Lecture 19, April 2, 2007

Synchronous Counters

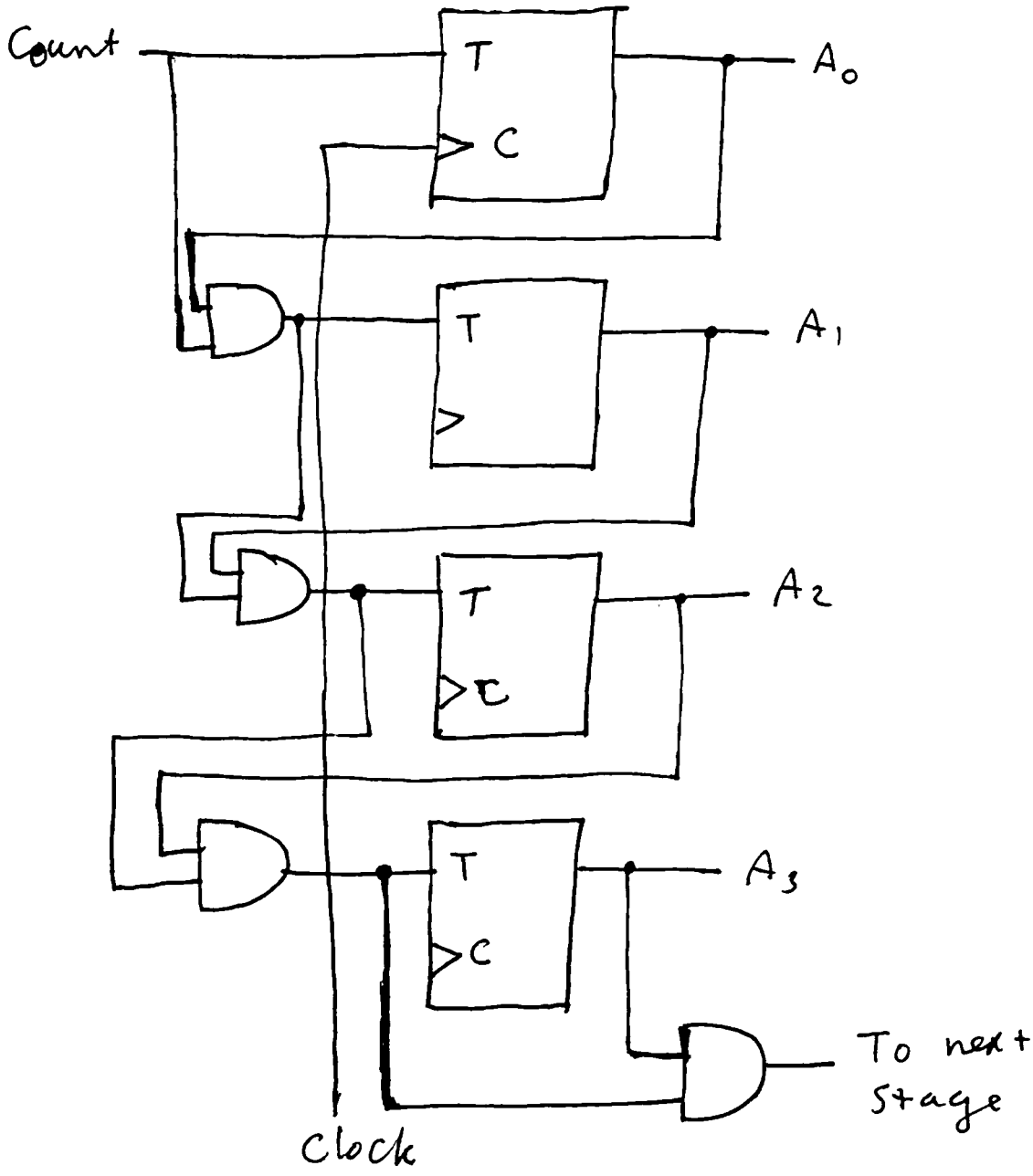
The difference between synchronous counters and ripple counters is that, in a synchronous counter, a common clock is applied to all flip-flops. The way the content of each flip-flop changes is determined by the logic at its data inputs, e.g., JK, D or T inputs, depending on the type of flip-flops used in implementing the counter.

