Synchronous Counters

Sparsh Mittal

Synchronous counter

Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.

A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter.

Data inputs determine whether a flip-flop is to be complemented at the time of the clock edge.

If $T = 0$ or $J = K = 0$, the flip-flop does not change state.

If $T = 1$ or $J = K = 1$, the flip-flop complements.

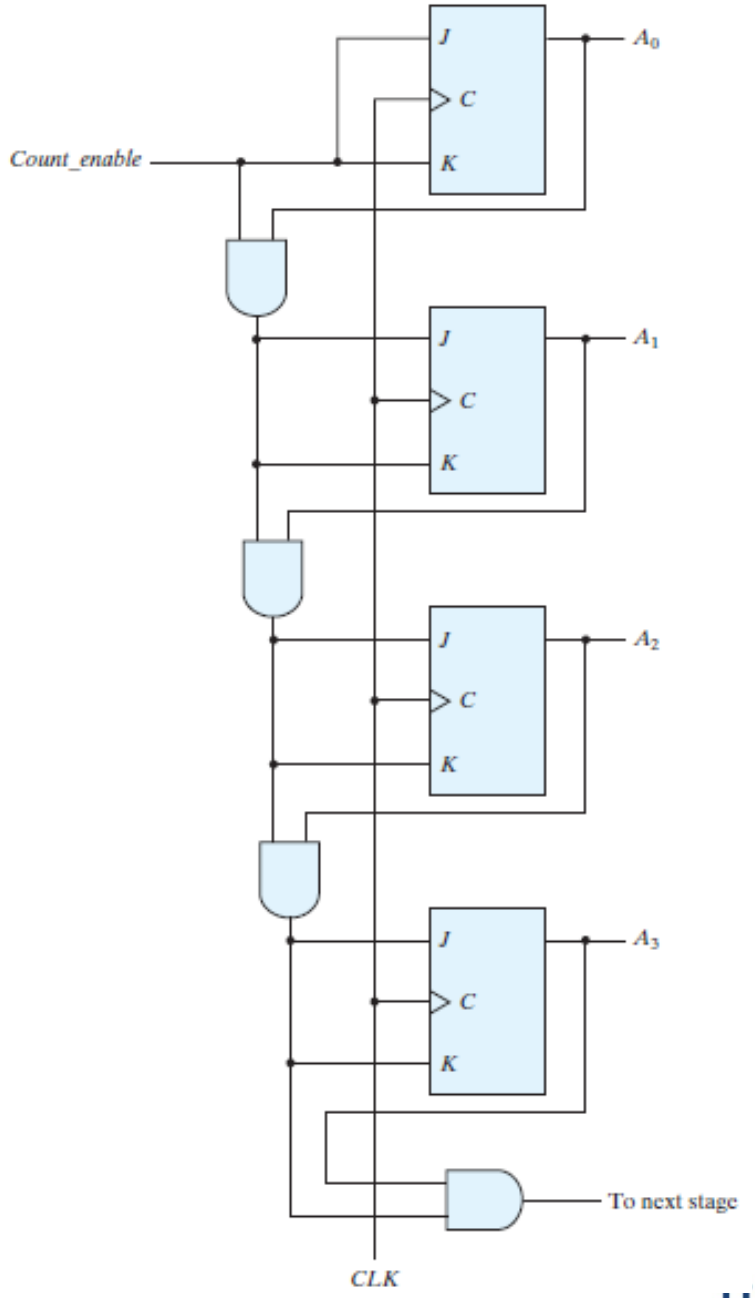
Four-bit synchronous binary counter

Flip-flop in LSB position is complemented with every pulse.

Other flip-flop is complemented when all the bits in lower significant positions are equal to 1.

The C inputs of all flip-flops are connected to a common clock.

Counter is enabled by Count_enable.



