Faculty of Engineering

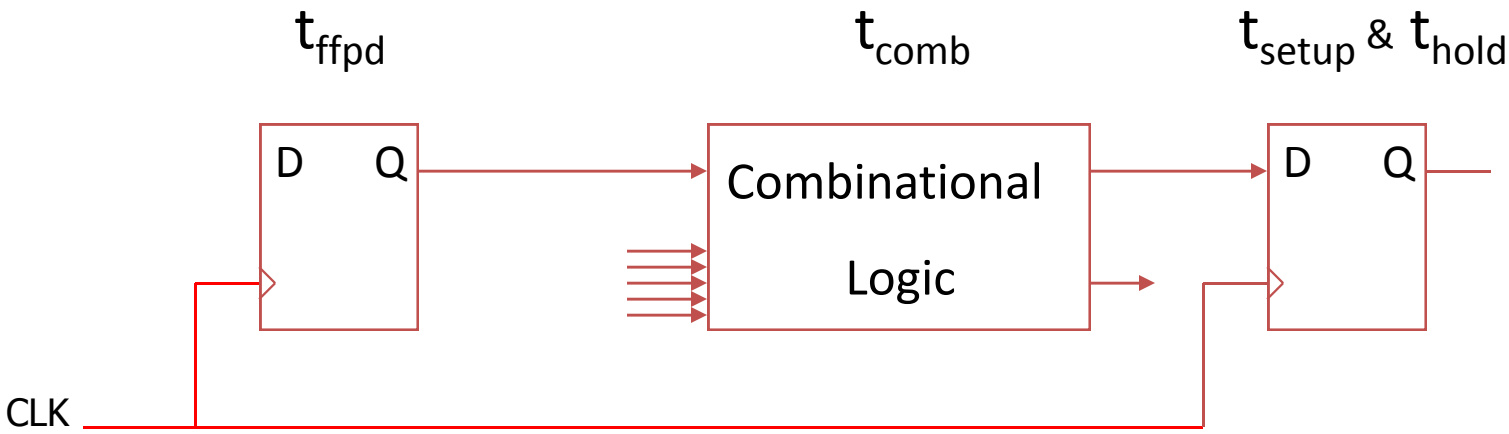
CSE115: Digital Design

Lecture 25:
Timing, Counters and Shift Registers

Suggested Reading

- Sections 8.1.4, 8.2.5, 8.4, 8.5

Synchronous System - Detailed Timing



