Mod-N **Asynchronous counter**

To design mod-N ripple counter first we need to find no.of flip flops required.

If no.of flip flops is n then $2^n \geq N$

n = no. of flip flops

N = No. of states

Example mod 6 $\Rightarrow 2^n \geq 6 \Rightarrow n = 3$

mod 5 $\Rightarrow 2^n \geq 5 \Rightarrow n = 3$

mod 11 $\Rightarrow 2^n \geq 11 \Rightarrow n = 4$

mod 10 $\Rightarrow 2^n \geq 10 \Rightarrow n = 4$

whenever $2^n = N$ then it is an n-bit Asynchronous Counter

$n = 3 \Rightarrow$ 3bit Asynchronous $\Rightarrow$ 8 states $\Rightarrow 2^3 = 8$

If $2^n > N$ then it is not not n-bit Asynchronous counter

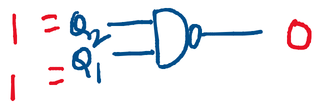
To realize mod-N (where $2^n > N$) using clear terminals refer to the expected state to the ckt with Nth clock pulse

whenever the flip flops, which are having the value as '1' at expected Nth clock pulse are the inputs to the NAND gate and NAND gate output is connected to clear terminals of all flip flops connected to clear terminals of all flip flops

A	B	NAND $\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

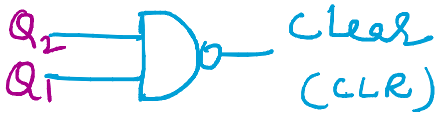
$Q_2 Q_1 Q_0$

6 → Binary = 110



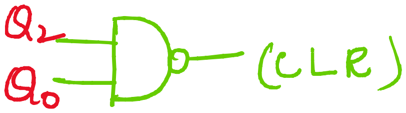
Mod 6

CLK	Q_2	Q_1	Q_0
6	1	1	0



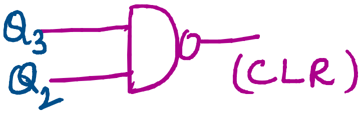
Mod 5

CLK	Q_2	Q_1	Q_0
5	1	0	1



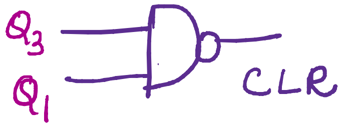
Mod 12

CLK	Q_3	Q_2	Q_1	Q_0
12	1	1	0	0

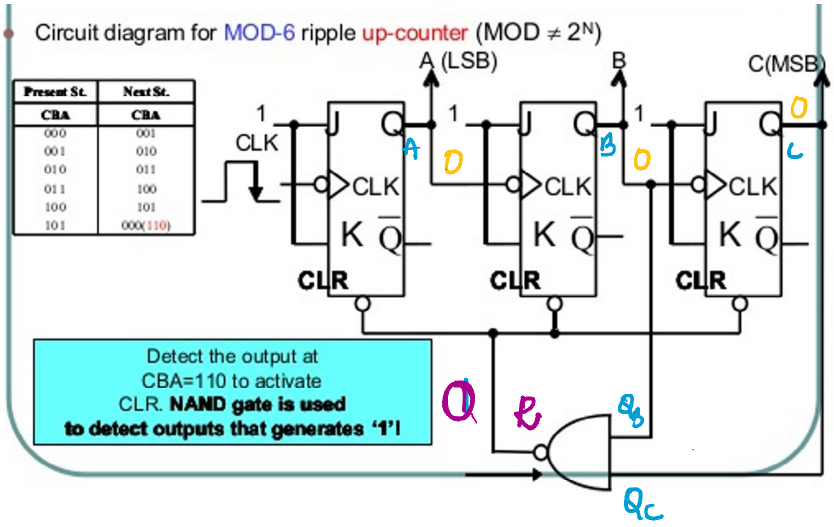


Mod 10

CLK	Q_3	Q_2	Q_1	Q_0
10	1	0	1	0



Mod-6 ripple up counter


Circuit diagram for MOD-6 ripple up-counter (MOD ≠ 2^N)

Present St.	Next St.
CBA	CBA
000	001
001	010
010	011
011	100
100	101
101	000(110)

Detect the output at CBA=110 to activate CLR. **NAND gate is used to detect outputs that generates '1'!**

	Q_C	Q_B	Q_A	$R = \overline{Q_B Q_C}$
0	0	0	0	
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0 skip
7	0	0	0	
8	0	0	0	

