COUNTERS

Counters WİTH INPUTS

Kinds of Counters

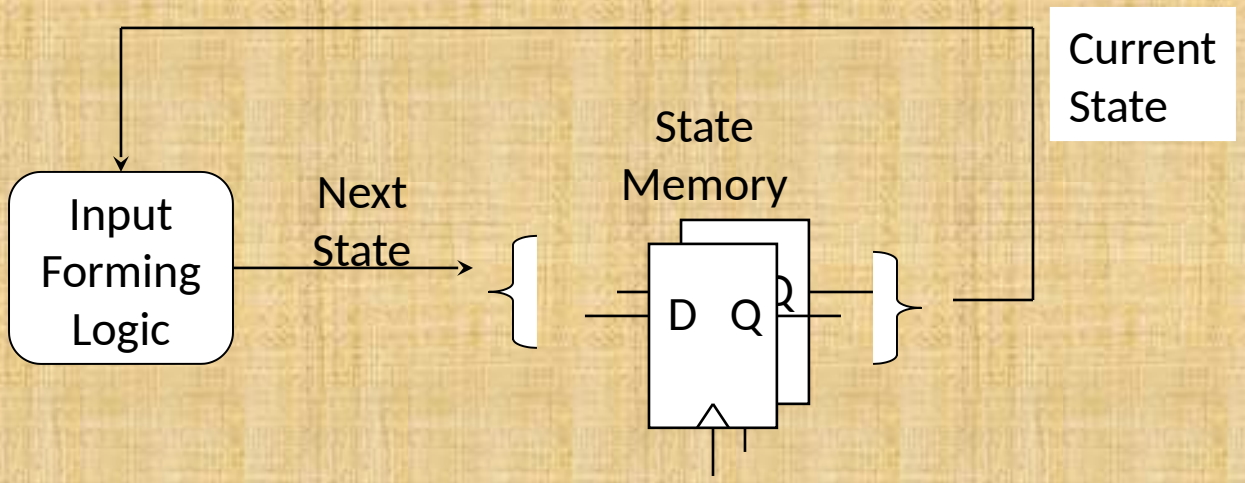
Synchronous Counters

Ripple Counters (Asynchronous counter)

Cascaded Counters (Modulo counter)

Hybrid counters

General Sequential Systems



Counters

- Counters are registers that go through a predefined sequence of states when inputs are applied
- Many counters follow the binary number sequence
  - For example, a 3-bit binary ripple counter goes through the following state transitions when the clock is asserted:

    000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000 …
- Counters are said to overflow when their sequence is complete
- Overflow causes counters to repeat a sequence of values over time

Types of Counters

- Two types of counters exist:
  1. Synchronous Counters
  2. Ripple Counters

  Synchronous counters are triggered by a common clock

  Ripple counters use flip-flop output transitions to serve as the trigger source for other flip-flops

  – For example, a 1 to 0 transition on flip-flop$_i$ triggers a toggling transition on flip-flop$_{i+1}$

  Ripple counters are not triggered by a common clock

Synchronous Counters

- In a synchronous counter, all flip flops are clocked by the same clock signal
  - They all change at the same time
- Synchronous counters can be cascaded to create larger counters that are also globally synchronous

Transition Table for 2-Bit Counter

Current State	Next State
00	01
01	10
10	11
11	00

Current State		Next State	
Q1	Q0	N1	N0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

It is the truth table for the input forming logic…

It describes what the *next state* values are as a function of the *current state* (clock is assumed)

Implementation of 2-Bit Counter

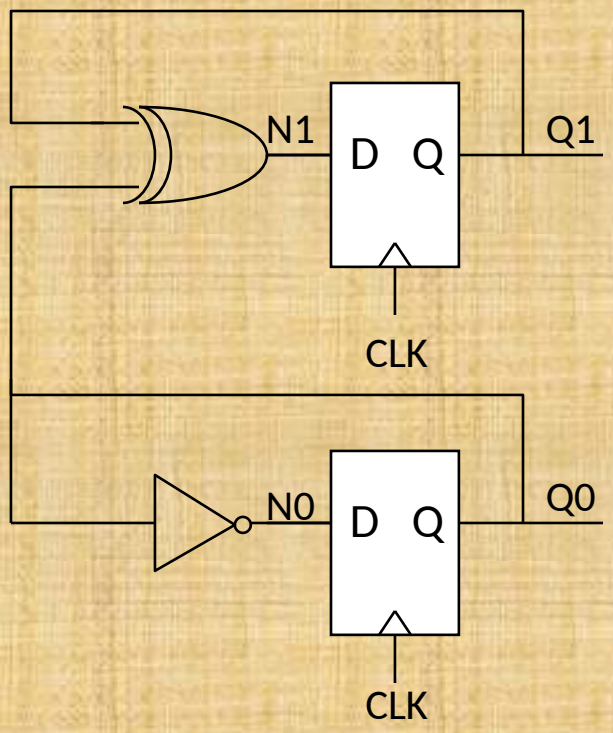
Current State		Next State	
Q1	Q0	N1	N0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Q0 \ Q1	0	1
0	1	1
1	0	0

N0 = Q0'

Q0 \ Q1	0	1
0	0	1
1	1	0

N1 =Q1 ⊕ Q0



Example 2 – A Gray Code Counter

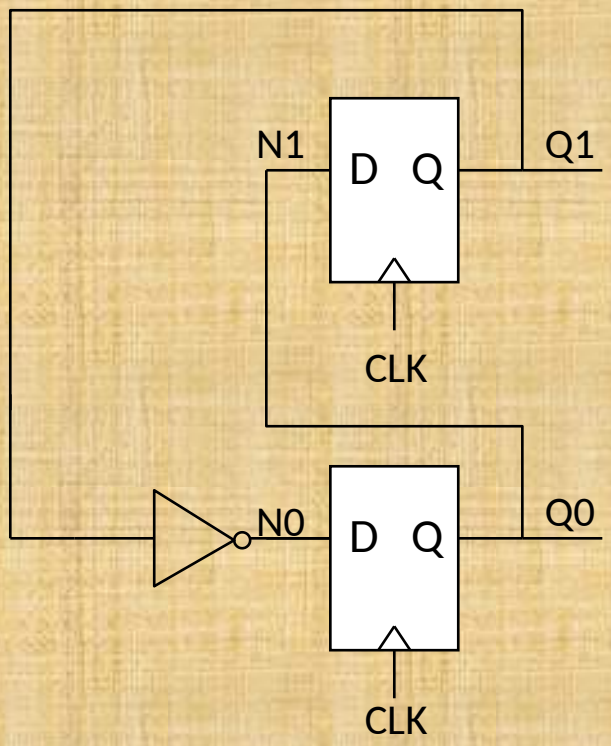
Q1	Q0	N1	N0
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

Q0 \ Q1	0	1
0	0	0
1	1	1

N1= Q0

Q0 \ Q1	0	1
0	1	0
1	1	0

N0 = Q1'



Example 3 – Not All Count Values Used

Desired count sequence = 00 – 01 – 11 - 00 …

Q1	Q0	N1	N0
0	0	0	1
0	1	1	1
1	0	?	?
1	1	0	0

What should next state for 10 be?

Example 3 – Not All Count Values Used

Q1	Q0	N1	N0
0	0	0	1
0	1	1	0
1	0	X	X
1	1	0	0

Q0 \ Q1	0	1
0	0	X
1	1	0

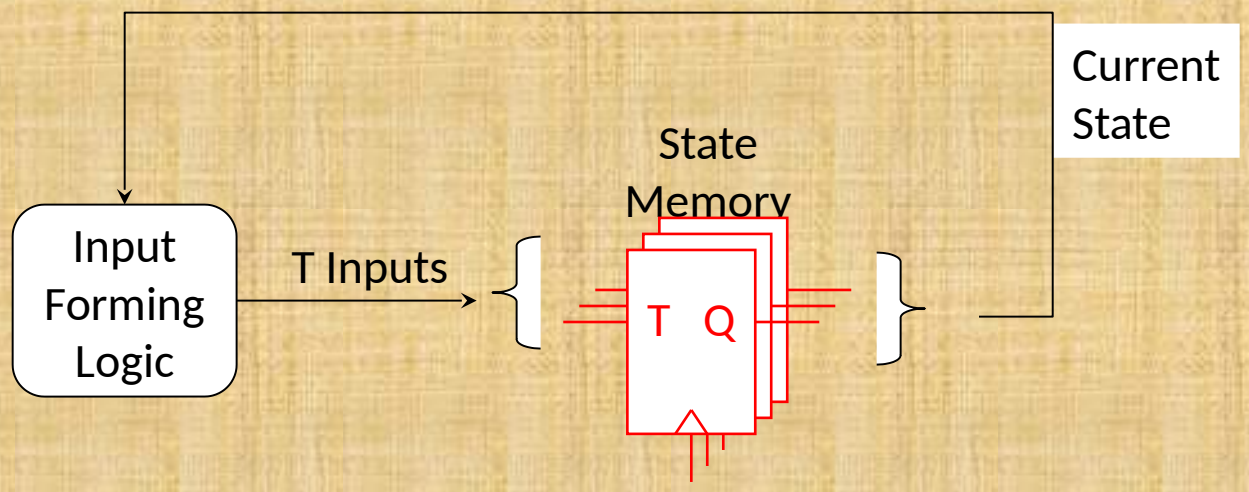
N1 = Q0•Q1'

Q0 \ Q1	0	1
0	1	X
1	0	0

N0 = Q0'

Do the normal K-map minimization with don't cares

Counters With Alternative FF's



Counter Design Procedure

Introduction

The process is a special case of the general sequential circuit design procedure.

no decisions on state assignment or transitions
current state is the output

Example: 3-bit Binary Upcounter

Present state			Next state		
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Decide to implement with Toggle Flipflops

What inputs must be presented to the T FFs to get them to change to the desired state bit?

We need to use the T FF excitation table to translate the present/next state values to FF inputs

TFF Counter Design Using Augmented Transition Table

Current State			Next State			TFF Inputs		
Q2	Q1	Q0	N2	N1	N0	T2	T1	T0
0	0	0	0	0	1			
0	0	1	0	1	0			
0	1	0	0	1	1			
0	1	1	1	0	0			
1	0	0	1	0	1			
1	0	1	1	1	0			
1	1	0	1	1	1			
1	1	1	0	0	0			

Next state values

Inputs to apply to achieve desired next state

TFF Counter Design Using Augmented Transition Table

Current State			Next State			TFF Inputs		
Q2	Q1	Q0	N2	N1	N0	T2	T1	T0
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	0	1
1	1	1	0	0	0	1	1	1

Next state values

Inputs to apply to achieve desired next state

T2

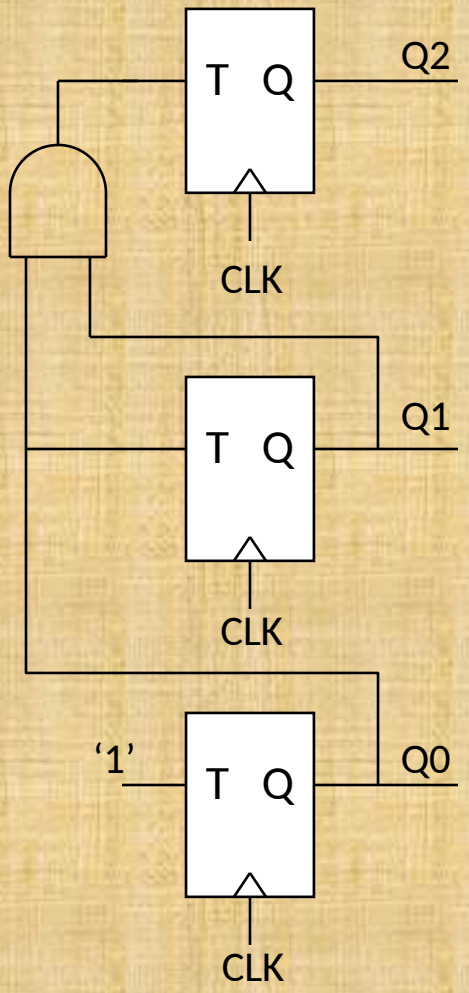
Q_1Q_0 \ Q_2	0	1
00		
01		
11	1	1
10		

T2 = Q1•Q0

T1 = Q0

T0 = '1'

TFF Counter Design



4-Bit Synchronous Binary Counter

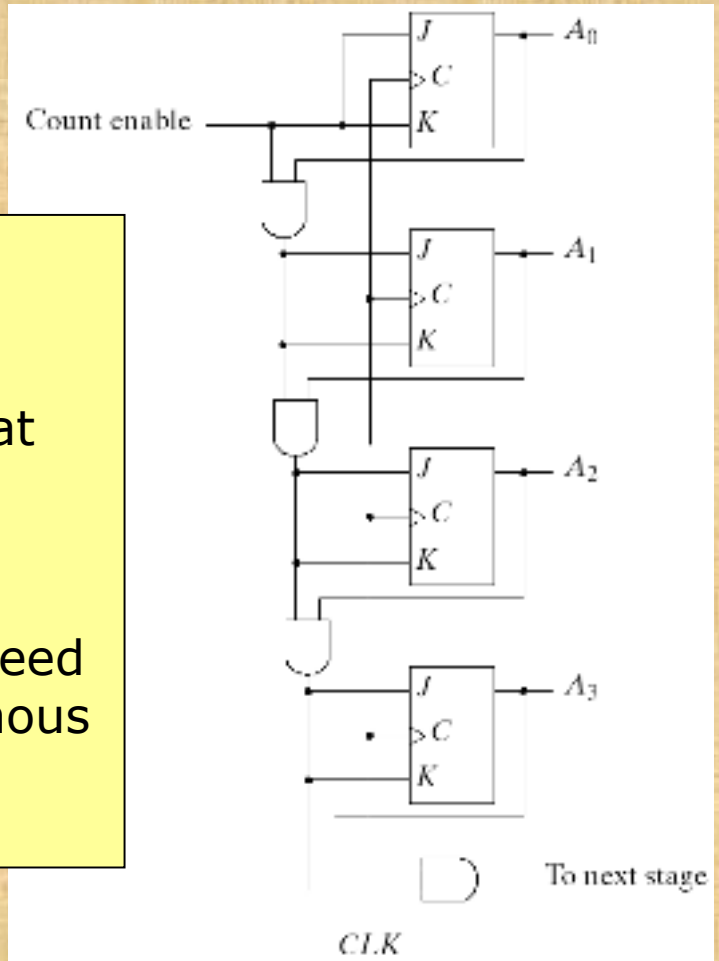


Fig. 6-12 4-Bit Synchronous Binary Counter

NOTE:

This is a well-designed circuit. It can operate at high clock frequencies. The maximum clock frequency can be easily calculated. For high-speed digital design, synchronous binary counters are preferred.

JKFF Gray Code Counter Design

Q2	Q1	Q0	N2	N1	N0	J2	K2	J1	K1	J0	K0
0	0	0	0	0	1						
0	0	1	0	1	1						
0	1	0	1	1	0						
0	1	1	0	1	0						
1	0	0	0	0	0						
1	0	1	1	0	0						
1	1	0	1	1	1						
1	1	1	1	0	1						

Next state values

Inputs to apply to achieve desired next state

JKFF Gray Code Counter Design

Q2	Q1	Q0	N2	N1	N0	J2	K2	J1	K1	J0	K0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	1	1	0	1	X	X	0	0	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	1	X	0	X	1	X	0

Next state values

Inputs to apply to achieve desired next state

J2 = Q1●Q0'

K2 = Q1'●Q0'

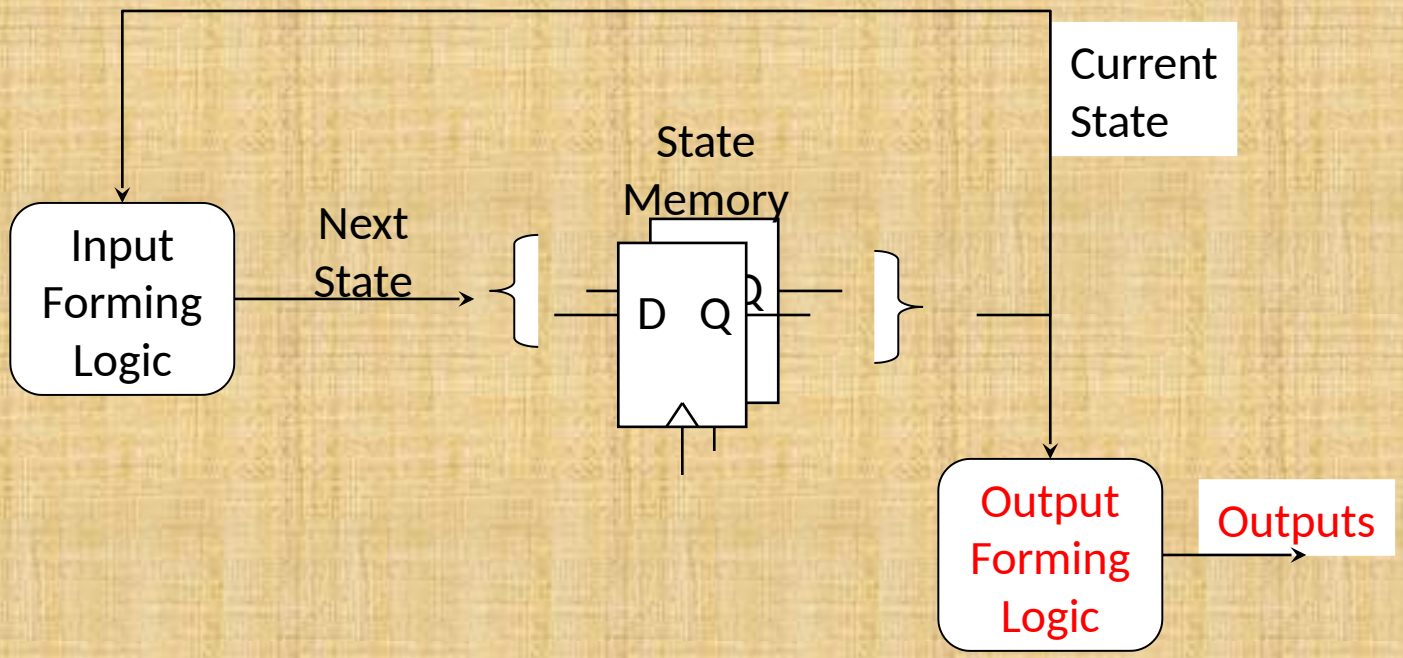
J1 = Q2'●Q0

K1 = Q2●Q0

J0 = Q2●Q1 + Q2'●Q1' = K0'

K0 = Q2'●Q1 + Q2●Q1' = Q2 ⊕ Q1

Counters With Outputs



Outputs = f(CurrentState)

Counters With Outputs

Z=1 when count={0,3,6}

Q2	Q1	Q0	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Z is called a *Moore* or *static* output.
It is a function only of the current state.