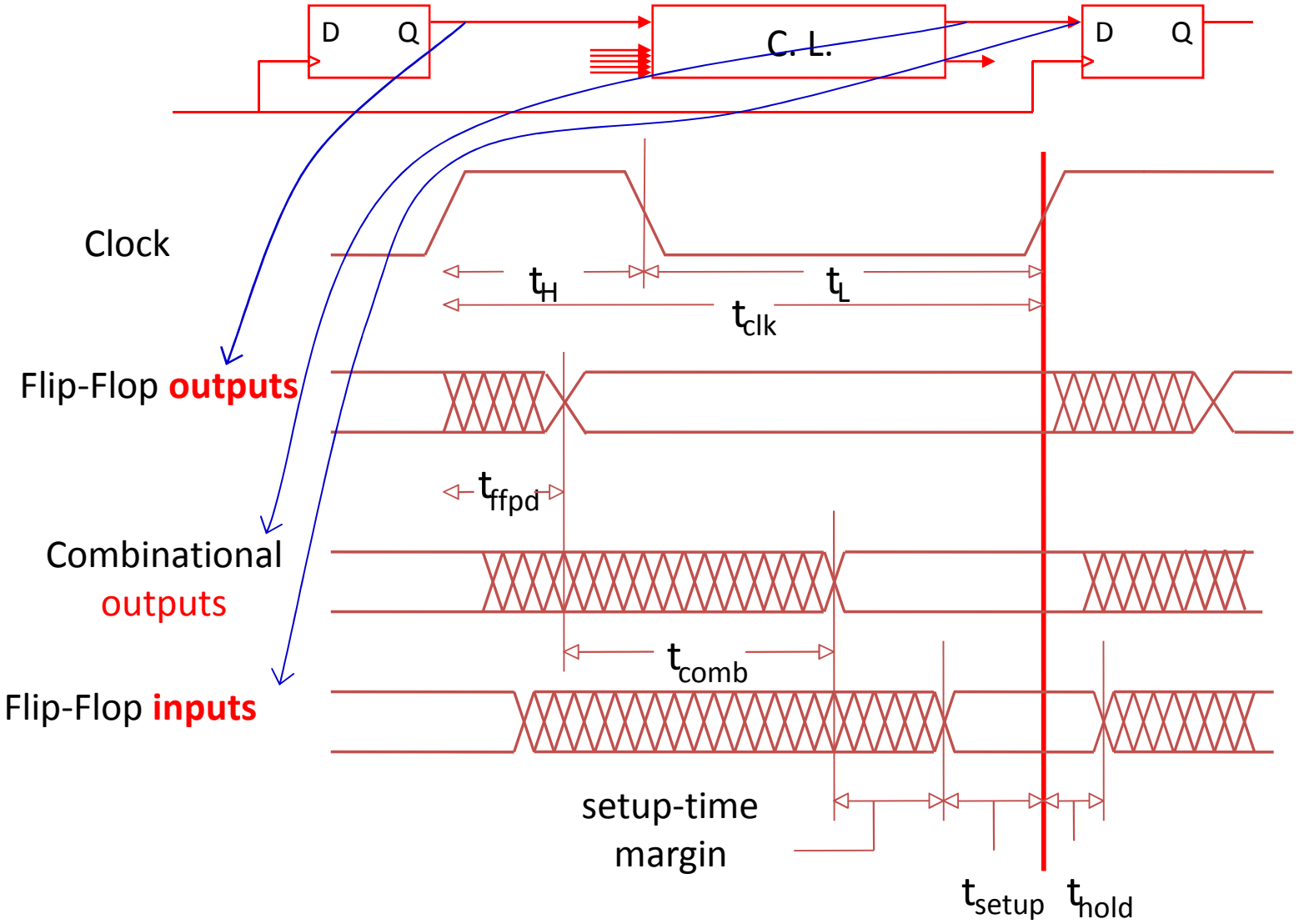
t_{ffpd} - CLK to Q, FF propagation delay (min, max)

t_{comb} - combinational logic delay (min, max)

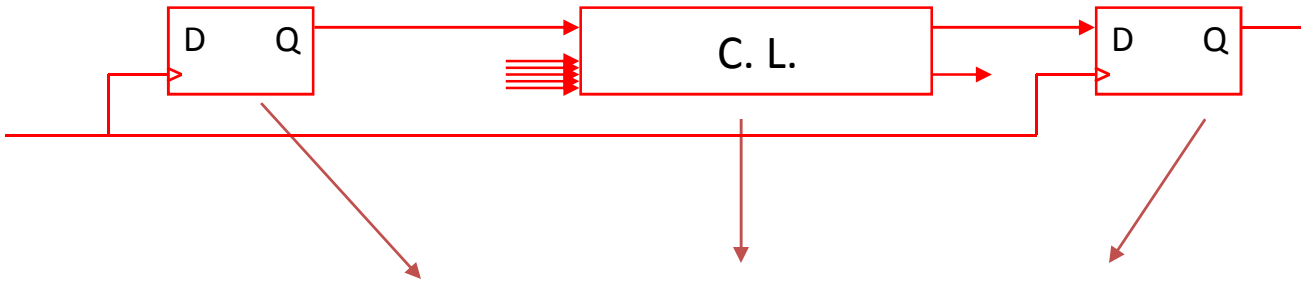
t_{setup} - input stable before clock (min)

t_{hold} - input stable after clock (min)