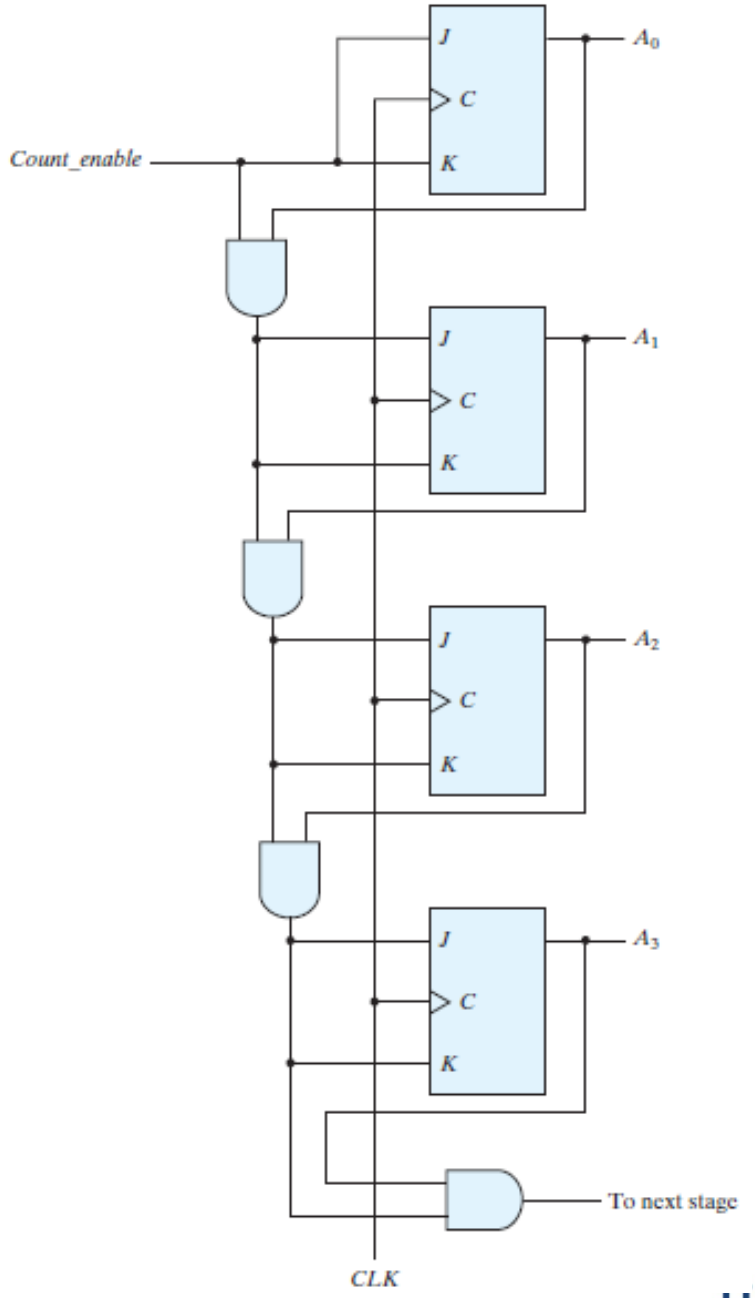
Four-bit synchronous binary counter

If the enable input is 0, all J and K inputs are equal to 0 and the clock does not change the state of the counter.

The first stage, A0, has its J and K equal to 1 if the counter is enabled.

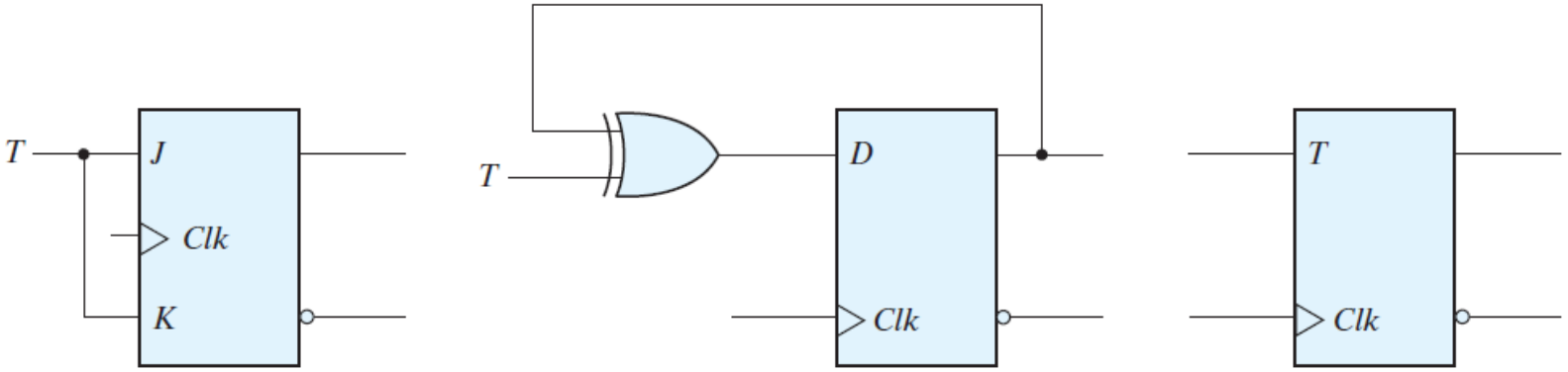
The other J and K inputs are equal to 1 if all previous least significant stages are equal to 1 and the count is enabled.

The chain of AND gates generates the required logic for the J and K inputs in each stage.



Comments

- Notice that flip-flops trigger on positive edge of clock.
- Synchronous counter can be triggered with either the positive or the negative clock edge.
- Complementing flip-flops can be of either the JK type, the T type, or the D type with XOR gates.