Combined Transition Table

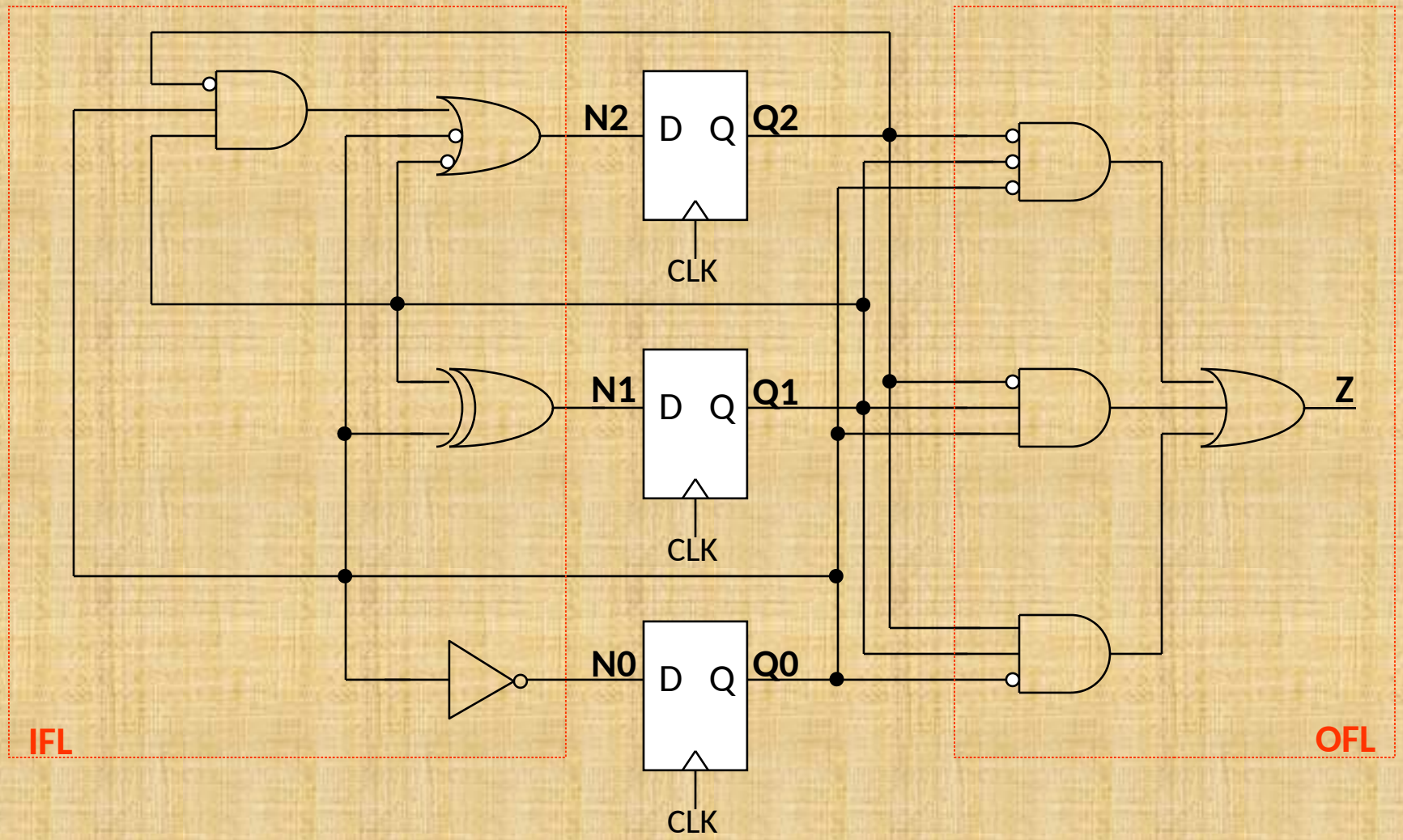
Q2	Q1	Q0	N2	N1	N0	Z
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	1	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0

Current state Next state Output

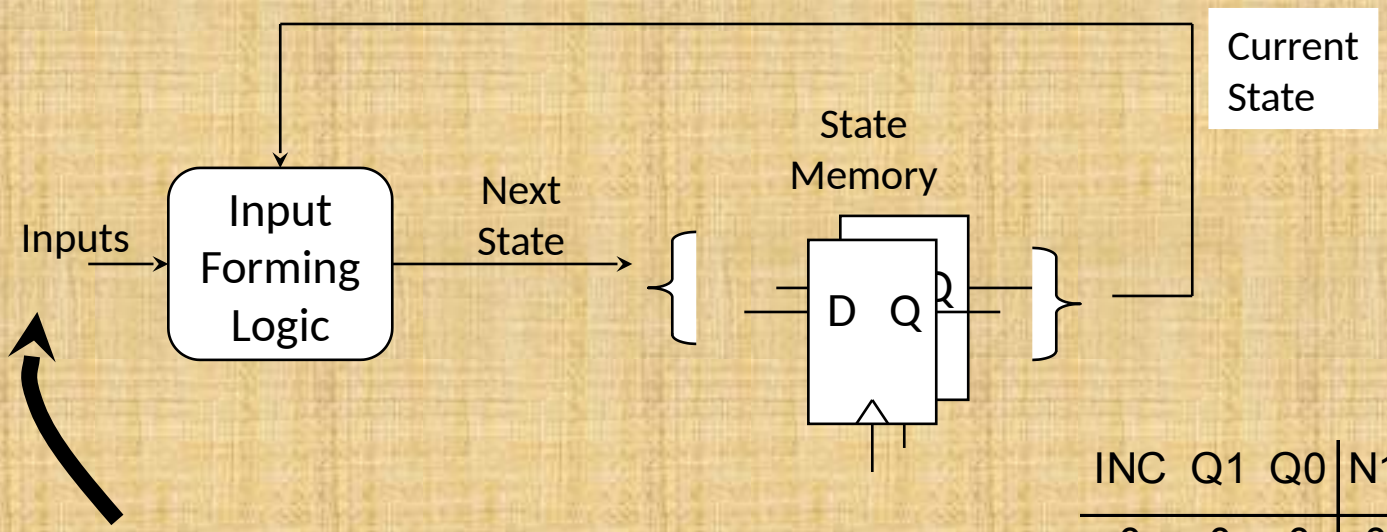
Z = Q2'•Q1'•Q0' + Q2'•Q1•Q0 + Q2•Q1•Q0'

(implement OFL with gates)

Counter With A Moore Output



An Incrementable Counter



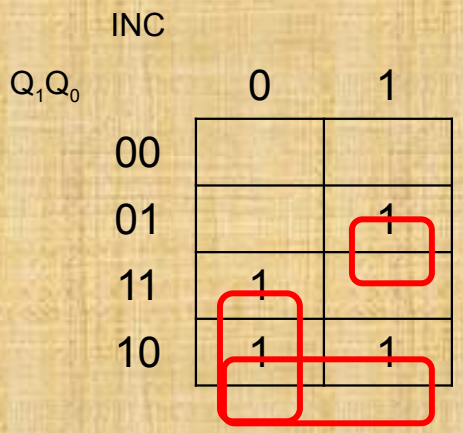
Note that the next state is a function of the input 'INC' as well as the current state.

INC	Q1	Q0	N1	N0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

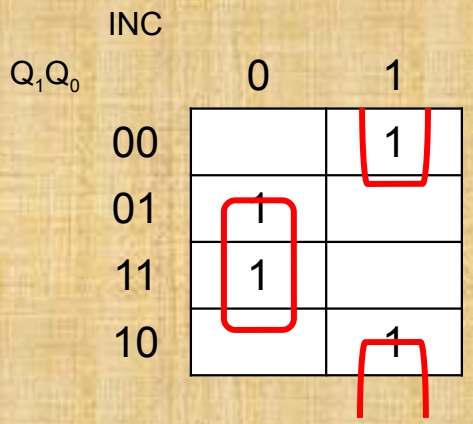
Incrementable Counter Derivation

INC	Q1	Q0	N1	N0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Doing the KMaps for this results in:



N1

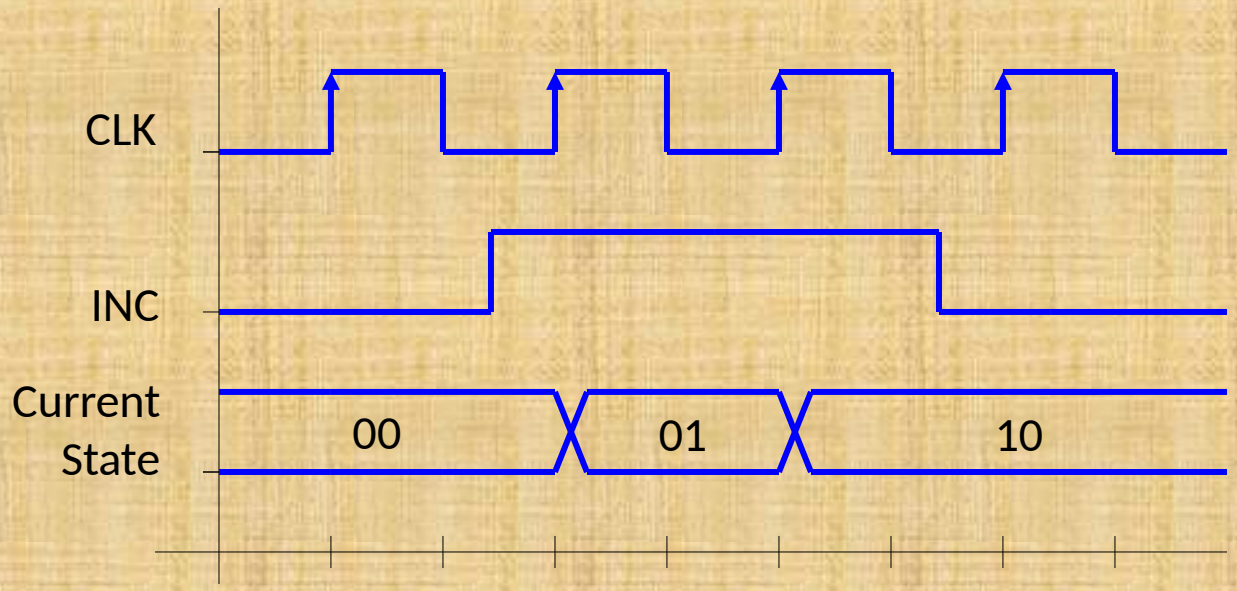


N0

N1 = INC•Q1'•Q0 + INC'•Q1 + Q1•Q0'

N0 = INC'•Q0 + INC•Q0' = INC⊕ Q0

Incrementable Counter Behavior



The counter increments on the clock edge only when INC is asserted

Counters With More Inputs

	CLR	INC	Q1	Q0	N1	N0
CLR = INC = 0 No state transition	0	0	0	0		
	0	0	0	1		
	0	0	1	0		
	0	0	1	1		
CLR = 0 INC = 1 Counter Increments	0	1	0	0		
	0	1	0	1		
	0	1	1	0		
	0	1	1	1		
CLR = 1 INC = 0 Counter resets to '00'	1	0	0	0		
	1	0	0	1		
	1	0	1	0		
	1	0	1	1		
CLR = 1 INC = 1 What should it do?	1	1	0	0		
	1	1	0	1		
	1	1	1	0		
	1	1	1	1		

Counters With More Inputs

	CLR	INC	Q1	Q0	N1	N0
CLR = INC = 0 No state transition	0	0	0	0	0	0
	0	0	0	1	0	1
	0	0	1	0	1	0
	0	0	1	1	1	1
CLR = 0 INC = 1 Counter Increments	0	1	0	0	0	1
	0	1	0	1	1	0
	0	1	1	0	1	1
	0	1	1	1	0	0
CLR = 1 INC = 0 Counter resets to '00'	1	0	0	0	0	0
	1	0	0	1	0	0
	1	0	1	0	0	0
	1	0	1	1	0	0
CLR = 1 INC = 1 What should it do?	1	1	0	0	?	?
	1	1	0	1	?	?
	1	1	1	0	?	?
	1	1	1	1	?	?

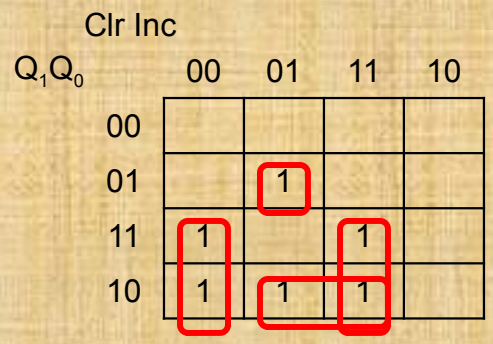
Precedence of INC vs. CLR?

1. Could do nothing
2. Could give INC precedence
3. Could give CLR precedence
4. Assume INC=CLR=1 will never occur

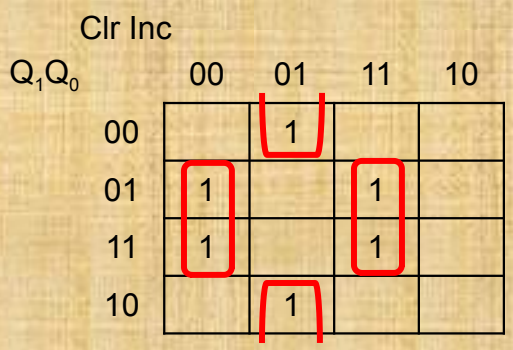
You decide when you draw the transition table!

Case 1 – Do Nothing

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	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	1



N1



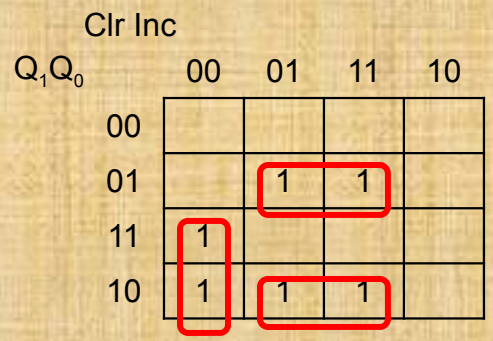
N0

N1 = CLR'•INC•Q1'•Q0 +
CLR'•INC'•Q1 +
CLR•INC•Q1 +
INC•Q1•Q0'

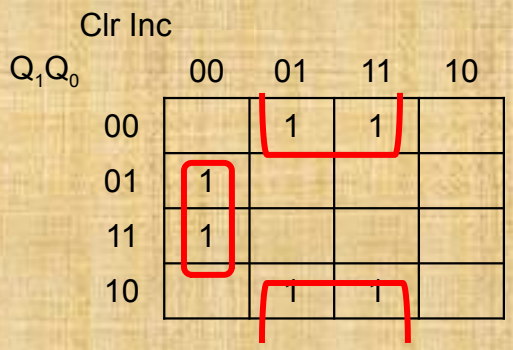
N0 = CLR'•INC'•Q0 +
CLR•INC•Q0 +
CLR'•INC•Q0'

Case 2 – Give INC Precedence

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	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	1
1	1	1	1	0	0



N1



N0

N1 = INC•Q1'•Q0 + CLR'•INC'•Q1 + INC•Q1•Q0'

N0 = CLR'•INC'•Q0 + INC•Q0'

Case 3 – Give CLR Precedence

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	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Clr Inc

Q_1Q_0	00	01	11	10
00				
01		1		
11	1			
10	1	1		

N1

Clr Inc

Q_1Q_0	00	01	11	10
00		1		
01	1			
11	1			
10		1		

N0

N1 = CLR'•INC•Q1'•Q0 +
CLR'•INC'•Q1 +
CLR'•Q1•Q0'

N0 = CLR'•INC'•Q0 +
CLR'•INC•Q0'

Case 4 – Assume INC=CLR=1 Will Never Occur

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	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X

Clr Inc

Q_1Q_0	00	01	11	10
00			X	
01		1	X	
11	1		X	
10	1	1	X	

N1

Clr Inc

Q_1Q_0	00	01	11	10
00		1	X	
01	1		X	
11	1		X	
10		1	X	

N0

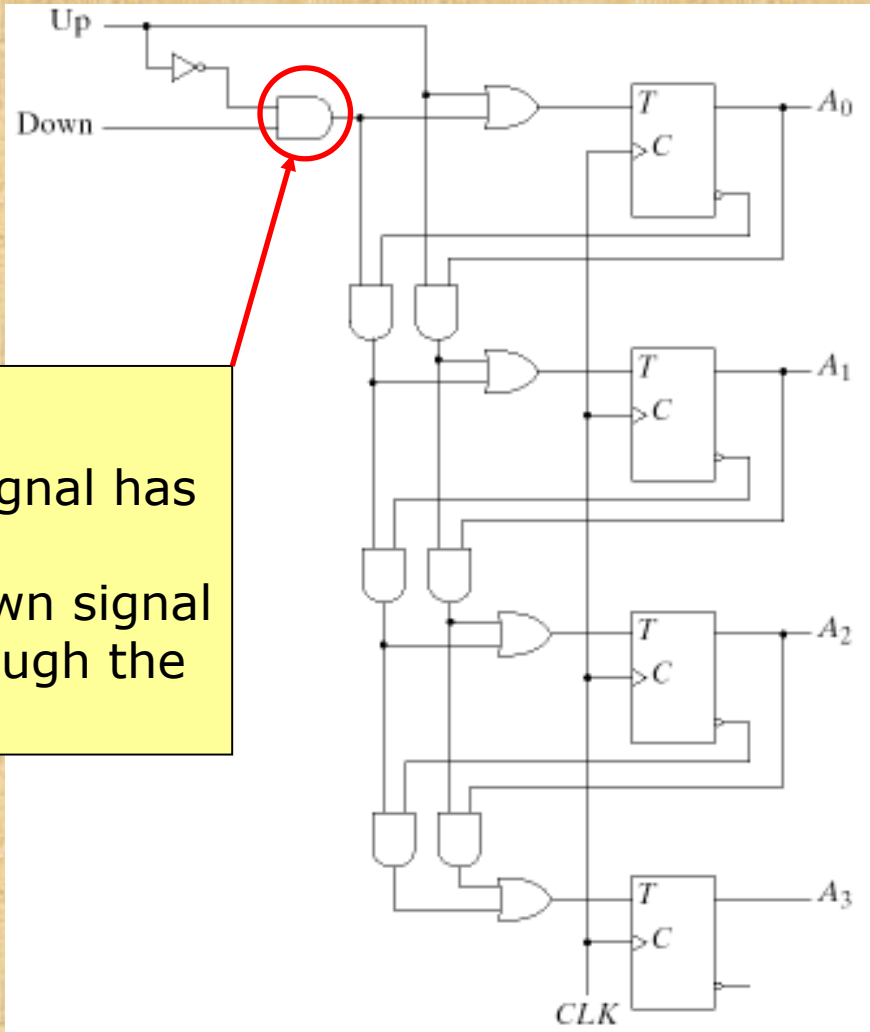
N1 = CLR'•INC'•Q1 + INC•Q1'•Q0 + CLR'•Q1•Q0'

N0 = CLR'•INC'•Q0 + INC•Q0'

What happens in the real circuit when INC=CLR=1?

