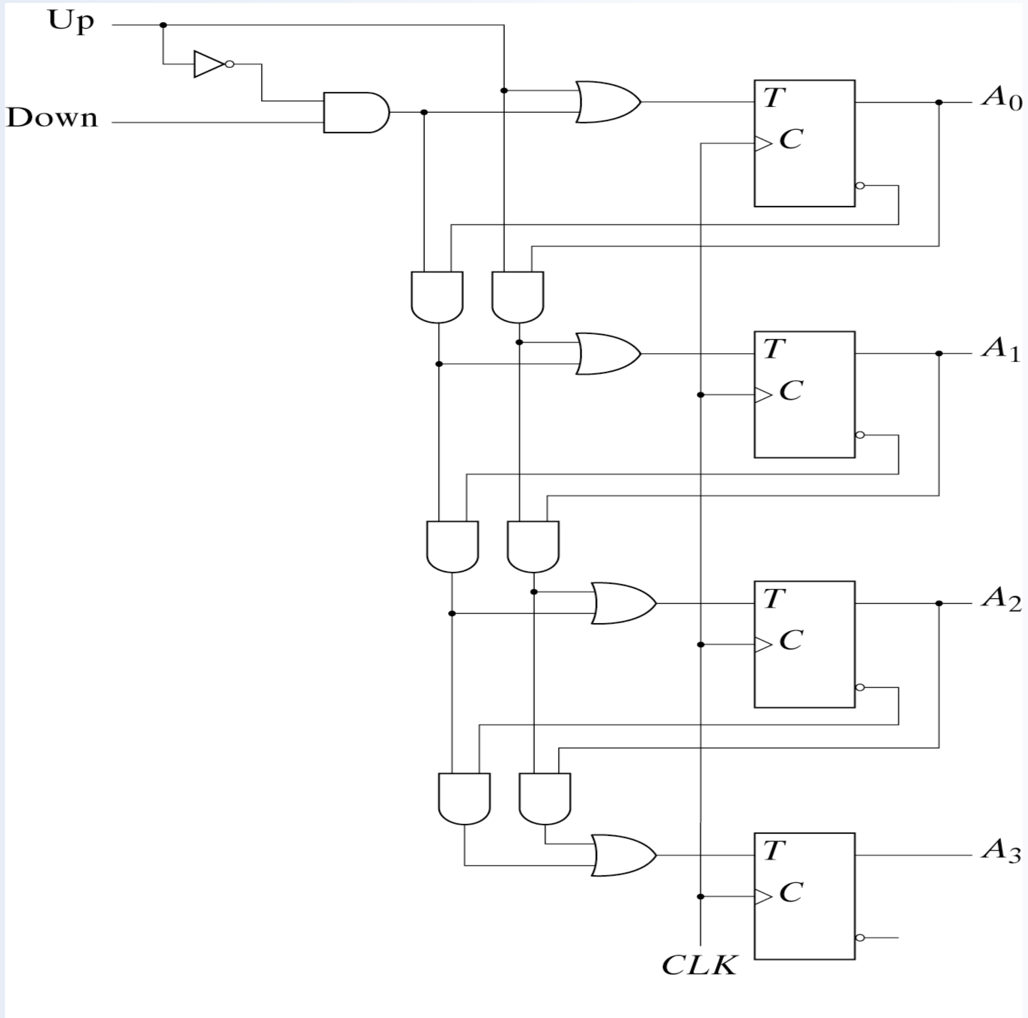


Fig. 6-13 4-Bit Up-Down Binary Counter

BCD counter with parallel load

In part a, 1001 is detected. In part b, 1010 is detected.
In part a, LOAD is set to 1 and effective next cycle. In part b, counter is immediately cleared