Synchronous Binary Counter

As an example consider a binary counter. Following the sequence of states 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, ..., 1111, and back to 0000 (for the example of a 4-bit counter) one observes that a bit changes only when all the previous bits are equal to 1 and a count signal is applied. That is, the

first flip-flop changes its content at all counts. The second flip-flop changes only when the first bit is 1. The third flip-flop changes if the content of the first two flip-flops is 11, i.e., we have 0100 after 0011 and also, we have 1000 after 0111. The third bit (the one with an arrow under it) has changed from a 0 to a 1 in the first case and from a 1 to a zero in the second case since the first two bits have been 11. Assume that we implement this counter using a T flip-flop. Then we need to connect the Count signal to the T input of the first flip-flop. To connect Count AND A_0 to the input of the second flip-flop and to connect Count. A_0. A_1 to the T input of the third flip-flop and so on. The example in the book using a JK flip-flop can be justified similarly noting that connecting the J and K inputs results in a T flip-flop.

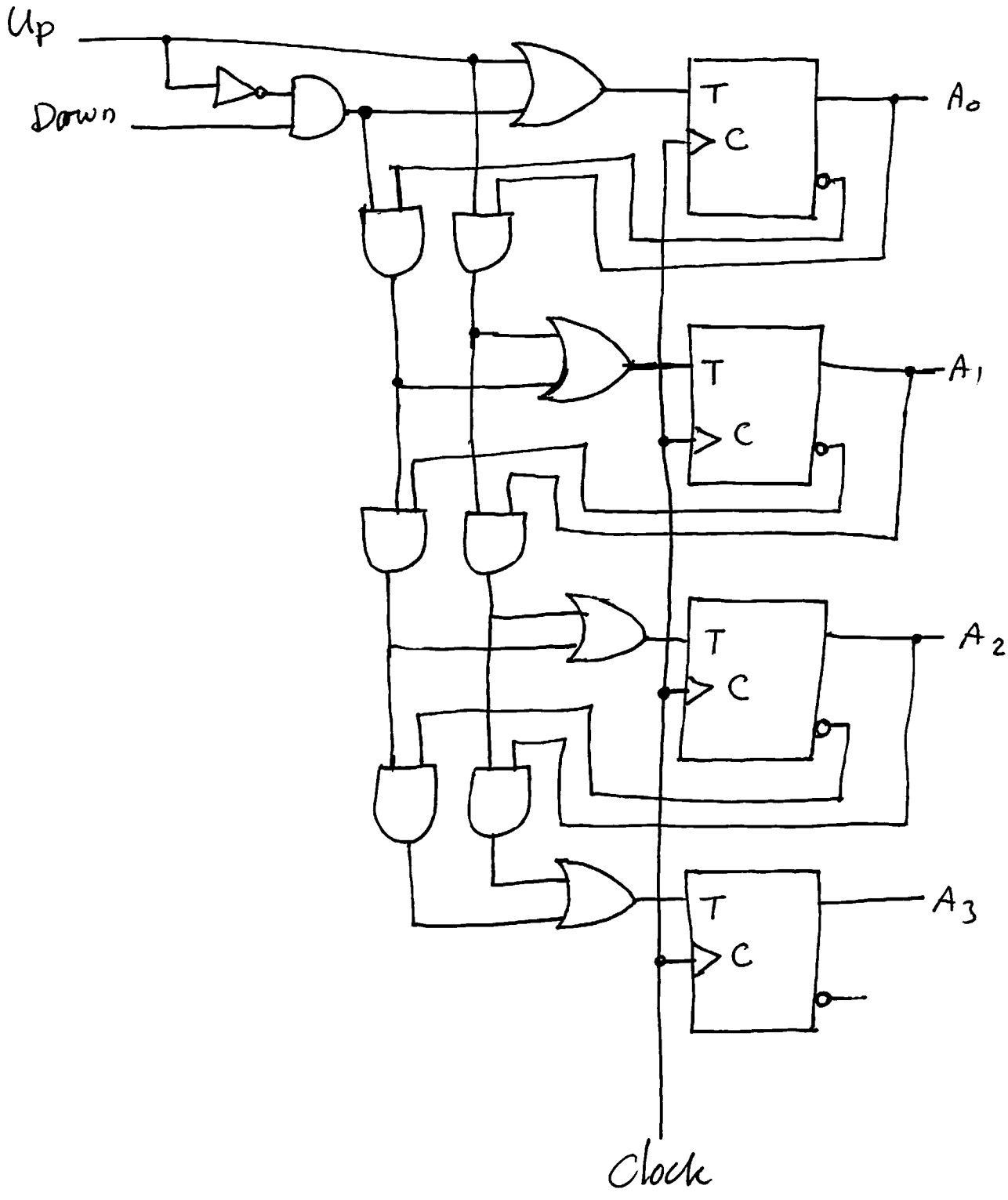
Following is the circuit diagram of a binary counter using T flip-flop. We have shown only four stages (four bits). It can be extended to as many bits as we require.