+ve −ve +ve −ve −ve +ve +ve −ve +ve −ve −ve

CLR is the active low Asynchronous

CLR = 0 → enabled

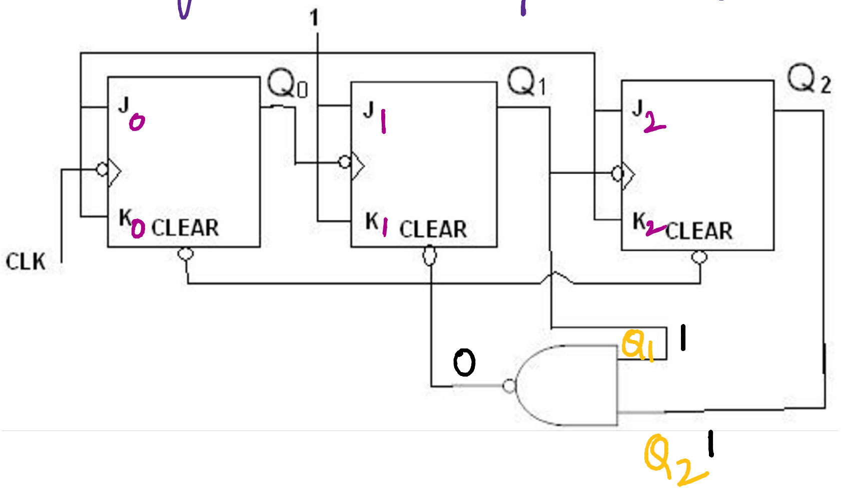
CLR = 1 → disabled

Here, R value (NAND gate o/p) is logic '1' it will perform normal clocked operation.

If NAND gate o/p is logic '0', ckt will perform asynchronous input operation. i.e all flip flops will be cleared i.e all flip flops will be cleared

(CLR = 0) → Asynchronous i/p

Mod-6 Asynchronous up counter



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

An **Asynchronous counter** can have 2n-1 possible counting states e.g. MOD-16 for a 4-bit counter, (0-15) making it ideal for use in Frequency Division applications. But it is also possible to use the basic asynchronous counter configuration to construct special counters with counting states less than their maximum output number. For example, modulo or MOD counters.

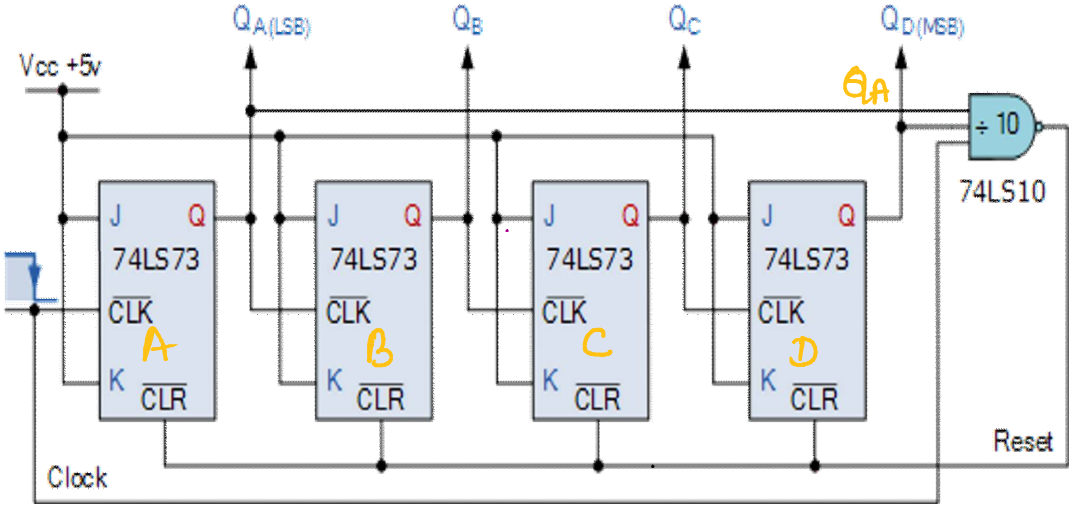
This is achieved by forcing the counter to reset itself to zero at a pre-determined value producing a type of asynchronous counter that has truncated sequences. Then an n-bit counter that counts up to its maximum modulus (2n) is called a full sequence counter and a n-bit counter whose modulus is less than the maximum possible is called a **truncated counter**.

But why would we want to create an asynchronous truncated counter that is not a MOD-4, MOD-8, or some other modulus that is equal to the power of two. The answer is that we can by using combinational logic to take advantage of the asynchronous inputs on the flip-flop.