It depends on the final equations…

4-Bit Up-Down Binary Counter



NOTE:

The Up control signal has priority. If Up is asserted, the Down signal will not pass through the AND gate.

4-Bit Binary Counter with Parallel Load

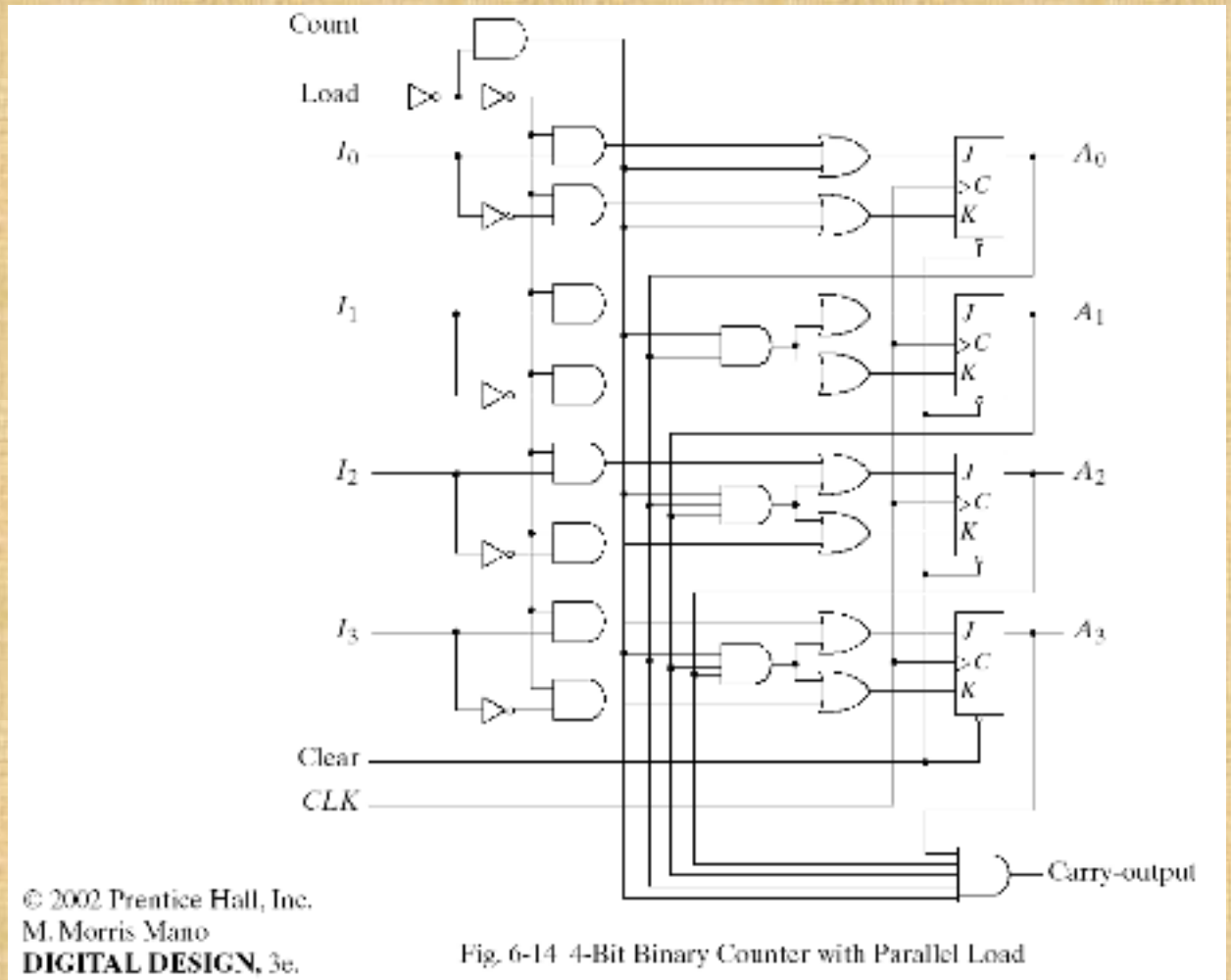


Fig. 6-14 4-Bit Binary Counter with Parallel Load

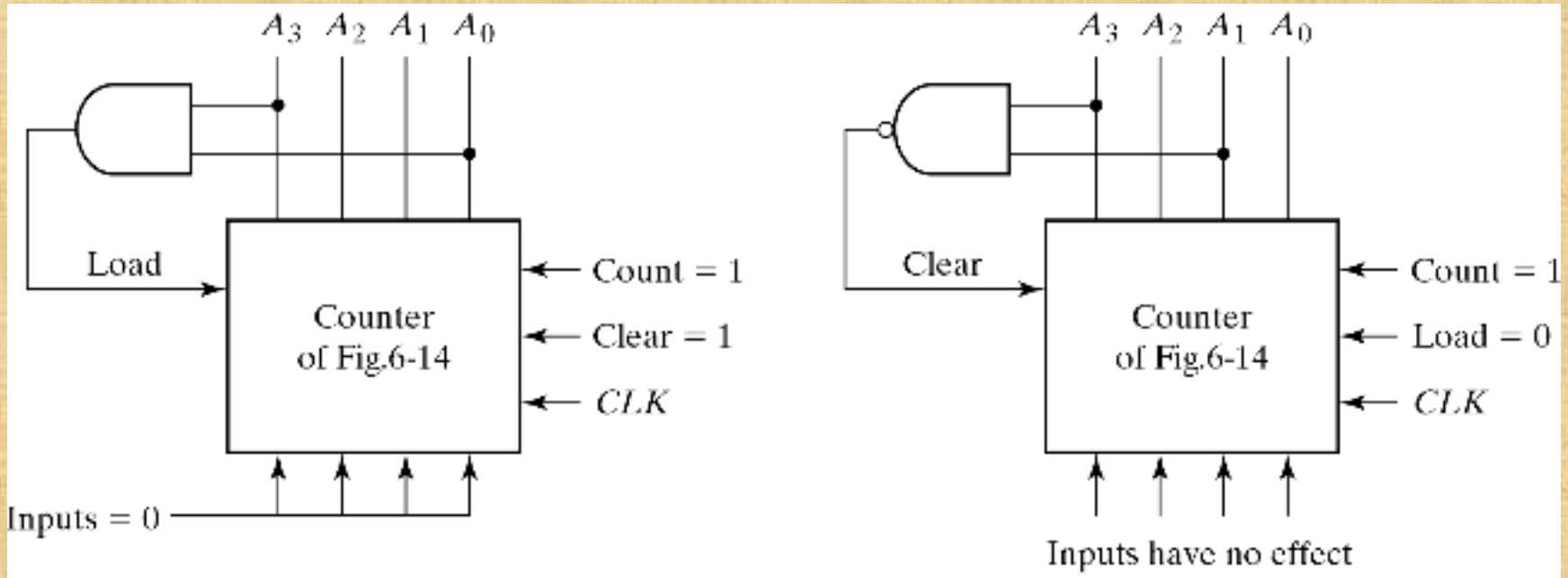
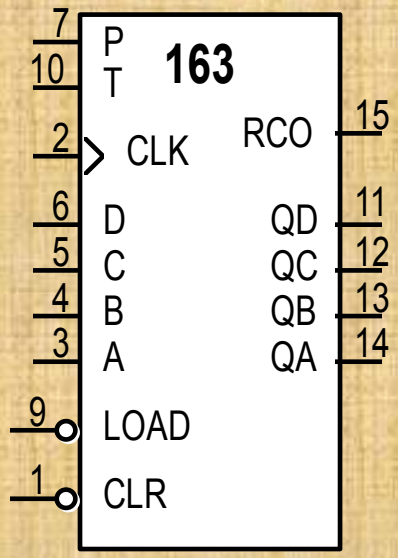


Fig. 6-15 Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

A common 4-bit counter



74163 Synchronous
4-Bit Upcounter

Synchronous Load and Clear Inputs

Positive Edge Triggered FFs

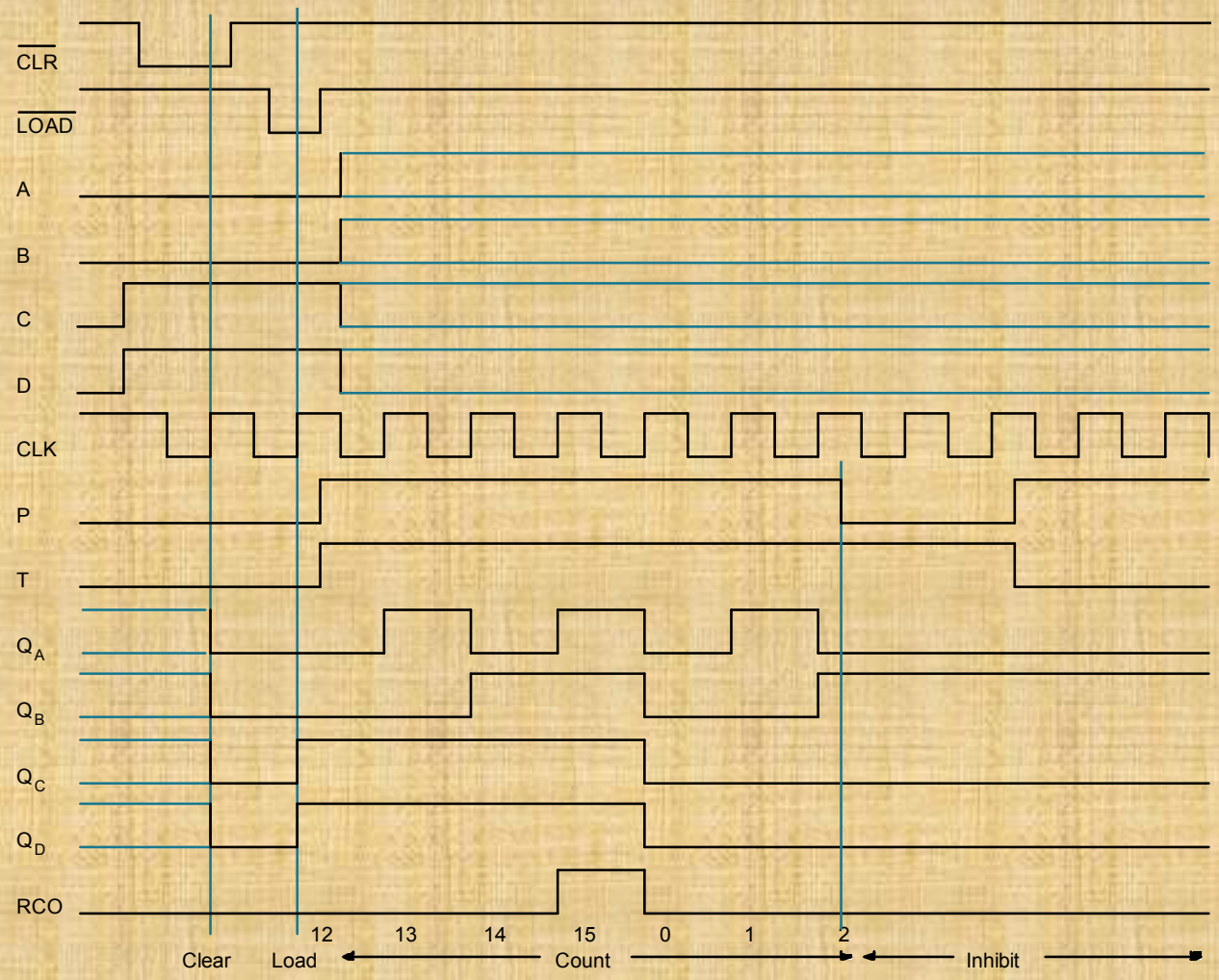
Parallel Load Data from D, C, B, A

P, T Enable Inputs: both must be asserted to enable counting

RCO: asserted when counter enters its highest state 1111, used for cascading counters *"Ripple Carry Output"*

74161: similar in function, asynchronous load and reset

74163 Detailed Timing Diagram



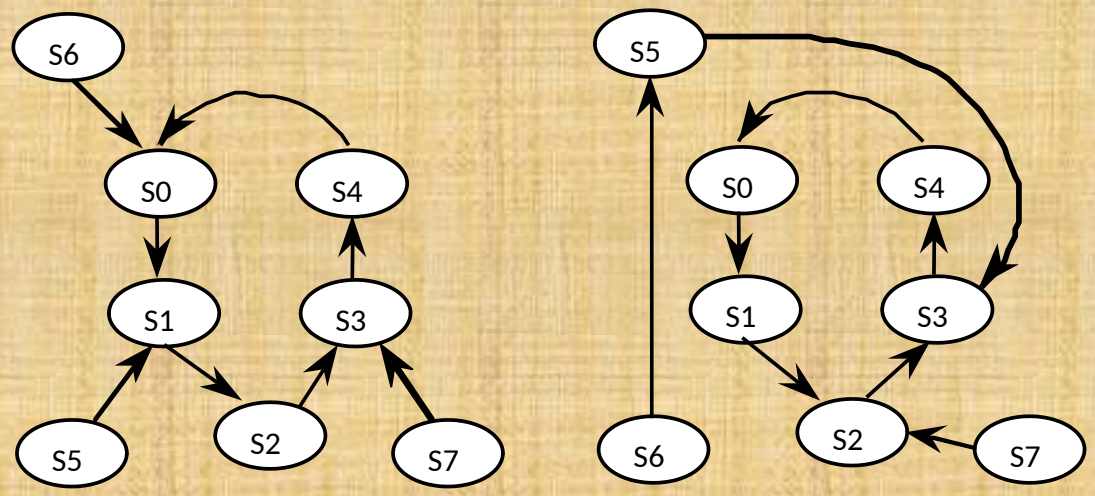
Self-Starting Counters

Start-Up States

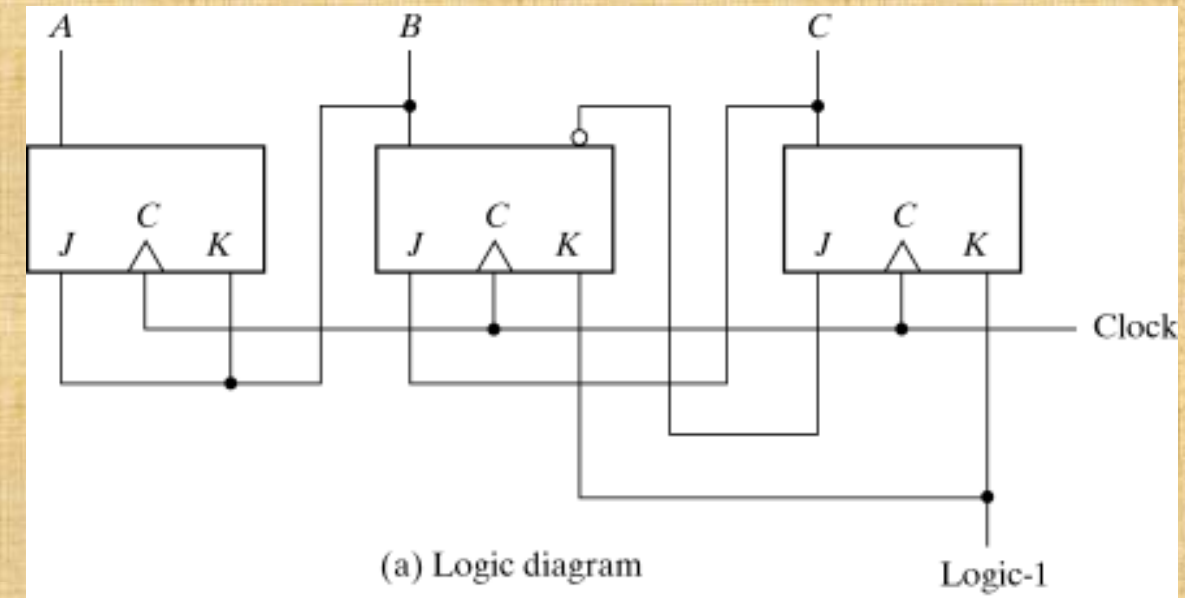
At power-up, counter may be in any possible state
Designer must guarantee that it (eventually) enters a valid state
Especially a problem for counters that validly use a subset of states

Self-Starting Solution:

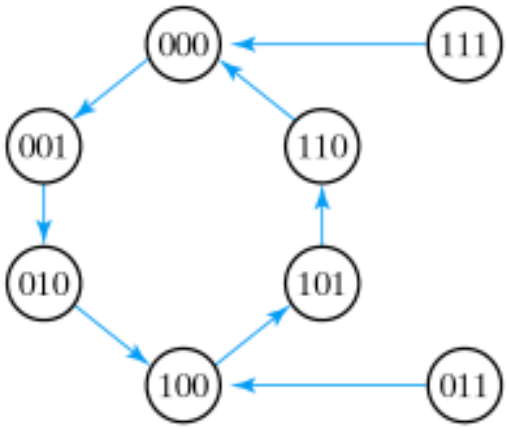
Design counter so that even the invalid states eventually transition to valid state