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

FIGURE 5.13
T flip-flop

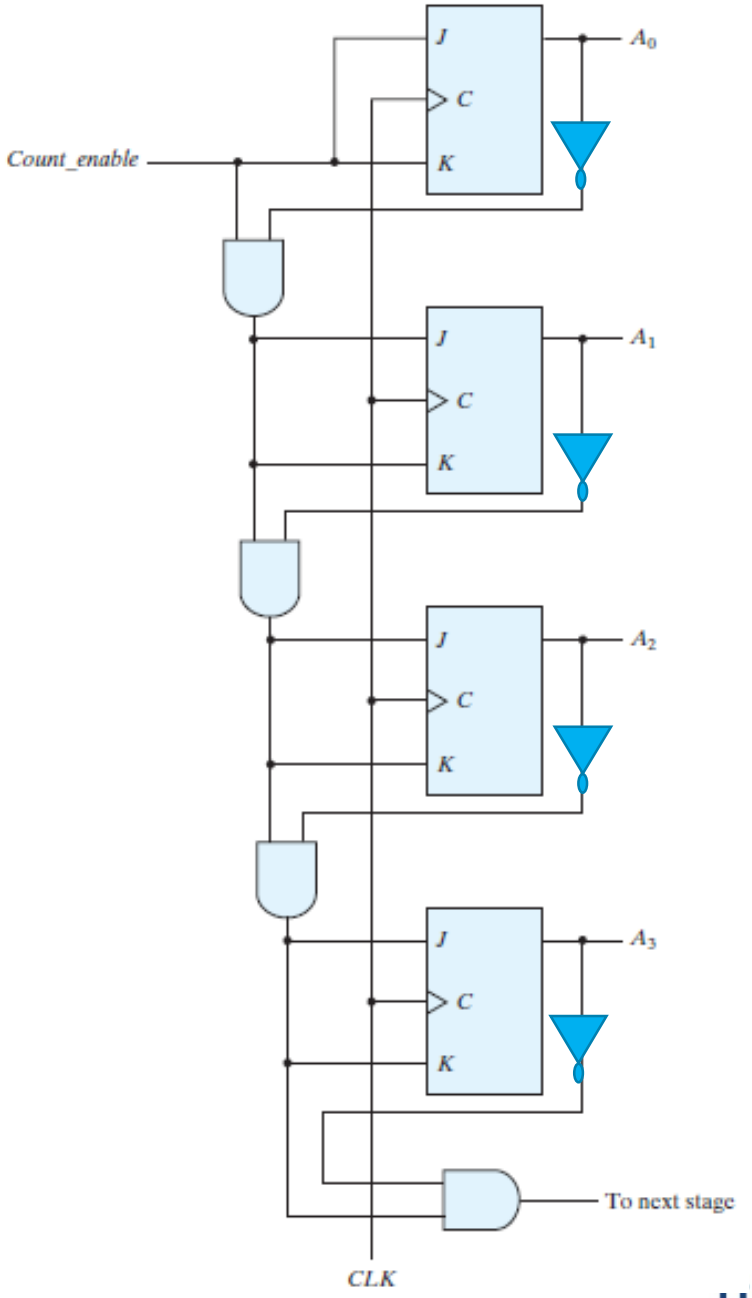
Synchronous countdown binary counter

The bit in the least significant position is complemented with each pulse.

A bit in any other position is complemented if all lower significant bits are equal to 0.

A countdown binary counter can be constructed as shown earlier, except that inputs to AND gates must come from the complemented outputs, instead of the normal outputs, of the previous flip-flops.

