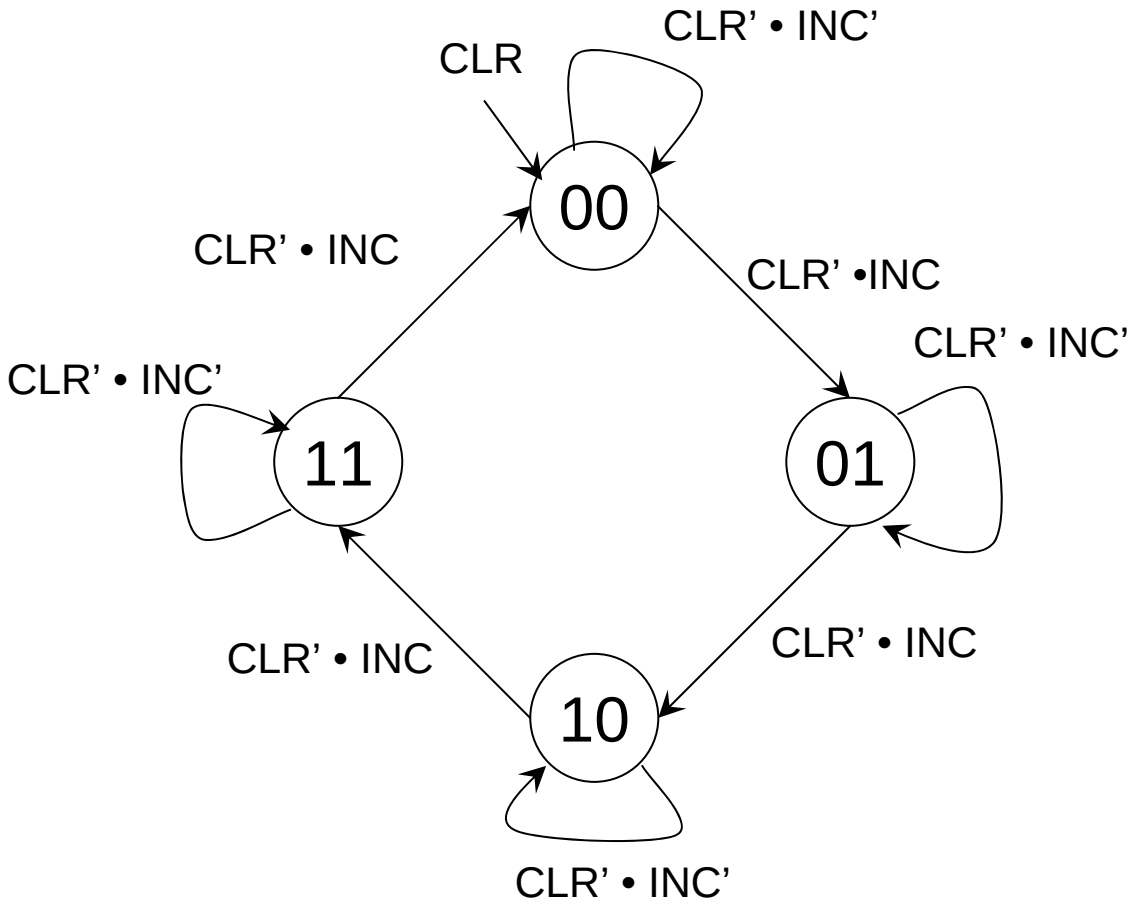
Cascaded Counters

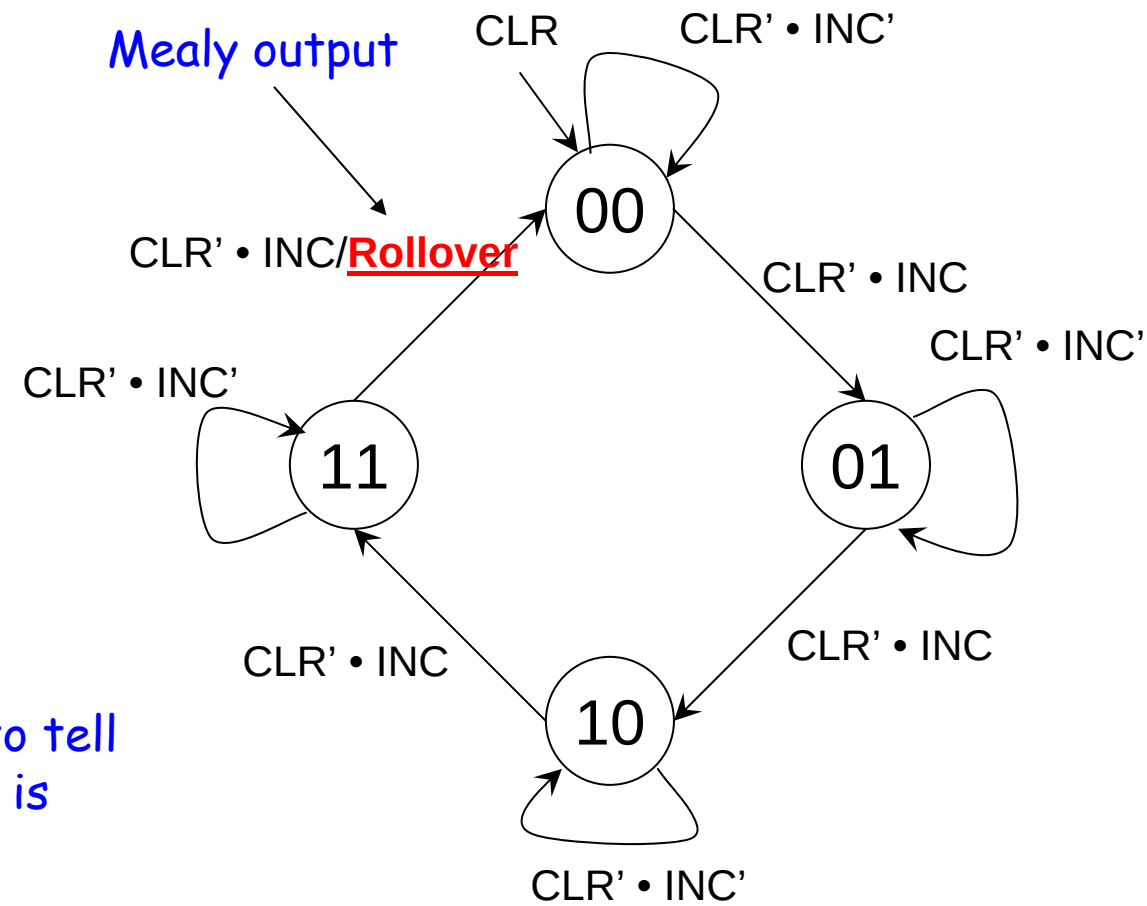
A Mod4 Counter
(A 2-bit counter)

CLR	INC	Q1	Q0	N1	N0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	0
1	-	-	-	0	0



A Mod4 Counter With a *Rollover* Signal

CLR	INC	Q1	Q0	N1	N0	Rollover
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	-	-	-	0	0	0



Signal Rollover can be used to tell other circuitry that counter is rolling over to all 0's

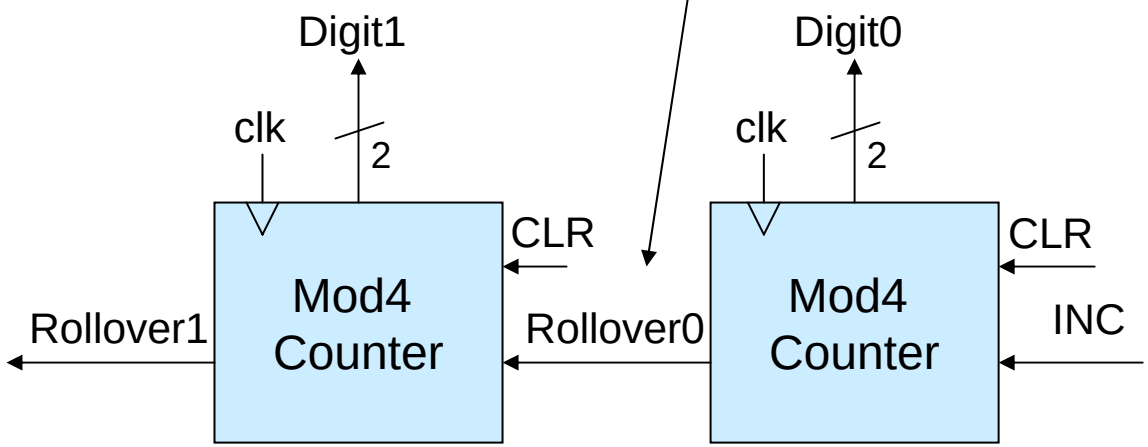
Cascading 2 Mod4 Counters

Digit1 Digit0

00
01
02
03
10
11
12
13
20
21
22
23
30
31
32
33
00

Count sequence

Increment higher digit's counter when lower digit's counter is rolling over

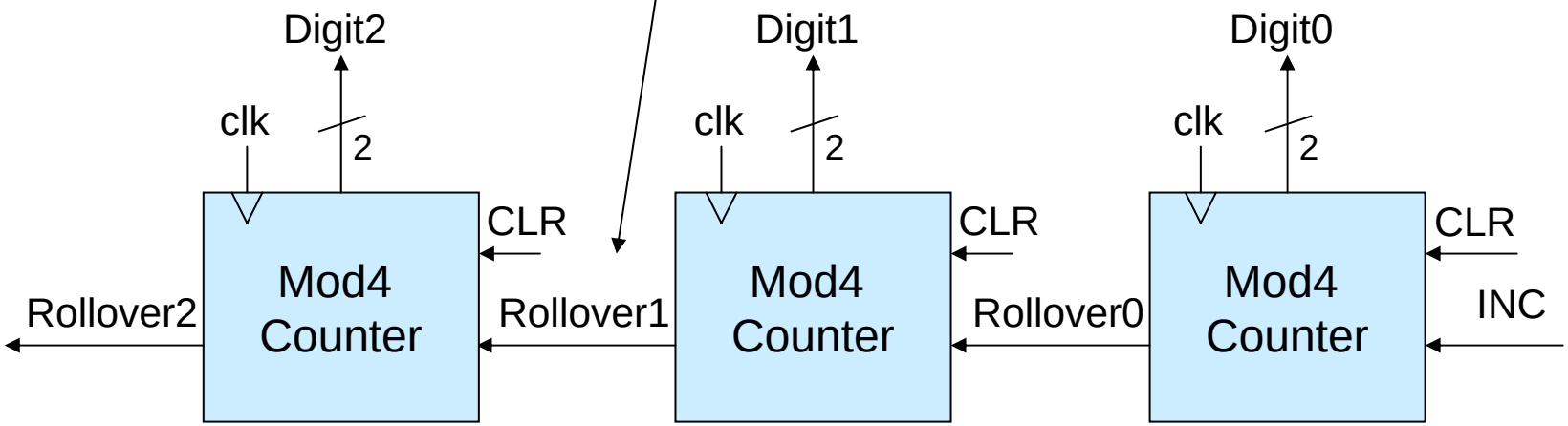


3 Digits' Worth

Could build a digital watch or clock circuit this way with Mod60 and Mod24 counters

Increment higher digit's counter when lower digit's counter is rolling over

Can do this with *any* counter that has a Rollover output



Cascading Counters

- A counter will increment only when
  - The counter below it is at its terminal count and it is being incremented
    - That *is* the definition of the Rollover signal
- Some people try to tie Rollover to the clk input of the next higher counter
  - Bad idea… Very bad idea…
  - Violates our Globally Synchronous policy
  - Doesn't work as intended