Two Self-Starting State Transition Diagrams

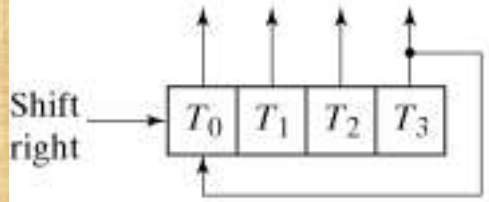


(a) Logic diagram

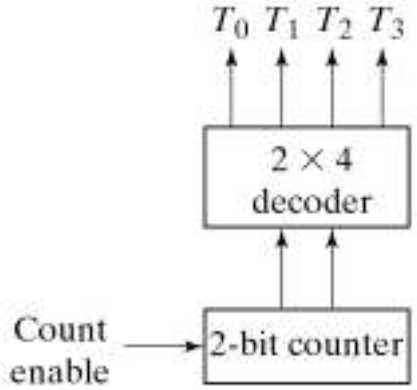


(b) State diagram

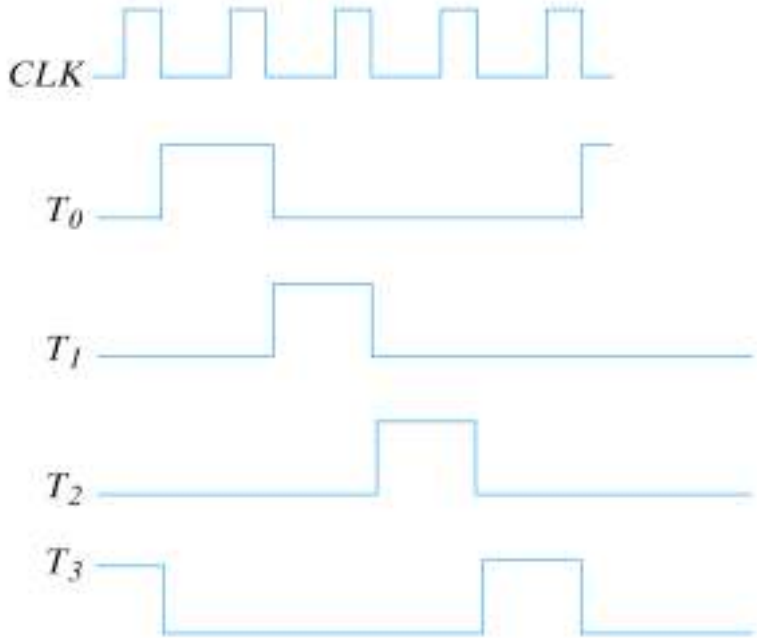
Fig. 6-16 Counter with Unused States



(a) Ring-counter (initial value = 1000)

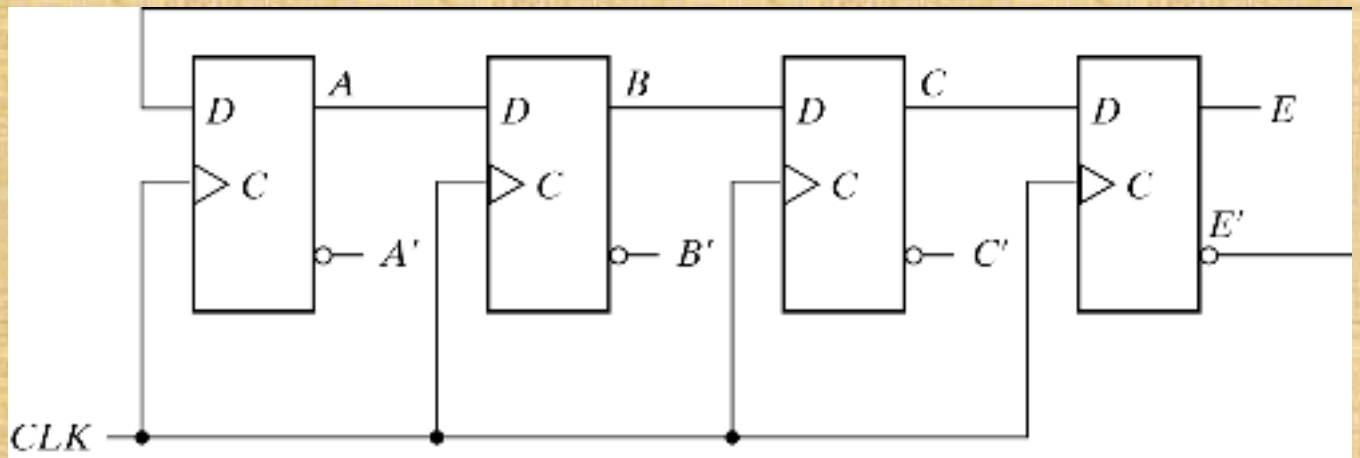


(b) Counter and decoder



(c) Sequence of four timing signals

Fig. 6-17 Generation of Timing Signals



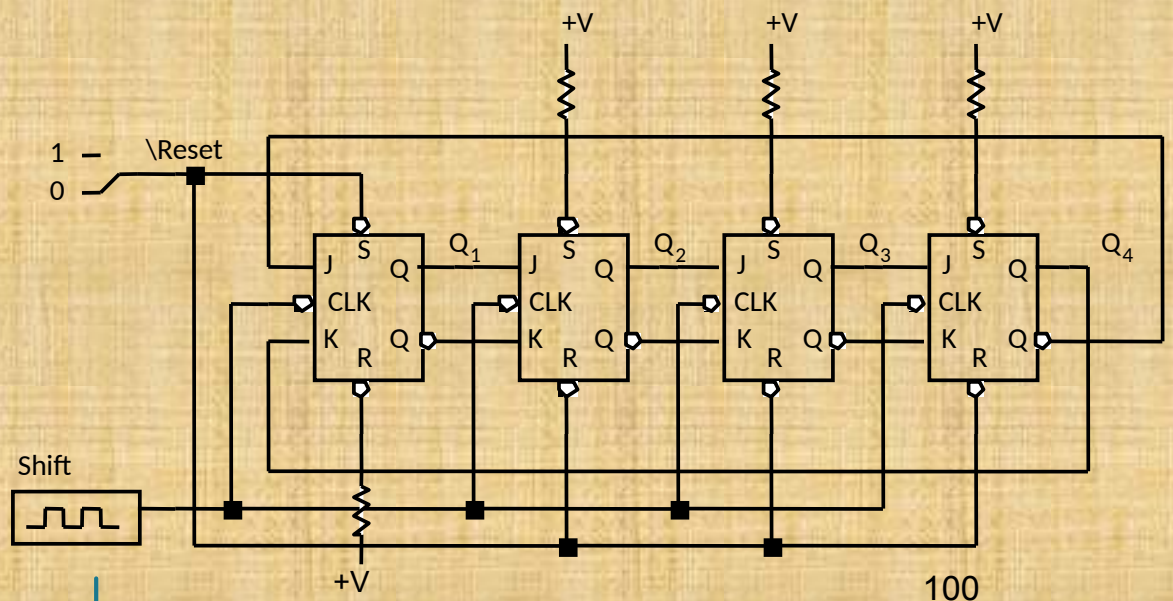
(a) Four-stage switch-tail ring counter

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	$C'E$

(b) Count sequence and required decoding

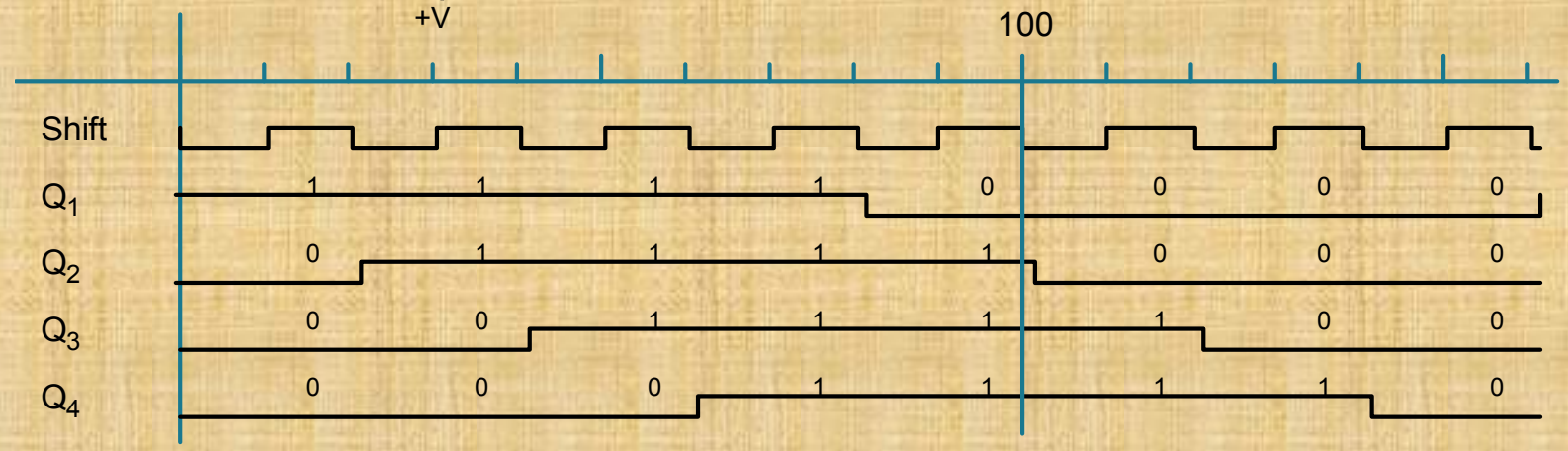
Fig. 6-18 Construction of a Johnson Counter

Twisted Ring (Johnson, Mobius) Counter

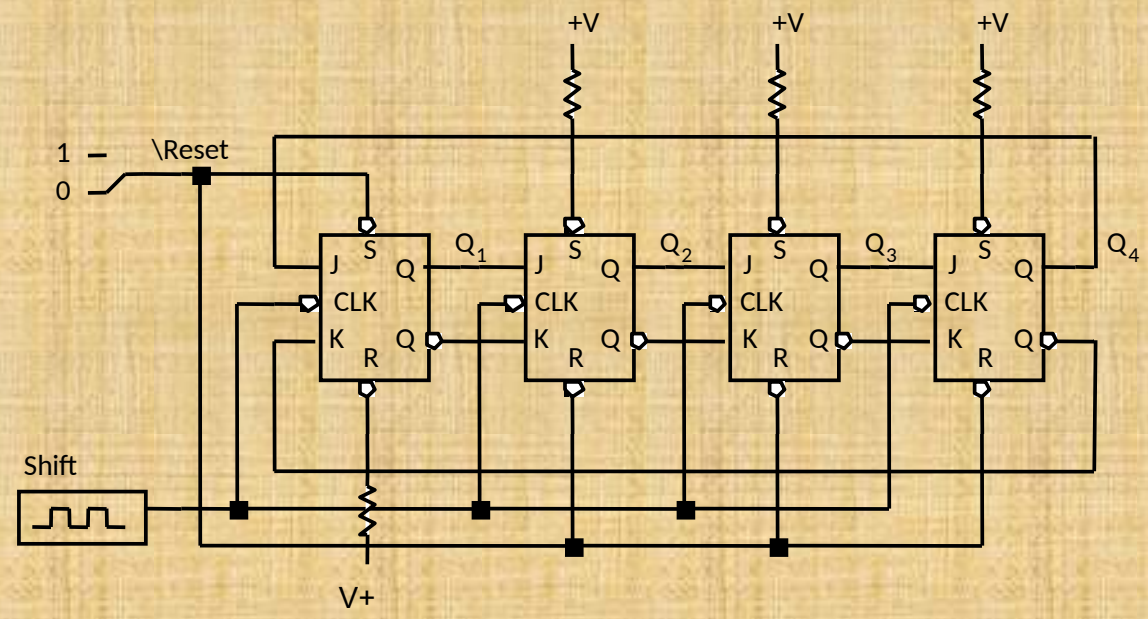


Inverted End-Around

8 possible states, single bit change per state, useful for avoiding glitches



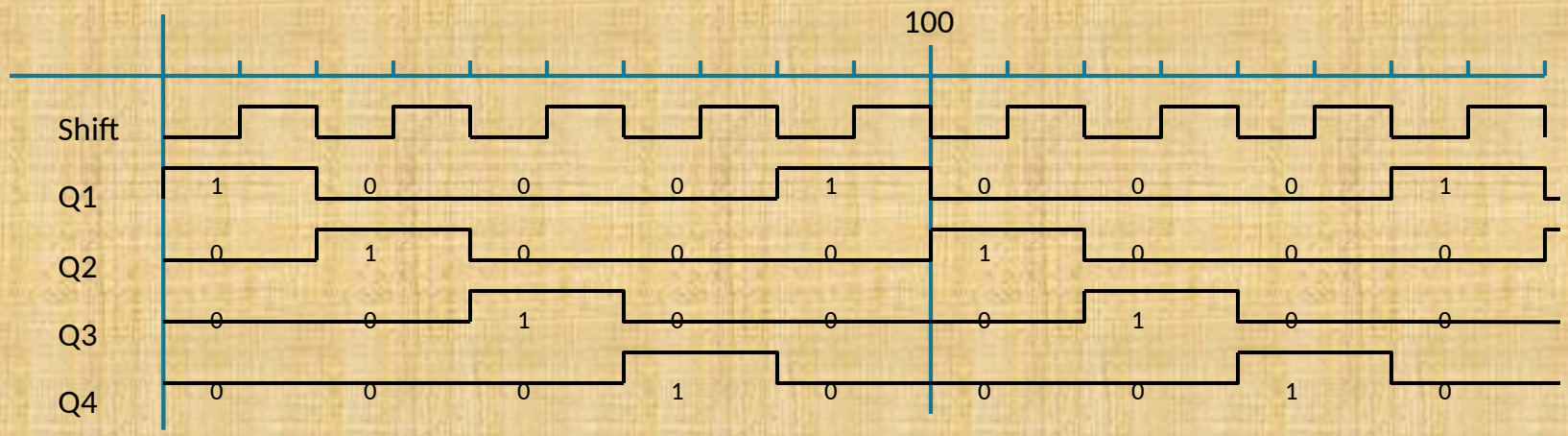
Ring Counter



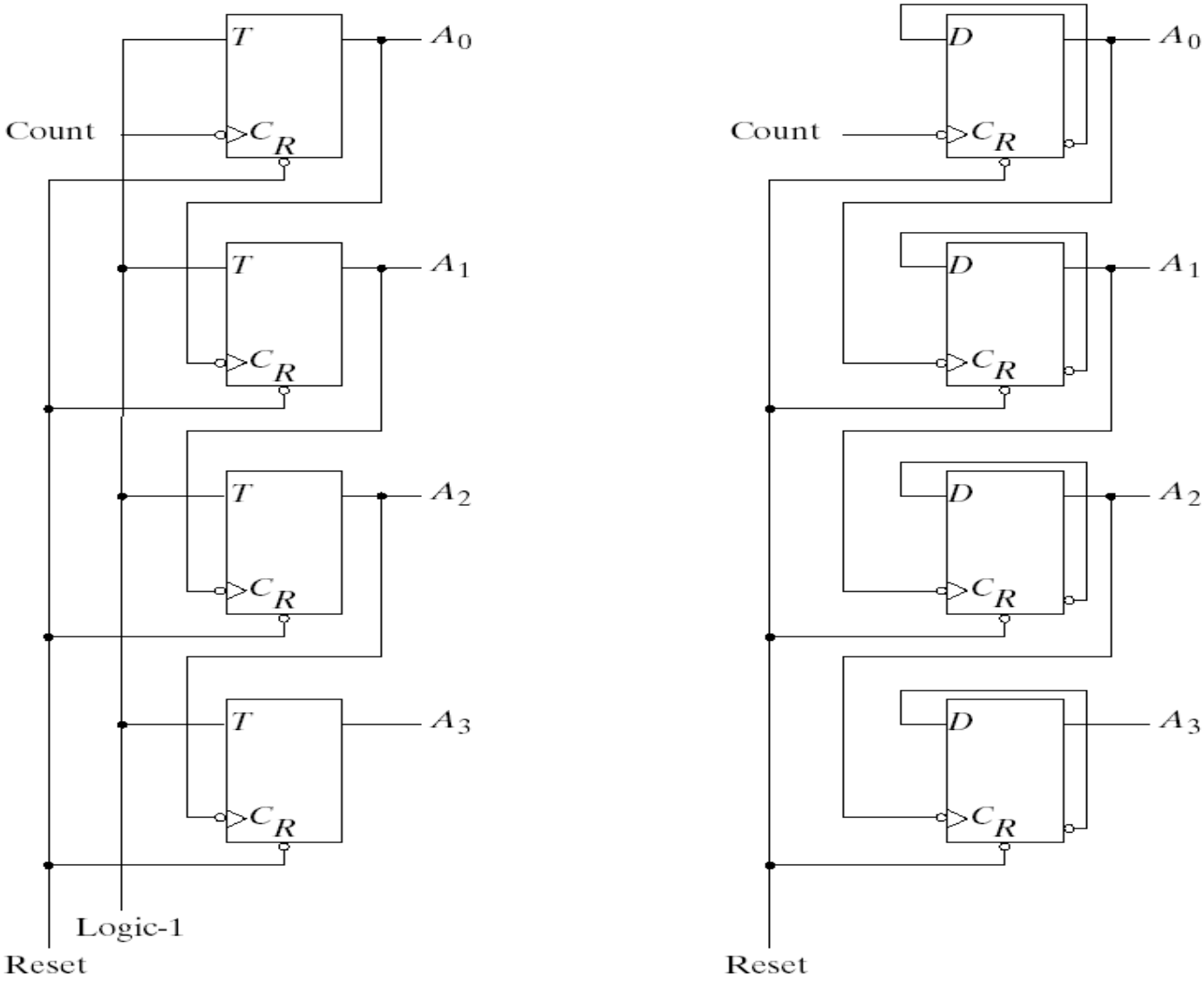
End-Around

4 possible states, single bit change per state, useful for avoiding glitches

Must be initialized



4-Bit Binary Ripple Counter



(a) With T flip-flops

(b) With D flip-flops

Fig. 6-8 4-Bit Binary Ripple Counter

BCD Ripple Counter

- It is possible to build counters that go through any fixed sequence of binary numbers
- For example, a BCD ripple counter can be specified by the following state diagram:

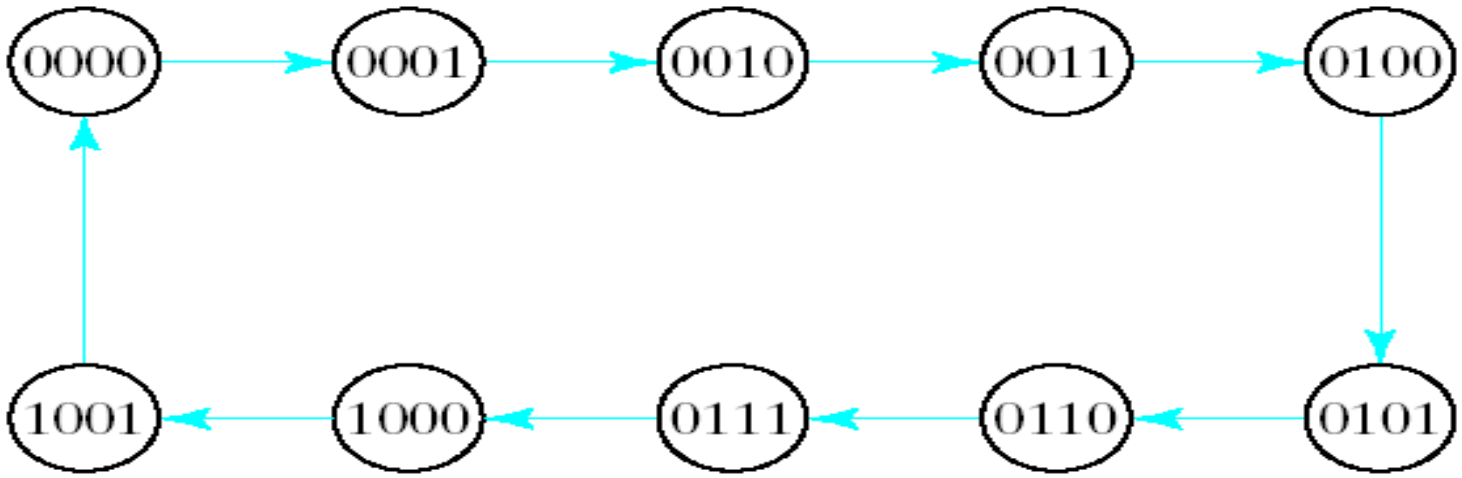
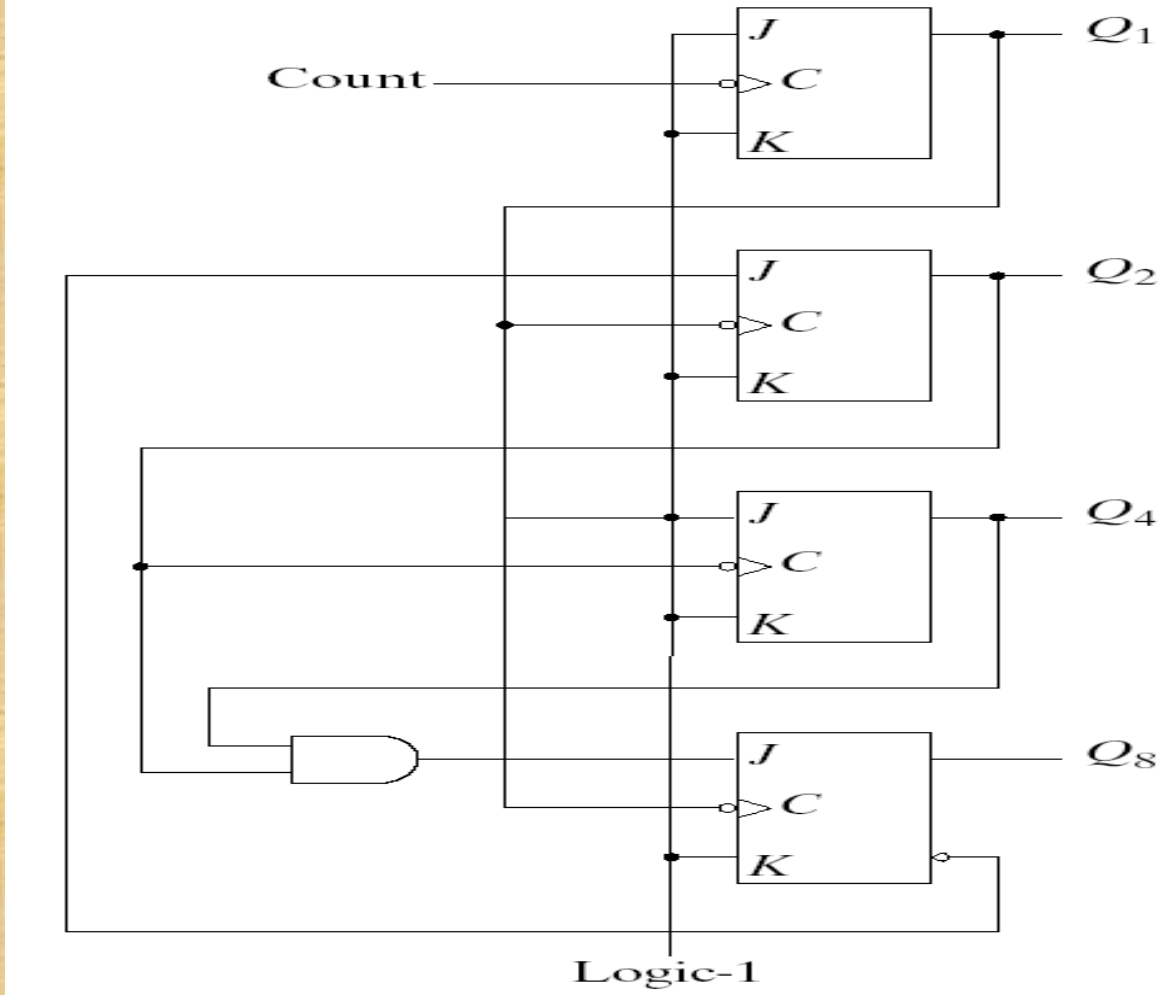


Fig. 6-9 State Diagram of a Decimal BCD-Counter

4-Bit BCD Ripple Counter



12-Bit BCD Counter

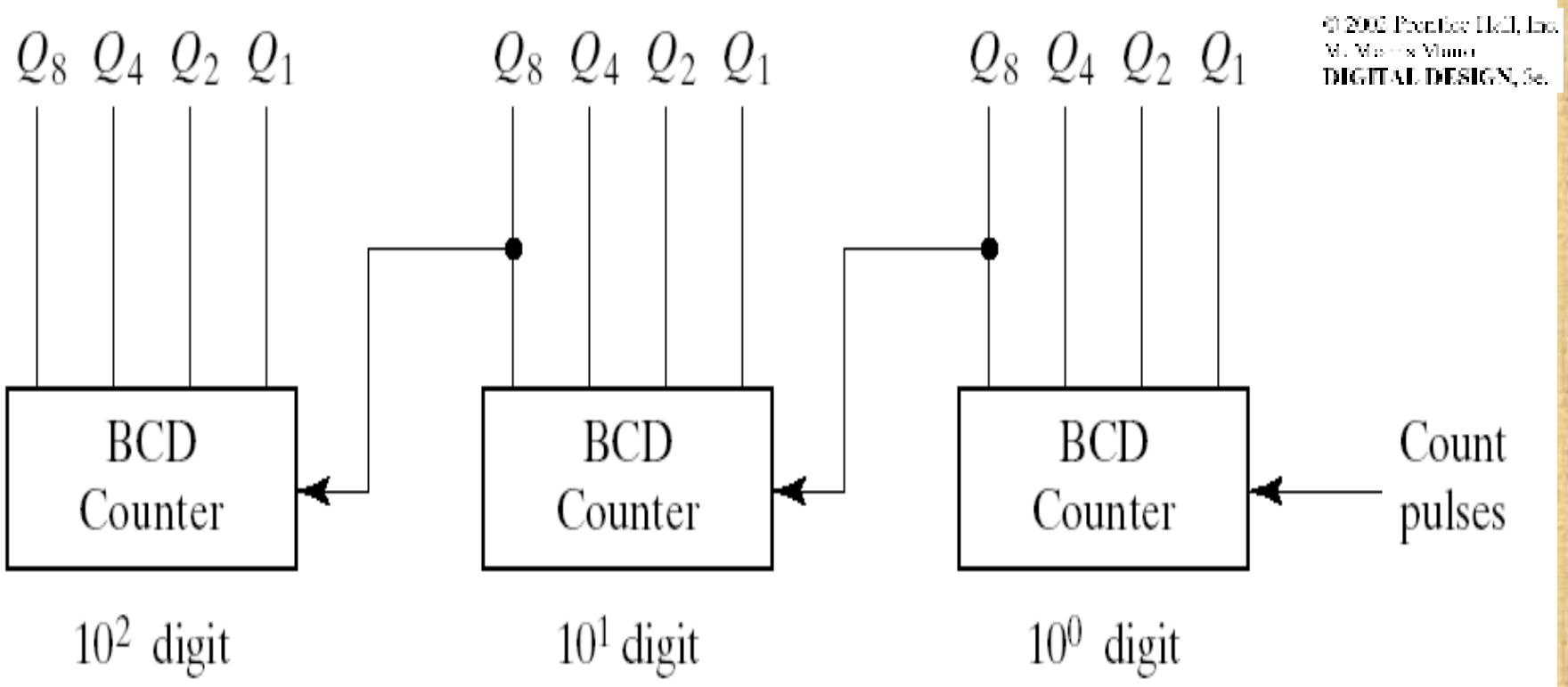


Fig. 6-11 Block Diagram of a Three-Decade Decimal BCD Counter

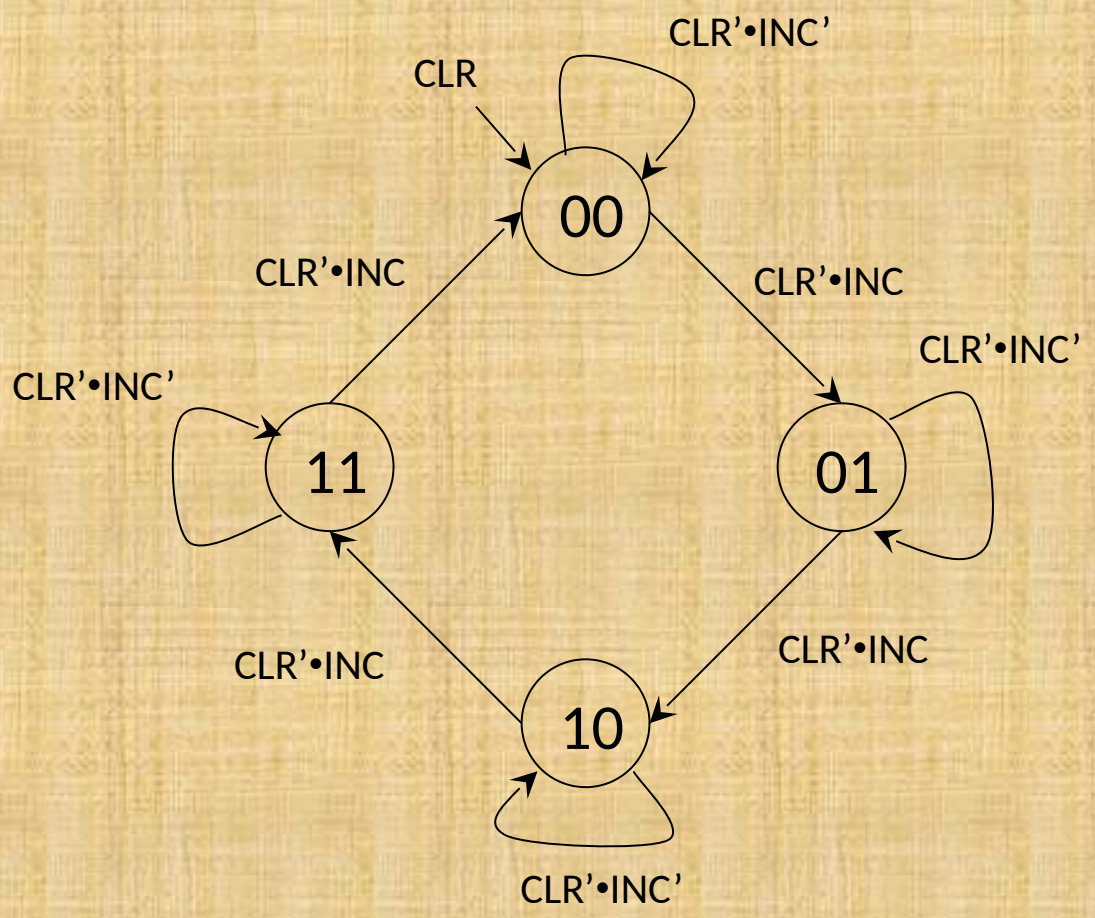
Mod-N Counters

- Generally we are interested in counters that count up to specific count values
  - Not just powers of 2
- A mod-N counter has N states
  - Counts from 0 to N-1 then rolls over
  - Requires $\lceil \log_2 N \rceil$ flip flops
- For example…
  - A 4-bit binary counter is a mod-16 counter
  - A counter that counts from 0-9 is a mod-10 counter

A Mod-4 Counter

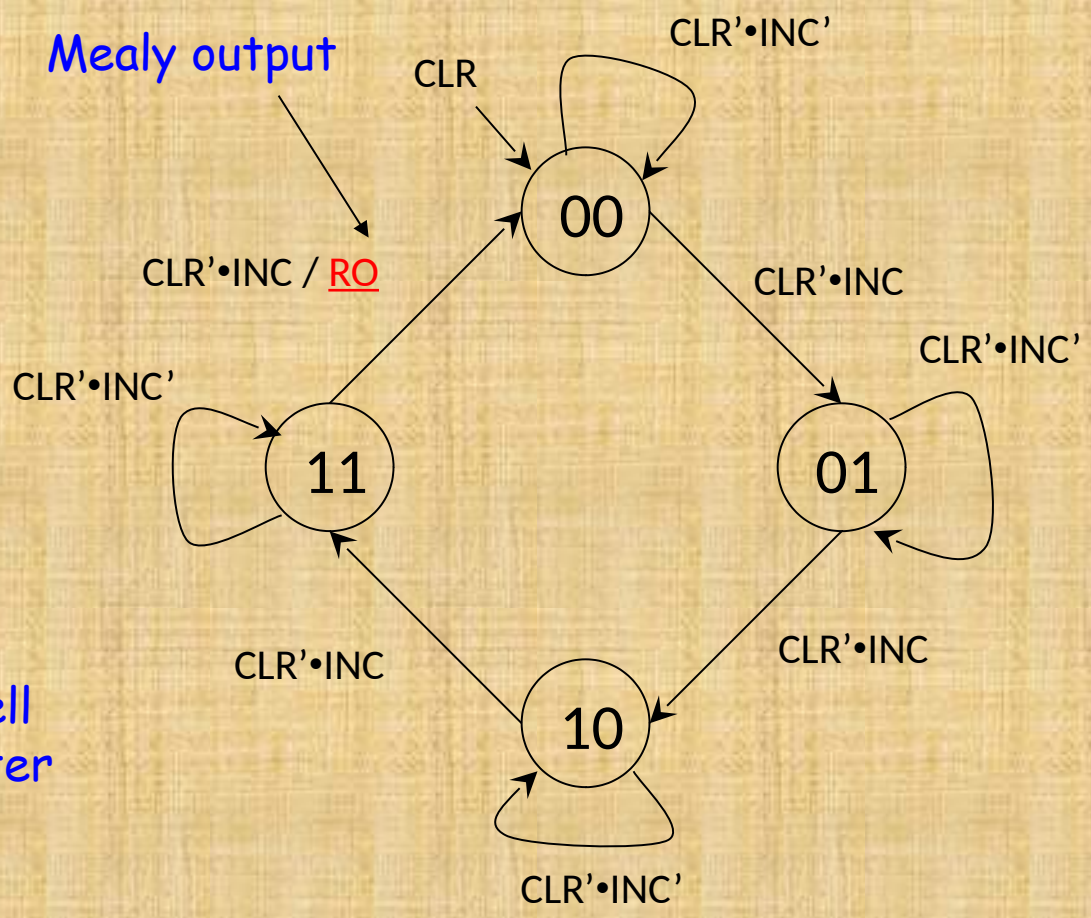
A.K.A. 2-bit counter

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 Mod-4 Counter With Rollover Signal

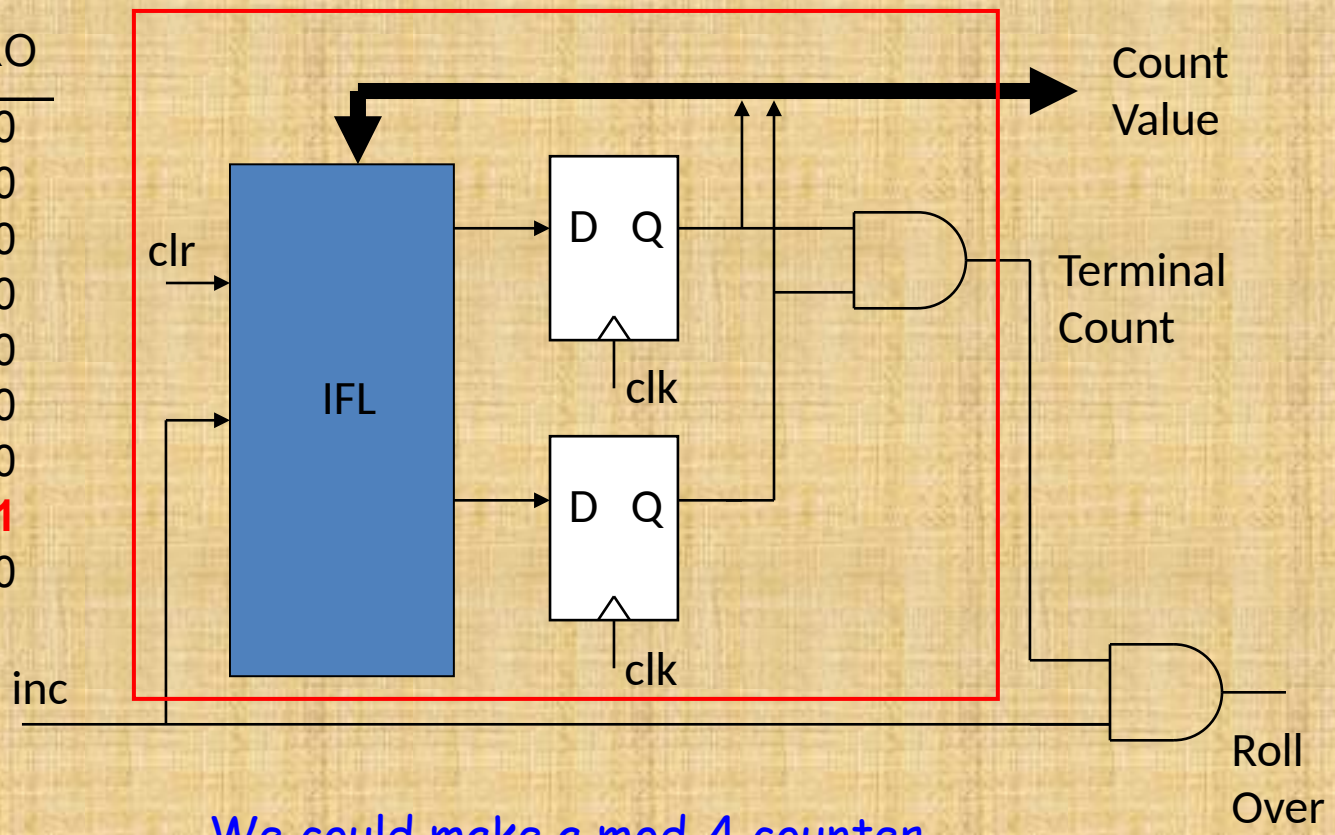
CLR	INC	Q1	Q0	N1	N0	RO
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	-	-	-	0	0	0



The ROLL signal is used to tell other circuitry that the counter is rolling over to all 0's.