Synchronous System - Detailed Timing



Synchronous System - Detailed Timing



Required

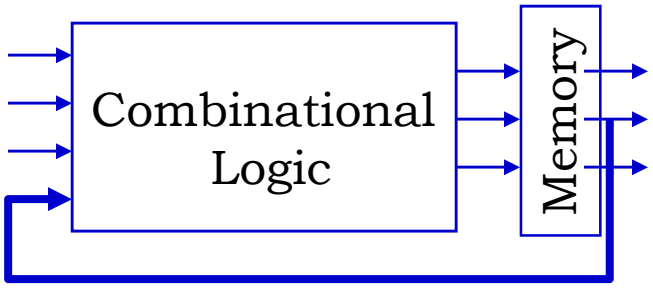
$$t_{clk,min} > t_{ffpd,max} + t_{comb,max} + t_{setup,min}$$

Difference → Setup time margin

$$t_{ffpd,\ min} + t_{comb,min} > t_{hold,min}$$

Difference → Hold time margin

Synchronous System Example



t_{comb} = 2 ns, min and 20 ns, max

t_{ffpd} = 3 ns, min and 15 ns, max

t_{setup} = 5 ns, min

t_{hold} = 2 ns, min

Setup Margin: $t_{clk,min} > t_{ffpd,max} + t_{comb,max} + t_{setup,min}$

Hold Time Margin: $t_{ffpd,\ min} + t_{comb,min} > t_{hold,min}$

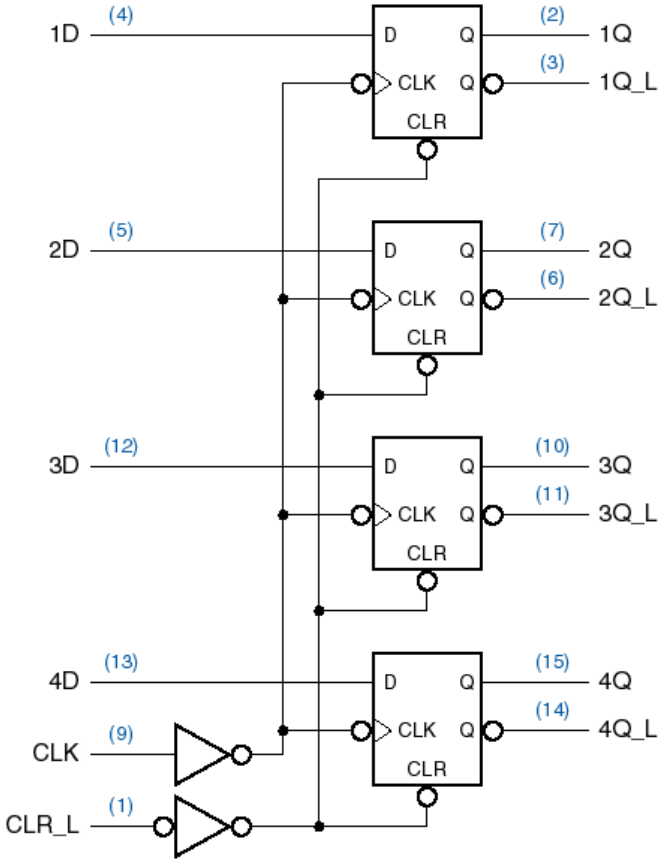
At 10 MHz clk

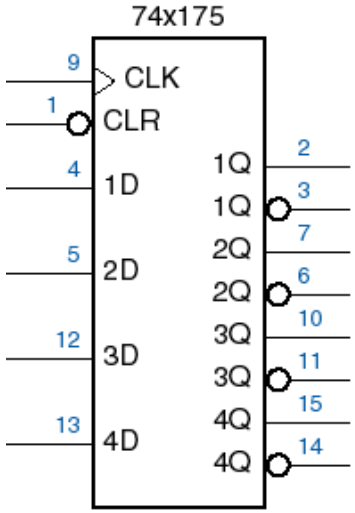
Setup margin: 100 - (15 + 20 + 5) = 60 ns