Sequence should be:
…-12-13-20-21-22-23-30-…

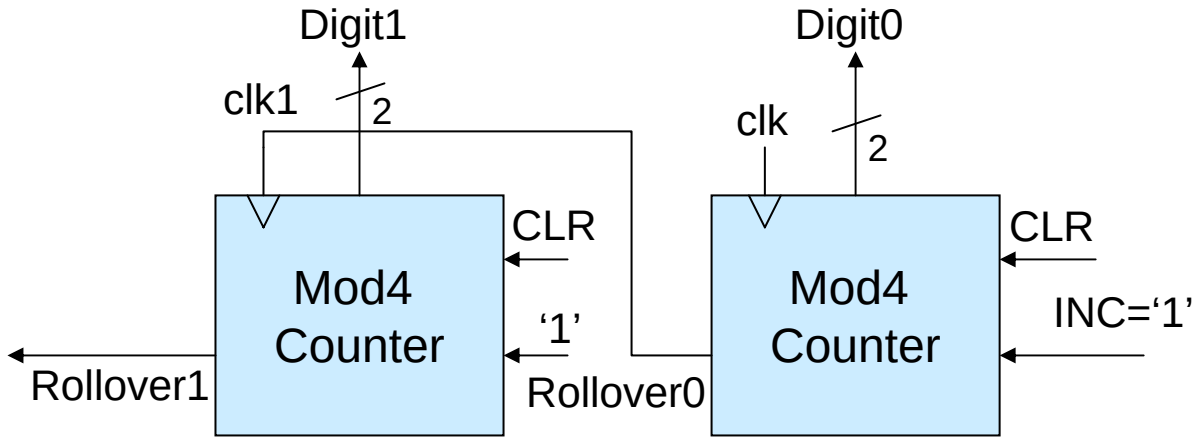
but we get:
…-12-23-20-21-22-33-30-…

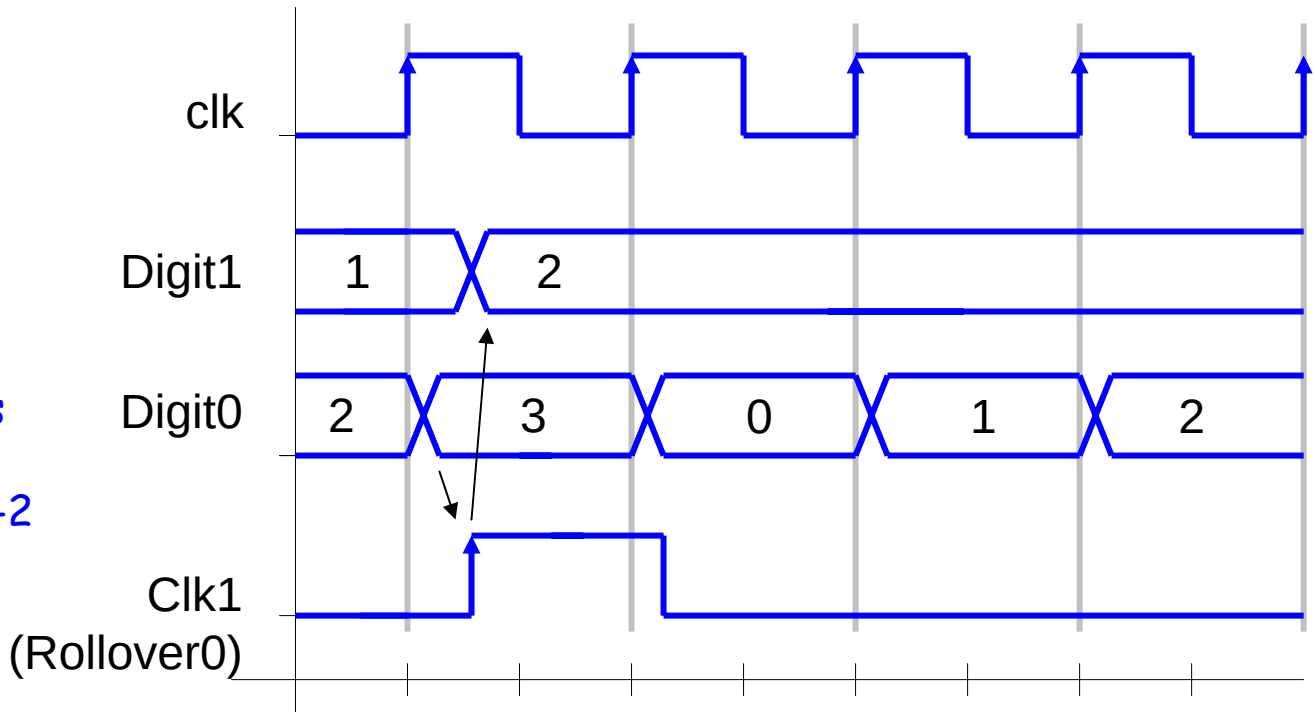
?????

DO NOT TIE CLK inputs on modules to anything but the clock !!!!!!

Even if you *tinker* until you get the right count sequence, you must guarantee that signal Rollover0 has no hazards

Digit0 transition from 1-2 makes this difficult if not impossible





Sequence should be:
…-12-13-20-21-22-23-30-…

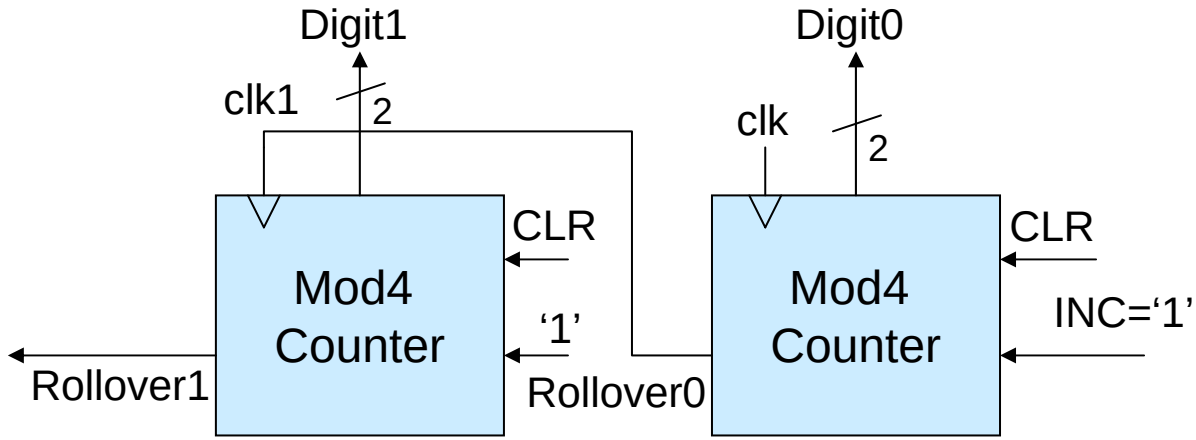
but we get:
…-12-23-20-21-22-33-30-…

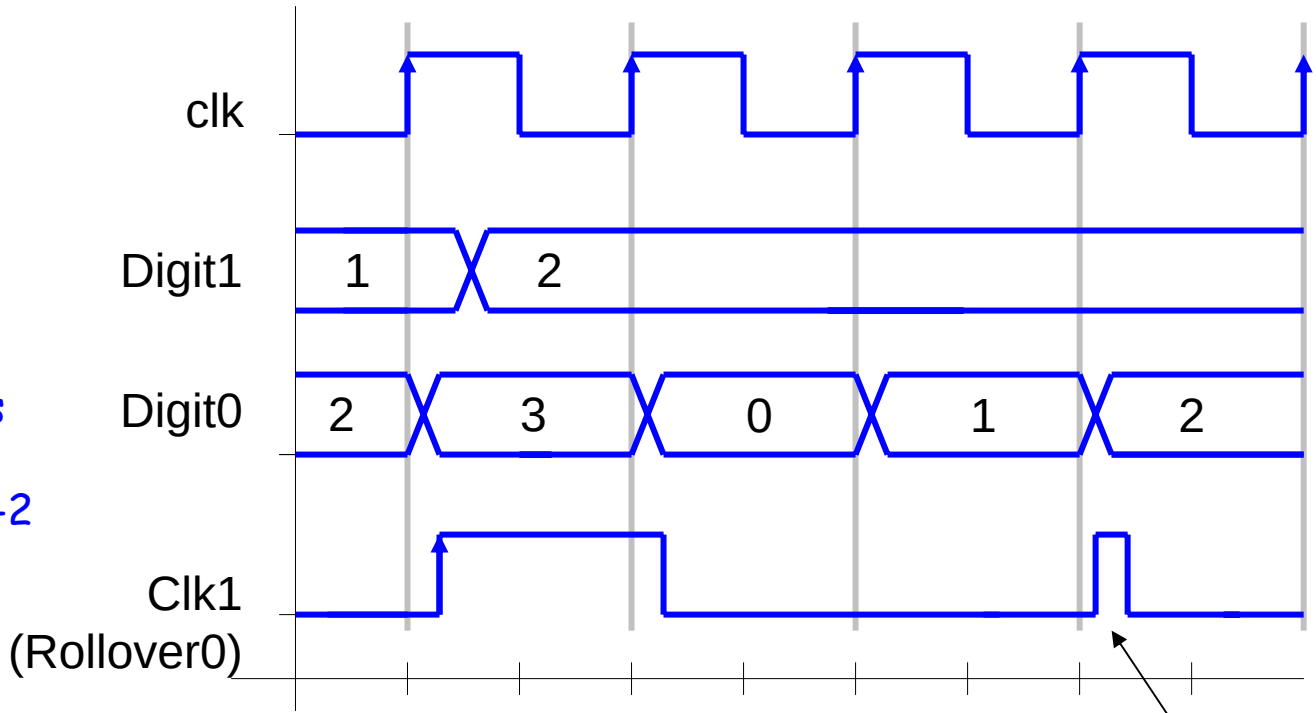
?????

DO NOT TIE CLK inputs on modules to anything but the clock !!!!!!

Even if you *tinker* until you get the right count sequence, you must guarantee that signal Rollover0 has no hazards

Digit0 transition from 1-2 makes this difficult if not impossible





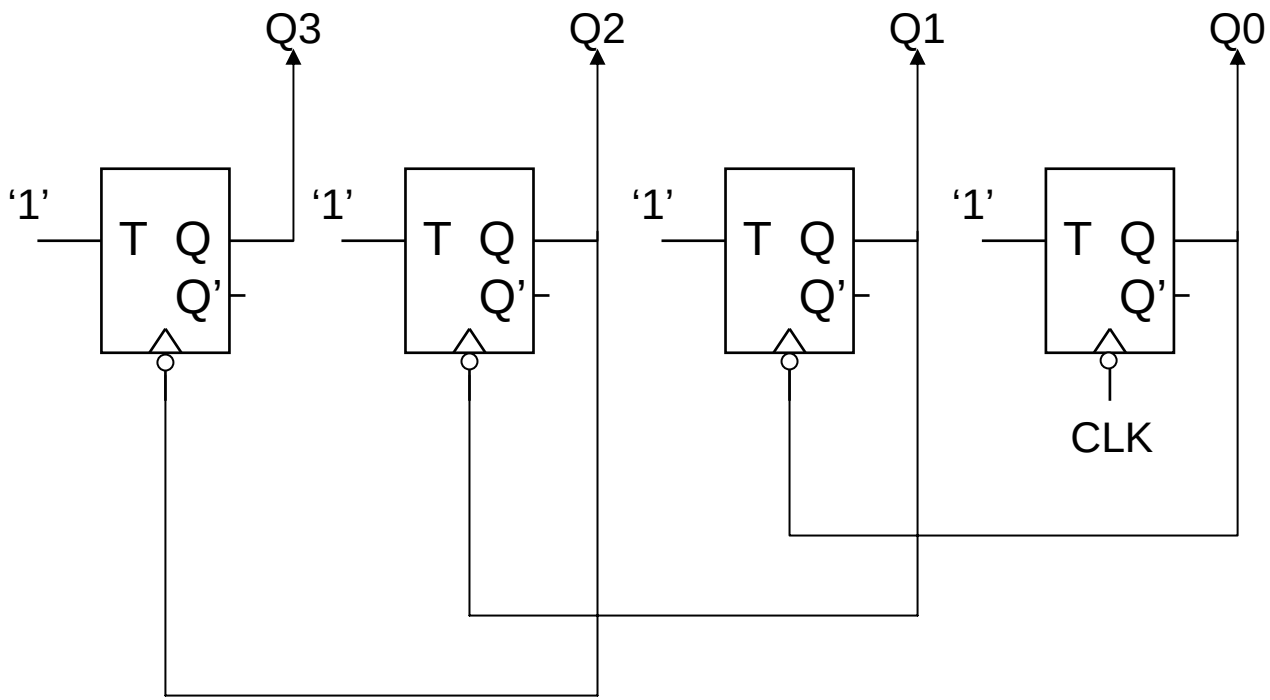
Ripple Counters

- When you tie a rollover-like signal to a clock on the next higher digit ⇔ ripple counter
- A ripple counter is an ASYNCHRONOUS counter
  - Transitions are not all synchronized to the clock
  - Different flip flops change at different times
  - Similar to gated clocks (seen earlier)
- Asynchronous circuits are an advanced topic

Another Common Ripple Counter

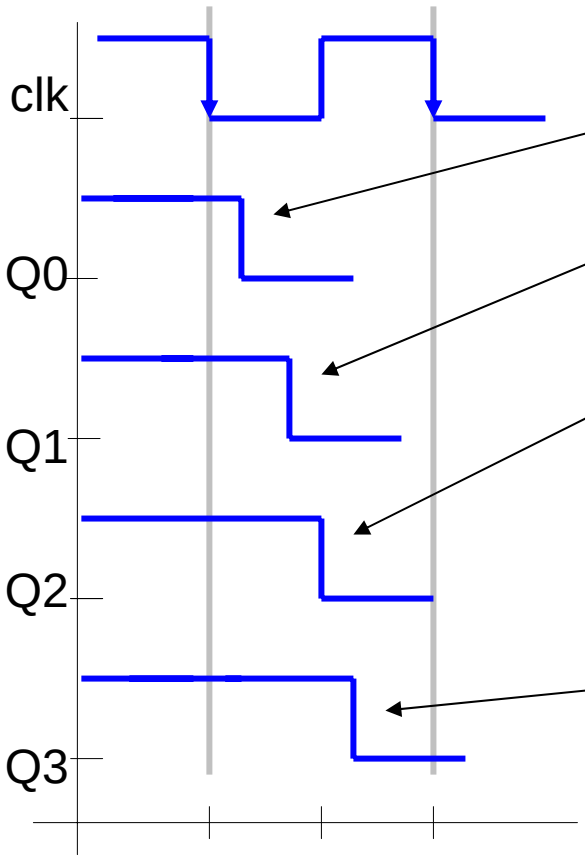
Sequence is:

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0000



So what is the problem?