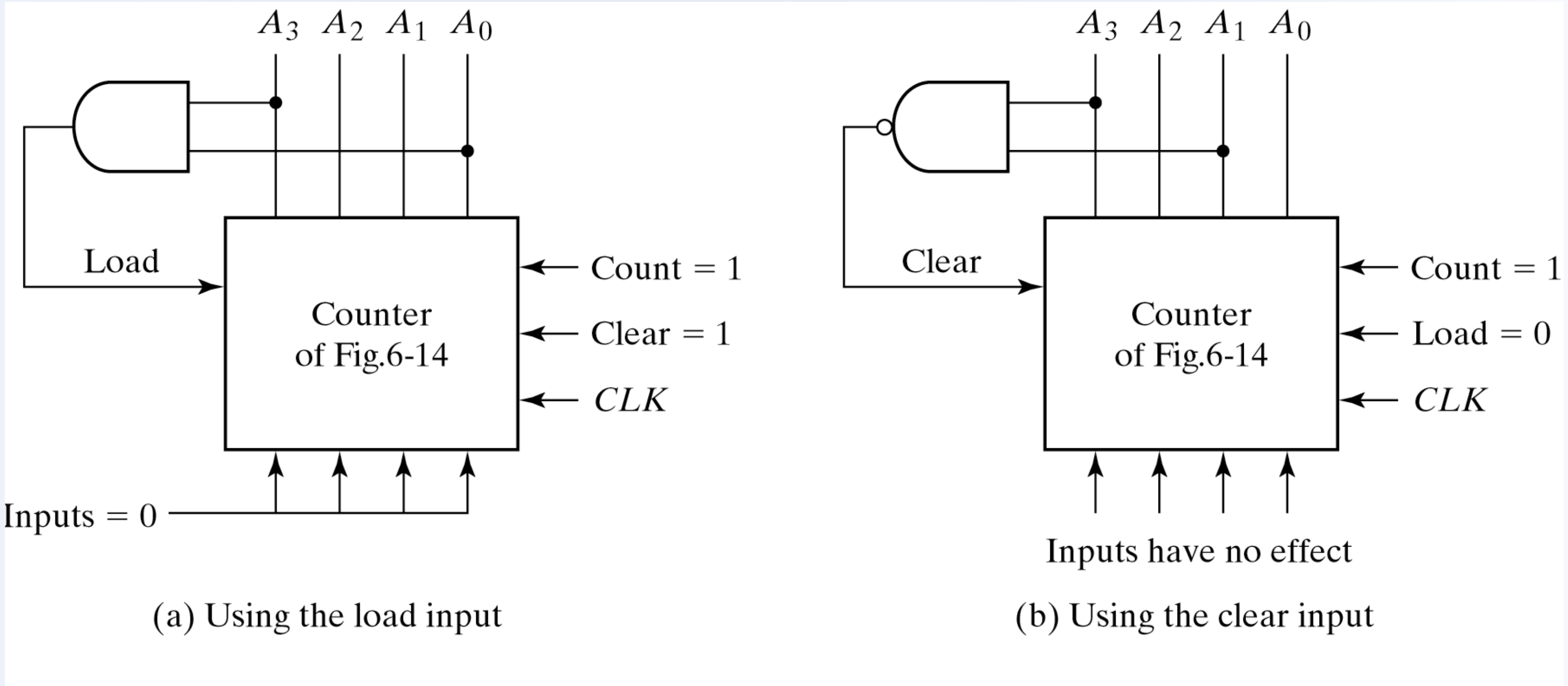
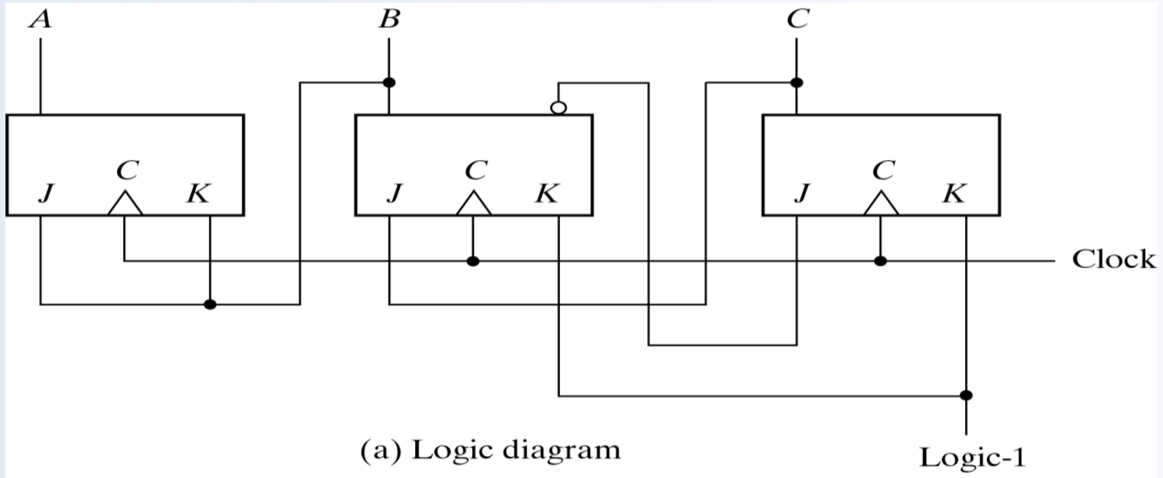


Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

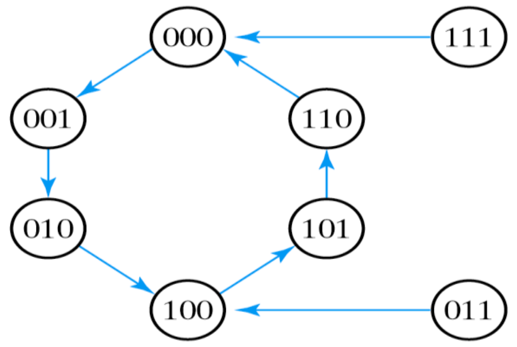
Other Counters: Counters with unused states

Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	JA	KA	JB	KB	JC	KC
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

JA=KA=B
JB=C, KB=1
JC=B' KC=1



(a) Logic diagram



(b) State diagram

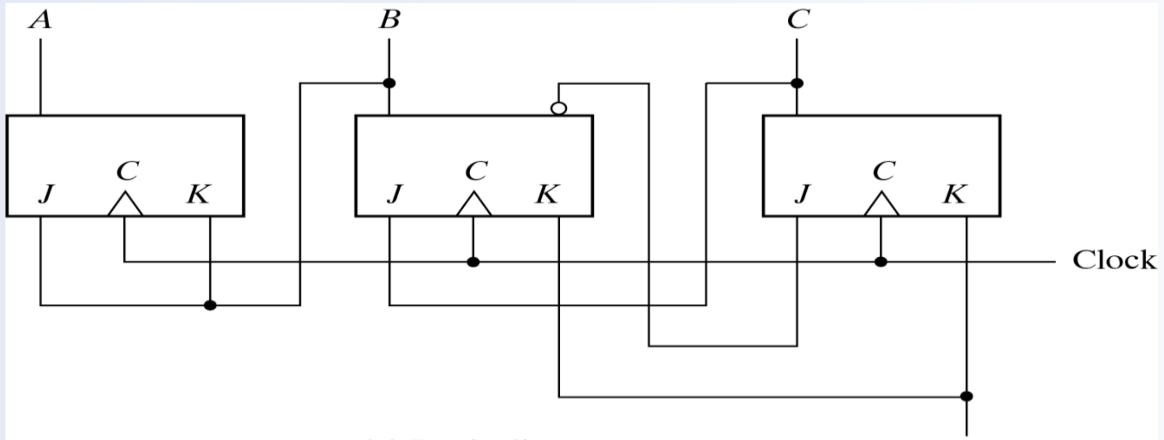
Fig. 6-16 Counter with Unused States

Other Counters: Counters with unused states

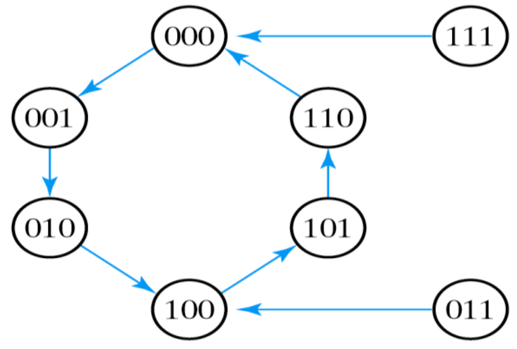
What happens if we fall in unused states?

In this case, 111 results in 000. 011 results in 100.

The Counter is self-correcting.



(a) Logic diagram



(b) State diagram

Fig. 6-16 Counter with Unused States

Other Counters: Ring Counter

A ring counter is a counter with ONLY 1 flip-flop set to 1 at any particular time, all other are cleared.