Up-down counter

A down-counter counts from 1111 to 0000. Writing down the contents of the flip-flops, i.e., 1111, 1110, 1101, ..., 0000 and back to 1111, we observe that the first bit changes with each occurence of the Count input. The other bits change, when the count occurs, only if all lower bits are . Using this observation, we can design an up-down counter. This counter needs two count inputs: an UP and a Down Count enable signal. We either Connect all the outputs of the previous (less significant) bits to the input T of the flip-flop (when UP is enabled) or the product of all inverted outputs (Q's) of the previous flip-flops to the T input of a flip-flop (when Down is enabled.

The circuit diagram for an UP-Down counter is shown in the next page.


Up
Down
T
C
A_0
T
C
A_1
T
C
A_2
T
C
A_3
Clock

BCD Counter

A BCD counter counts from 0000 to 1001 and then moves to 0000 and counts again to 1001 and repeats this.

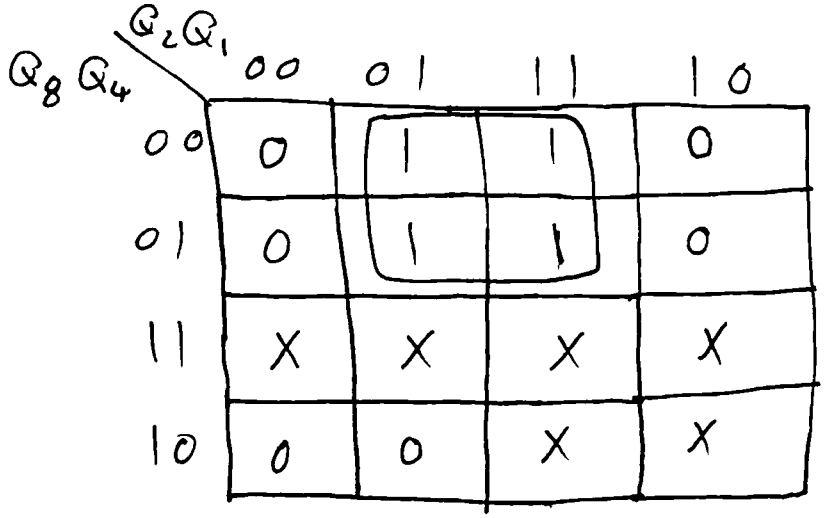
Consider the implementation of a BCD counter using T flip-flop. The following table shows the Present state, the next state and the Flip-flop inputs. We also have added an output, y, that signals the next counter (next decades counter) to start counting.

Present State				Next State				output	FF inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	T_8	T_4	T_2	T_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

So, we have:

$$T_1 = 1$$

For T_2, we have,



Q_8Q_4 \ Q_2Q_1	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	X	X	X	X
10	0	0	X	X