Timing Diagram



Q0 changes in response to clock edge

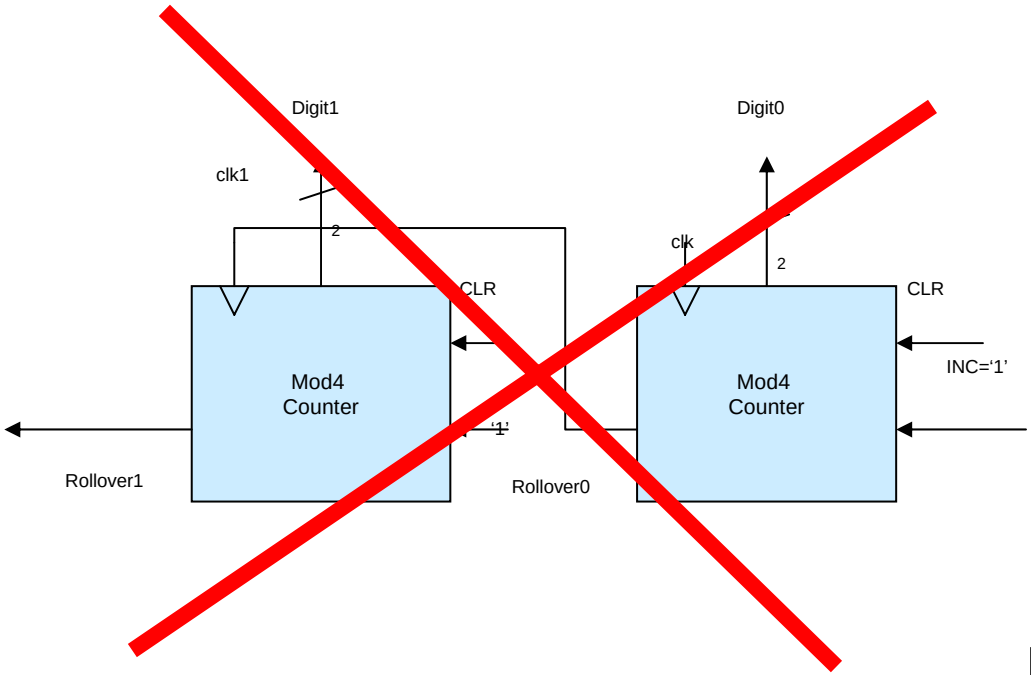
Only after it changes does Q1's FF get a clock

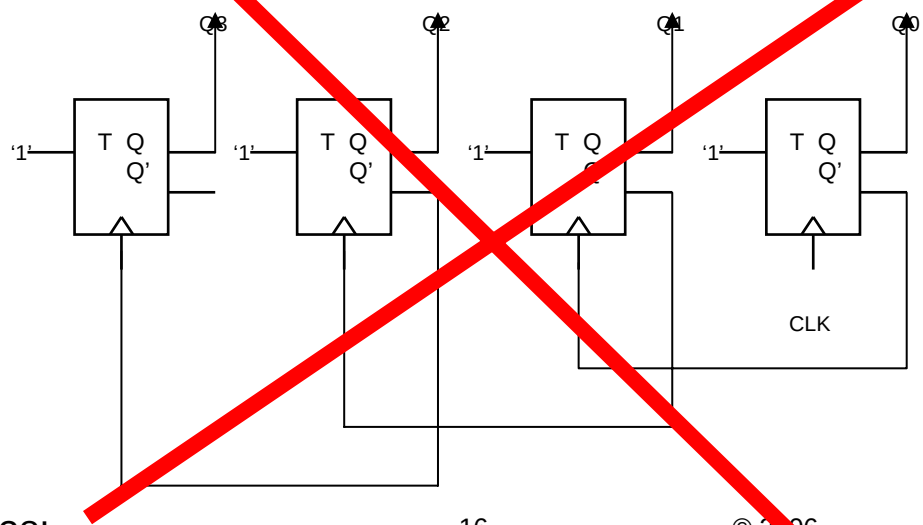
Only after that does Q2's FF get a clock

Net effect is that all the FF's change at different times

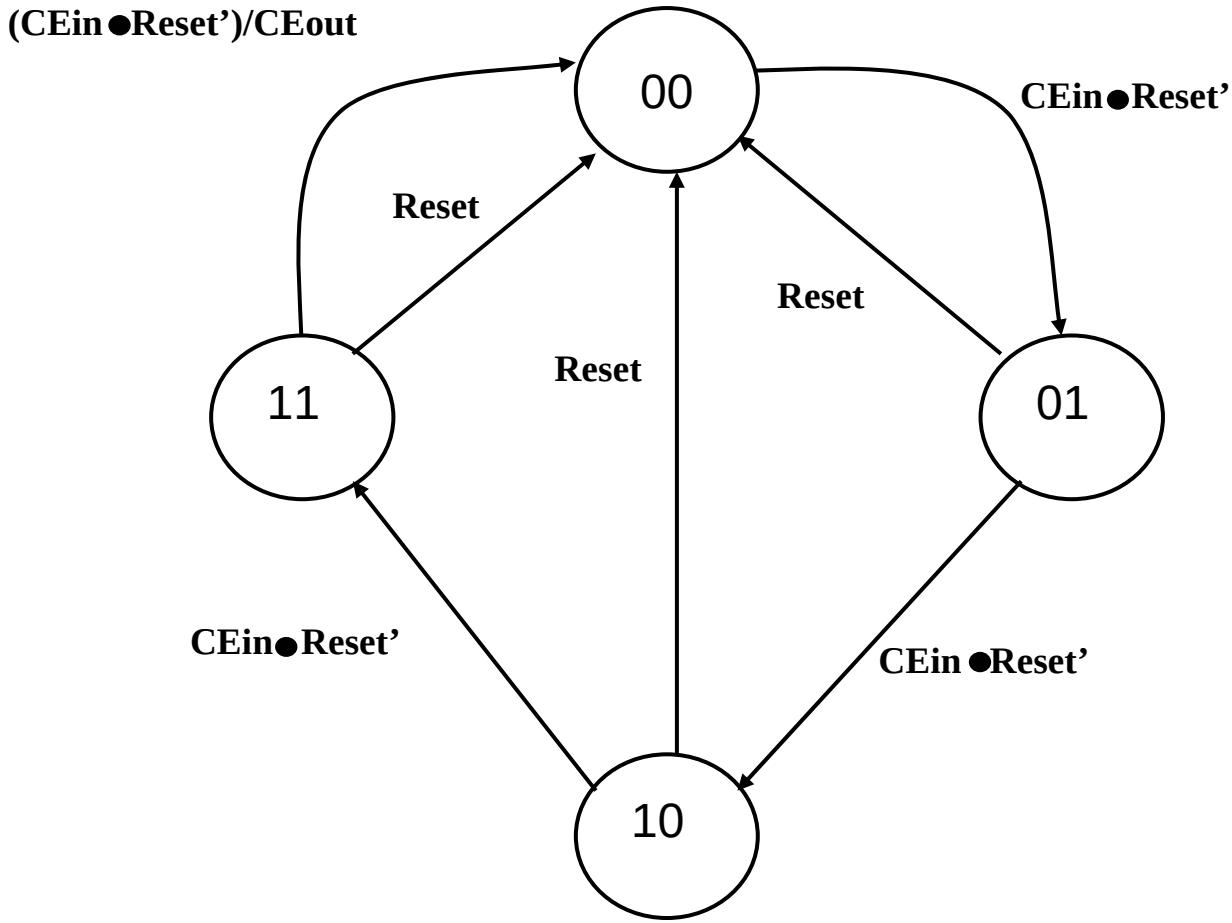
Logic depending on Q3 has very little time to react before next clock edge