Synchronous countdown binary counter



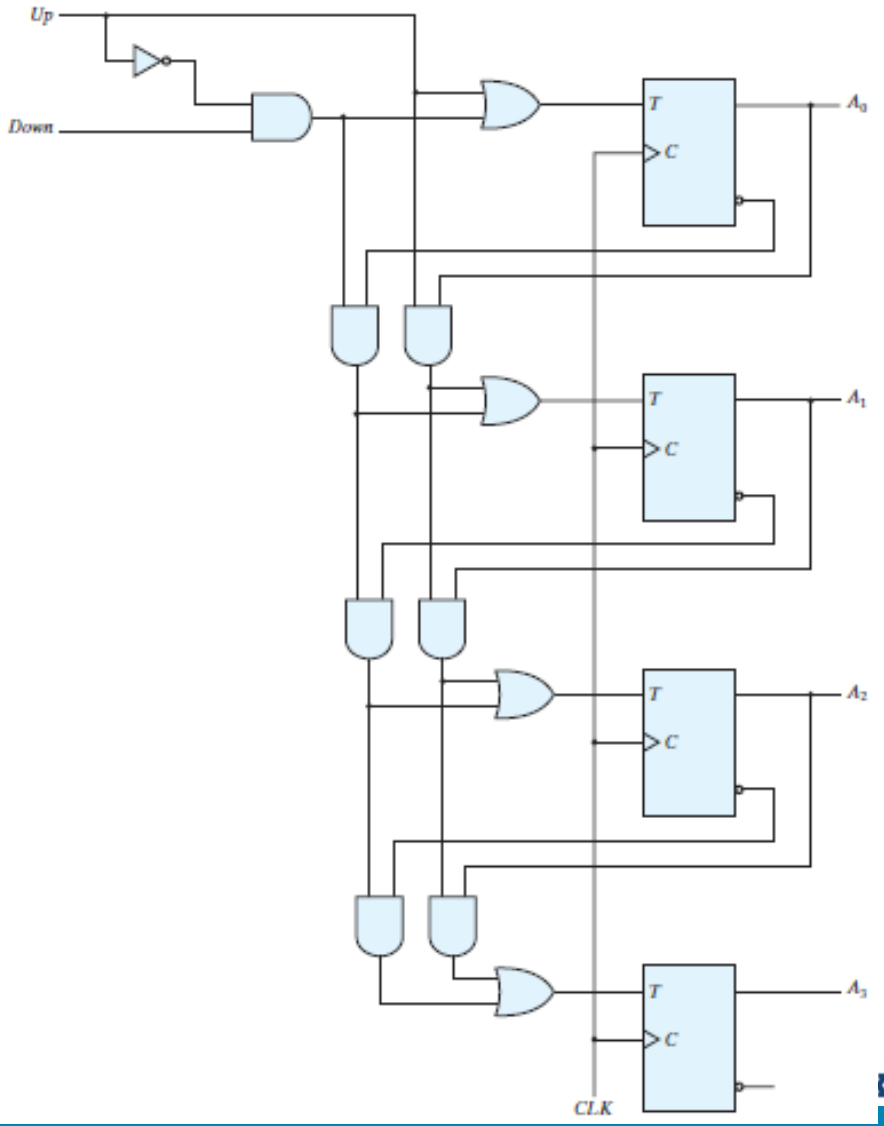
Synchronous Up–down binary counter

Up=1 ➔ circuit counts up, since the *T* inputs receive their signals from the values of the previous normal outputs of the flip-flops.

Down=1 and Up=0 ➔ Circuit counts down, since complemented outputs of previous flip-flops are applied to *T* inputs

Down=Up=0 ➔ no change in state

Down=Up=1 ➔ counts up [Up signal has priority]



BCD Counter

- Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern, unlike a straight binary count.
- To derive the circuit of a BCD synchronous counter, it is necessary to go through a sequential circuit design procedure.
- An output y is equal to 1 when the present state is 1001.
- Thus, y can enable
  - count of the next-higher significant decade and
  - switching the present decade from 1001 to 0000.

BCD Counter

State Table for BCD Counter

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

The circuit can be drawn with four T flip-flops, five AND gates, and one OR gate.

$$T_{Q1} = 1$$

$$T_{Q2} = Q'_8 Q_1$$

$$T_{Q4} = Q_2 Q_1$$

$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$

$$y = Q_8 Q_1$$

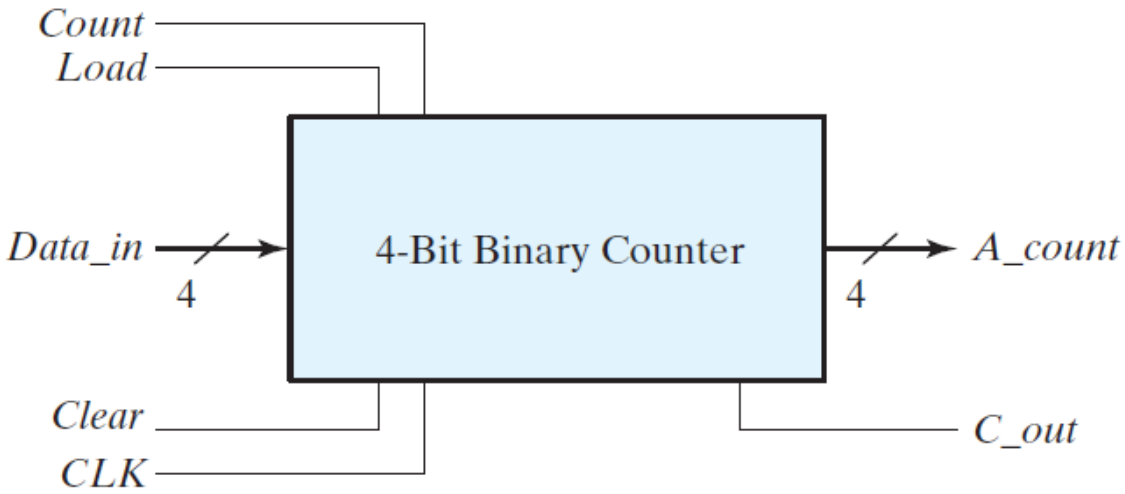
Binary Counter with Parallel Load

Binary Counter with Parallel Load

Counters employed in digital systems often require a parallel-load capability for transferring an initial binary number into the counter prior to the count operation.