So

$$T_2 = Q_8' Q_1$$

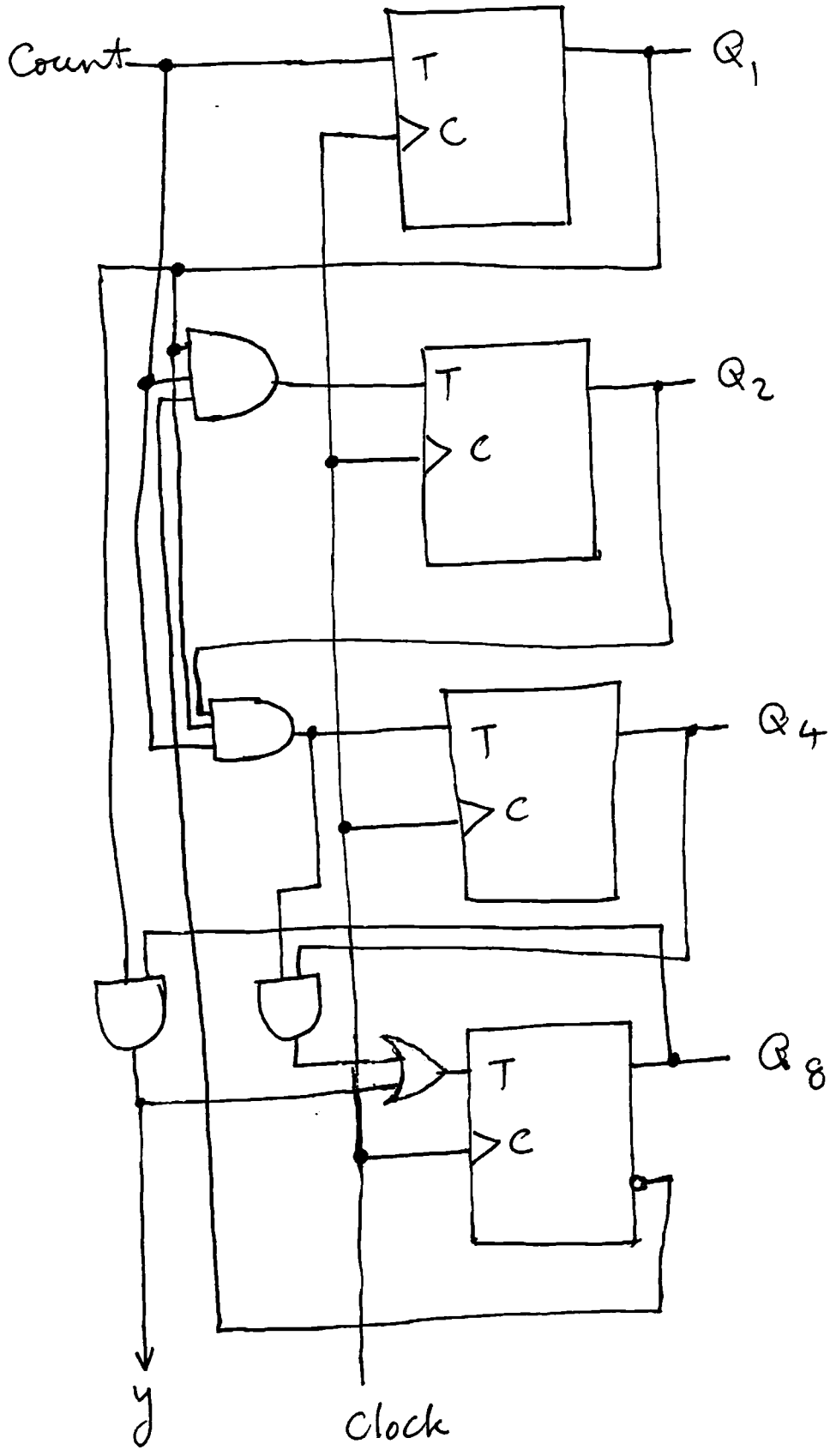
Similarly, you may find

$$T_4 = Q_2 Q_1$$

$$T_8 = Q_8 Q_1 + Q_4 Q_2 Q_1$$

and $$y = Q_8 Q_1$$

Note that we have used don't care for entries corresponding to 1010 to 1111.