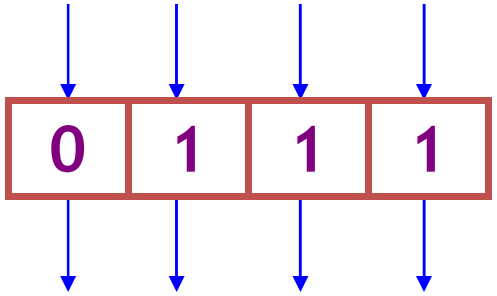
Hold Margin: (3 + 2) -2 = 3ns

Max Frequency: $t_{clk,min} \geq 40$ ns → $f_{max} \leq 25$ MHz

74x175: Multibit Registers and Latches

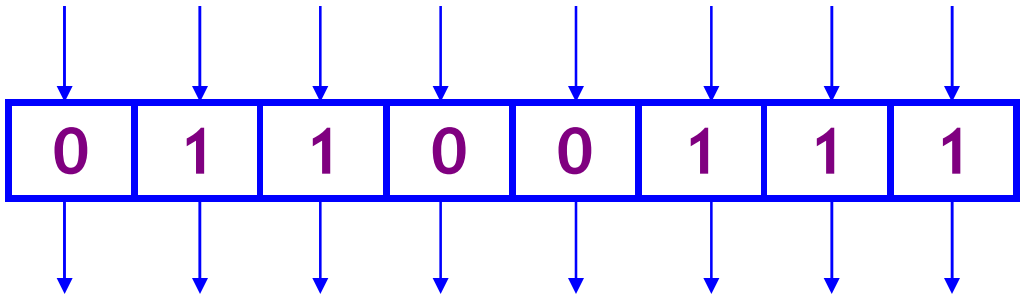






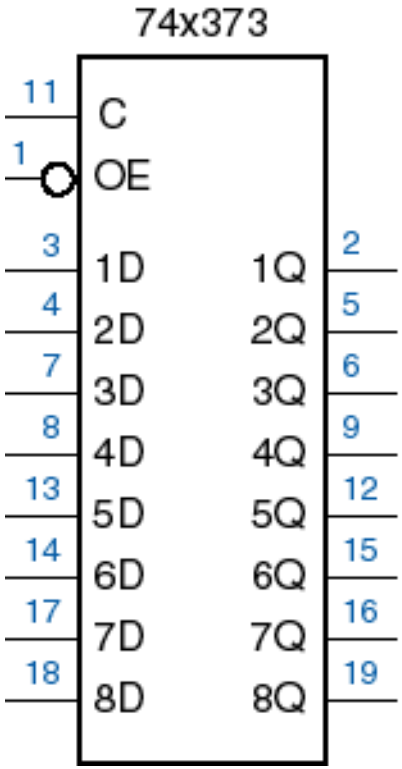
74x373: Octal Latch

Latch: output follows input when C is asserted



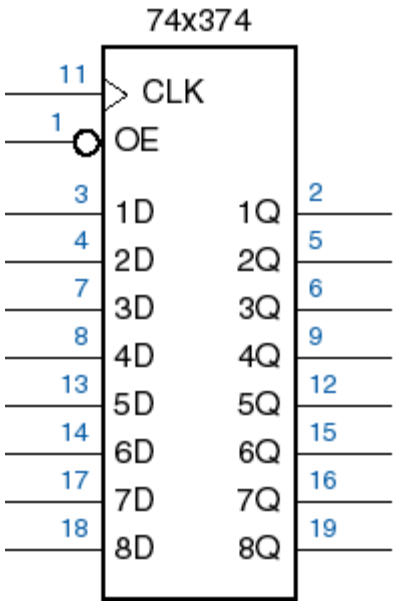
Latch-enable input "C"