Do Not Use Asynchronous or Ripple Counters

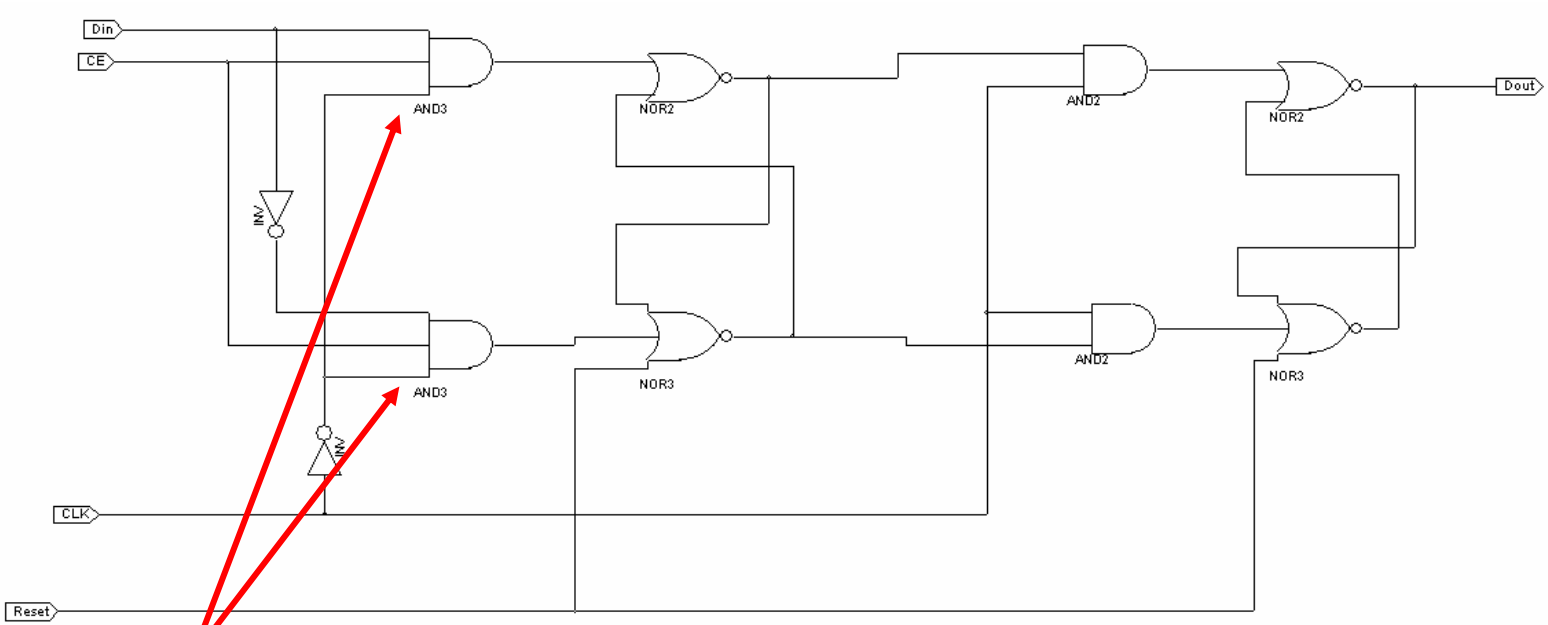




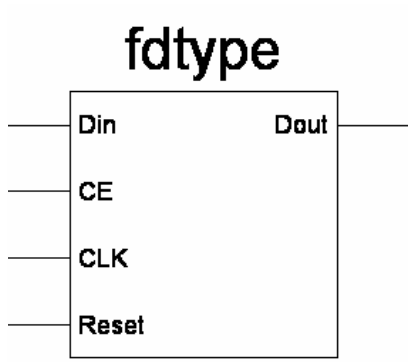
Mod4 Counter



The State Flip- Flop

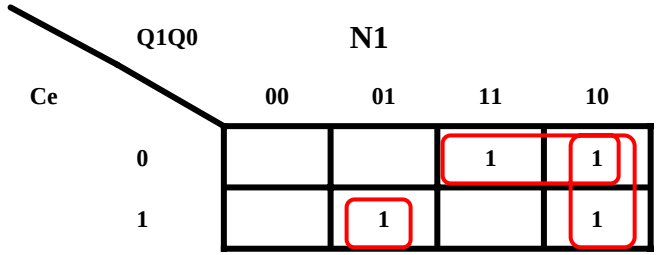


Note: Both CLK and CE are required to change the flip-flop

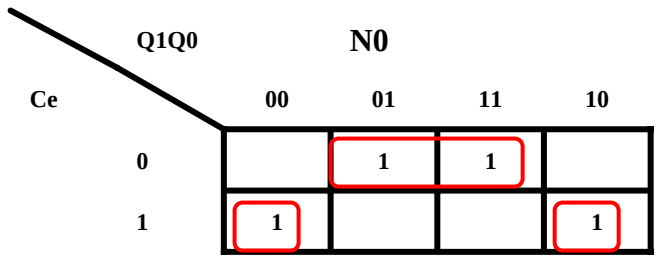


Mod4 Counter

CEin	Q1	Q0	N1	N0	CEout
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	1



$N1 = Ce'Q_1 + Q_1Q_0' + CeQ_1'Q_0$

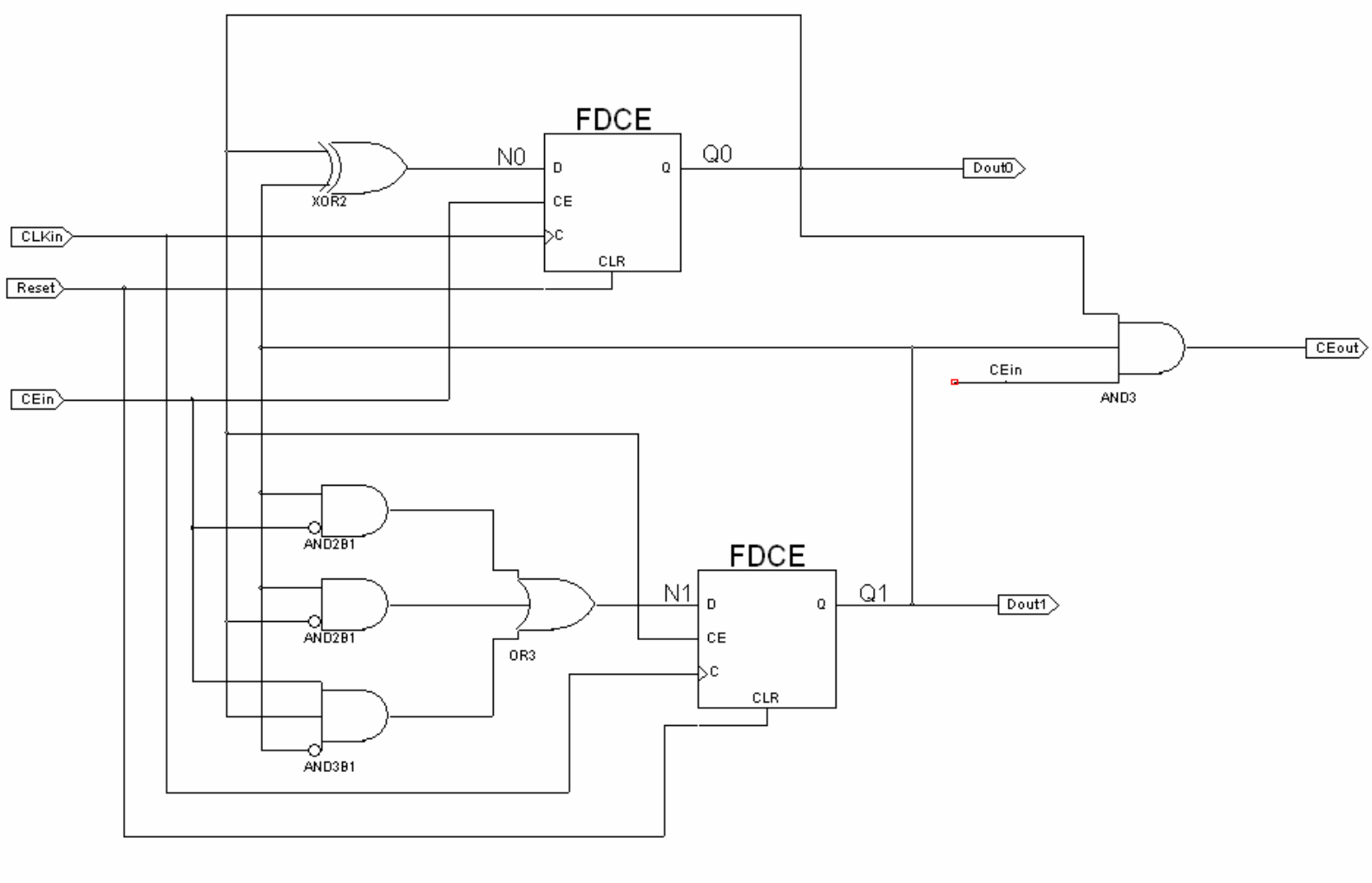


Note: No change when CEin=0

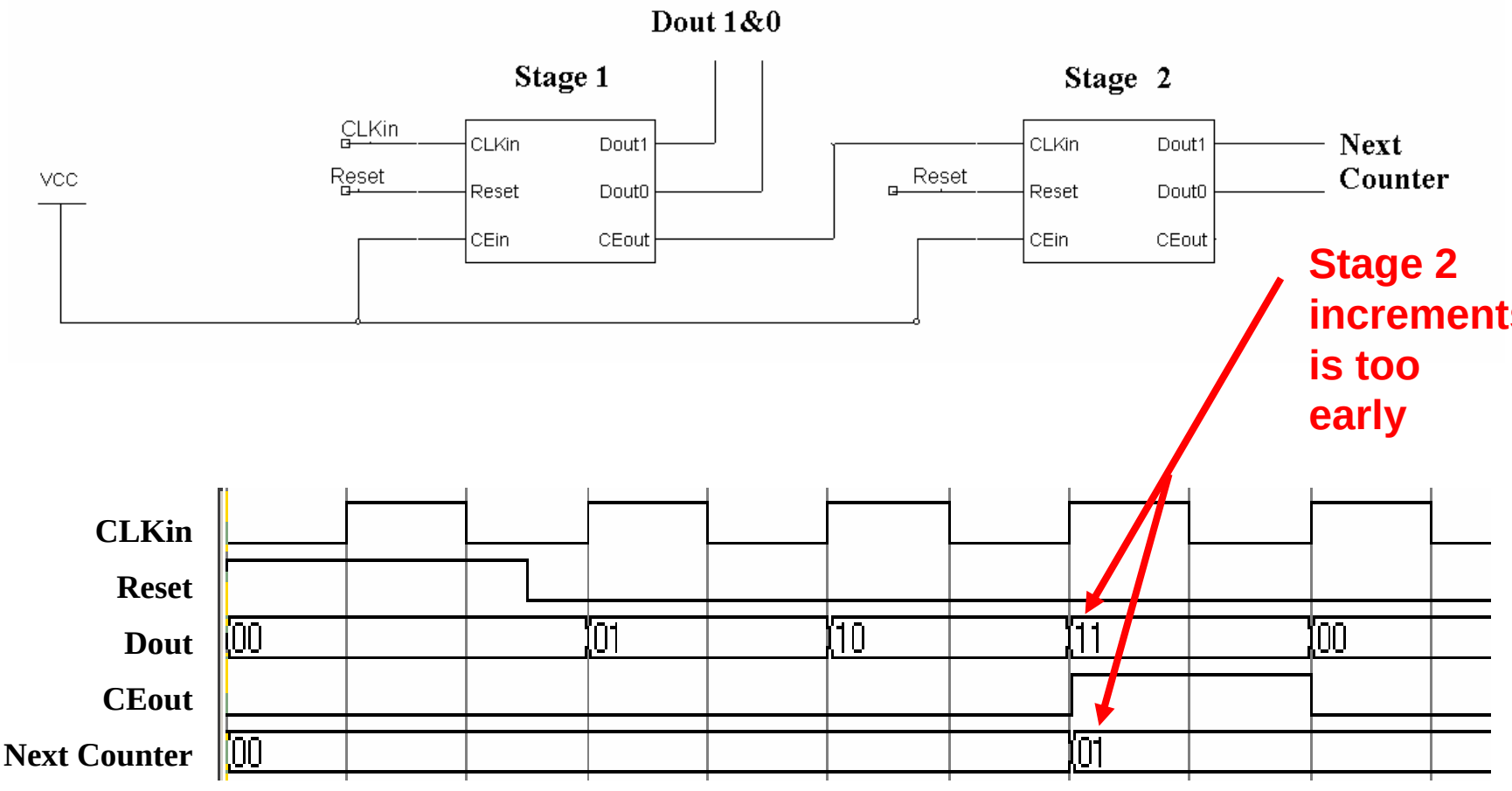
$CEout = CeQ_1Q_0$

$N0 = Ce'Q_0 + CeQ_0'$

Mod4 Counter

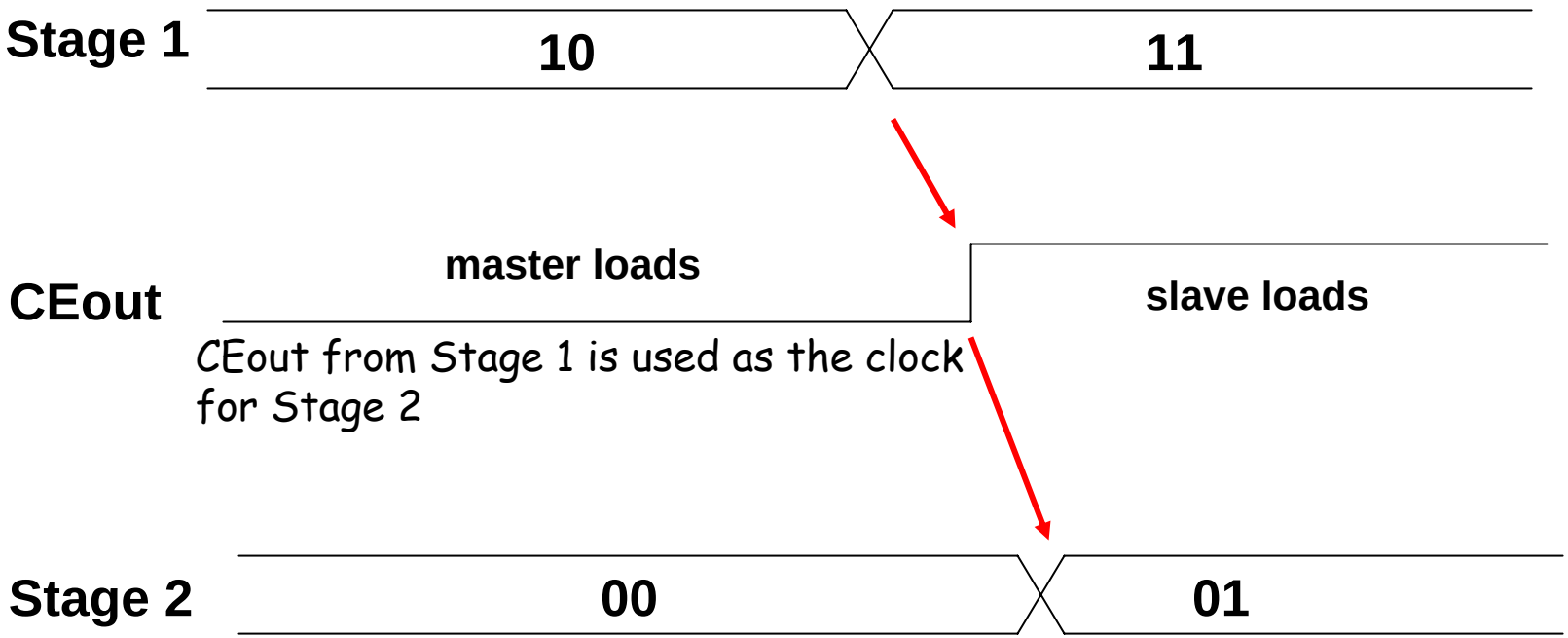


Cascaded Ripple Counter



Cascaded Ripple Counter

Why is Next Stage increment too early?

