Registers and Counters

A circuit with flip-flops is considered a sequential circuit even in the absence of combinational logic. Circuits that include flip-flops are usually classified by the function they perform. Two such circuits are *registers* and *counters*:

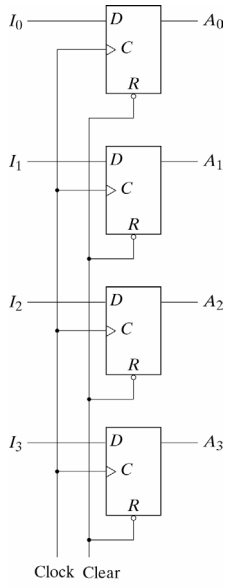
- **Register** – is a group of flip-flops. Its basic function is to hold information within a digital system so as to make it available to the logic units during the computing process. However, a register may also have additional capabilities associated with it.
- **Counter** – is essentially a register that goes through a predetermined sequence of states. The gates in the counter are connected in such a way as to produce the prescribed sequence of binary states.

Applications of registers include serial addition, convolutional encoders for error-control coding, and pseudo-random binary sequence generators.

Counters are primarily used as pattern generators.

1. Registers

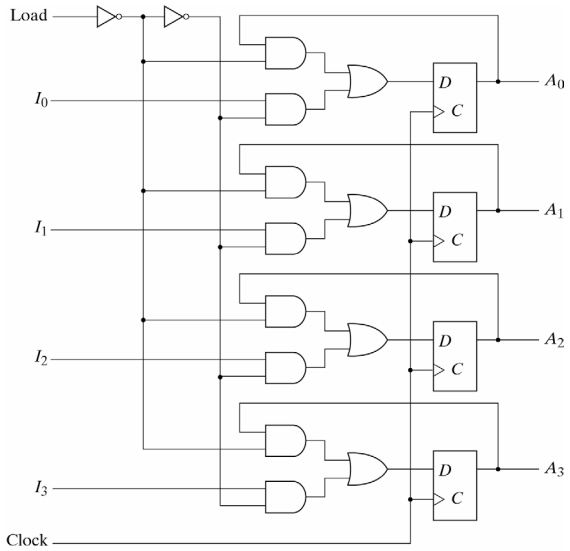
Various types of registers are available on the market. A simple 4-bit register is shown below:



The common clock input triggers all flip-flops and the binary data available at the four inputs are transferred into the register. The clear input is useful for clearing the register to all 0's output.

1.1 Register with Parallel Load

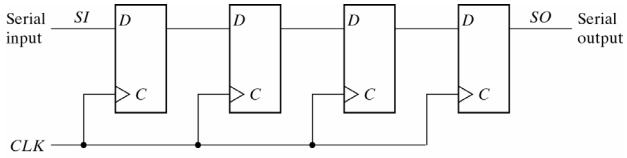
If all the bits in a register are loaded at the same time, the loading is done in *parallel*. A 4-bit register with a load control input is shown below:



The *Load* input determines the action to be taken with each clock pulse. The feedback connection from output to input is necessary because the *D* flip-flop does not have a "no change" condition.

2. Shift Registers

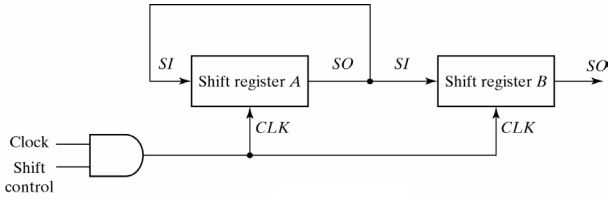
Registers capable of shifting their binary contents in one or both directions. A *unidirectional* 4-bit shift register that uses only flip-flops is as follows:



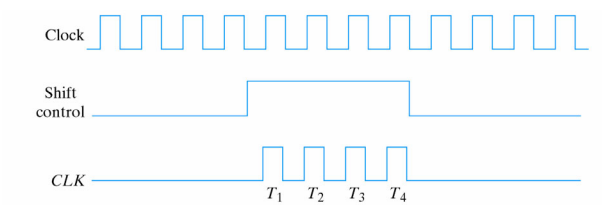
Each clock pulse shifts the contents of the register one bit position to the right.

