PATRIoT Lab

Lecture 22 – Registers and Counters 3

Dr. Aftab M. Hussain,

Assistant Professor, PATRIoT Lab, CVEST

Chapter 6

Synchronous counter - binary

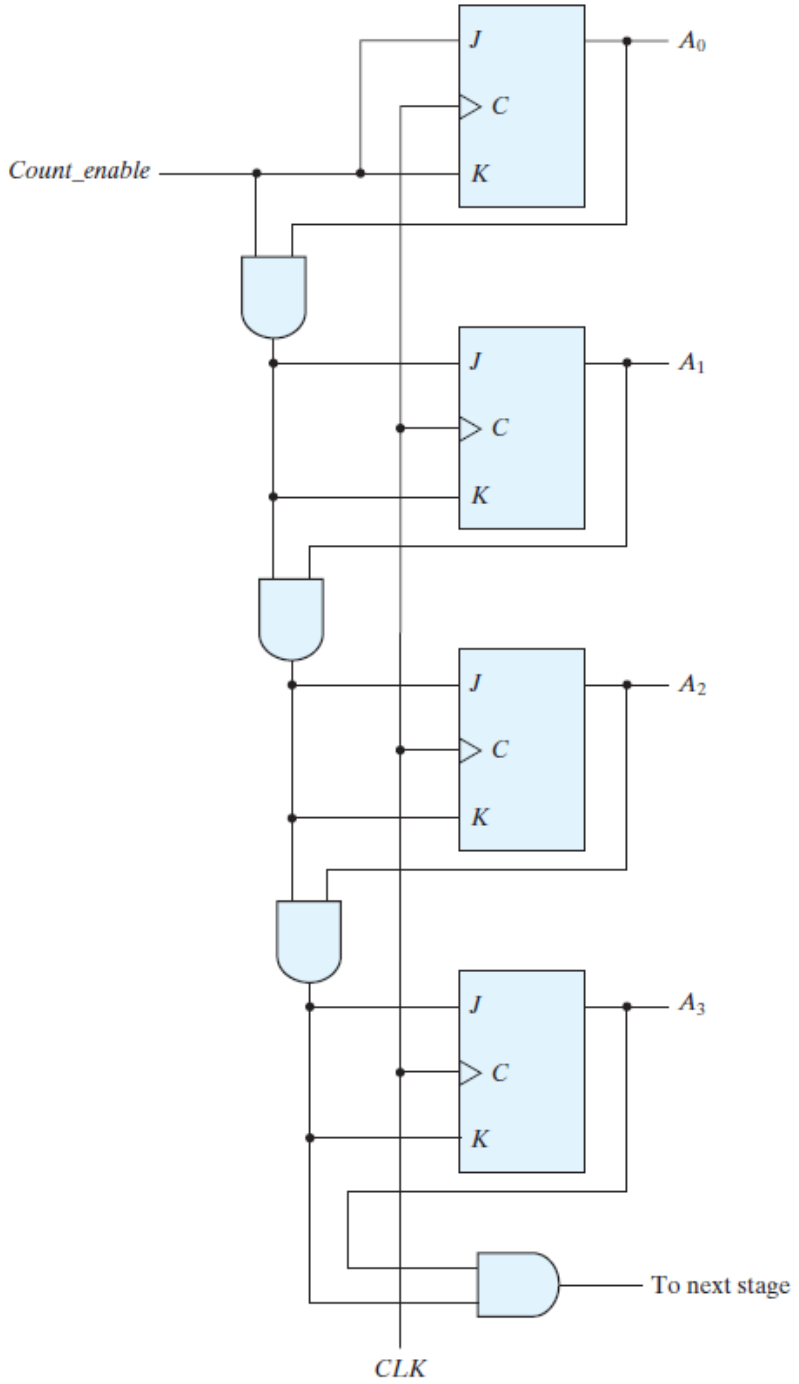
- The key problem with ripple counters is that they do not update their states immediately after the arrival of the clock
- Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops
- A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter
- The decision whether a flip-flop is to be complemented is determined from the values of the data inputs, such as *T* or *J* and *K* 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