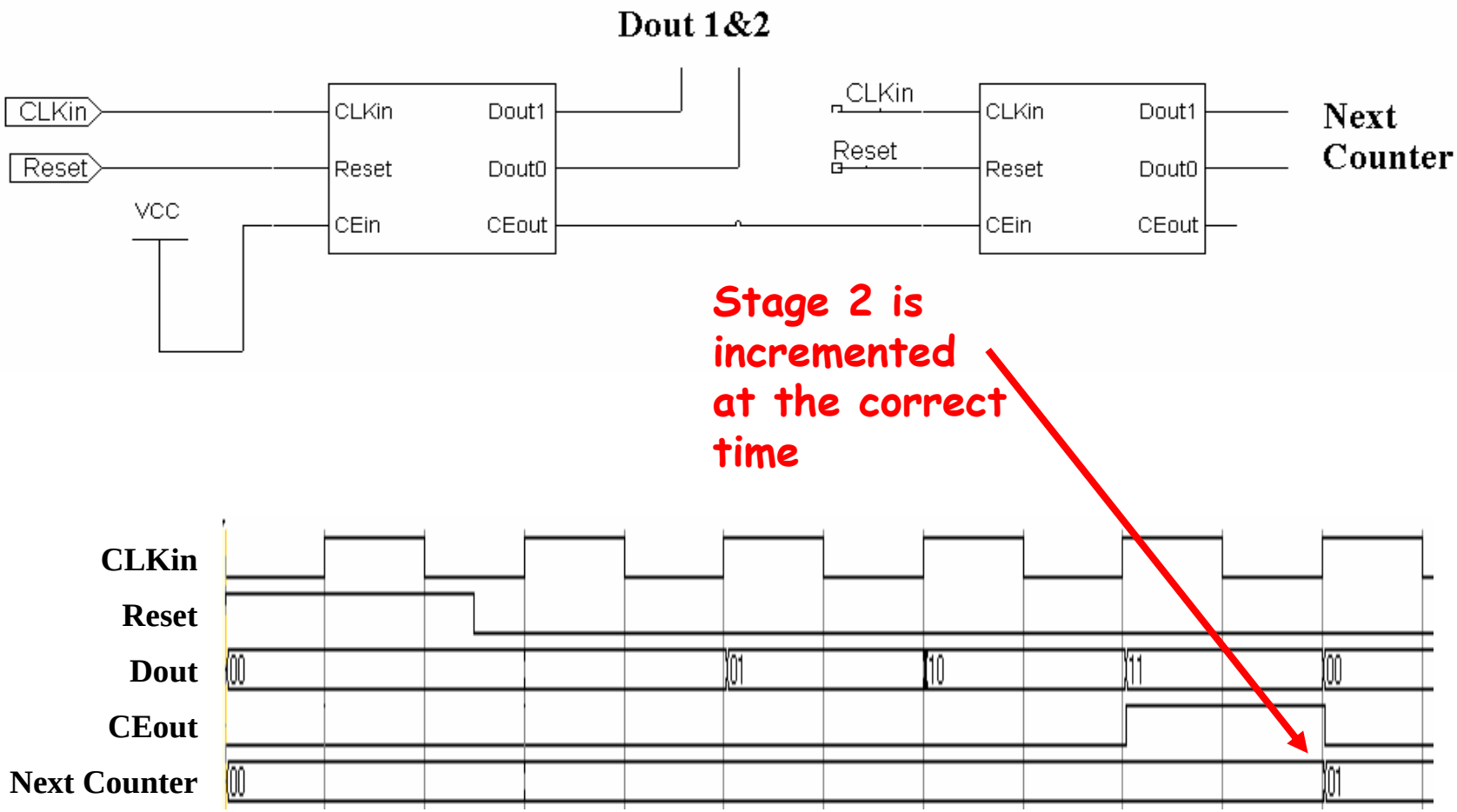


Cascaded Ripple Counter

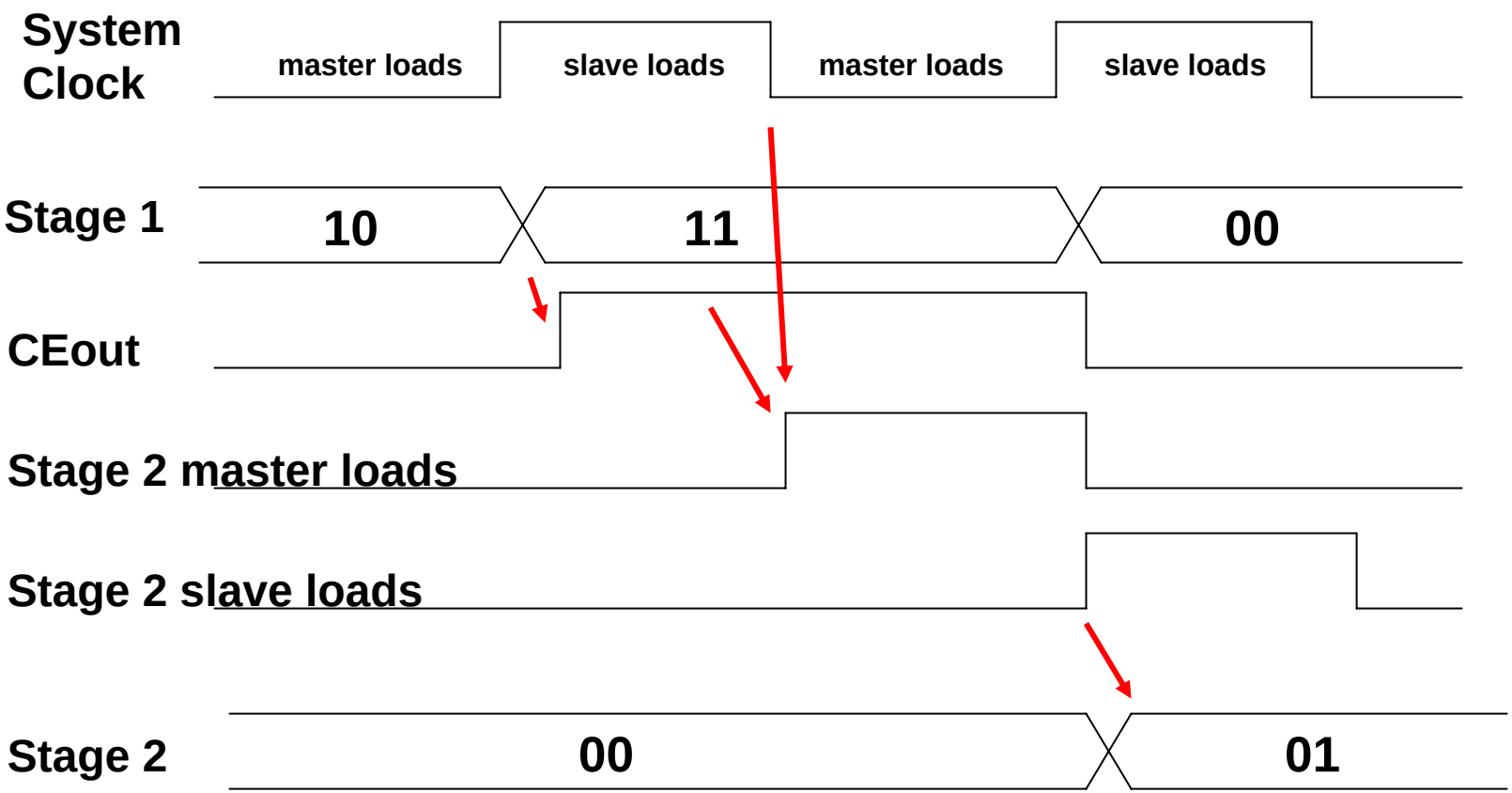
A Ripple Counter is an ASYNCHRONOUS counter since it's transitions are NOT synchronized to the system clock.

In a SYNCHRONOUS counter, all stages transition with the system clock

Cascaded Synchronous Counter

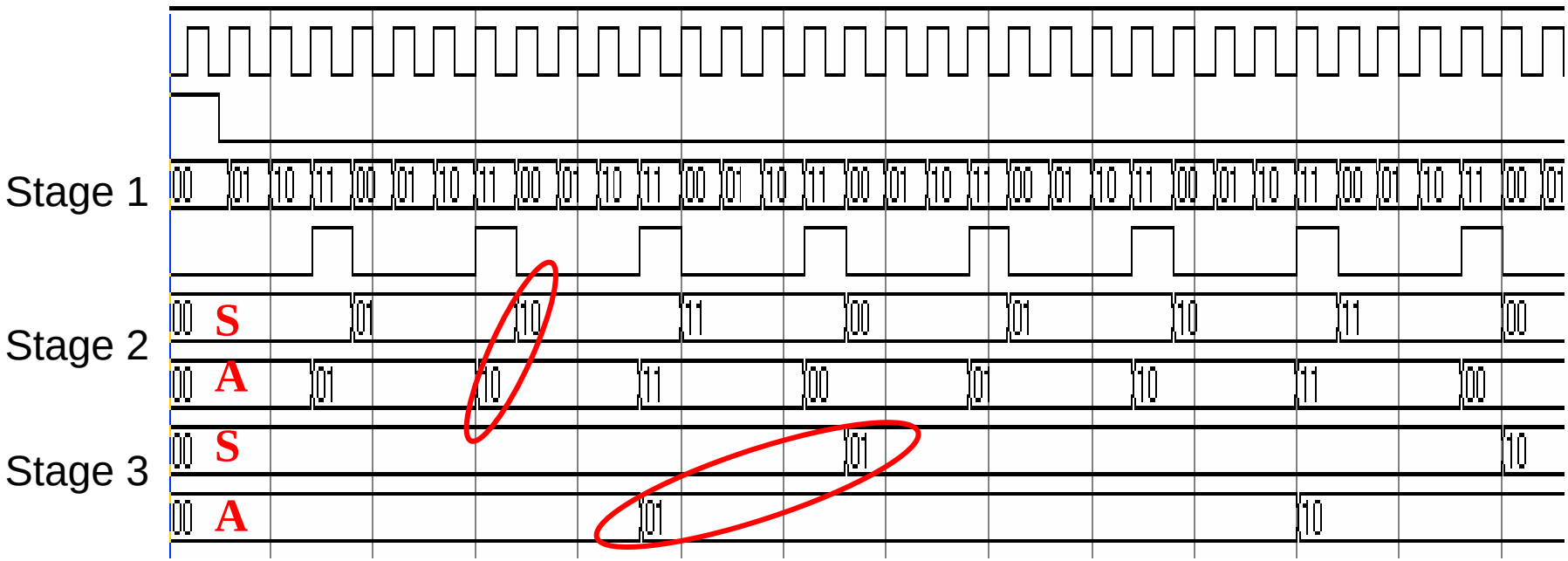


Cascaded Synchronous Counter



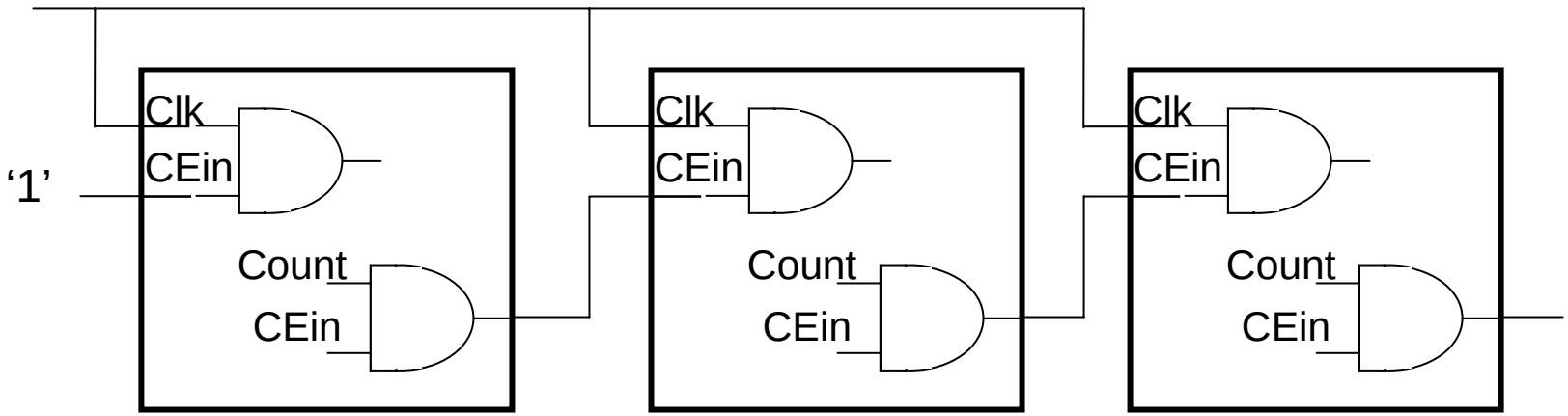
Cascaded Counters

The more stages you add to the counter, the bigger the discrepancy between Synchronous and Asynchronous counters