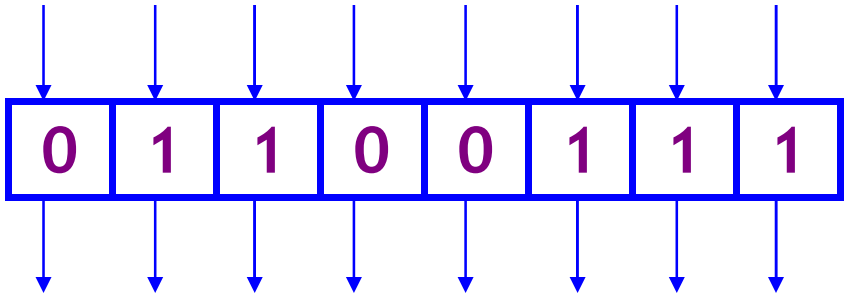
Output Enable

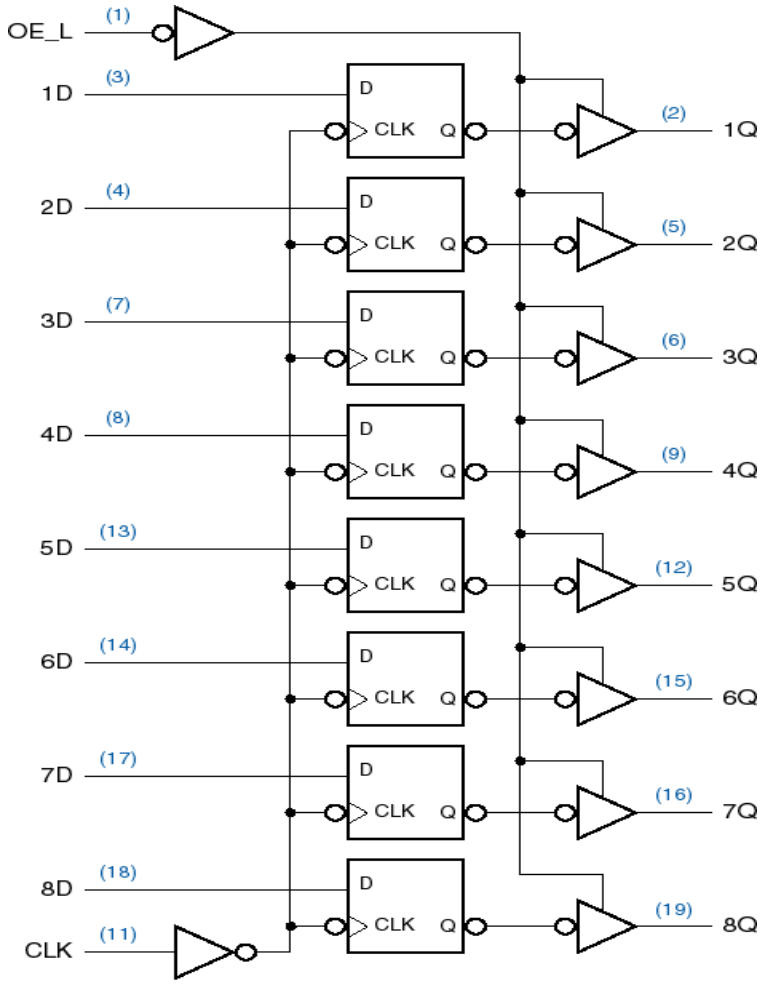