2.1 Serial Transfer

Serial transfer of information from register *A* to register *B* is done with shift registers:



Suppose the shift registers have four bits each. The control unit that supervises the transfer must be designed such that it enables the shift registers, via the *shift control* signal, for a fixed time of four clock pulses:



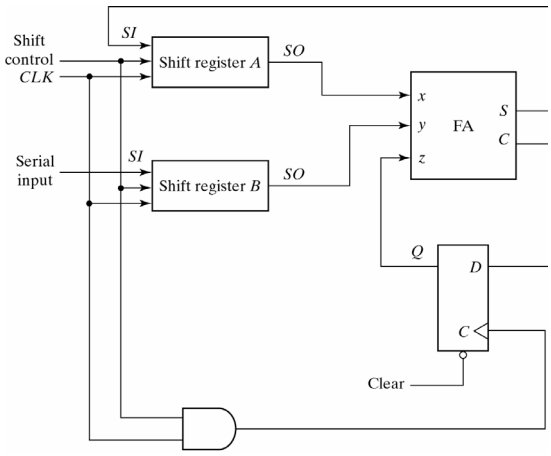
Assume that the binary content of *A* before the shift is 1011 and that of *B* is 0010. The serial transfer occurs in four steps as shown in the table below:

Timing Pulse	Shift Register *A*	Shift Register *B*
Initial value	1 0 1 1	0 0 1 0
After T_1	1 1 0 1	1 0 0 1
After T_2	1 1 1 0	1 1 0 0
After T_3	0 1 1 1	0 1 1 0
After T_4	1 0 1 1	1 0 1 1

2.2 Serial Addition

Operations in digital computers are usually done in parallel because of speed requirements. Serial operations are slower but need less components.

A *n*-bit serial adder may be realized by using two shift registers:

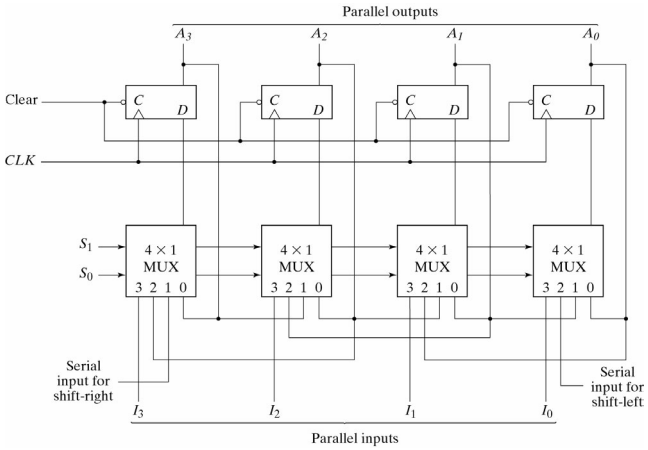


Bits in the two registers are added one pair at a time through a single full adder (FA) circuit. The carry out of the FA is transferred to a *D* flip-flop. The output of this flip-flop is then used as the carry input for the next pair of significant bits.

2.3 Universal Shift Register

A second general classification of shift registers consists of *bidirectional* shift registers. These type of registers are capable of shifting their contents either left or right depending upon the signals present on appropriate control input lines.

A 4-bit *universal shift register* is shown below:

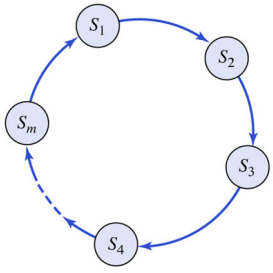


Mode Control		
s_1	s_0	*Register* Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

3. Counters

A counter is an example of a register. Its primary function is to produce a specified output pattern sequence. This sequence might correspond to the number of occurrences of an event or it might be used to control various parts of a digital system.

The counting sequence is often depicted by a graph called a *state diagram*. A *modulus-m* counter (i.e., a counter with *m* states) has the following state diagram:

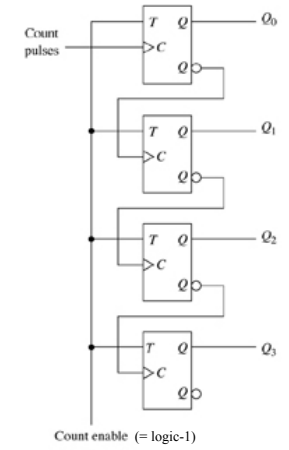


Each node S_i denotes the states of the counter and the arrows in the graph denote the order in which the states occur.

Counters are available in two categories: *ripple counters* and *synchronous counters*.