Cascaded Synchronous Counter

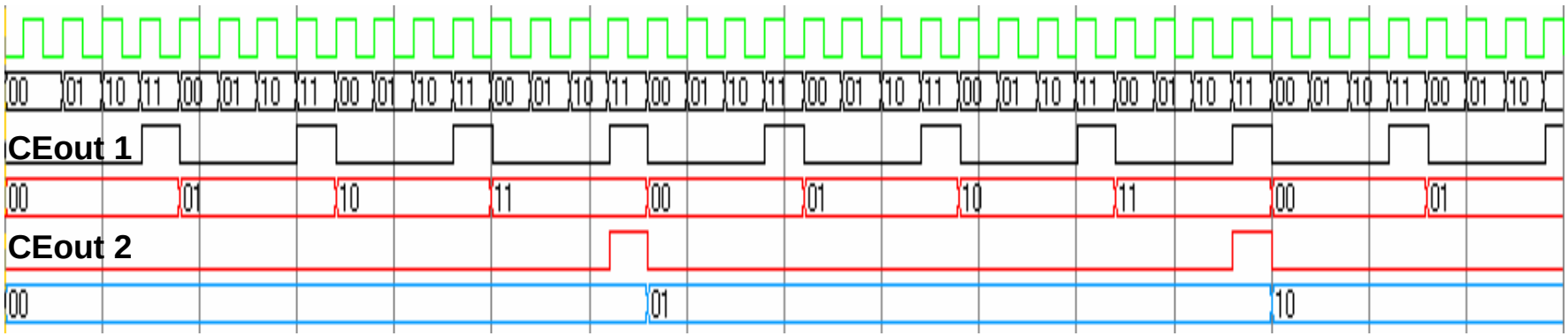
CEin controls when the S.M. can change



CEin also controls when the CEout will be generated

Cascaded Synchronous Counter

Note that the CEout from each stage is synchronized with the system clock ***and*** with each other. The stage transitions are all synchronized with the system clock



Color Code:

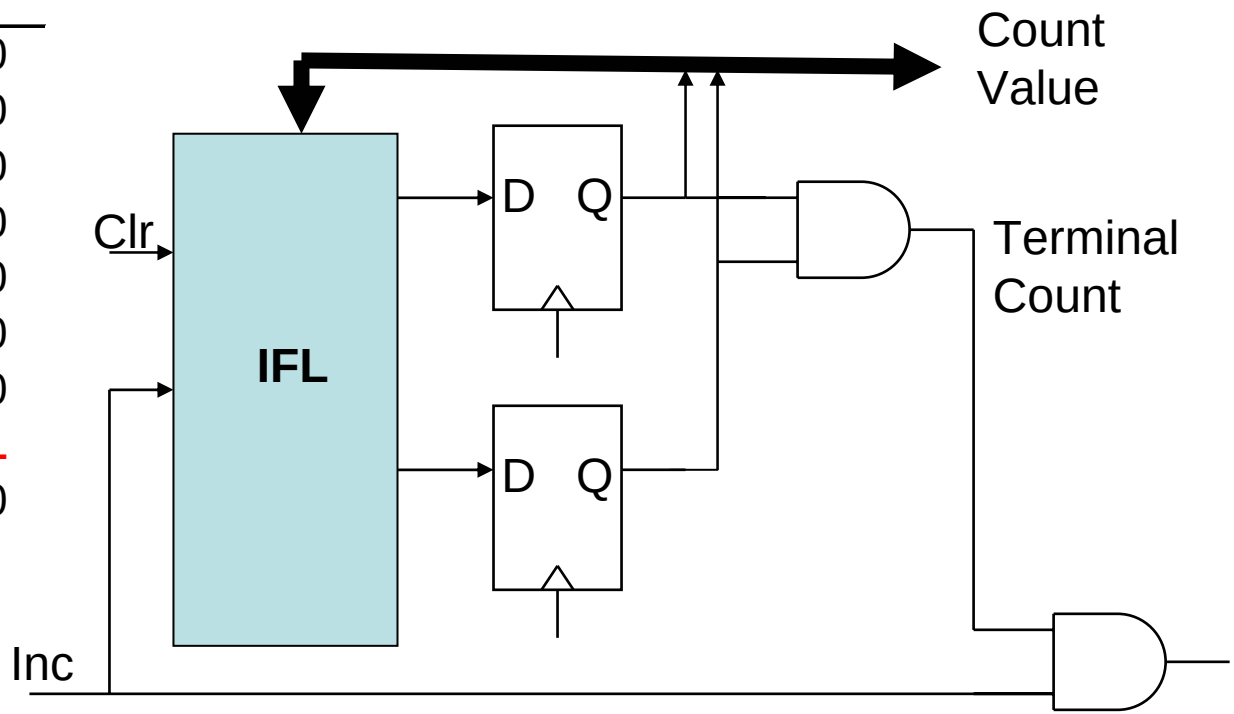
Clock
Stage 1
Stage 2
Stage 3

Cascaded Synchronous Counter

Lesson : Always use SYNCHRONOUS counters and timers.

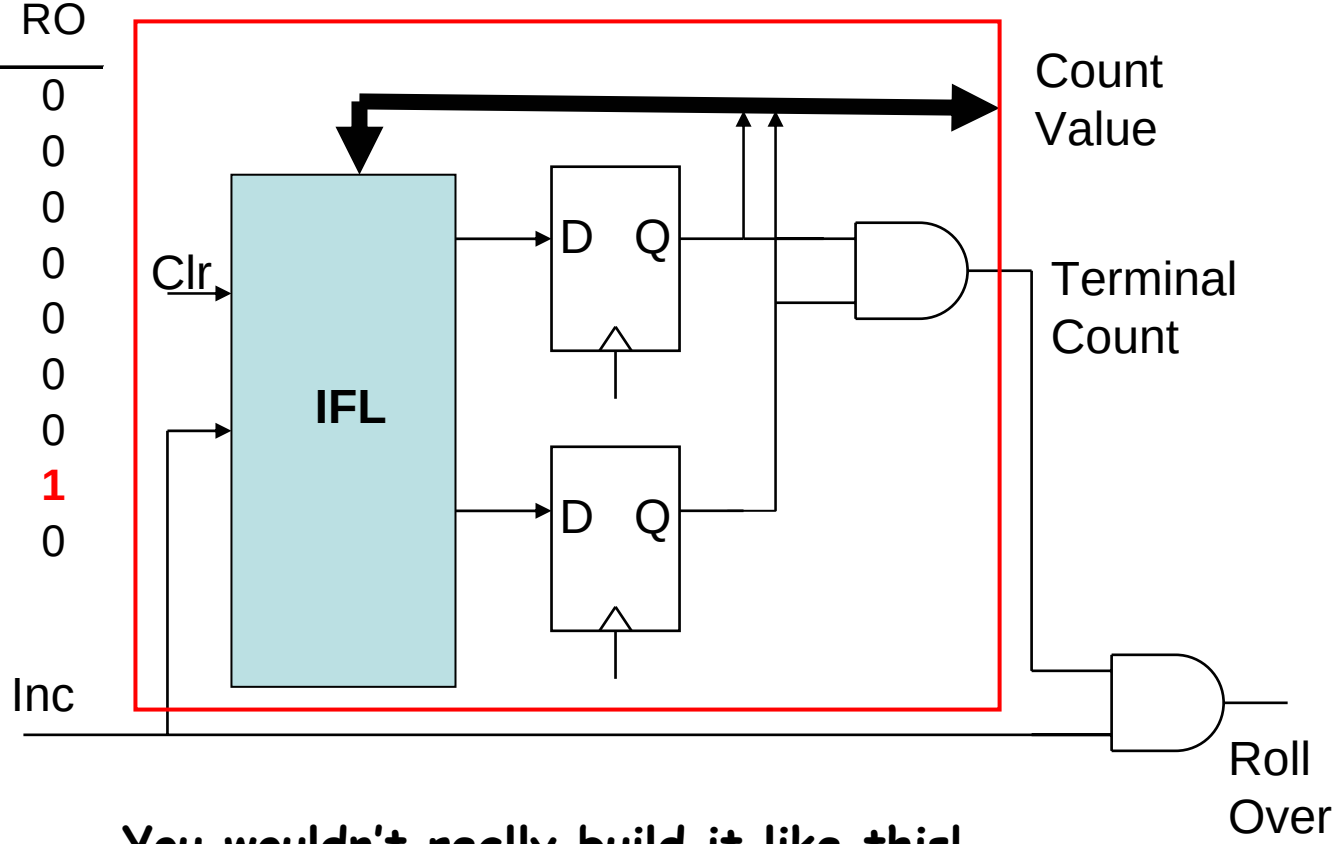
A Mod4 Counter

CLR	INC	Q1	Q0	N1	N0	RO
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	**1**
1	-	-	-	0	0	0



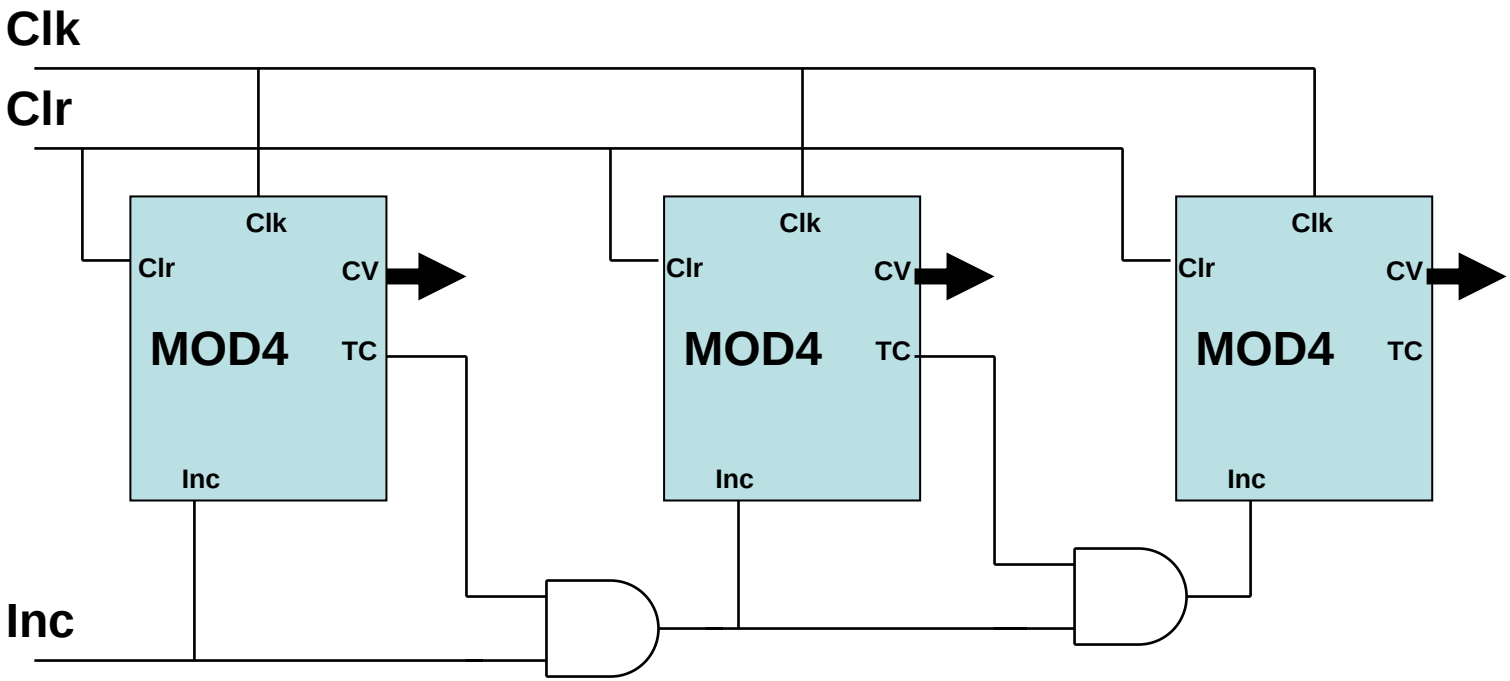
A Mod4 Counter

CLR	INC	Q1	Q0	N1	N0	RO
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	**1**
1	-	-	-	0	0	0