Count
T
C
Q_1
T
C
Q_2
T
C
Q_4
T
C
Q_8
y
Clock

Binary counter with parallel load

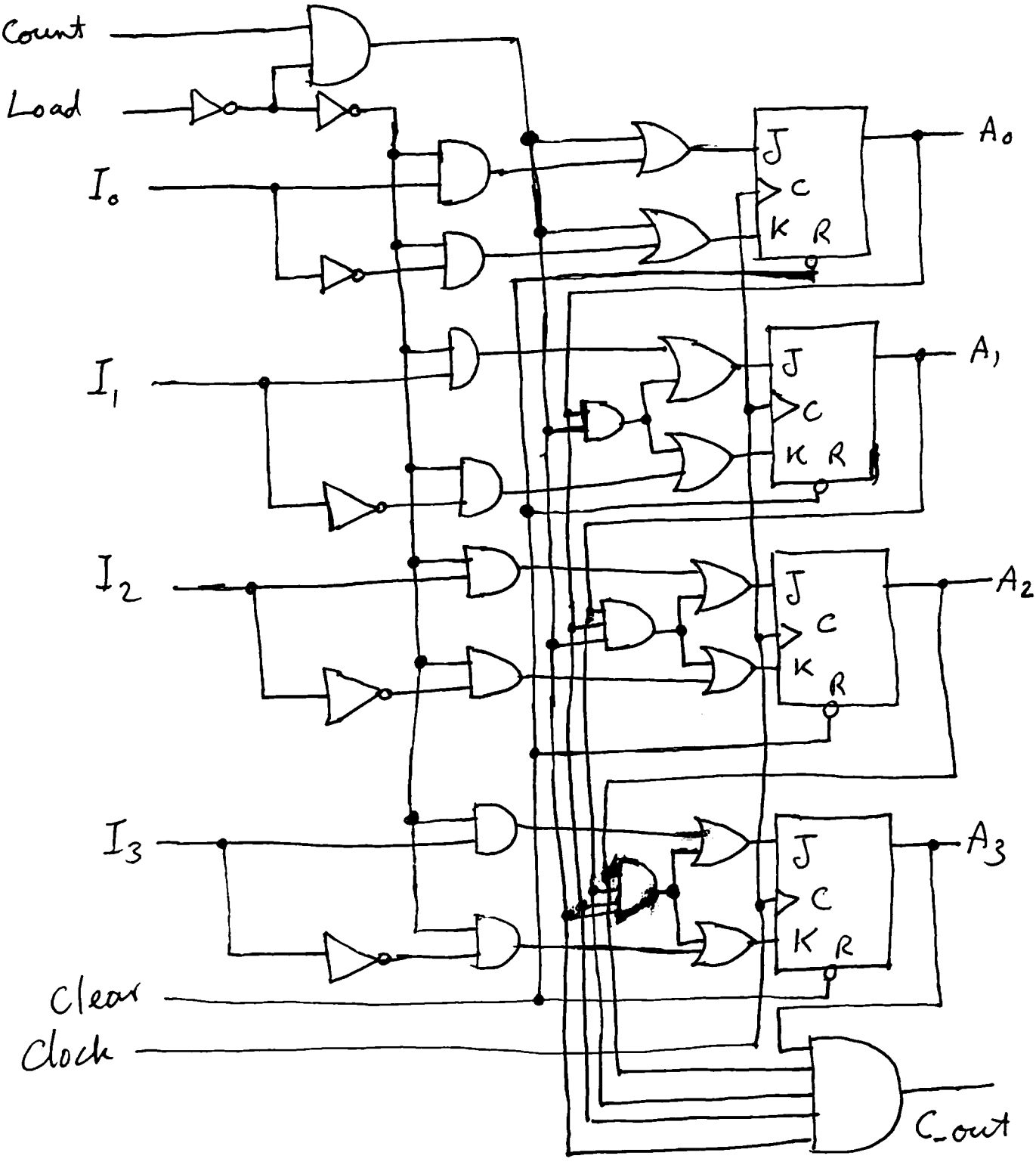
Sometimes it is necessary to establish the value from which the count starts. For this reason, it is required that we allow the Counter to be loaded with a value prior to Count. In such a case the counter will have two control inputs: a Count input and a Load input.

When both Count and Load are zero, the state of the counter does not change.

When the load is low, with each Count the counter is incremented.

When the load is one regardless of the value of the flip-flops and the Count input, the Counter will be loaded with the bits on parallel inputs $I_0, I_1, I_2, I_3, \ldots$

The circuit diagram of a binary counter with parallel load is shown on the next page.



Clear input when equal to zero forces the Counter to reset to 0000.

C_out is a carry output enabling extension

of the Counter to more than 4 bits.

Counter with unused states

A Counter with n flip-flops can have 2^n states. It is possible that in an application not all of these states are necessary. We have already seen the example of BCD Counter that uses only 10 out of $2^4 = 16$ possible states.

To design a Counter with unused states, we can treat the unused states as don't care condition or some specific next states. Since, it is possible that as a result of interference, the circuit go to one of these unused states, after design of the circuit, we need to check to make sure that if the circuit end up in an unused state it is capable of eventually going to a valid state. The following example makes the idea clear.

Example: Consider a Counter following the sequence 000, 001, 010, 100, 101, 110. That is, the Counter skips 011 and 111.

