Counters

CS207 Chapter A

James YU yujq3@sustech.edu.cn

Department of Computer Science and Engineering Southern University of Science and Technology

Jul. 19, 2021



Counters

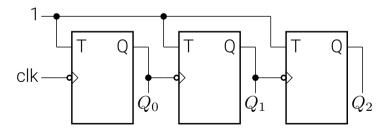


- Counters are one of the simplest types of sequential networks.
- A counter is usually constructed from one or more flip-flops that change state in a prescribed sequence when input pulses are received.
- Since the clock pulses occur at known intervals, the counter can be used as **an instrument for measuring time** and therefore period of frequency.
 - Asynchronous and synchronous counters.
 - Single and multimode counters.
 - Modulus counters.

Aynschronous counters



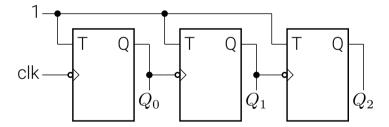
- A.k.a. Serial or ripple counters.
- The simplest counter circuit can be built using T flip-flops because the toggle feature is naturally suited for the implementation of the counting operation.
- All the flip-flops are not driven by the same clock pulse.
 - The successive flip-flop is triggered by the output of the previous flip-flop.
 - Hence the counter has cumulative settling time, which limits its speed of operation.



Aynschronous counters



- The clock inputs of the three flip-flops are connected in cascade.
- The T input of each flip-flop is connected to a constant 1, which means that the state of the flip-flop will toggle (reverse) at each negative edge of its clock.
- We are assuming that the purpose of this circuit is to count the number of pulses that occur on the primary input CLK (Clock).
- Thus the clock input of the first flip-flop is connected to the CLK line.



Asynchronous counters



- The counter has 8 different states.
 - It is a MOD-8 asynchronous counter.
- The Modulus (or MOD-number) of a counter is the total number of unique states it passes through in each of the complete cycles.
- The maximum binary number that can be counted by the counter is 2^n-1 .

State	Q_2	Q_1	Q_0	
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

Asynchronous counters

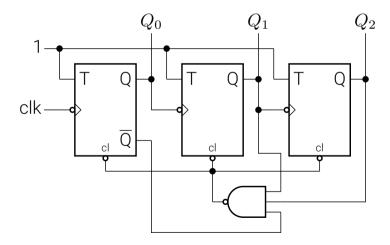


- The input consists of a sequence of pulses of frequency f.
- Q_0 changes only when the clock makes a transition from 1 to 0.
 - Two input pulses will result in a single pulse in Q_0 .
 - The frequency of Q_0 is f/2.
- Similarly, the frequency of Q_1 signal will be half that of Q_0 signal, i.e., f/4.
- The frequency of Q_2 signal will be half that of Q_1 signal, i.e., f/8.
- Frequency divider.
 - If there are n flip-flops used in the circuit then the frequency will be divided by 2^n .

<u> </u>					
State	Q_2	Q_1	Q_0		
0	0	0	0		
1	0	0	1		
2	0	1	0		
3	0	1	1		
4	1	0	0		
5	1	0	1		
6	1	1	0		
7	1	1	1		

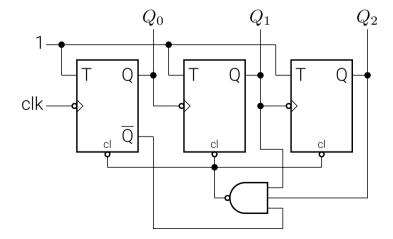


- In practice, it is often required to have a counter which has a MOD-number less than 2^n .
- In such cases, it is required that the counter will skip states that are normally a part of the counting sequences.



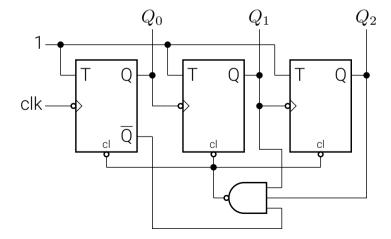


- Although the counter goes to the 110 state, it remains there only for a few nanoseconds before it recycles to the 000 state.
- Hence we may say that the counter counts from 000 to 101, it skips the states
 110 and 111
- Works as a MOD-6 counter.





- Although the counter goes to the 110 state, it remains there only for a few nanoseconds before it recycles to the 000 state.
- Hence we may say that the counter counts from 000 to 101, it skips the states
 110 and 111
- Works as a MOD-6 counter.