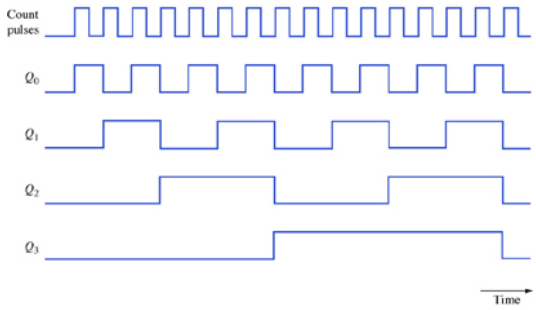
4. Ripple Counters

In a ripple counter, the flip-flop output transition serves as a source for triggering other flip-flops. A 4-bit binary ripple counter (mod-16) is as follows:



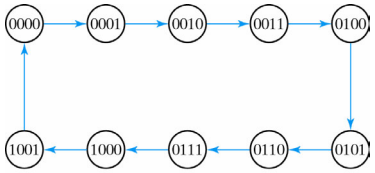
Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1
0	0	0	0

etc.

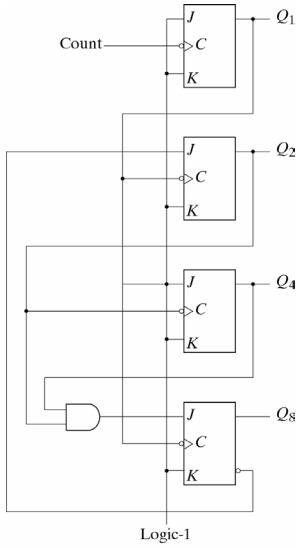


4.1 BCD Ripple Counter (Mod-10)

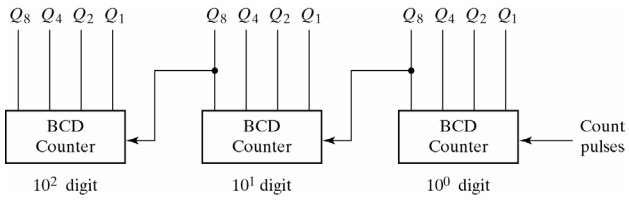
A decimal counter follows a pattern of 10 states:



The logic diagram of a BCD counter using *JK* flip-flops is shown below:



A multiple decade counter can be constructed by connecting BCD counters in cascade. A three-*decade* counter is shown below:



The inputs to the second and third decades come from Q_8 of the previous decade. When Q_8 in one decade goes from 1 to 0, it triggers the count for the next higher-order decade while its own goes from 9 to 0.

4.2 Settling Time of Ripple Counters

A ripple counter is also known as an *asynchronous counter*. The rippling behaviour affects the overall settling time.

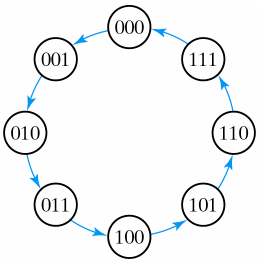
The worst-case delay occurs when the counter goes from its 11….1-state to its 00…0-state.

For an *n*-stage binary ripple counter, the worst-case setting time is $n \times t_{pd}$, where t_{pd} is the propagation delay associated with each flip-flop.

5. Synchronous Binary Counters

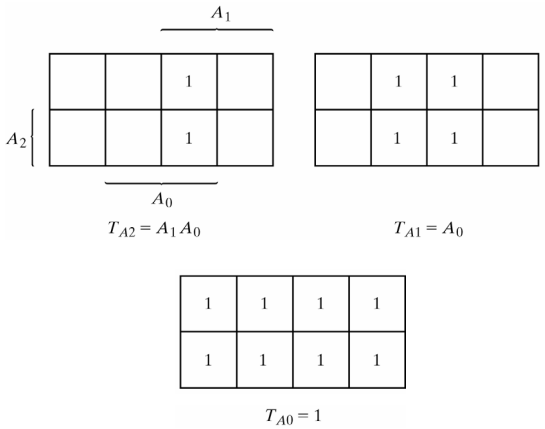
The settling time problem associated with ripple counters is avoided in *synchronous counters*. In these counters, the count pulses are applied directly to the control inputs *C* of all flip-flops.

The state diagram and state table of a 3-bit binary counter are:

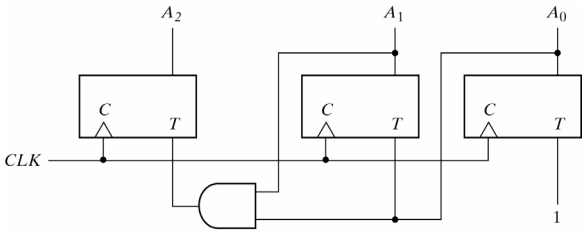


Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

The flip-flop input equations are specified by the *K*-maps:

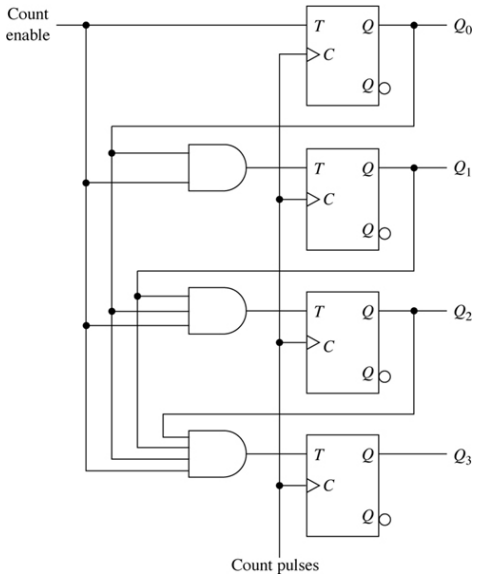


The input equations listed under the *K*-maps specify the combinational part of the counter. Including these functions with the three *T* flip-flops, the logic diagram of the counter is:



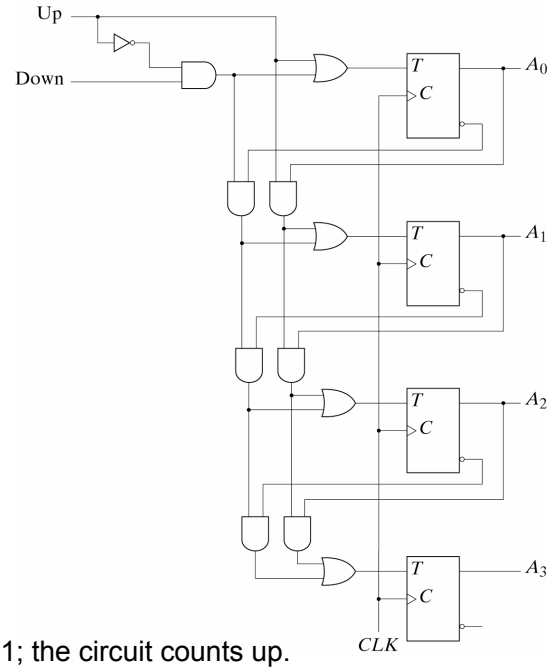
Synchronous counters have a regular pattern and can be constructed with complementing flip-flops and gates. The complementing flip-flops can be either of the *JK*-type or the *T*-type or the *D*-type with X-OR gates.

A 4-bit binary synchronous counter (count-down) with *count enable function* can be realized as shown below:



5.1 Up-Down Binary Counter

The circuit of a 4-bit up-down binary counter with *T* flip-flops is:



Up = 1; the circuit counts up.
Down = 1, *Up* = 0; the circuit counts down.
Up = 0, *Down* = 0; the circuit doesn't change state.
Up = 1, *Down* = 1, the circuit counts up.

5.2 BCD Counter

It counts from 0000 to 1001 (decimal 0 to 9) and back to 0000. The state table for a BCD counter is listed below:

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 flip-flop input conditions are obtained from the present and next state conditions. The function of the output *y* is to enable the count of the next-higher decade while the same pulse switches the present decade from 1001 to 0000.

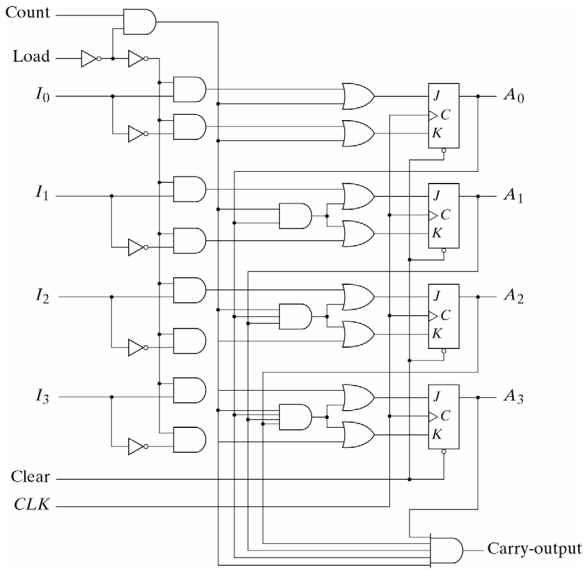
The flip-flop input equations can be minimized by means of *K*-maps. The simplified functions are:

$$T_{Q1} = 1, \qquad T_{Q2} = Q_8' Q_1$$
$$T_{Q4} = Q_2 Q_1, \quad T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

Its logic diagram requires four *T* flip-flops, five AND gates and one OR gate.