Synchronous counter - binary

- The design of a synchronous binary counter is very simple
- In a synchronous binary counter, the flip-flop in the least significant position is complemented with every pulse
- *A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1*
- For example, if the present state of a four-bit counter is $A_3A_2A_1A_0 = 0011$, the next count is 0100
- A_0 is always complemented
- A_1 is complemented because the present state of $A_0 = 1$
- A_2 is complemented because the present state of $A_1A_0 = 11$
- However, A_3 is not complemented, because the present state of $A_2A_1A_0 = 011$, which does not give an all-1's condition

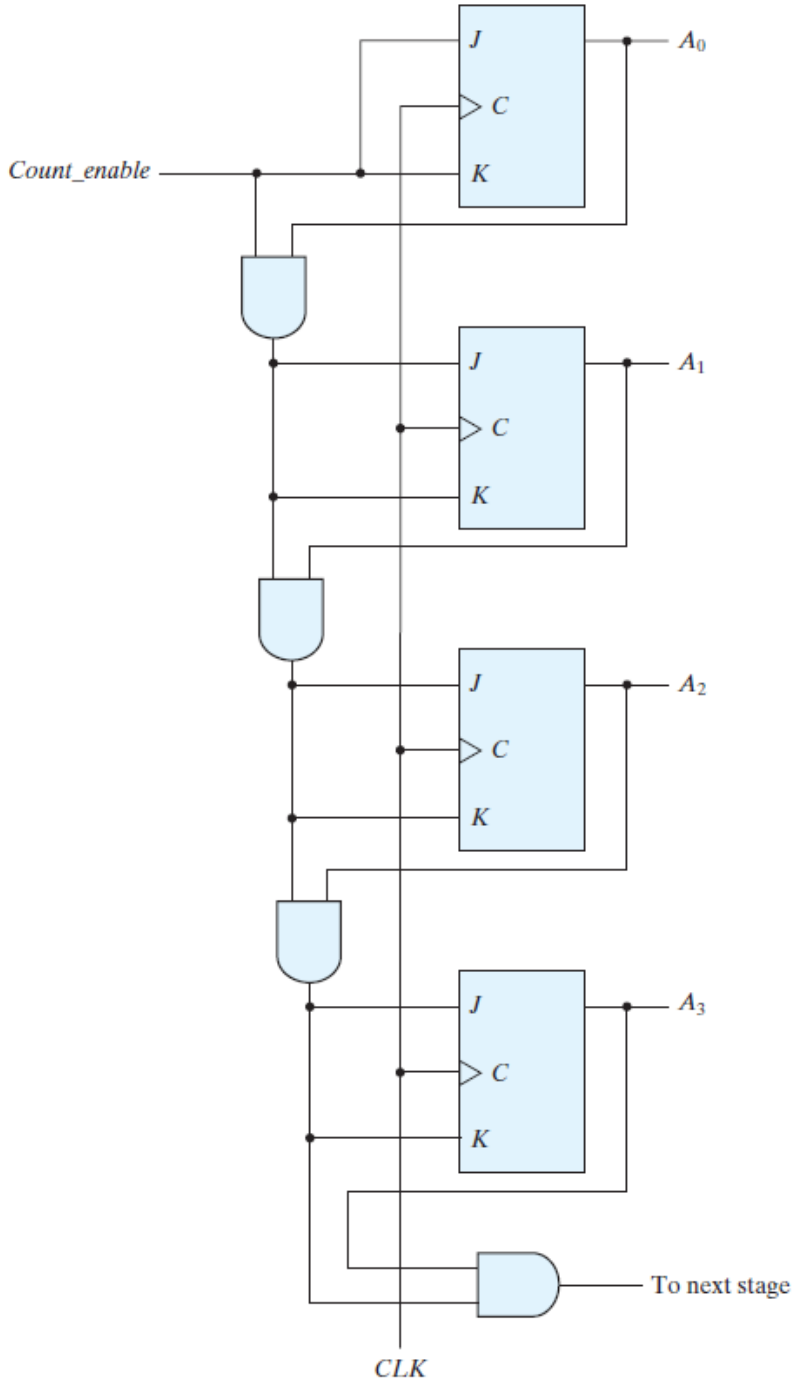
Synchronous counter - binary

- Synchronous binary counters have a regular pattern and can be constructed with complementing flip-flops and gates
- The regular pattern can be seen from the four-bit counter
- The *C* inputs of all flip-flops are connected to a common clock
- The counter is enabled by *Count_enable*
- 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, A_0, 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