74x374: Octal Register

Register: Edge-Triggered Behavior



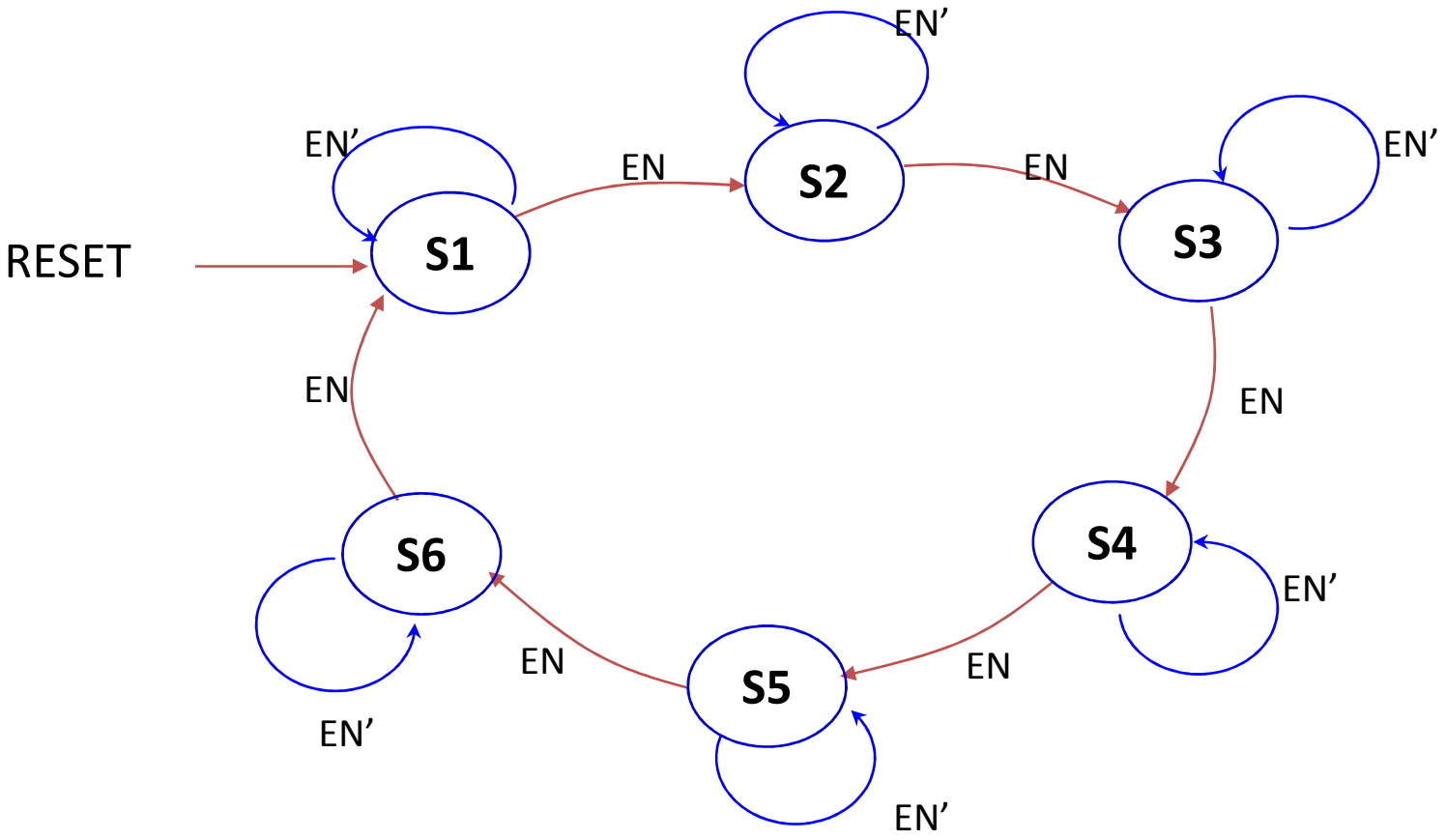




Counters