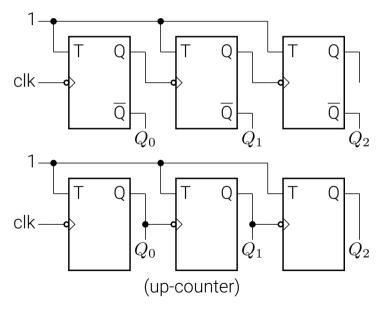


- To construct any MOD-N counter, the following general steps are to be followed.
 - Find the number of flip-flops n required for the desired MOD-number using the equation $2^n 1 < N < 2^n$.
 - Then connect all the n flip-flops as a ripple counter.
 - Find the binary number for N.
 - Connect all the flip-flop outputs, for which Q=1, as well as Q'=1, when the count is N, as inputs to the NAND gate.
 - Connect the NAND gate output to the clear input of each flip-flop.
- When the counter reaches the N-th state, the output of the NAND gate goes low, resetting all flip-flops to \emptyset .

Asynchronous down-counter



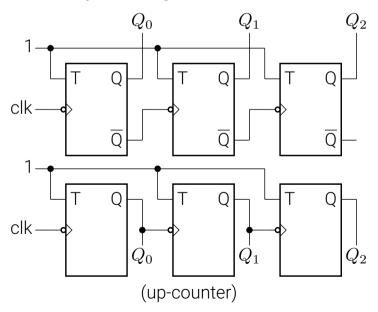
- A down-counter using n flip-flops counts downward starting from a maximum count of 2^n-1 to zero.
- Such a down-counter may be designed in three different ways as follows.



Asynchronous down-counter



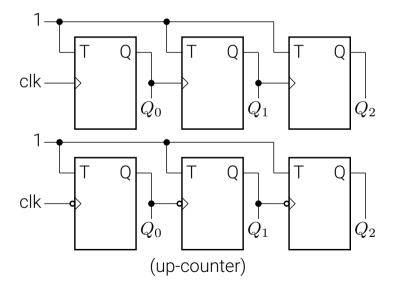
- A down-counter using n flip-flops counts downward starting from a maximum count of 2^n-1 to zero.
- Such a down-counter may be designed in three different ways as follows.



Asynchronous down-counter



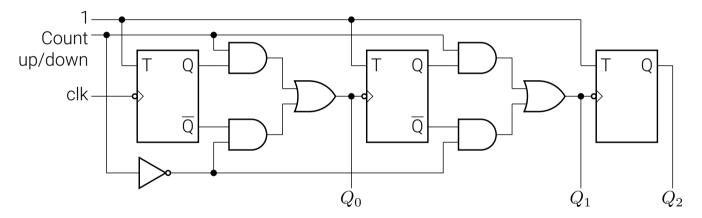
- A down-counter using n flip-flops counts downward starting from a maximum count of 2^n-1 to zero.
- Such a down-counter may be designed in three different ways as follows.



Asynchronous up-down counter



- We have already considered up-counters and down-counters separately.
- But both of the units can be combined in a single *up-down counter*.
 - Such a counter is also called a *multimode counter*.



Synchronous counters

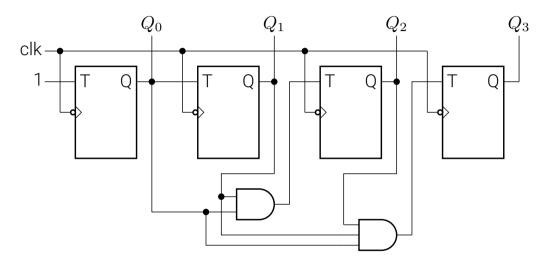


- The ripple or asynchronous counter is the simplest to build, but its highest operating frequency is **limited because of ripple action**.
 - Each flip-flop has a delay time.
 - In ripple counters these delay times are additive and the total "settling" time for the counter is approximately the product of the delay time of a single flip-flop and the total number of flip-flops.
 - There is the possibility of glitches occurring at the output of decoding gates used with a ripple counter.
- Both of these problems can be overcome, if all the flip-flops are clocked synchronously.
- The resulting circuit is known as a synchronous counter.