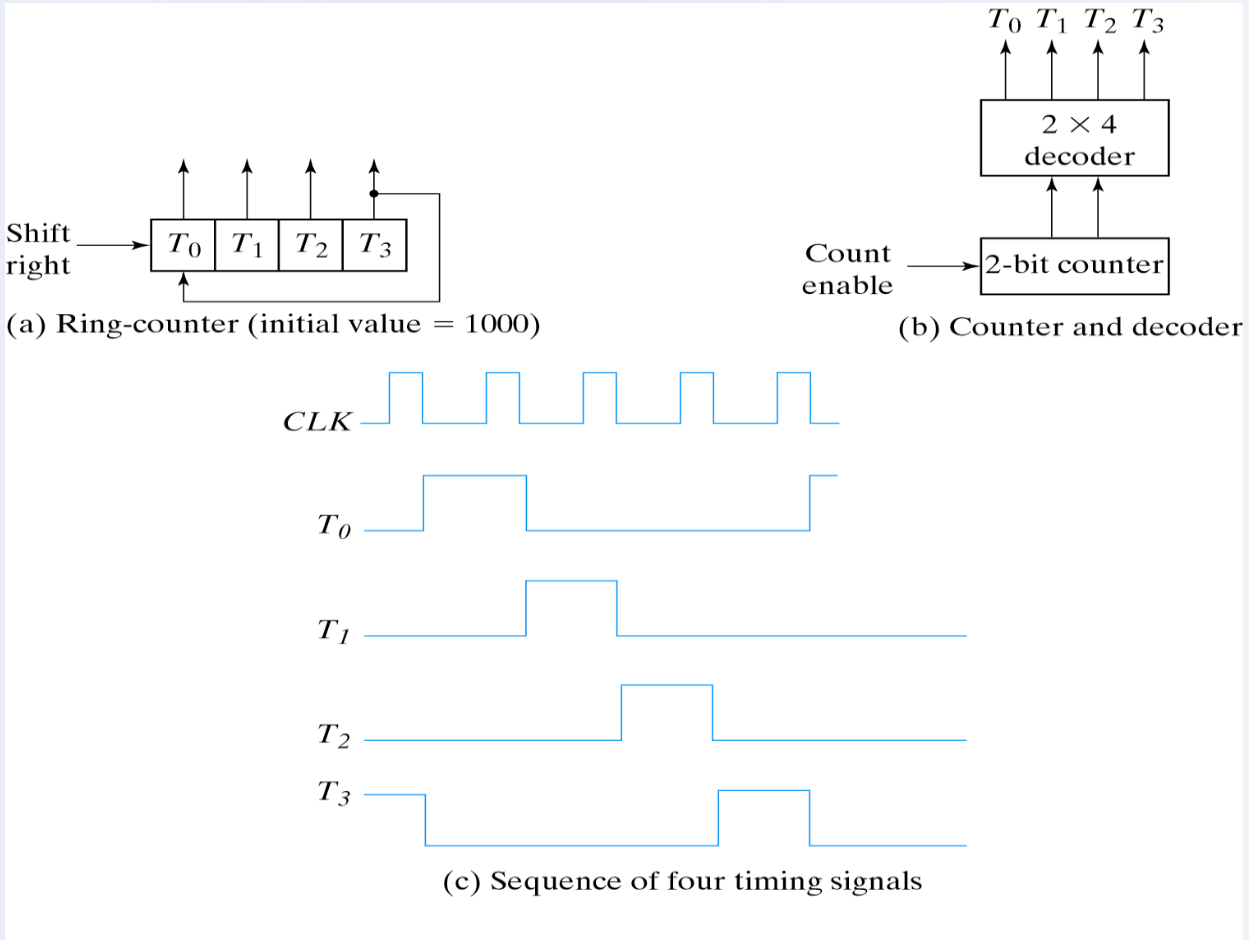
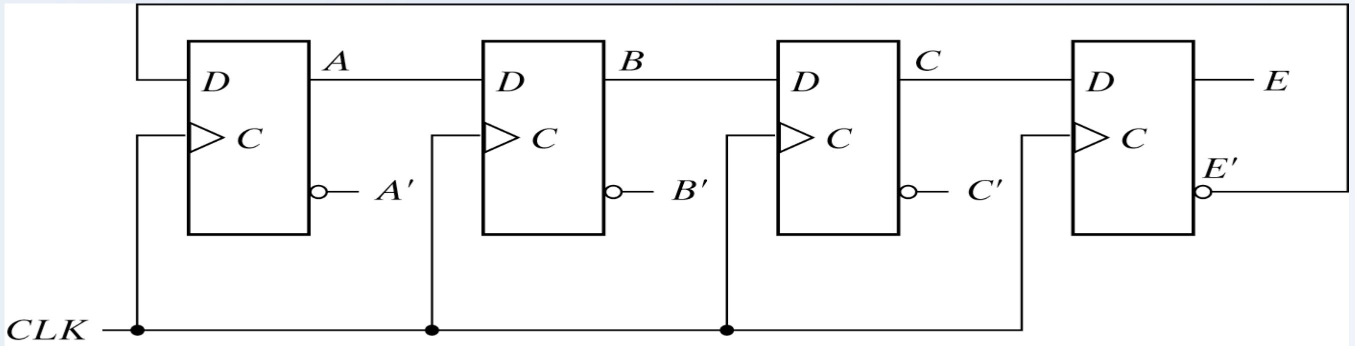


Fig. 6-17 Generation of Timing Signals

Other Counters: Johnson Counter

A 4 flip-flop ring counter that produces 8 states (not 4).