Synchronous counter - binary

- Note that the flip-flops trigger on the positive edge of the clock
- The polarity of the clock is not essential here, but it is with the ripple counter
- The synchronous counter can be triggered with either the positive or the negative clock edge
- The complementing flip-flops in a binary counter can be of either the *JK* type, the *T* type, or the *D* type with XOR gates



Synchronous counter – down counter

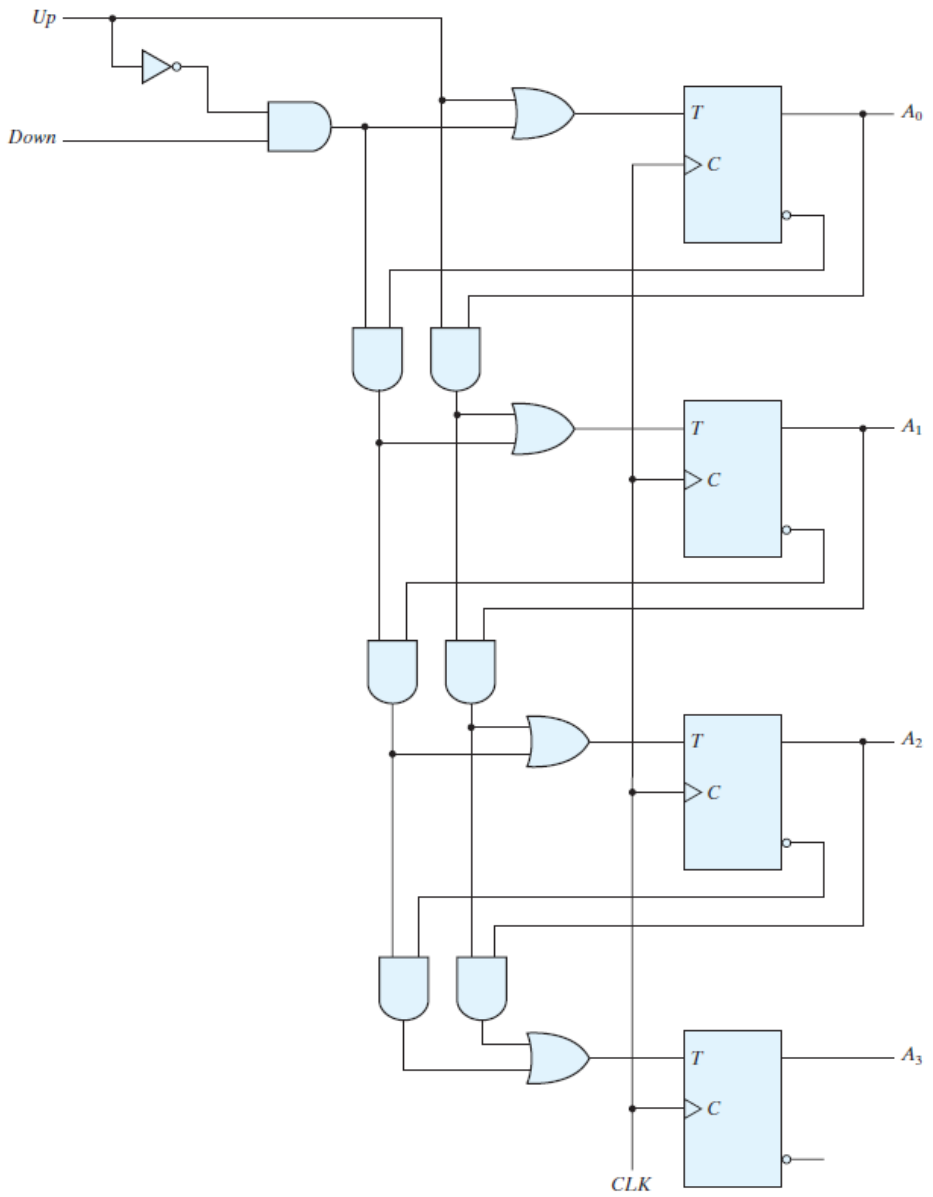
- A synchronous countdown binary counter goes through the binary states in reverse order, from 1111 down to 0000 and back to 1111 to repeat the count
- It is possible to design a countdown counter in the usual manner, but the result is predictable by inspection of the downward binary count
- The bit 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*
- For example, the next state after the present state of 0100 is 0011
- The second significant bit is complemented because the first bit was 0
- The third significant bit is complemented because the first two bits were equal to 0
- But the fourth bit does not change, because not all lower significant bits are equal to 0
- A countdown binary counter can be constructed as a regular counter, except that the inputs to the AND gates must come from the complemented outputs, instead of the normal outputs, of the previous flip-flops