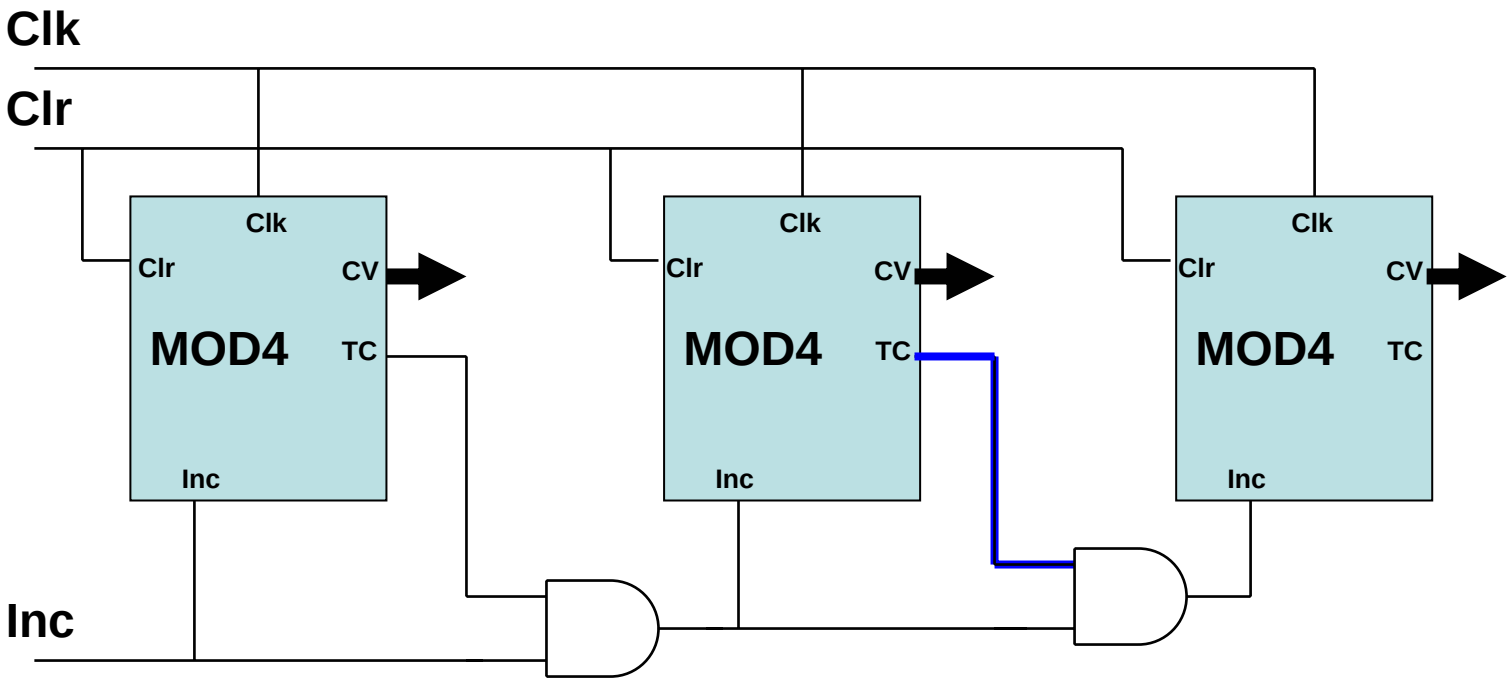


You wouldn't really build it like this!