(a) Four-stage switch-tail ring counter

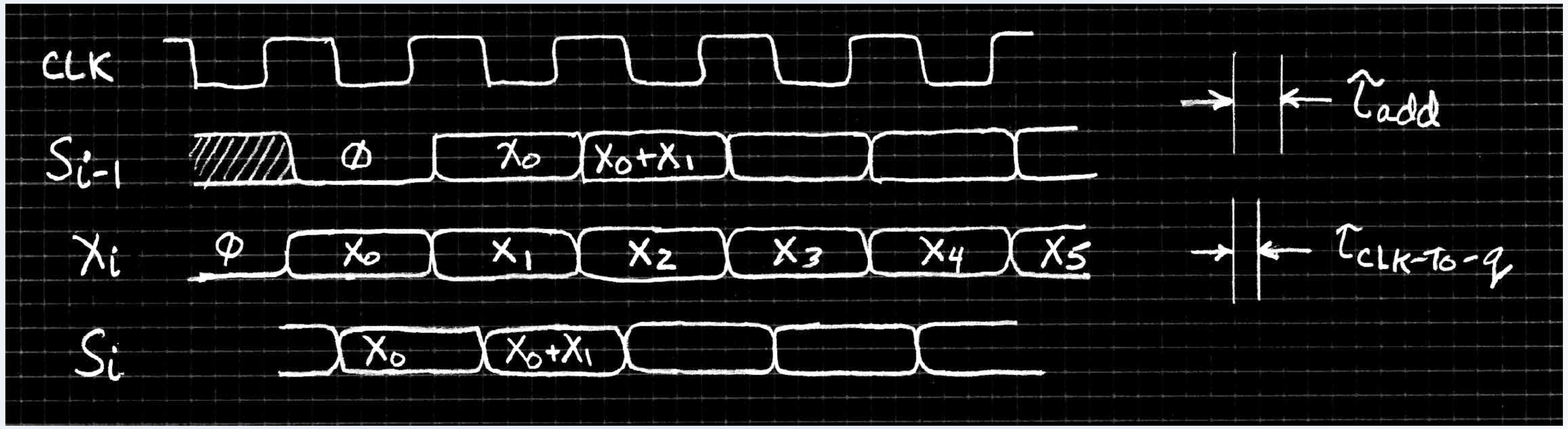
Sequence number	Flip-flop outputs A	B	C	E	AND gate required for output
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

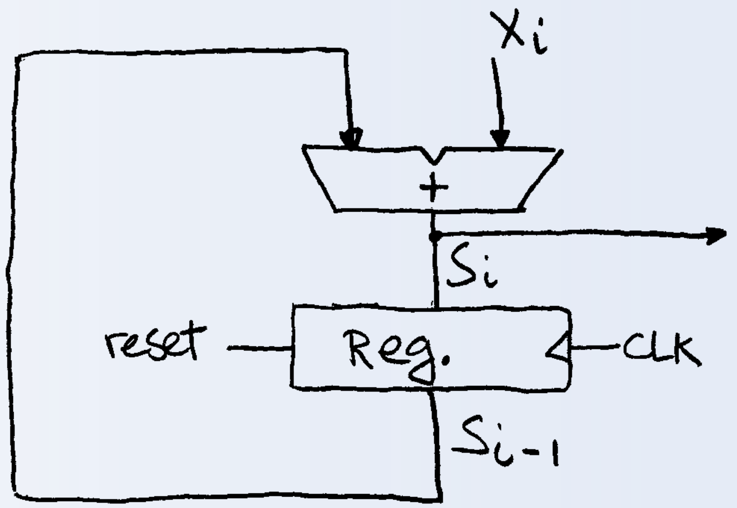
(b) Count sequence and required decoding

Fig. 6-18 Construction of a Johnson Counter

Accumulator Revisited

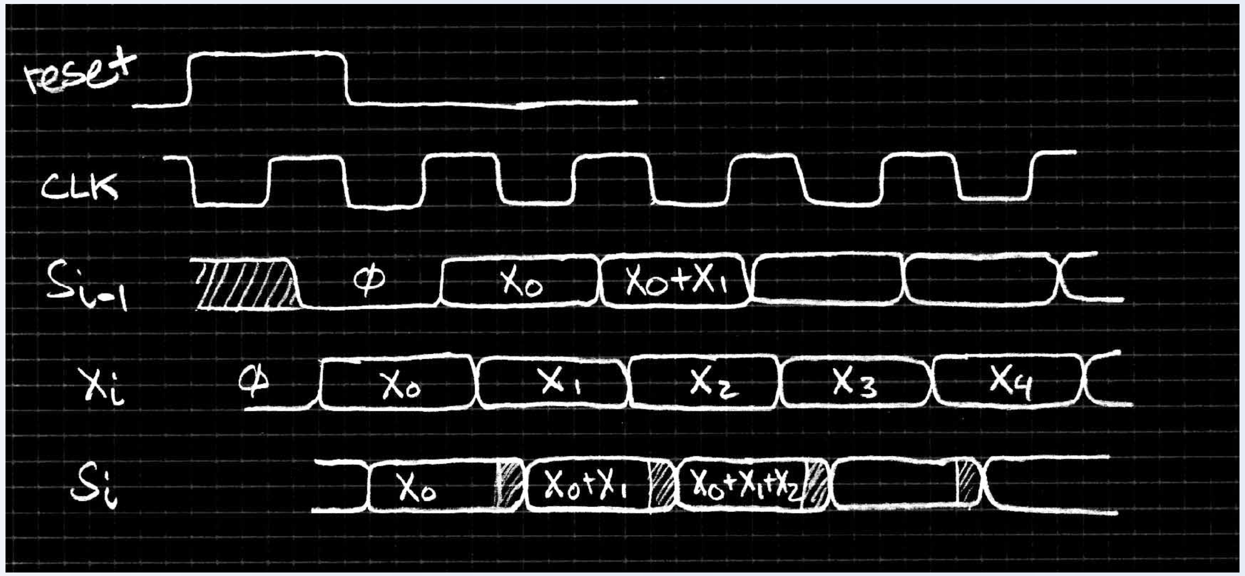
Accumulator Revisited (proper timing 1/2)