Load=1 ==> count is disabled and data is transferred from four data inputs into four flip-flops.

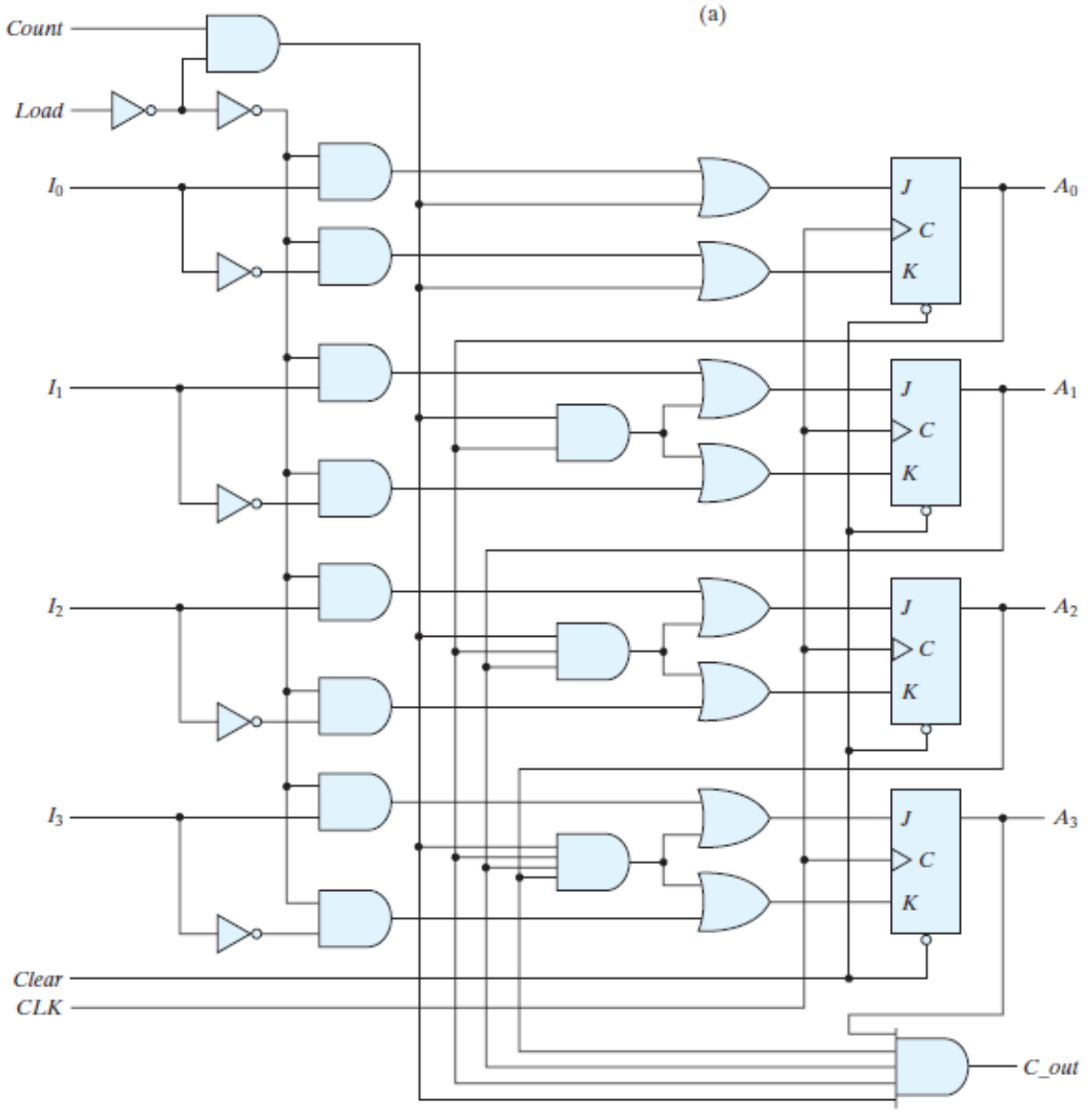
If both control inputs are 0, clock pulses do not change the state of the register.