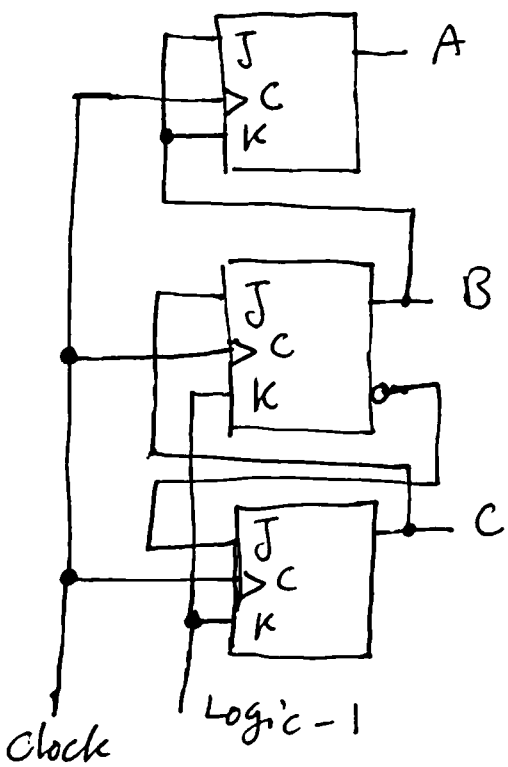
The state table for this Counter is:

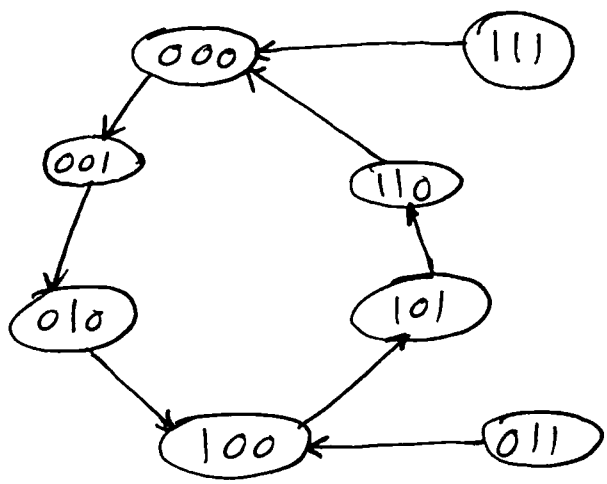
Present state			Next state			Flip-flop inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

we have $J_A = K_A = B$, $J_B = C$, $K_B = 1$

$J_C = B'$ and $K_C = 1$.

So,

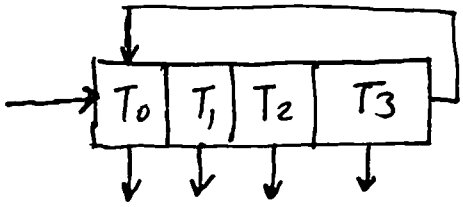


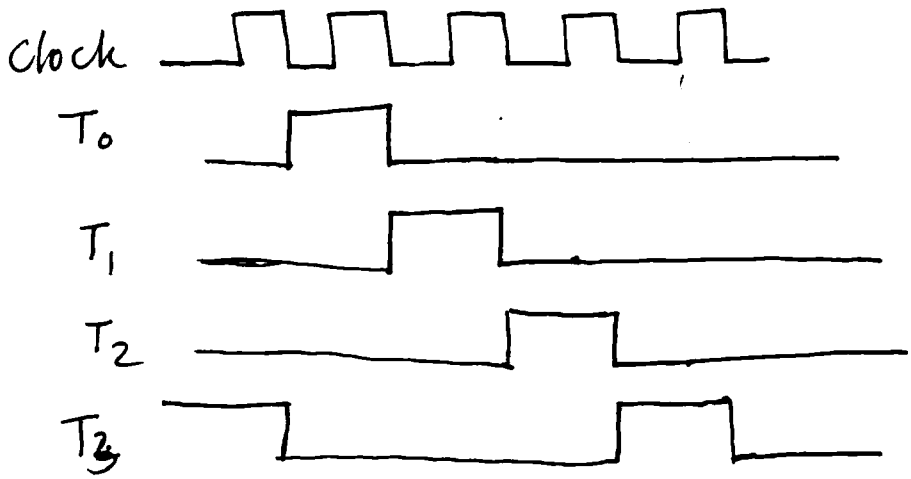


The state diagram shows that if the Counter is in unwanted states 011 or 111, it will move to valid states 100 and 000.