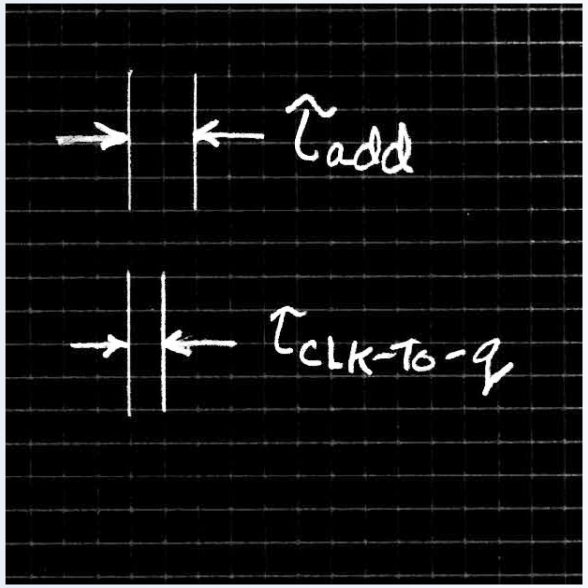


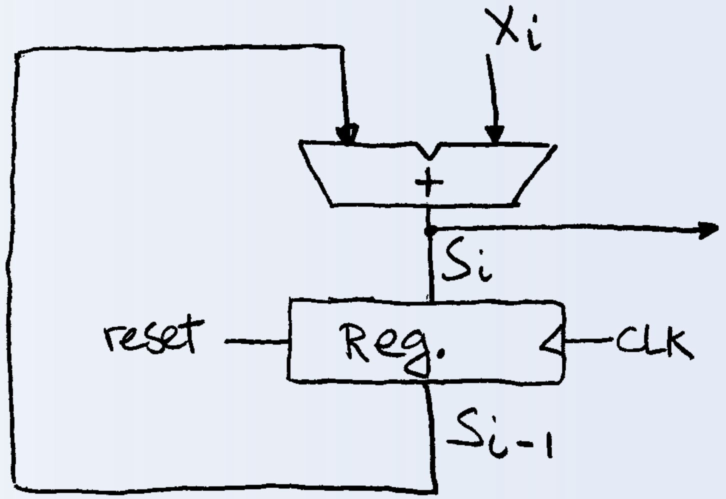


- Reset input to register is used to force it to all zeros (takes priority over D input).
- S_{i-1} holds the result of the i^{th}-1 iteration.
- Analyze circuit timing starting at the output of the register.

Accumulator Revisited (proper timing 2/2)