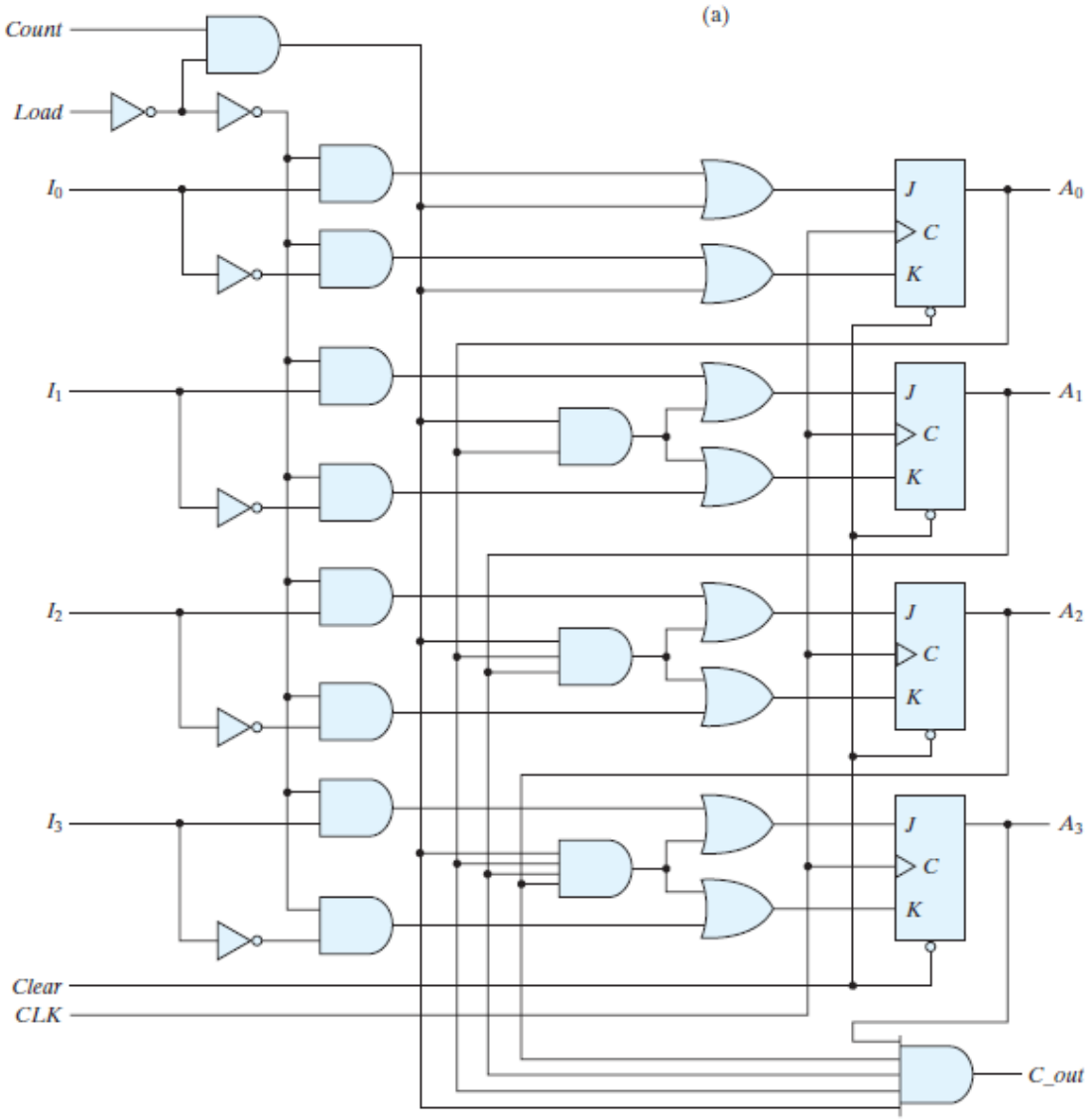
A four-bit register with parallel load capability and it can operate as a counter.