Synchronous counter – up/down counter

- Let us see if we can design a counter that counts up/down based on an input
- The counter should count up if U=1, D=0; down if U=0, D=1; no change if U=D=0; and up if U=D=1
- If we chose T flip-flops for this design, the inputs should be A if U is 1 (regardless of D), A’ if U’D is 1 and 0 if U=D=0
- We can cascade this logic to make the synchronous up/down counter

Synchronous counter – up/down counter

- Let us see if we can design a counter that counts up/down based on an input
- The counter should count up if U=1, D=0; down if U=0, D=1; no change if U=D=0; and up if U=D=1
- If we chose T flip-flops for this design, the inputs should be A if U is 1 (regardless of D), A’ if U’D is 1 and 0 if U=D=0
- We can cascade this logic to make the synchronous up/down counter



Synchronous counter – BCD

- BCD counter counts in binary-coded decimal from 0000 to 1001 and back to 0000
- 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
- The input conditions for the *T* flip-flops are obtained from the present- and next-state conditions
- Also shown in the table is an output *y*, which is equal to 1 when the present state is 1001
- In this way, *y* can enable the count of the next-higher significant decade while the same pulse switches the present decade from 1001 to 0000

Present State				Next State				Output
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y
0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	1	0
0	0	1	1	0	1	0	0	0
0	1	0	0	0	1	0	1	0
0	1	0	1	0	1	1	0	0
0	1	1	0	0	1	1	1	0
0	1	1	1	1	0	0	0	0
1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1

Synchronous counter – BCD

- The flip-flop input equations can be simplified by means of maps
- The unused states for minterms 10 to 15 are taken as don't-care terms
- The simplified functions are:

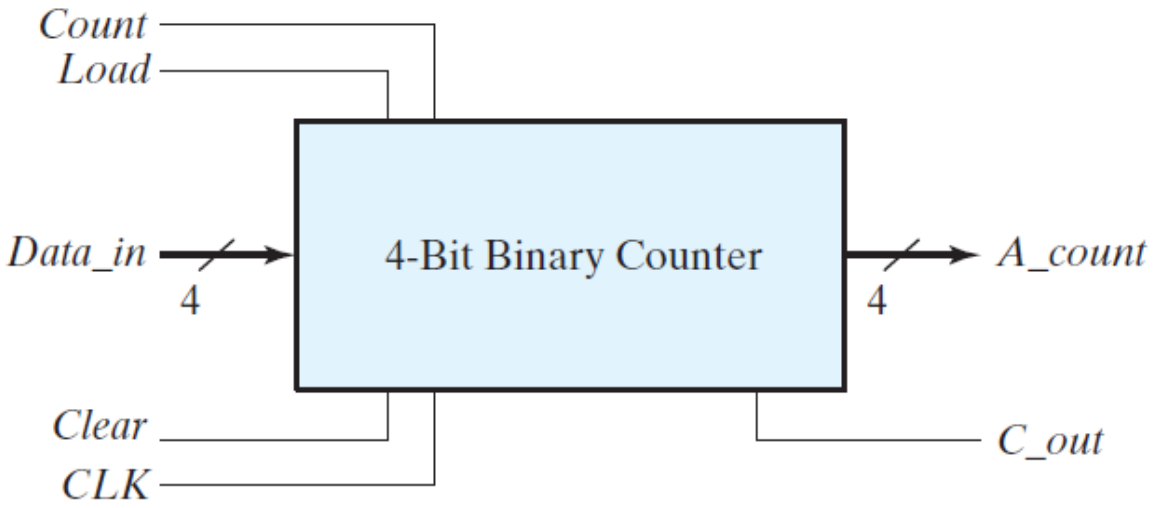
$$TQ_1 = 1$$
$$TQ_2 = Q_8'Q_1$$
$$TQ_4 = Q_2Q_1$$
$$TQ_8 = Q_8Q_1 + Q_4Q_2Q_1$$
$$y = Q_8Q_1$$