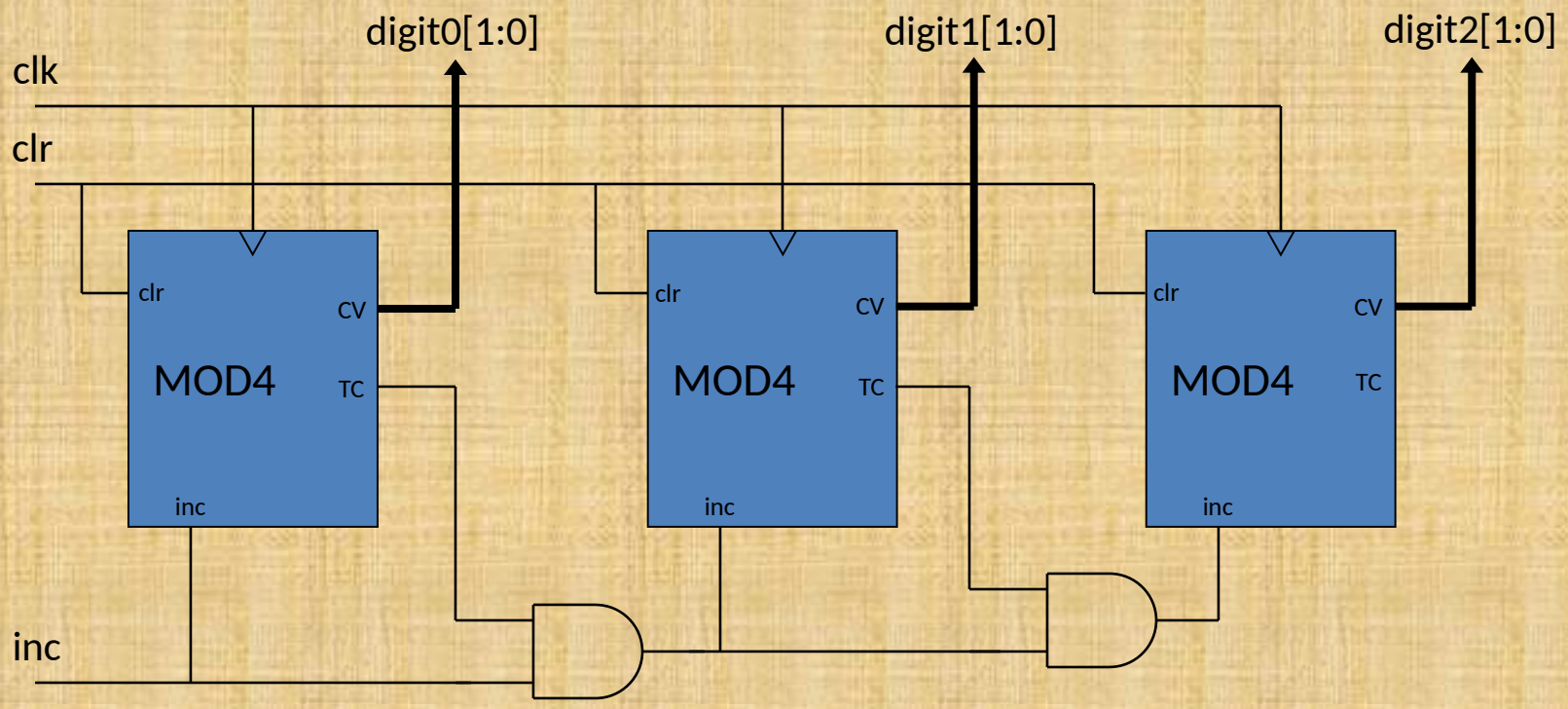
A Mod-4 Counter

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

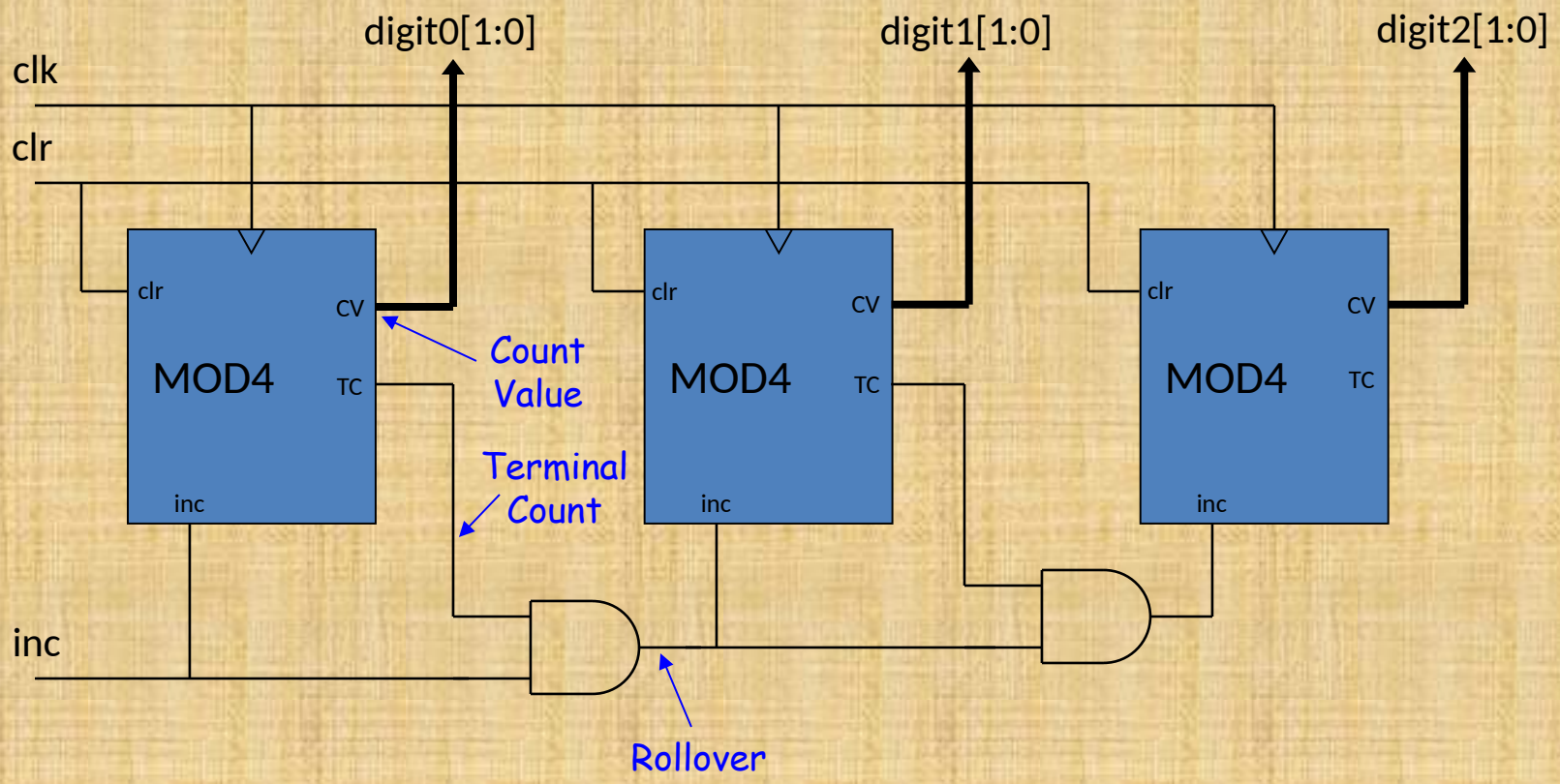


We could make a mod-4 counter from the block shown in red.

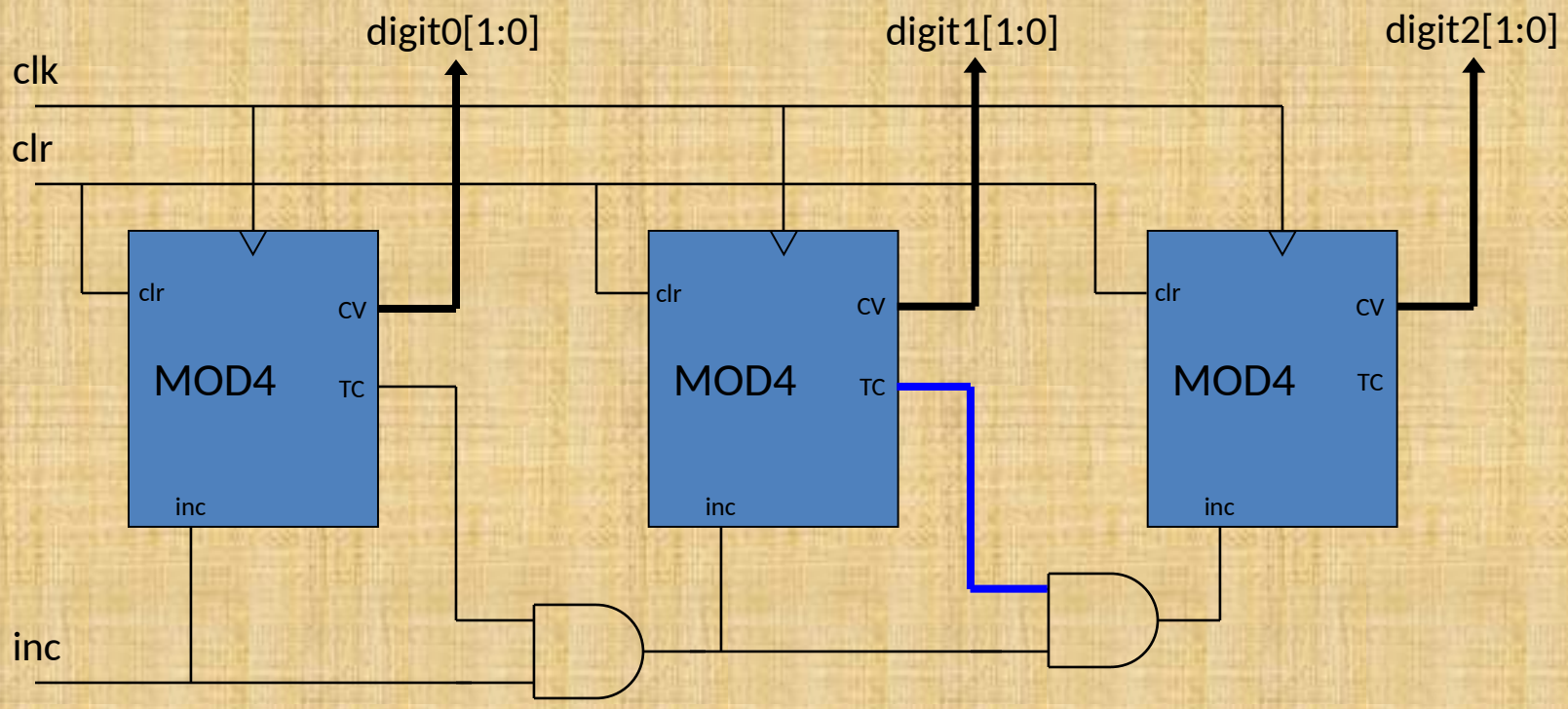
Cascaded Counters



Cascaded Counters

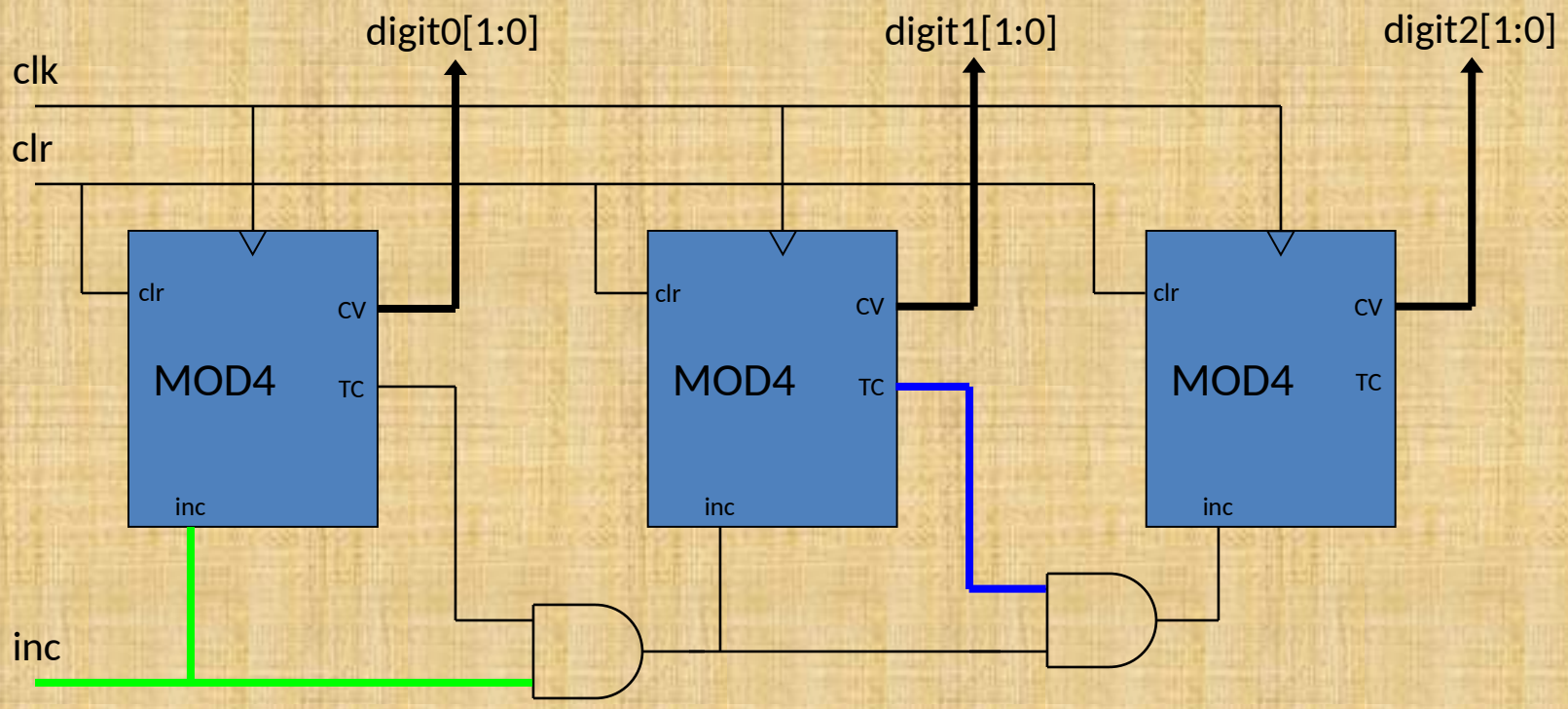


Cascaded Counters

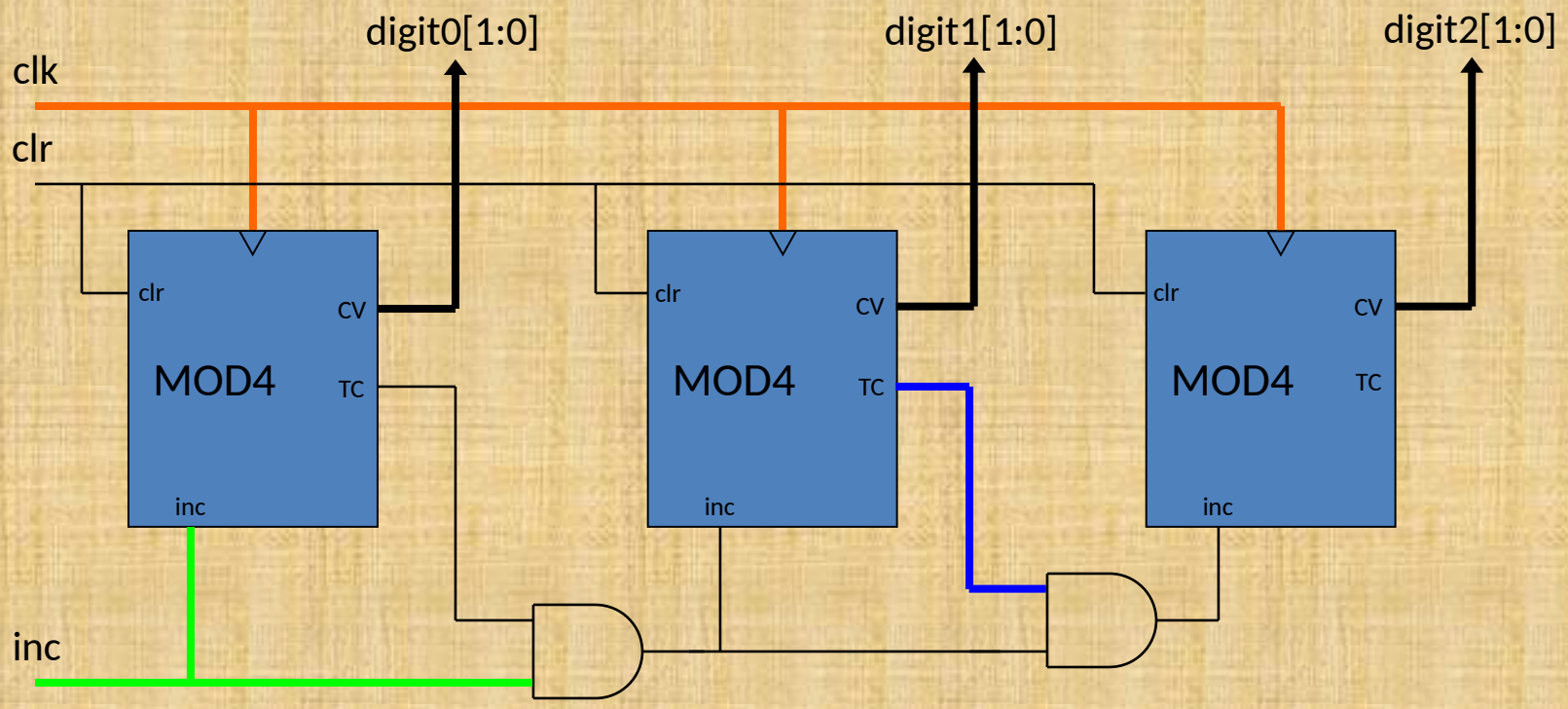


Assume that the second timer is already at the terminal count.