Binary Counter with Parallel Load

Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

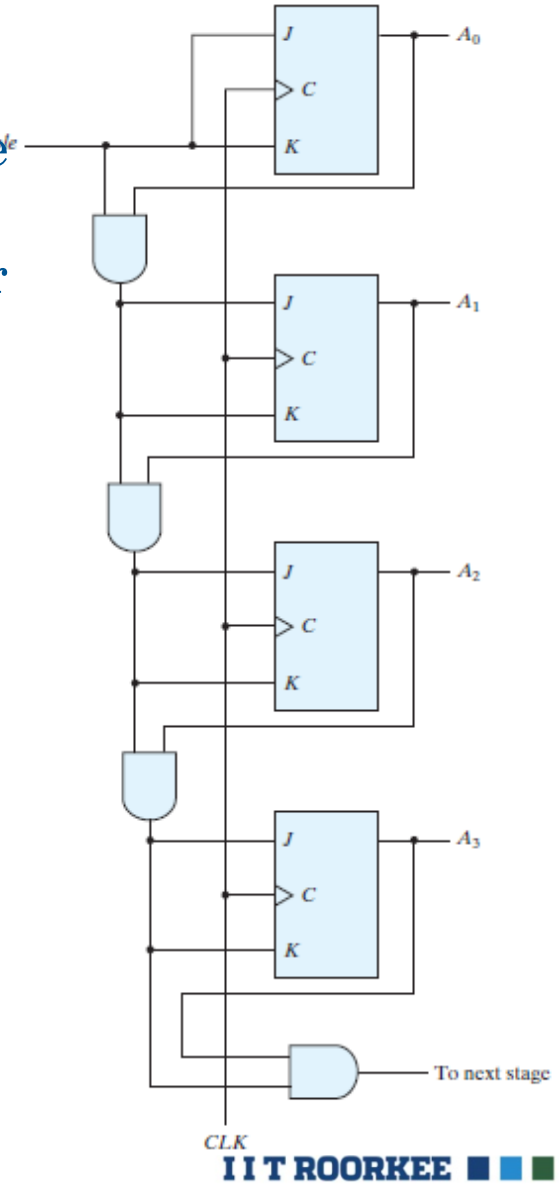


Comparison



In going from state 1111 to 0000:
Left: only one gate delay occurs
Right: four gate delays occur due to AND gate chain.
➔ Left design has higher counter speed because the delay to generate the carry bit is reduced.

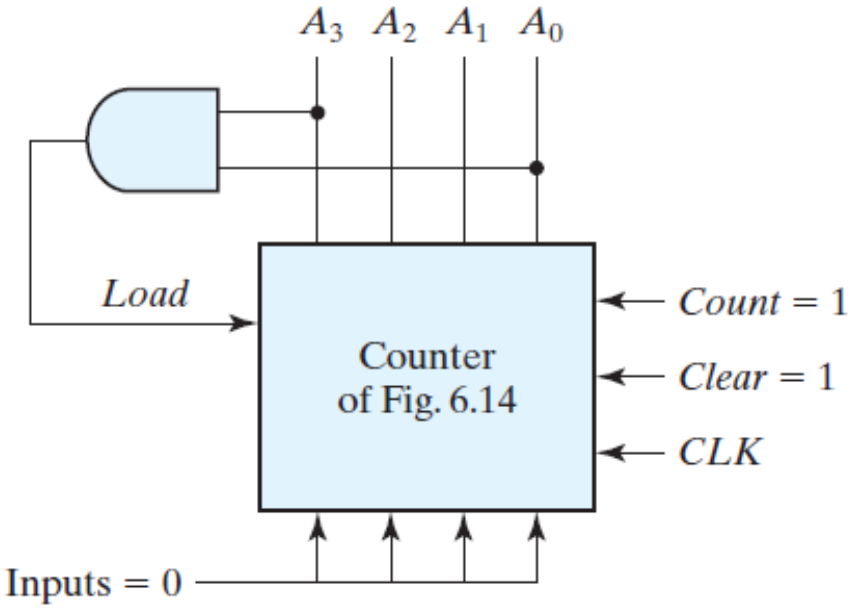
Left: each flip-flop is associated with an AND gate that receives all previous flip-flop outputs directly
Right: AND gates are connected in a chain.



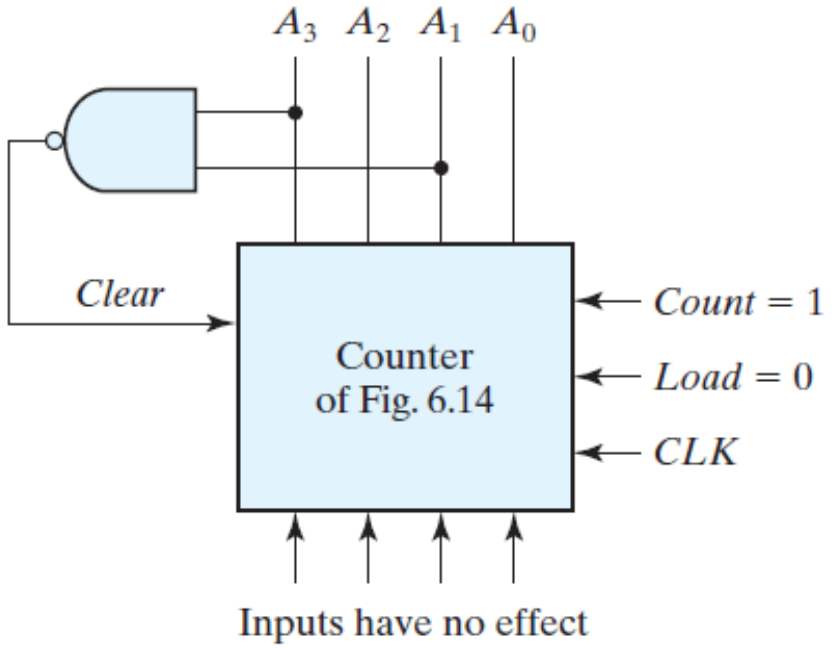
Versatility of binary counter with parallel load

A counter with a parallel load can be used to generate any desired count sequence.