- The circuit can be made using these expressions

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

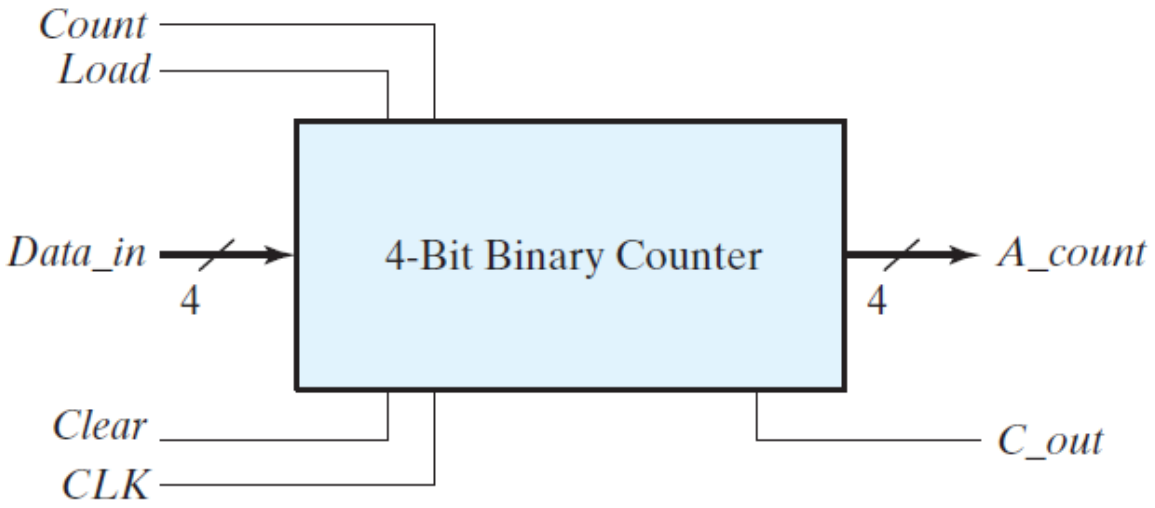
Synchronous counter – with parallel load

- Counters employed in digital systems quite often require a parallel-load capability for transferring an initial binary number into the counter prior to the count operation
- When equal to 1, the input load control disables the count operation and causes a transfer of data from the four data inputs into the four flip-flops
- If both control inputs are 0, clock pulses do not change the state of the counter



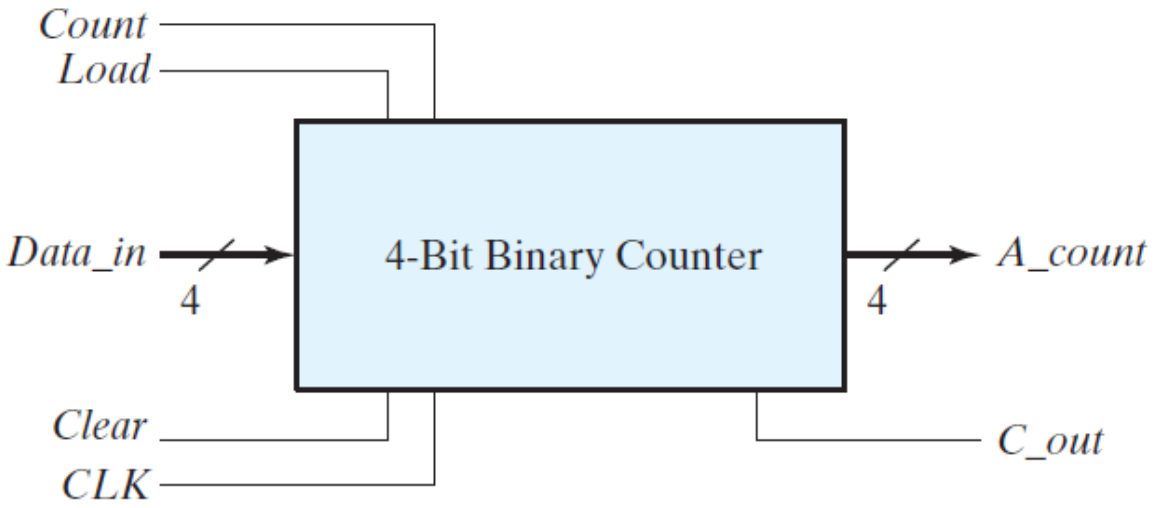
Synchronous counter – with parallel load