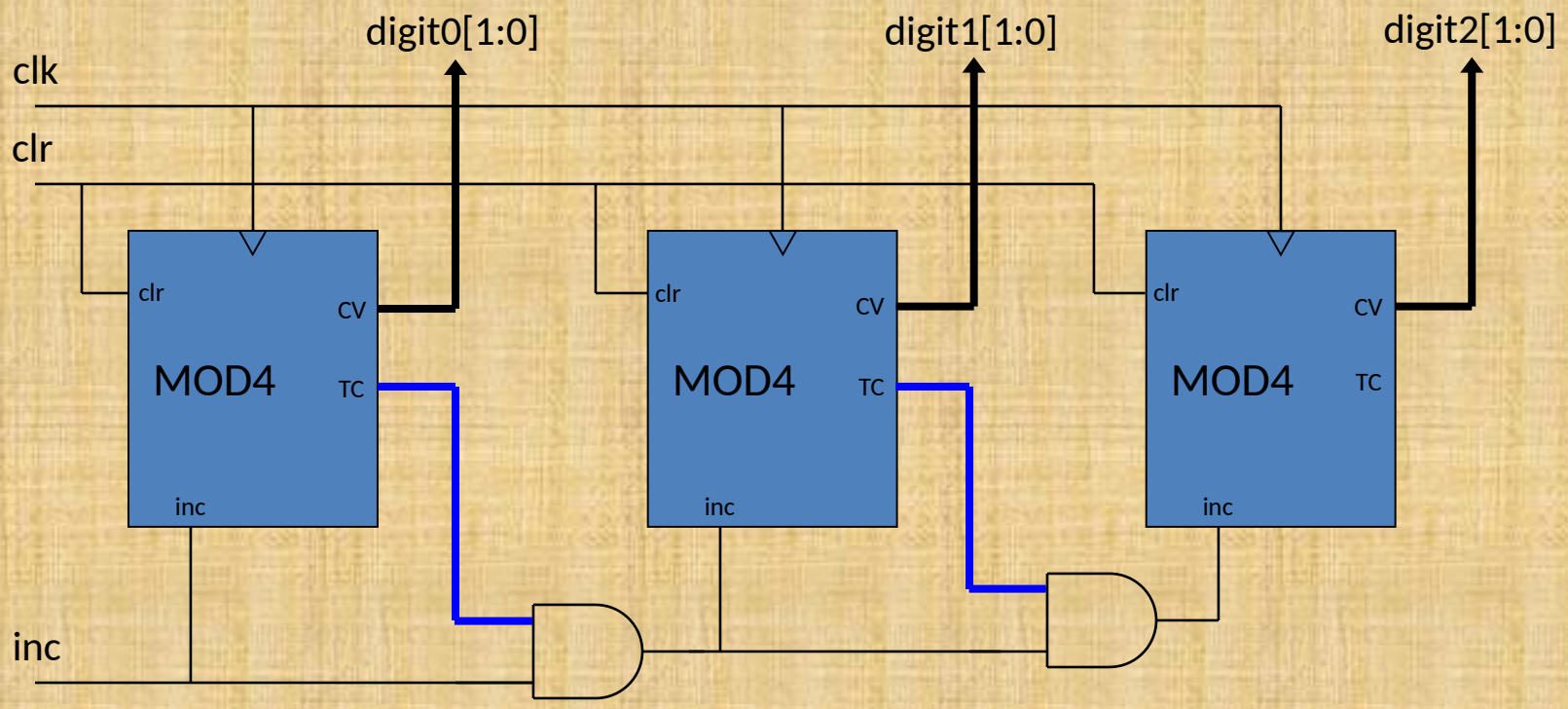
Cascaded Counters



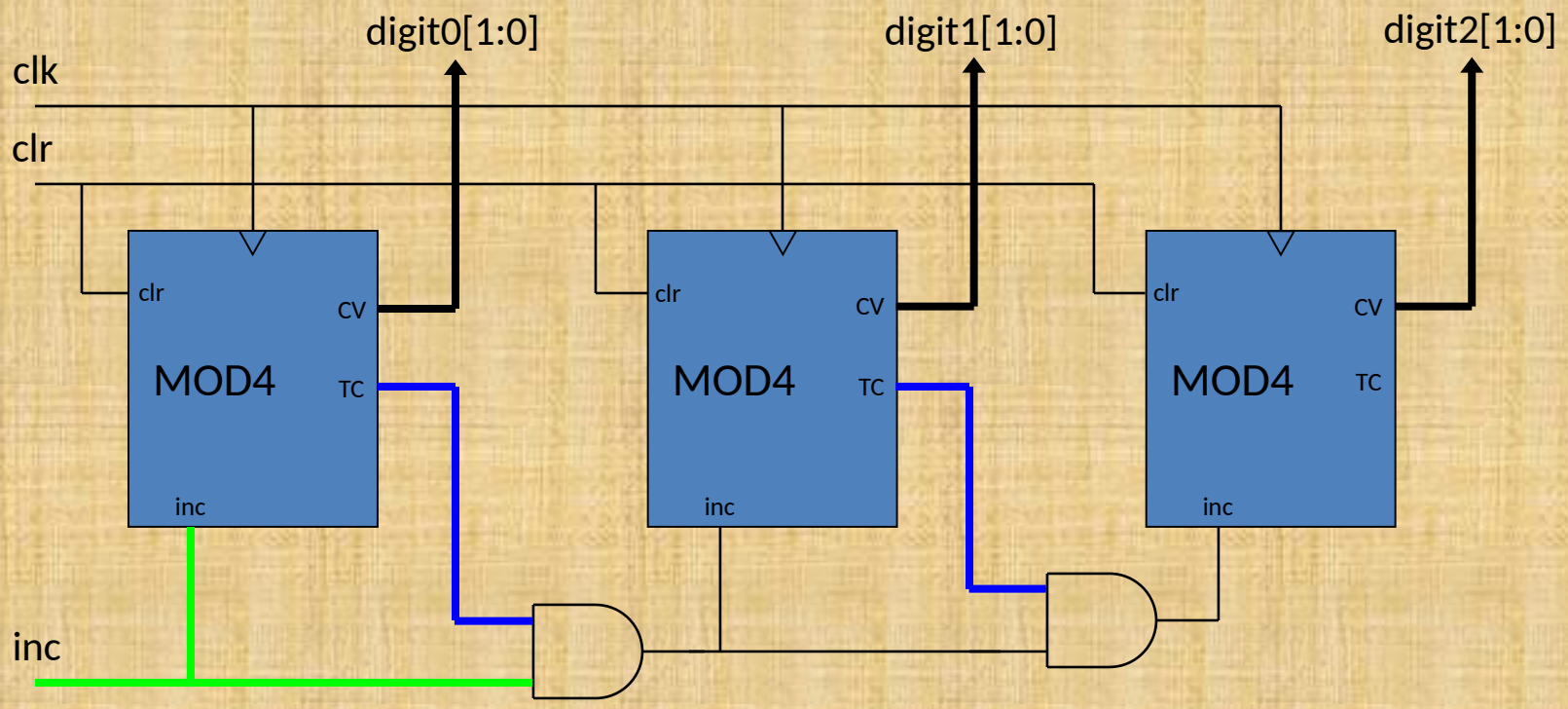
Cascaded Counters



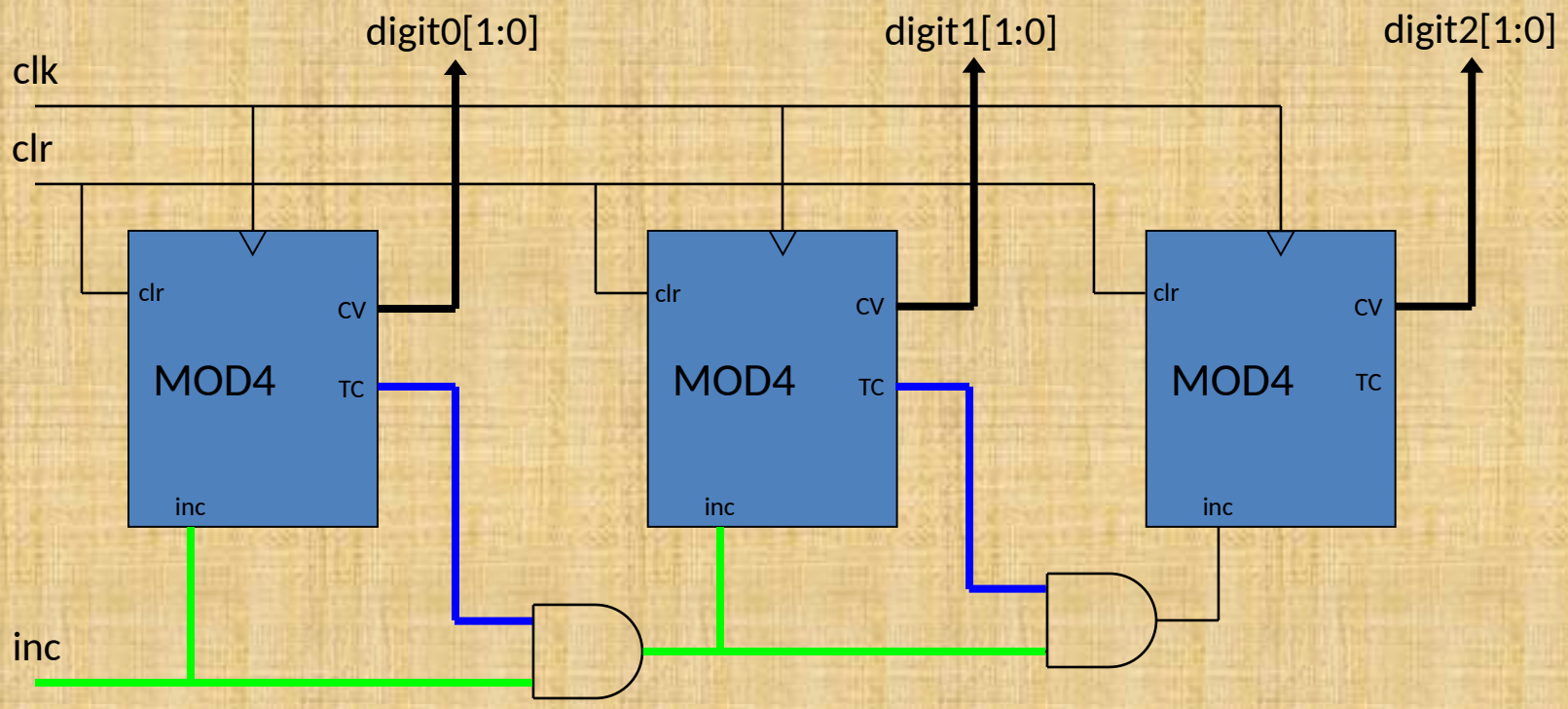
Cascaded Counters



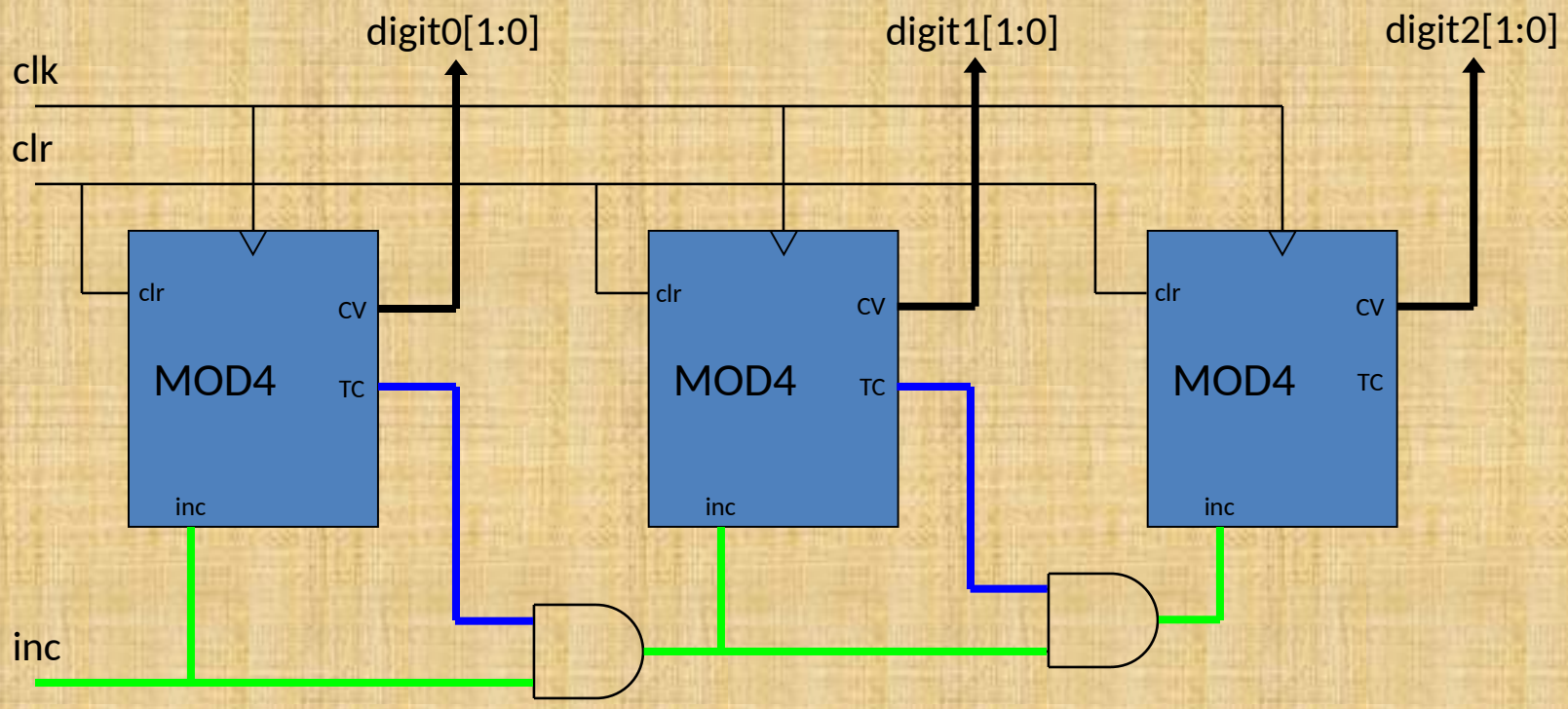
Cascaded Counters



Cascaded Counters



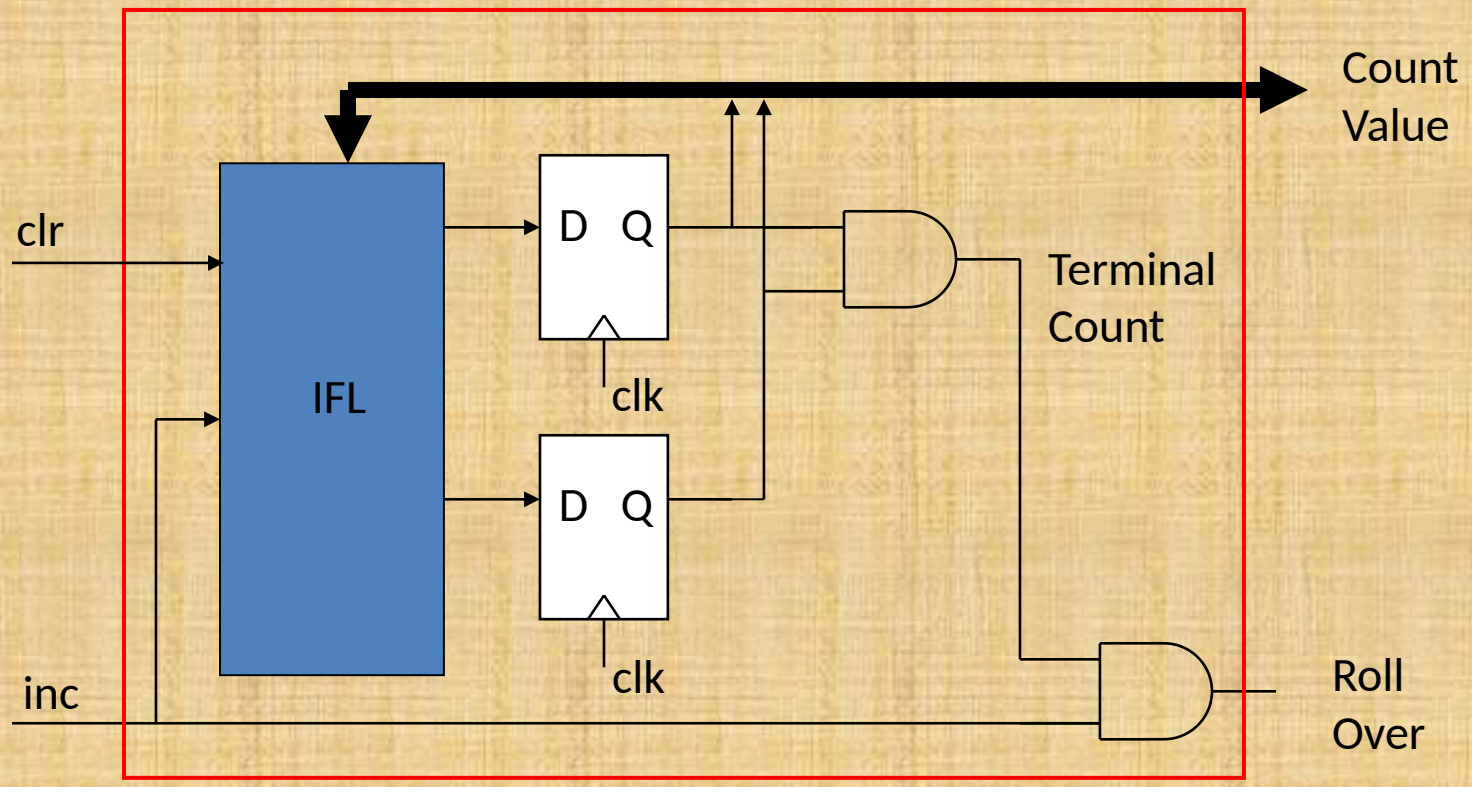
Cascaded Counters



It looks like the inc signal ripples from counter to counter.
How is this different from the ripple counter examples?