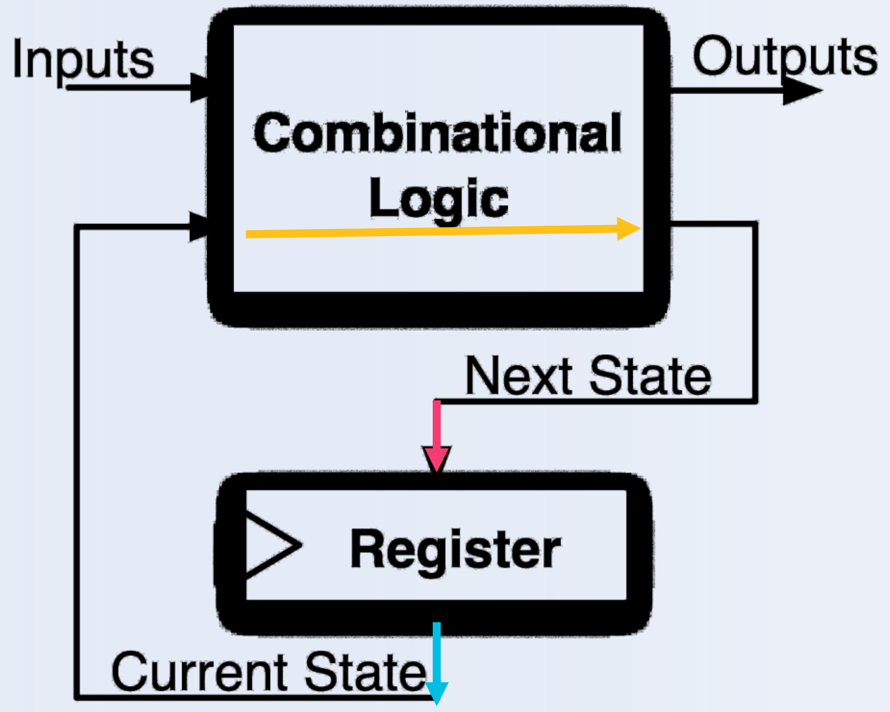


- reset signal shown.
- Also, in practice X might not arrive to the adder at the same time as S_{i-1}
- S_i temporarily is wrong, but register always captures correct value.
- In good circuits, instability never happens around rising edge of clk.

Pipelining for Performance

Maximum Clock Frequency

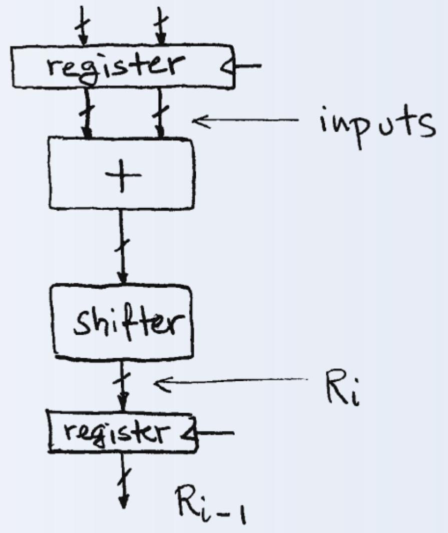
- What is the maximum clock frequency of this circuit? (Hint: Frequency = 1 / Period)



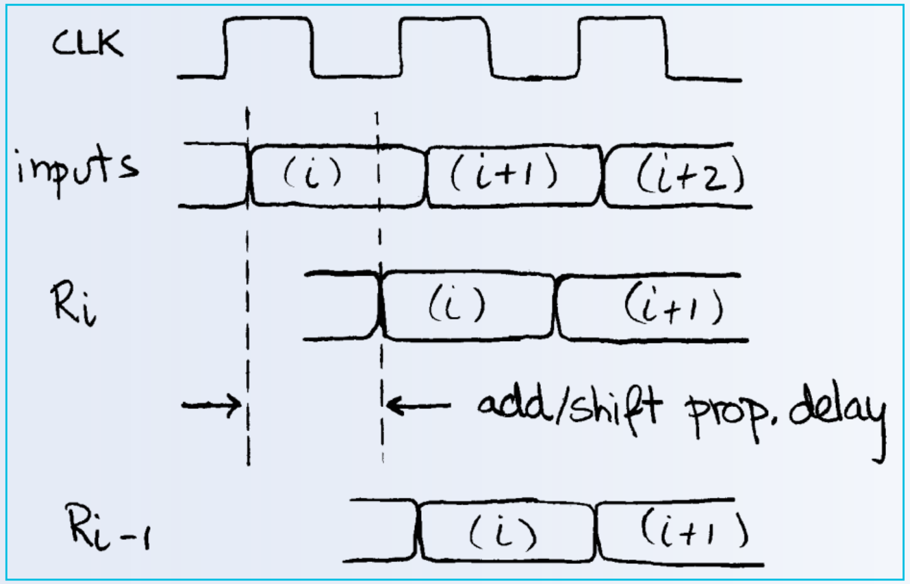
Max Delay = CLK-to-Q Delay + CL Delay + Setup Time

Pipelining to improve performance (1/2)

- Extra Registers are often added to help speed up the clock rate.