Synchronous counters



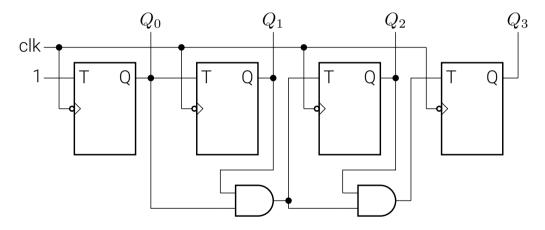
• Synchronous counter with parallel carry.



Synchronous counters



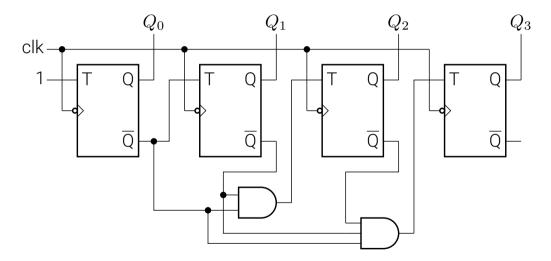
- Synchronous counter with ripple carry.
 - As the number of stages increases, the number of AND gates also increases, along with the number of inputs for each of those AND gates.



Synchronous down-counter



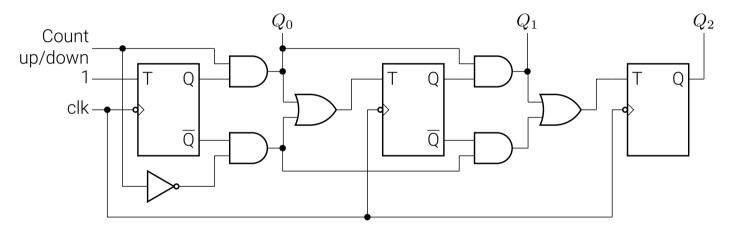
• A parallel down-counter can be made to count down by using the inverted outputs of flip-flops to feed the various logic gates.



Synchronous up-down counter



• Combining both the functions of up- and down-counting in a single counter, we can make a *synchronous up-down counter*.



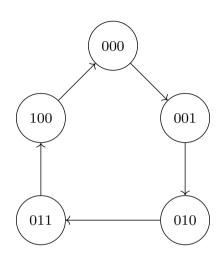
Design a synchronous counter



- Following certain general steps, synchronous counters of any given count sequence and modulus can be designed. The steps are listed below:
 - From the given word description of the problem, draw a state diagram that describes the operation of the counter.
 - From the state table, write the count sequences in the form of a table.
 - Find the number of flip-flops required.
 - Decide the type of flip-flop to be used for the design of the counter. Then determine the flip-flop inputs that must be present for the desired next state from the present state using the excitation table of the flip-flops.
 - Prepare K-maps for each flip-flop input in terms of flip-flop outputs as the input variables. Simplify the K-maps and obtain the minimized expressions.
 - Connect the circuit using flip-flops and other gates corresponding to the minimized expressions.

Design a MOD-5 counter





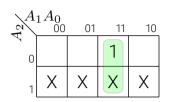
Design a MOD-5 counter

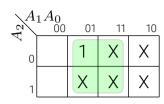


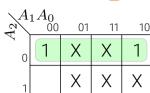
Present		Next		Flip-flip Inputs							
$\overline{A_2}$	A_1	$\overline{A_0}$	$\overline{A_2}$	A_1	$\overline{A_0}$	$\overline{J_{A2}}$	K_{A2}	$\overline{J_{A1}}$	K_{A1}	$\overline{J_{A0}}$	K_{A0}
0	0	0	0	0	1	0	Χ	0	Χ	1	Χ
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

Design a MOD-5 counter





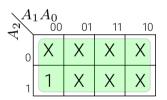




$$J_{A2} = A_1 A_0$$

$$J_{A1} = A_0$$

$$J_{A0} = A_2'$$



$$K_{A2} = 1$$

$$K_{A1} = A_0$$

$$K_{A0}=1$$

Lock out



- In the counters with modulus less than 2^n , it may happen that the counter by chance finds itself in any one of the unused states.
 - See 101, 110, 111 in the example above.
- If by chance the counter enters into any one of these unused states, its next state will not be known.
- It may be possible that the counter might go from one unused state to another and never arrive at a used state.
- A counter whose unused states have this feature is said to suffer from lock out.
- To ensure that lock out does not occur, we design the counter **assuming the next state to be the initial state, from each of the unused states**.