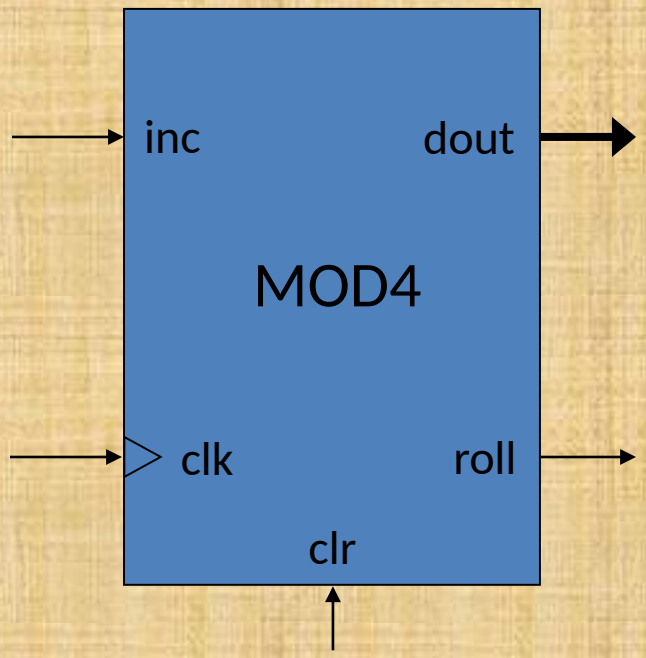
A Mod-4 Counter

With consolidated rollover logic



A good mod-4 counter includes the logic within the red block.

A Mod-4 Counter



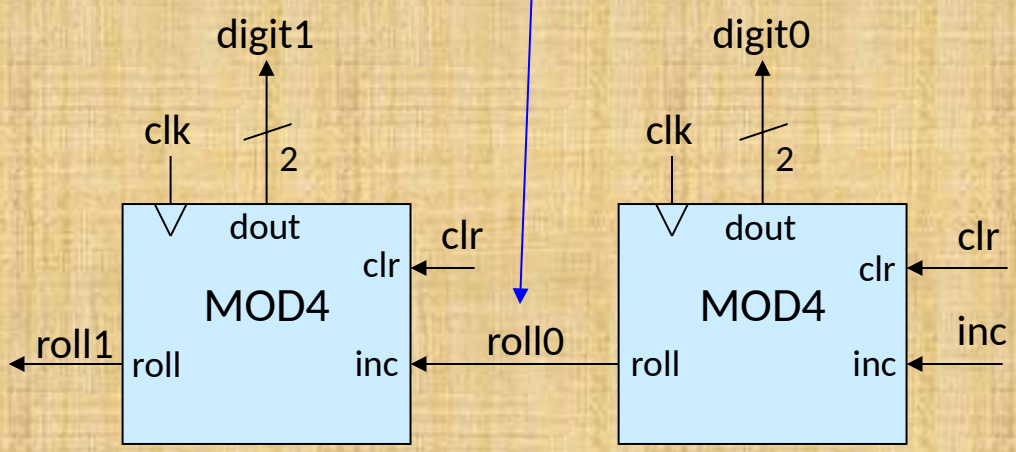
Cascading two Mod-4 Counters

Count Sequence:

digit1 digit0

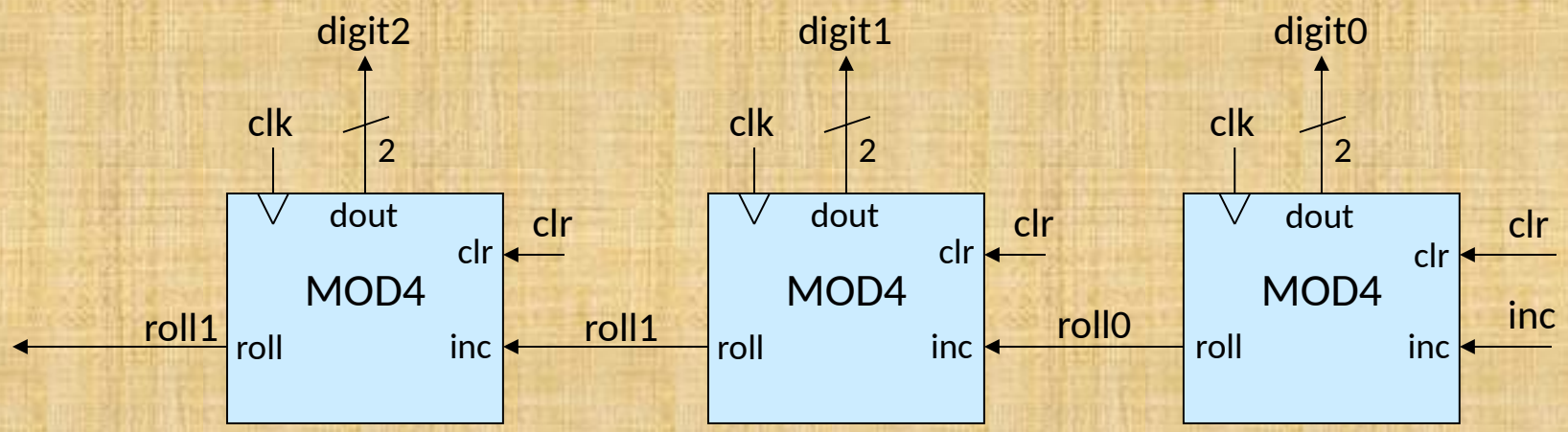
00
01
02
03
10
11
12
13
20
21
22
23
30
31
32
33
00
...

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



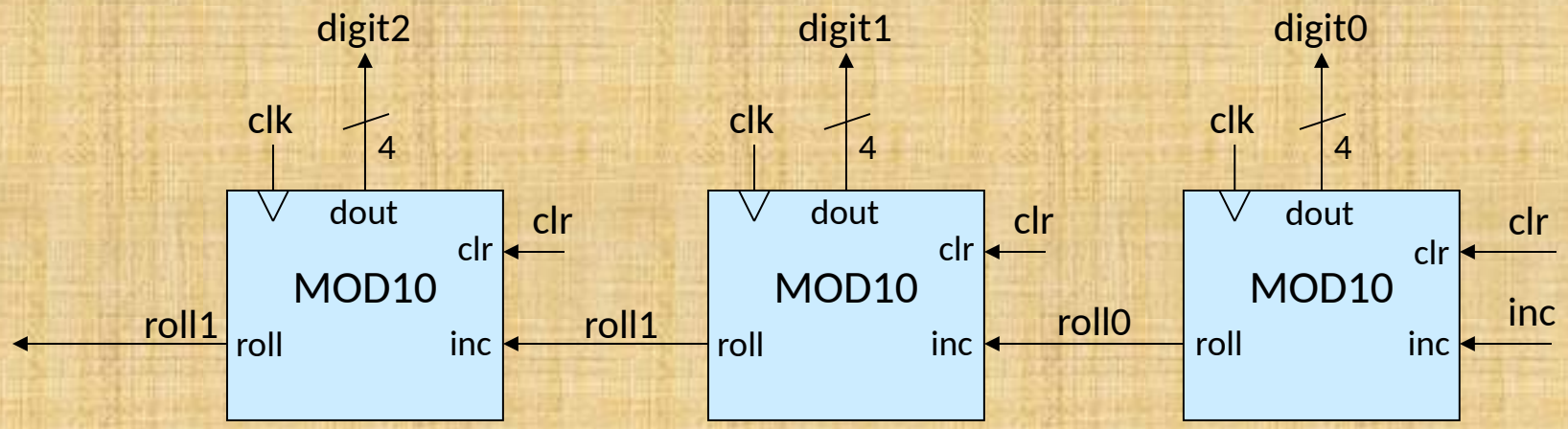
Three-digit Mod-4 Counter

- Can combine any counters that have a rollover signal to make larger counters
  - Combine two 16-bit counters to make a 32-bit counter
  - Combine three mod-4 counters to make a three-digit mod-4 counter

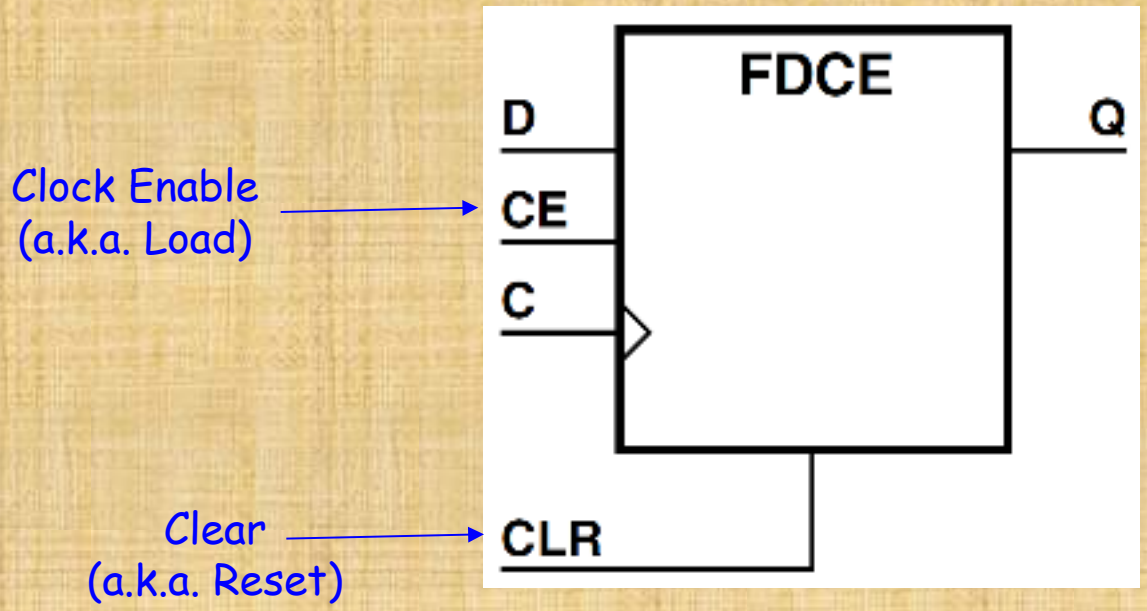


BCD Counter

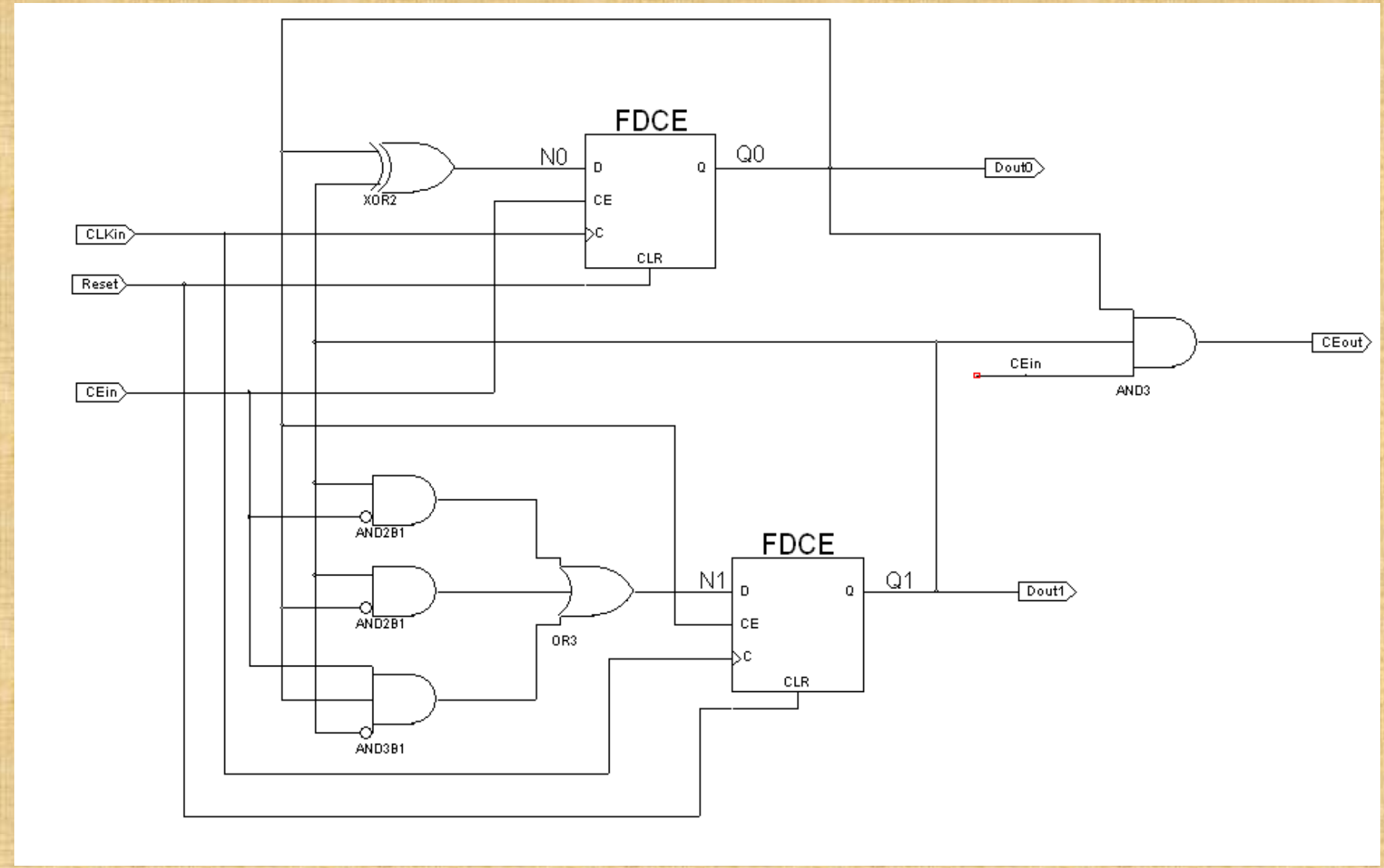
- Combine to create non-binary counters
  - BCD counter



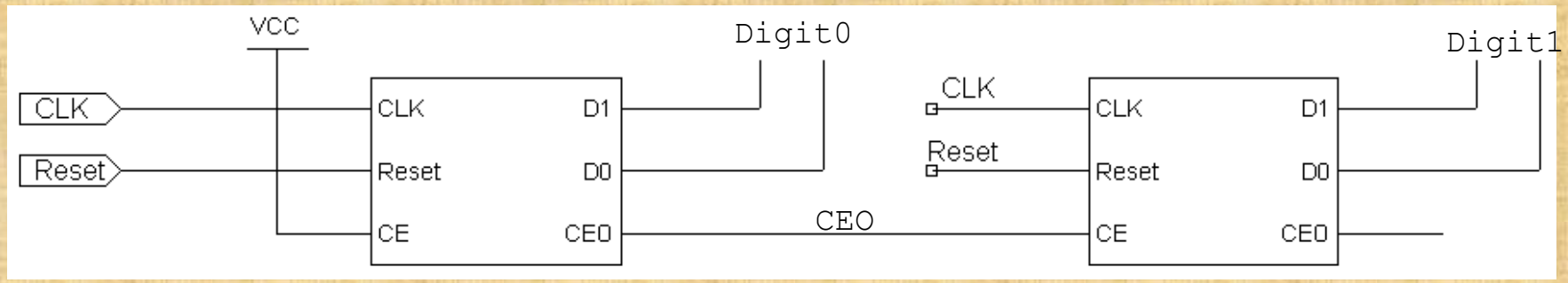
D Flip Flop with Asynchronous Clear and Clock Enable

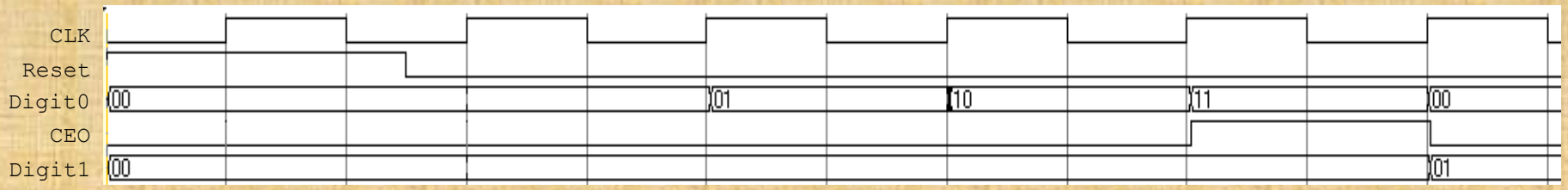


Mod-4 Counter



Cascaded Synchronous Counter





Hybrid Counters

- Can combine different kinds of mod counters
  - Combine an 8-bit counter with a 16-bit counter to create a 24-bit counter
  - Combine mod-24 and mod-60 counters to create a digital H:M:S clock