Example: generating BCD count.

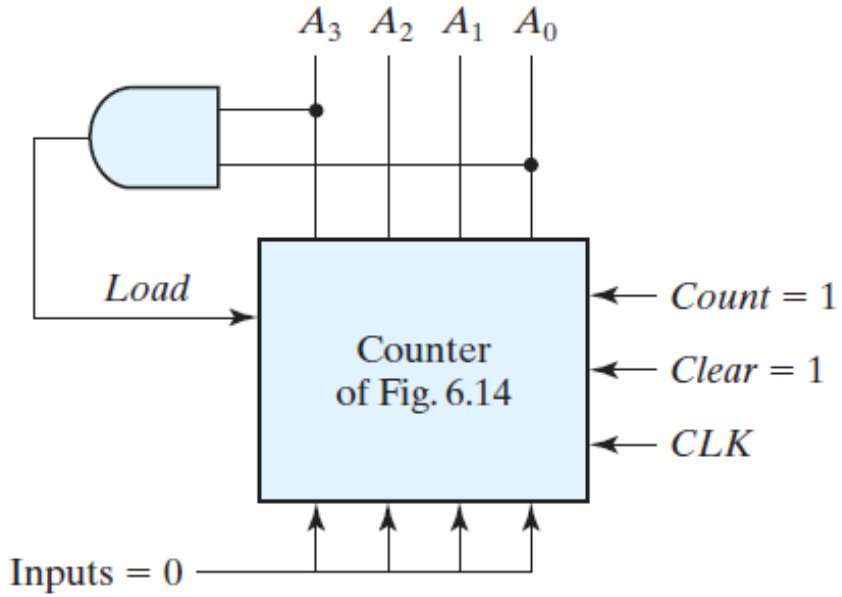


(a) Using the load input



(b) Using the clear input

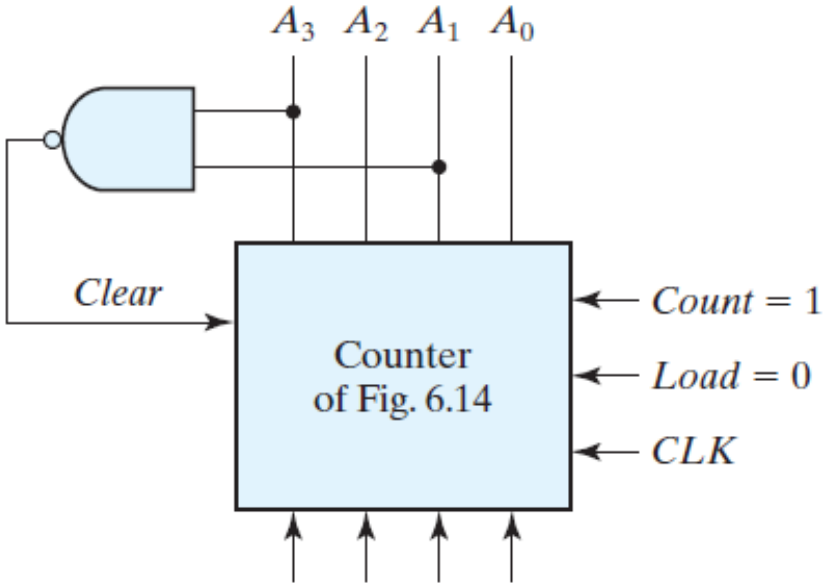
- Recall: Load control inhibits the count
- Recall: clear operation is independent of other control inputs.
- Initially counter=0. Then Clear and Count inputs are set to 1, so the counter remains active.
- As long as AND gate output is 0, each positive-edge clock increments counter by 1.
- When output reaches 1001, both A0 and A3 become 1, so AND gate output equals 1.
- This activates Load input ==> on next clock edge the register does not count, but is loaded from its four inputs.
- Since four inputs are all 0, after 1001, 0000 is loaded into register.



(a) Using the load input

AND gate detects occurrence of state 1001.

- NAND gate detects the count of 1010, but as soon as this count occurs, the register is cleared.
- The count 1010 has no chance of staying on for any appreciable time, because the register goes immediately to 0.
- A momentary spike occurs in output A_1.
- The spike may be undesirable ➔ this configuration is not recommended.
- If the counter has a synchronous clear input, it is possible to clear the counter with the clock after an occurrence of the 1001 count.



(b) Using the clear input

References

https://www.electronics-tutorials.ws/counter/count_3.html