- The four control inputs— *Clear*, *CLK*, *Load*, and *Count* —determine the next state
- The *Clear* input is asynchronous and, when equal to 0, causes the counter to be cleared regardless of the presence of clock pulses or other inputs
- With the *Load* and *Count* inputs both at 0, the outputs do not change, even when clock pulses are applied
- A *Load* input of 1 causes a transfer from inputs I_0 - I_3 into the register during a positive edge of *CLK*
- The *Load* input must be 0 for the *Count* input to control the operation of the counter

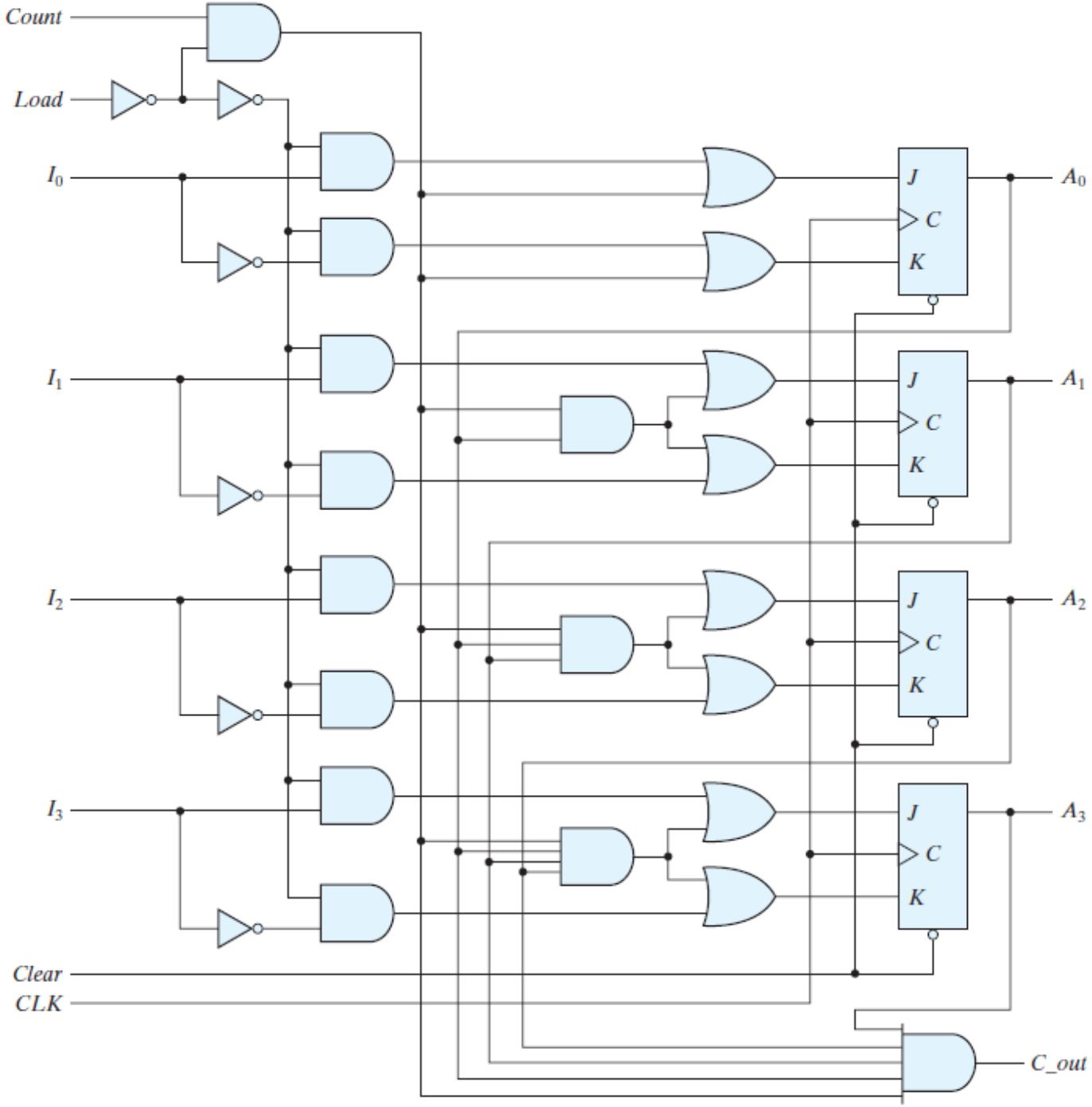


Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Synchronous 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



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