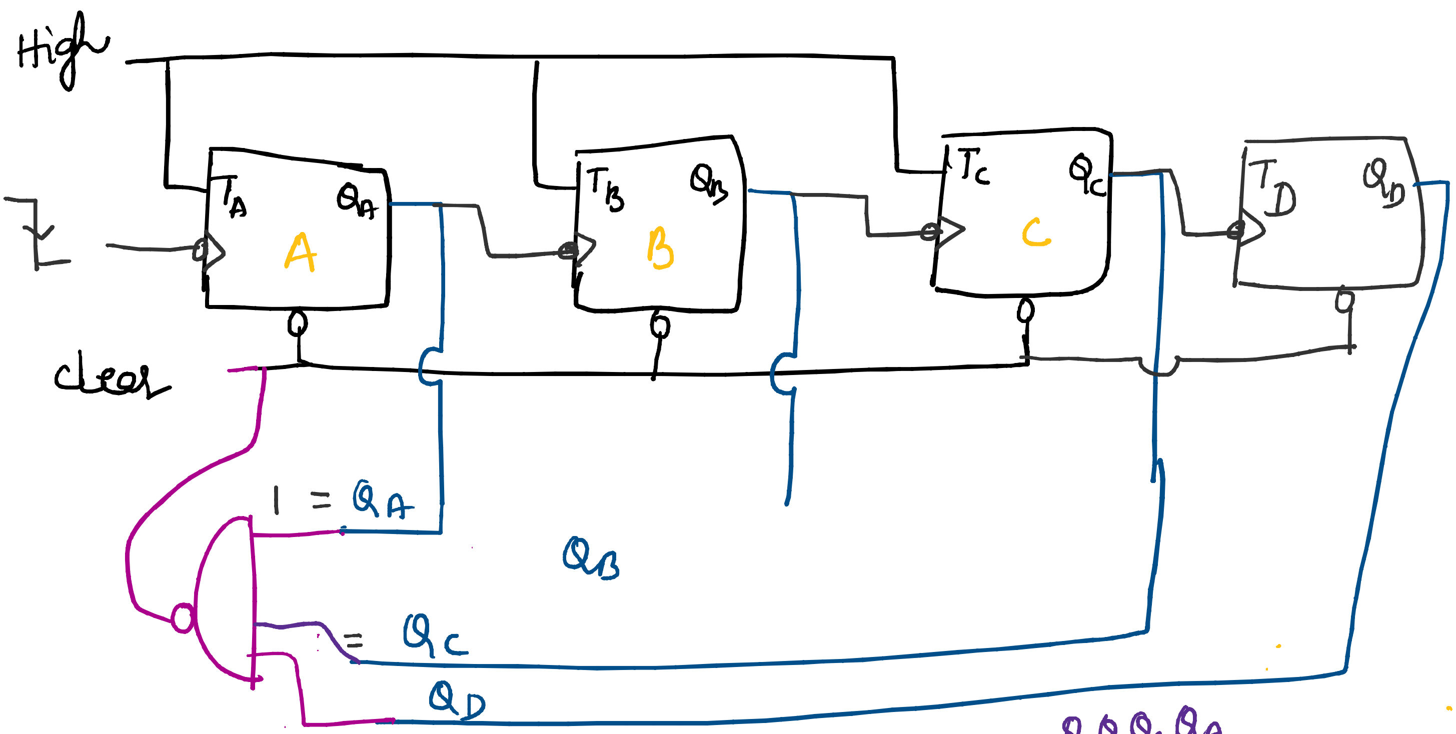
If we take the modulo-16 asynchronous counter and modified it with additional logic gates it can be made to give a decade (divide-by-10) counter output for use in standard decimal counting and arithmetic circuits.

Such counters are generally referred to as **Decade Counters**. A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and to do this we need to feed this condition back to the reset input. A counter with a count sequence from binary "0000" (BCD = "0") through to "1001" (BCD = "9") is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.