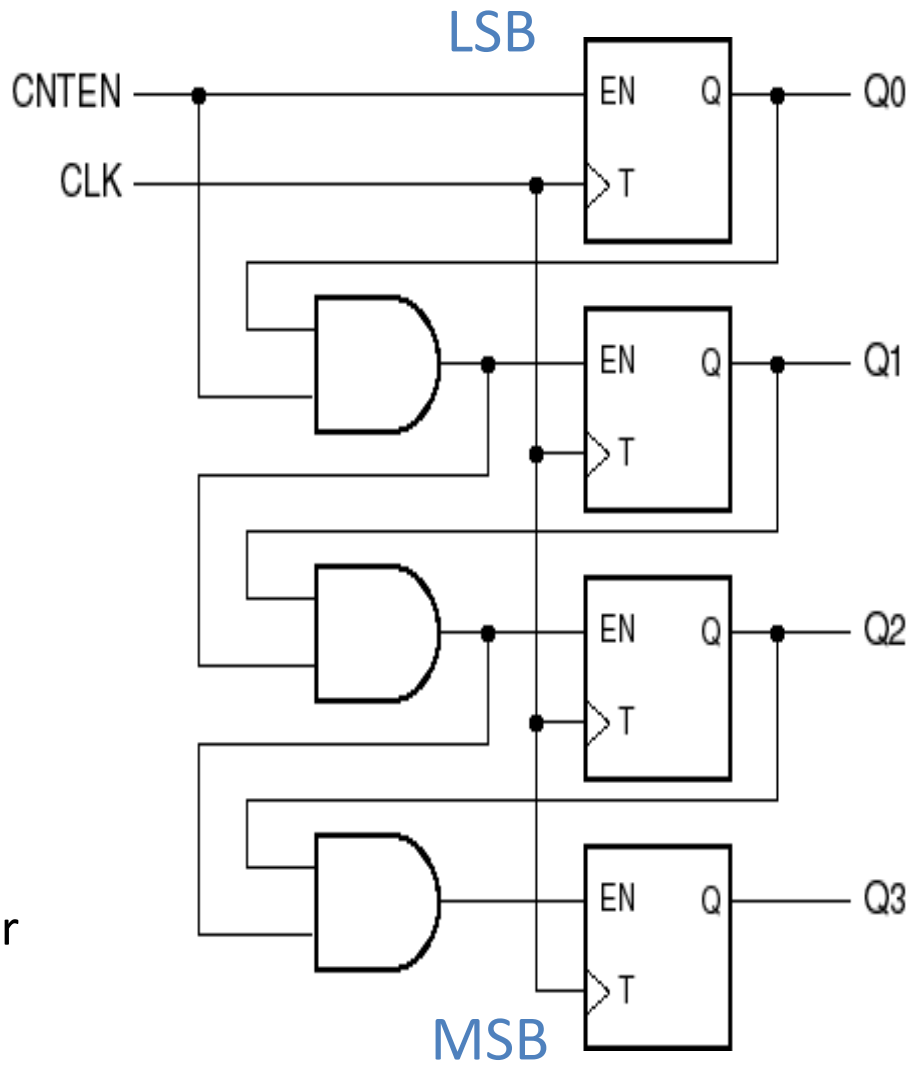
Clocked sequential circuit with single-cycle state diagram