Ring Counter

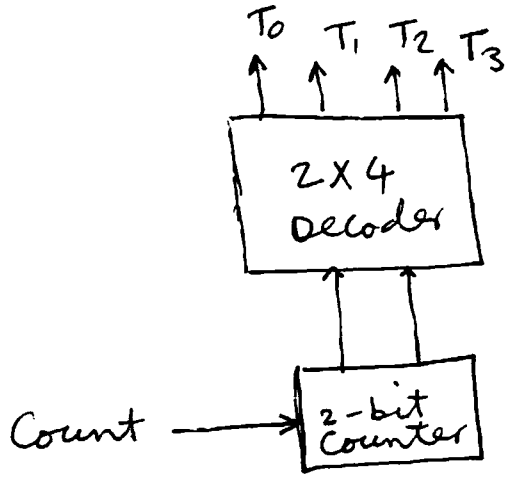
A Ring counter is a Counter that counts 1000, 0100, 0010, 0001, and 1000 again. That is only one flip-flop is set at any given time. A Ring counter is used for Controlling the sequence of operation in a digital System. It can be implemented using a shift register whose least significant output is connected to the input of the most significant flip-flop.





Another way to implement a ring Counter is to use a $\log_2 k$ bit Counter and a k-output

decoder. For the above example with 4-bit counter, we need a 2-bit counter and a 2 x 4 decoder.



Johnson Counter

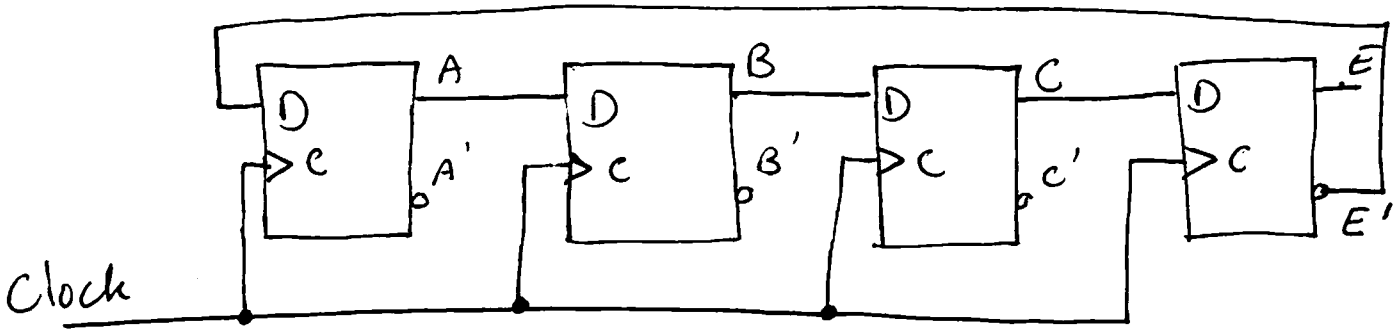
In order to generate k distinguishable states using a ring counter, we need k flip-flops.

Using a different feedback to the input of the left-most flip-flop, we can generate 2k states (2k control pulses) using k flip-flops.

To do this, we use switch-tail ring counter.

In a switch-tail ring counter, the feedback is from the inverted output of the right-most

flip-flop. Following is the logic diagram for a four-stage switch-tail ring counter.



The count sequence for the switch-tail ring counter is:

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	$C'E$

A Johnson counter is a switch tail ring with outputs (or inverted outputs) of flip-flops combined (by AND) to form $2k$ timing signals.

The decoding rule (forming AND gate inputs) is straightforward. For all zero pattern the two

extreme flip-flop outputs are used in inverted form. For all one pattern the regular outputs of the extreme flip-flops are used. For other patterns, the first two alternating bits, either 01 or 10 is used.

For example: in the above table for 0000 we use $A'E'$ for 1111 we use AE, for 1000 we use AB' for 10 at the begining of the pattern, for 1100, we use BC'. Similarly for 0111 we use $A'B$ and for 0011 we use $B'C$ and so on.

When a Johnson Counter goes to an unused state, it goes from one invalid state to another. To avoid this problem, we may disconnect the output of B flip-flop from the input D of C flip-flop and insted feed C flip-flop with:

$$D_C = (A + C)B .$$