Cascaded Counters

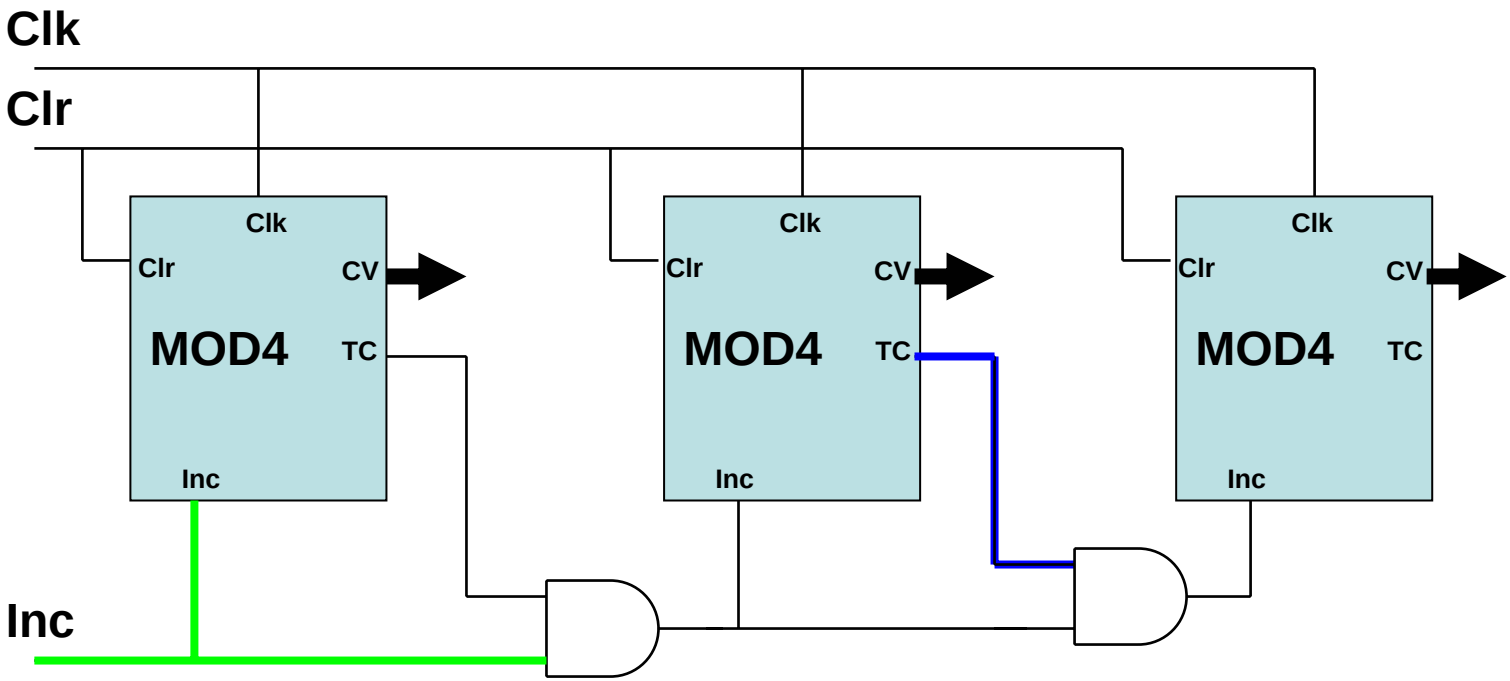


Cascaded Counters

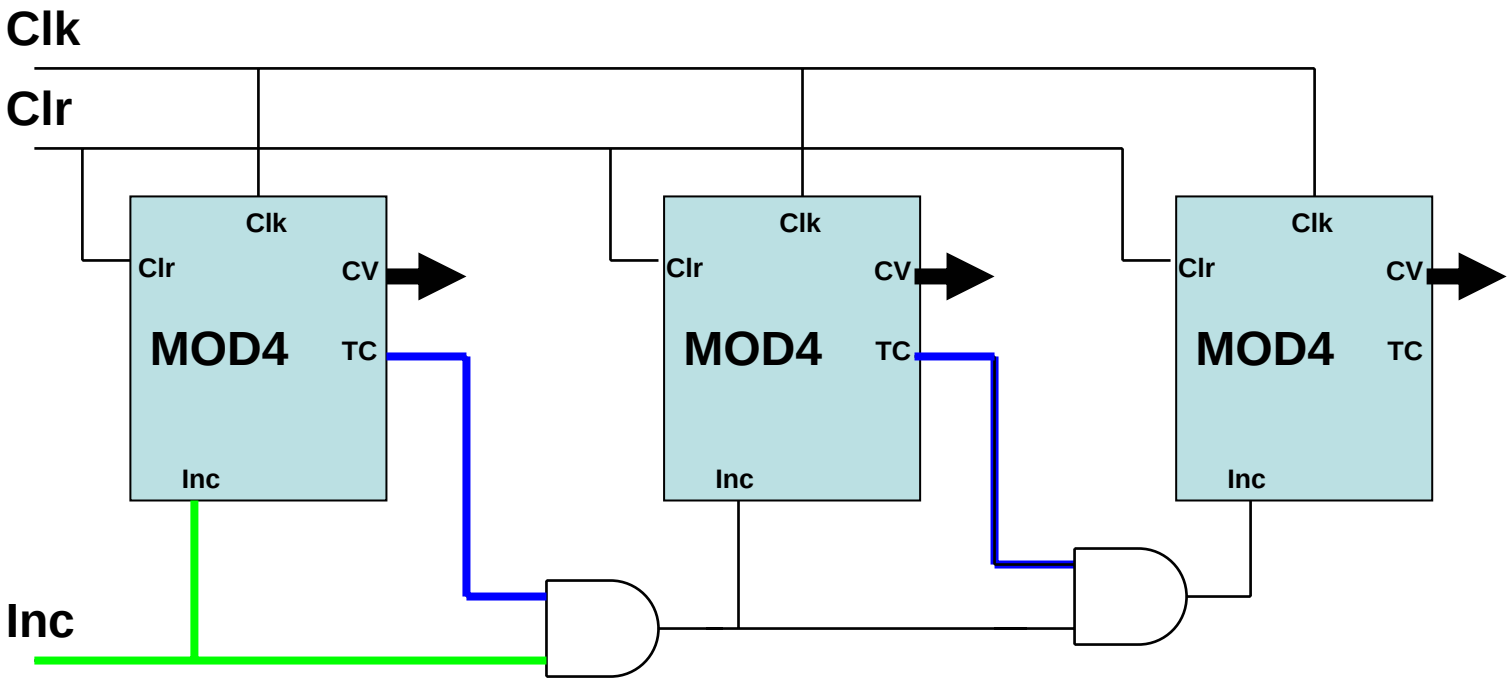


Assume that the second timer is already at the terminal count

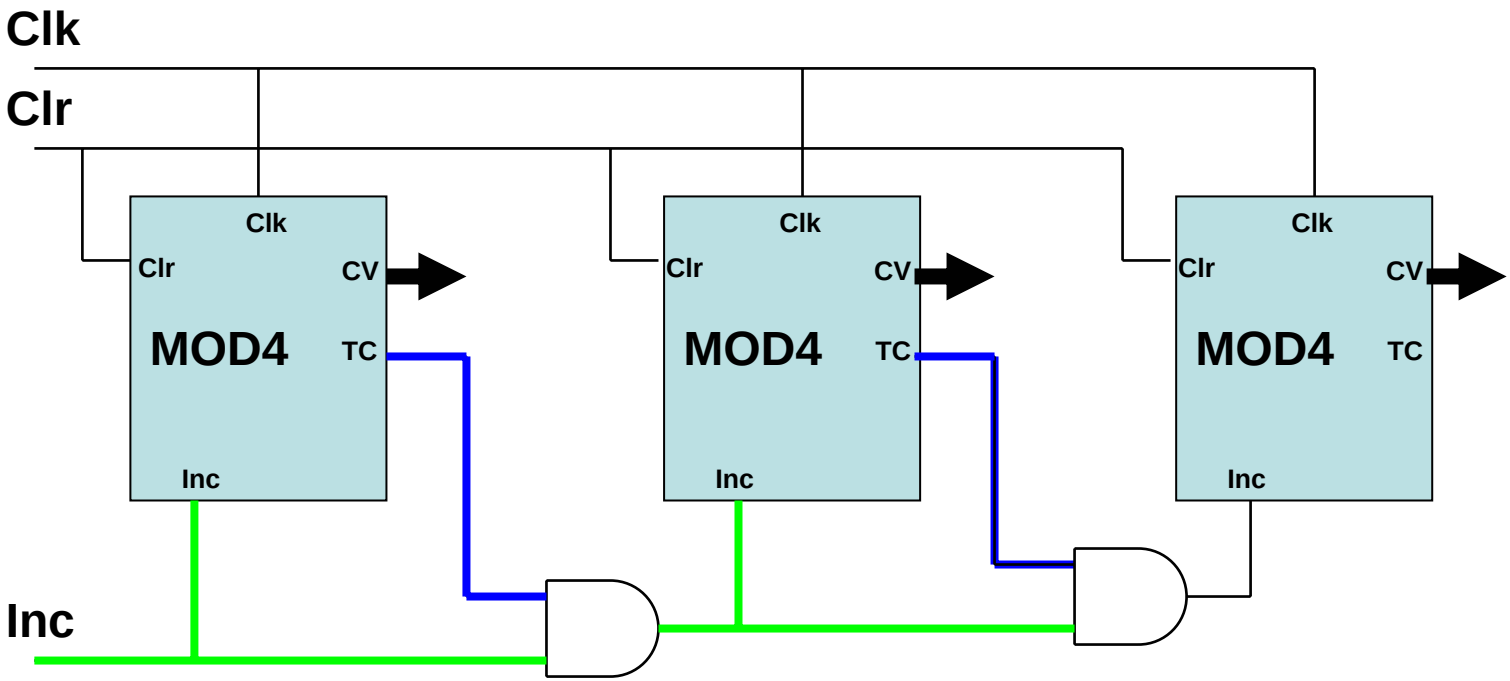
Cascaded Counters



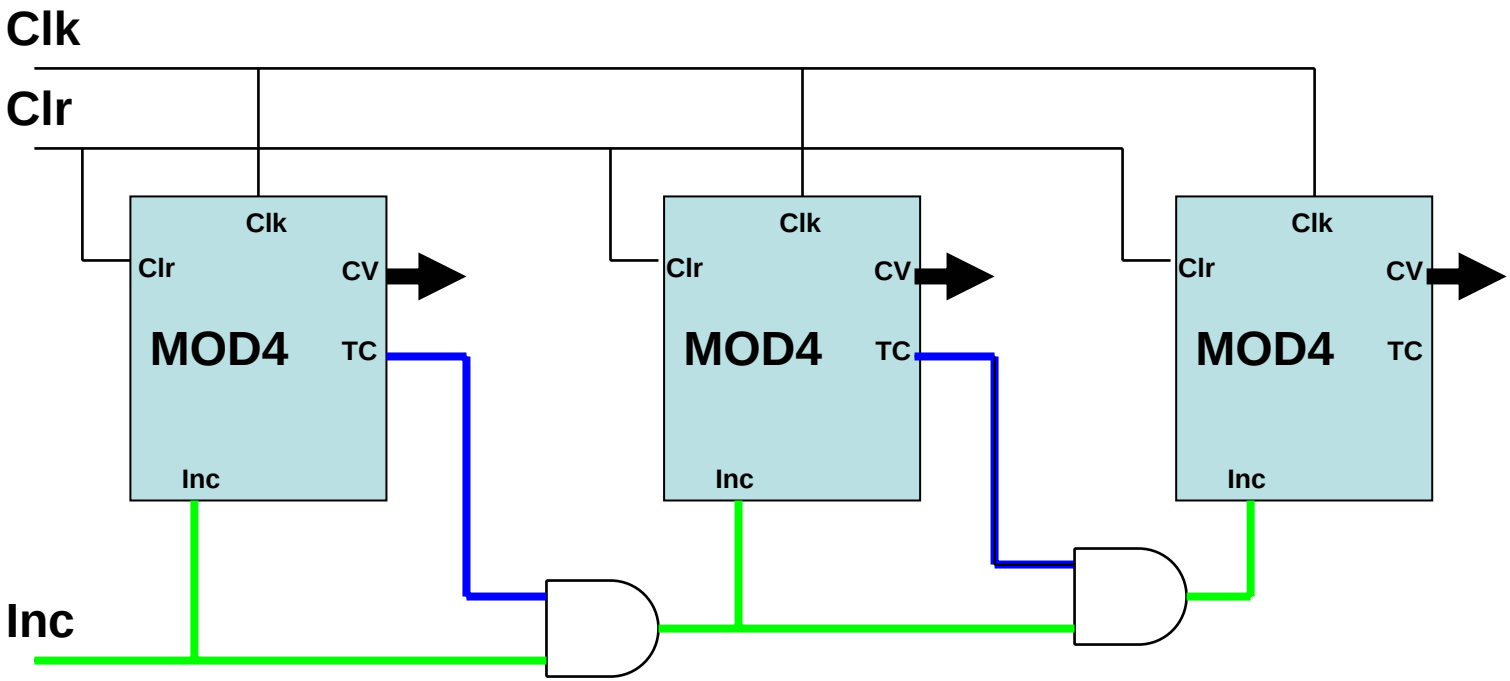
Cascaded Counters



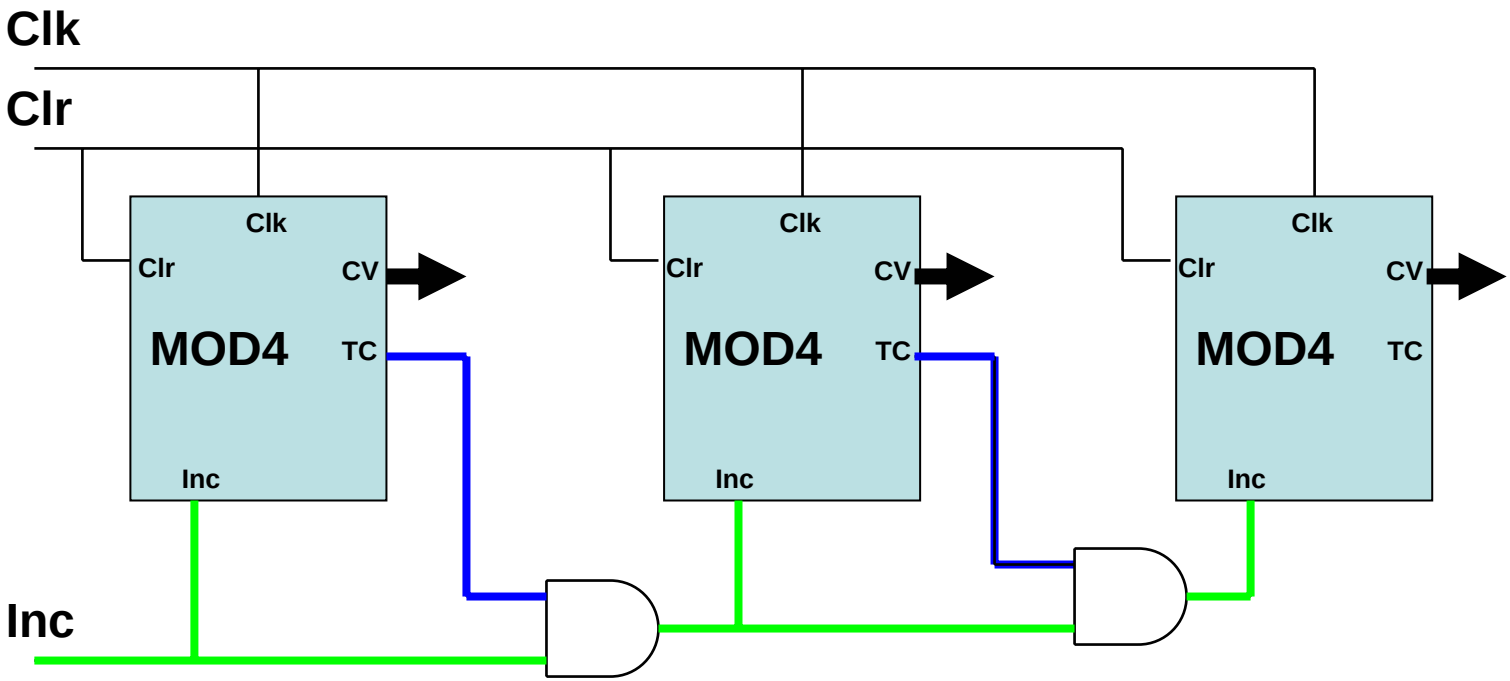
Cascaded Counters



Cascaded Counters



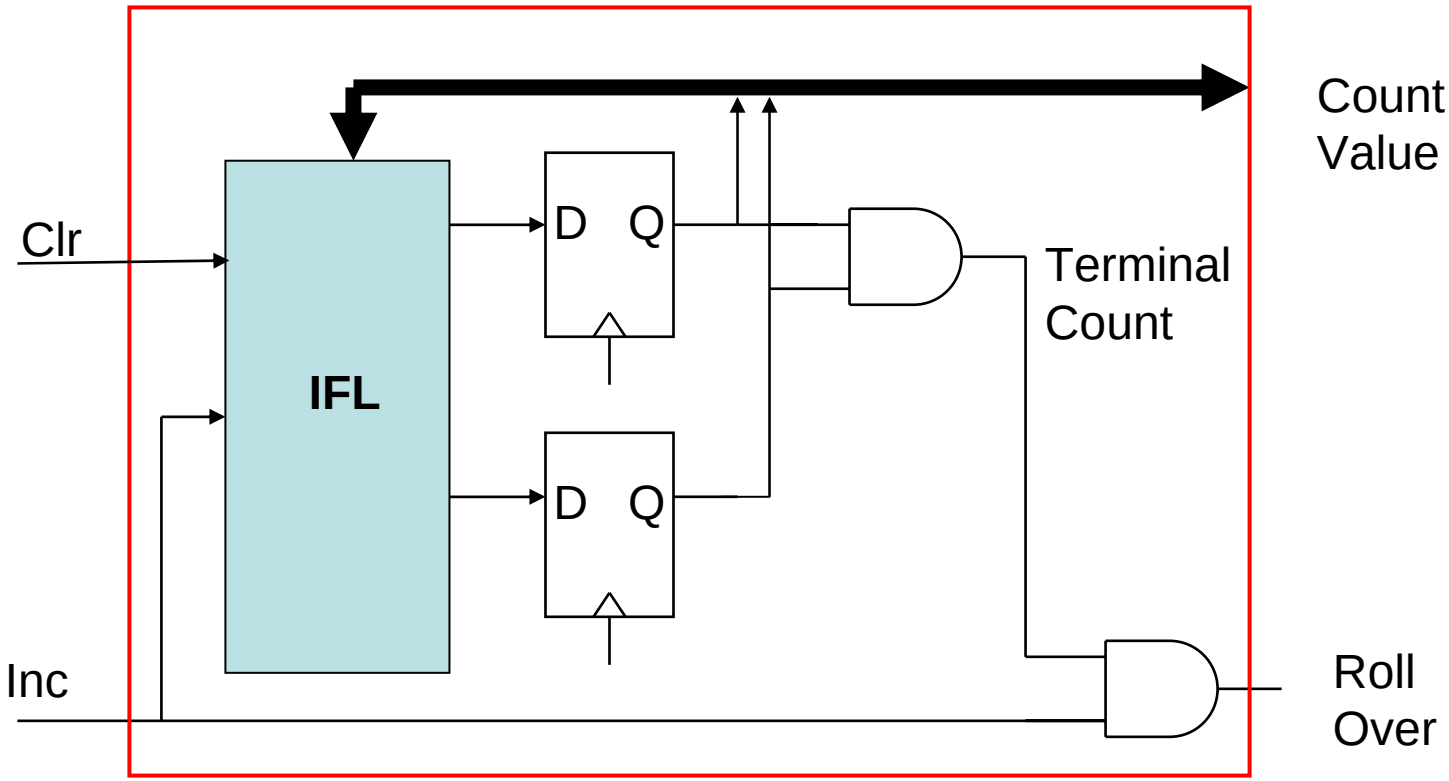
Cascaded Counters



It looks like the Inc signal will ripple from counter to counter.
Why is this any different from the toggle flip-flop example?

A Mod4 Counter

The right way!



A Mod4 Counter