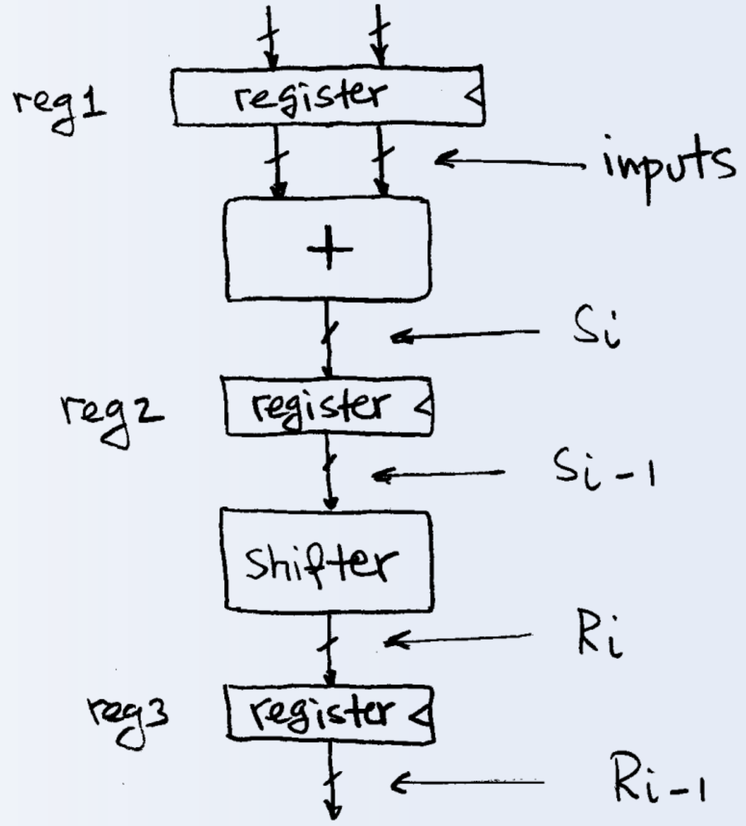
Timing



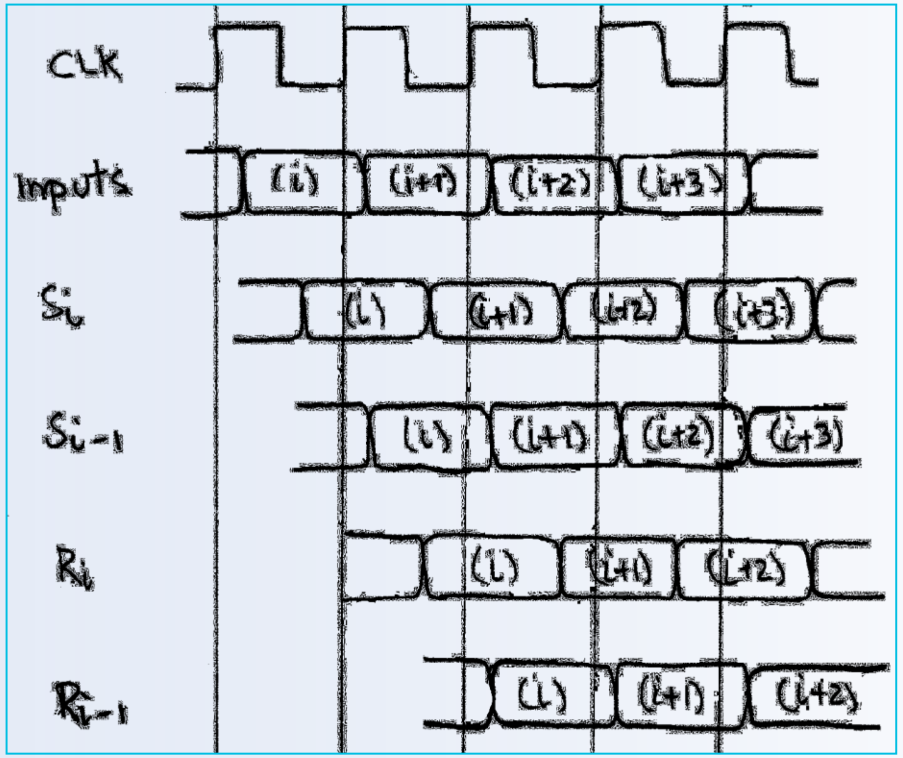
- Note: Delay of 1 clock cycle from input to output.
- Clock period limited by propagation delay of adder/shifter.

Pipelining to improve performance (2/2)

- Insertion of register allows higher clock frequency.
- More outputs per second.



Timing...



Recap of Timing Terms

- Clock (CLK) - steady square wave that synchronizes system
- Setup Time - when the input must be stable <u>before</u> the rising edge of the CLK
- Hold Time - when the input must be stable <u>after</u> the rising edge of the CLK
- “CLK-to-Q” Delay - how long it takes the output to change, measured from the rising edge of the CLK
- Flip-flop - one bit of state that samples every rising edge of the CLK (positive edge-triggered)
- Register - several bits of state that samples on rising edge of CLK or on LOAD (positive edge-triggered)

“And In conclusion…”

- State elements are used to:
  - Build memories
  - Control the flow of information between other state elements and combinational logic
- D-flip-flops used to build registers
- Clocks tell us when D-flip-flops change
  - setup and hold times are important
- We pipeline long-delay CL for faster clock

Thank You

NCDC NUST CHIP DESIGN CENTRE