Asynchronous Decade Counter



Mod 13 Asynchronous up counter



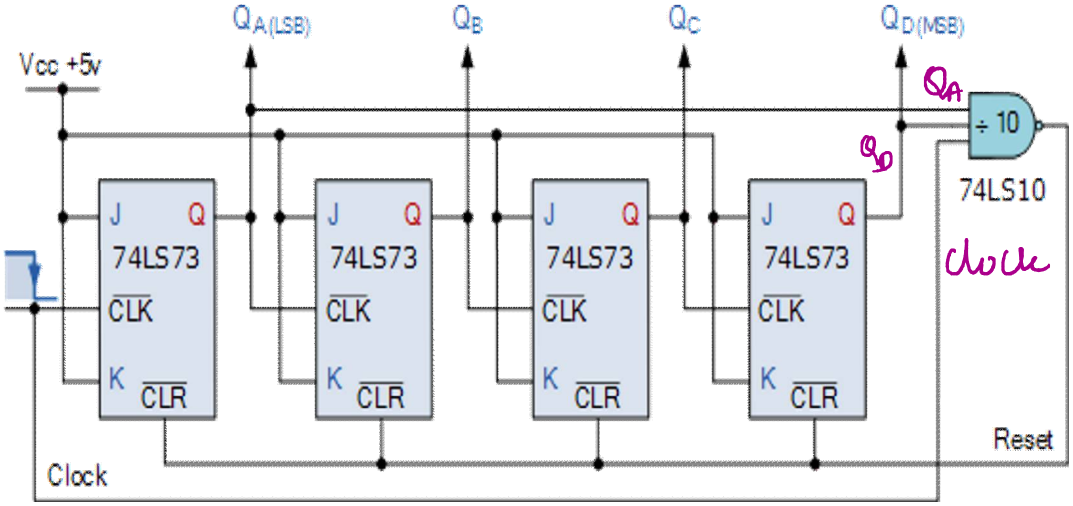
$Q_D Q_C Q_B Q_A$

1110 ⇒ 14

1100 = 12

1101 = 13

Mod-10 ripple Counter



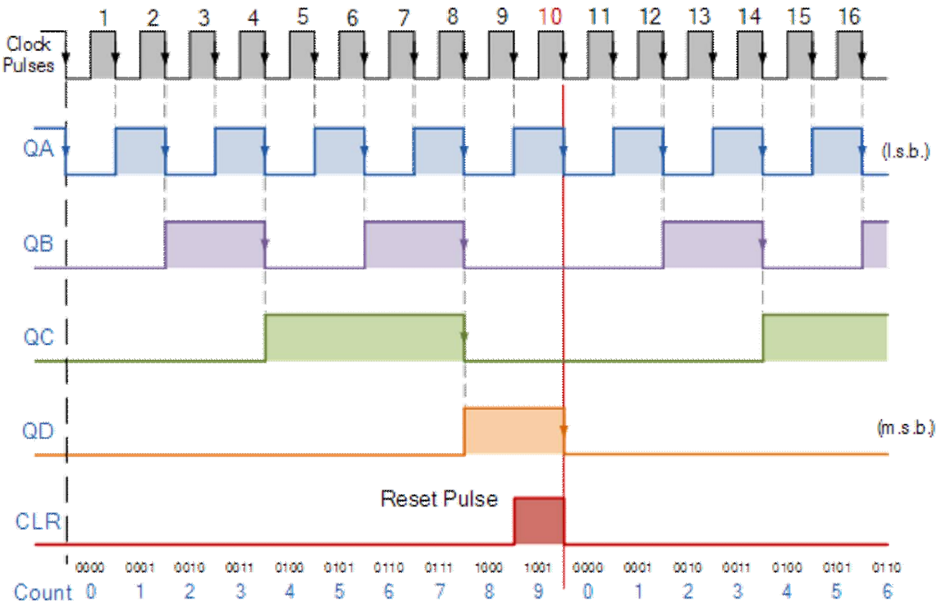
This type of asynchronous counter counts upwards on each trailing edge of the input clock signal starting from 0000 until it reaches an output 1001 (decimal 9). Both outputs QA and QD are now equal to logic “1”. On the application of the next clock pulse, the output from the 74LS10 NAND gate changes state from logic “1” to a logic “0” level.

As the output of the NAND gate is connected to the CLEAR (CLR) inputs of all the 74LS73 J-K Flip-flops, this signal causes all of the Q outputs to be reset back to binary 0000 on the count of 10. As outputs QA and QD are now both equal to logic “0” as the flip-flop’s have just been reset, the output of the NAND gate returns back to a logic level “1” and the counter restarts again from 0000. We now have a decade or Modulo-10 up-counter.

Decade Counter Truth Table

Clock Count	Output bit Pattern				Decimal Value
	QD	QC	QB	QA	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	2
4	0	0	1	1	3
5	0	1	0	0	4
6	0	1	0	1	5
7	0	1	1	0	6
8	0	1	1	1	7
9	1	0	0	0	8
10	1	0	0	1	9
11	Counter Resets its Outputs back to Zero				

Decade Counter Timing Diagram



By using the same idea of truncating counter output sequences, the above circuit could easily be adapted to other counting cycles be simply changing the connections to the inputs of the NAND gate or by using other logic gate combinations.