5.3 Binary Counter with Parallel Load

A 4-bit binary counter with parallel load capability:

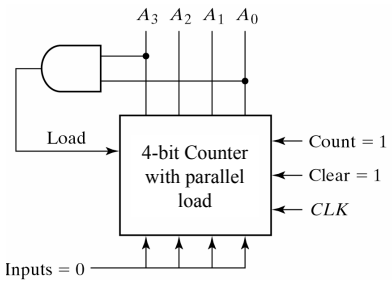


Its operation is summarized in the following table:

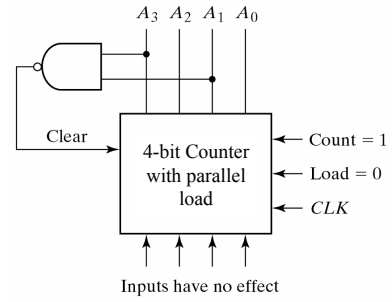
Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	*X*	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

A counter with parallel load can be used to create any desired count sequence. For example, the 4-bit counter with parallel load shown previously can be used to generate a BCD count in two ways:

1. Using the load input:



2. Using the clear input:



6. Other Counters

Counters can be constructed also by means of shift registers. Examples include the ring counter and the Jonhson counter.

6.1 Counter with Unused States

A circuit with *n* flip-flops has 2^n binary states. There are occasions when a sequential circuit uses less than 2^n states. The unused states may be treated as *don't care* conditions or may be assigned specific next states. Once the circuit is designed and realized, outside interference may cause it to enter one of the unused states. In that case it is important to ensure that the circuit can resume normal operation.

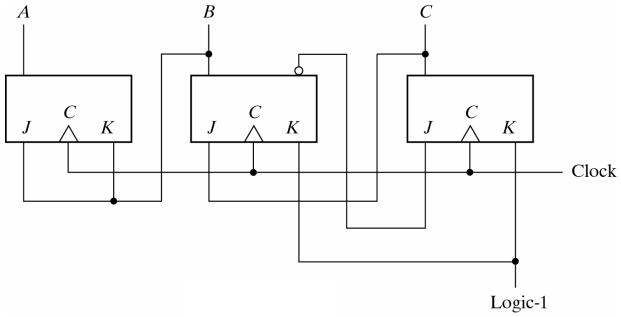
Consider the counter specified by the table:

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

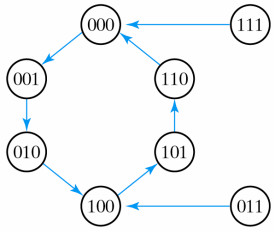
The flip-flop input equations (after simplification) are:

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = B' \qquad K_C = 1$$

The logic diagram of the counter is:

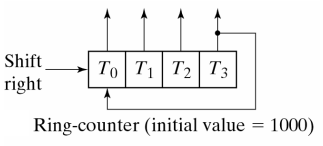


The state diagram including the effect of the unused states is:

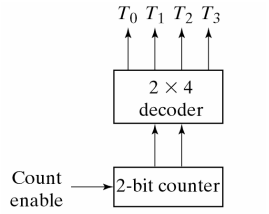


6.2 Ring Counter

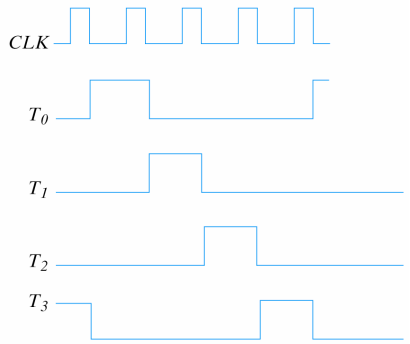
It is a circular shift register with only one flip-flop being set at any particular time, all others are cleared. The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals. A 4-bit shift register connected as a ring counter is shown below:



Ring-counter (initial value = 1000)



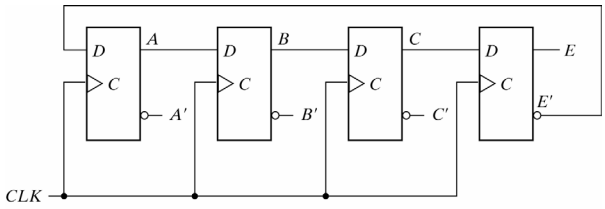
Counter and decoder



Sequence of four timing signals

6.3 Johnson Counter

An interesting variation of the ring counter is obtained if, instead of the *Q* output we take the *Q*′ of the last stage and feed it back to the first stage. A four-stage *switch-tail* counter is shown below:



Starting from a cleared state, the switch-tail counter goes through a sequence of eight states as listed below:

Sequence number	Flip-flop outputs				AND gate required for output
	A	*B*	*C*	*E*	
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 *k*-bit switch-tail counter with 2*k* decoding gates to provide outputs for 2*k* timing signals.