Modulo-m counter = **Divide**-by-m counter



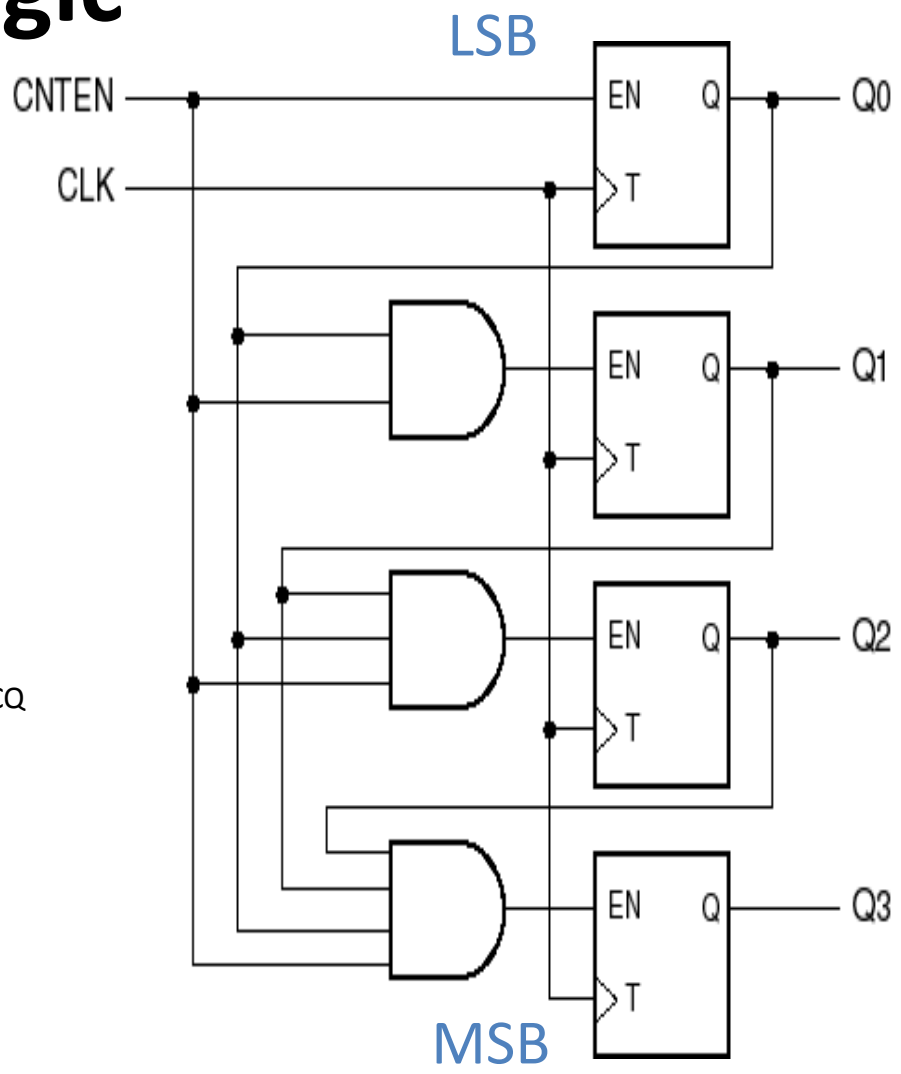
Synchronous Counter: Serial Enable Logic

1. Flip-flops enabled when all lower flip-flops = 1.
2. Enable propagates **serially** — limits speed. Requires: **$(n-1) \cdot t < T_{CLK}$**
3. All flip-flop outputs change simultaneously t_{CQ} after CLK
4. Most Frequently Used Type of Counter

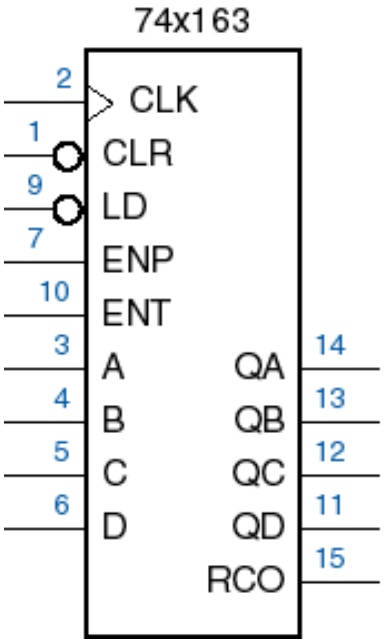


Synchronous Counter: Parallel Enable Logic

1. Single-level enable logic per flip-flop
2. Fastest and most complex type of counter. Requires $\mathbf{D\,t < T_{CLK}}$
3. All outputs change simultaneously t_{CQ} after CLK

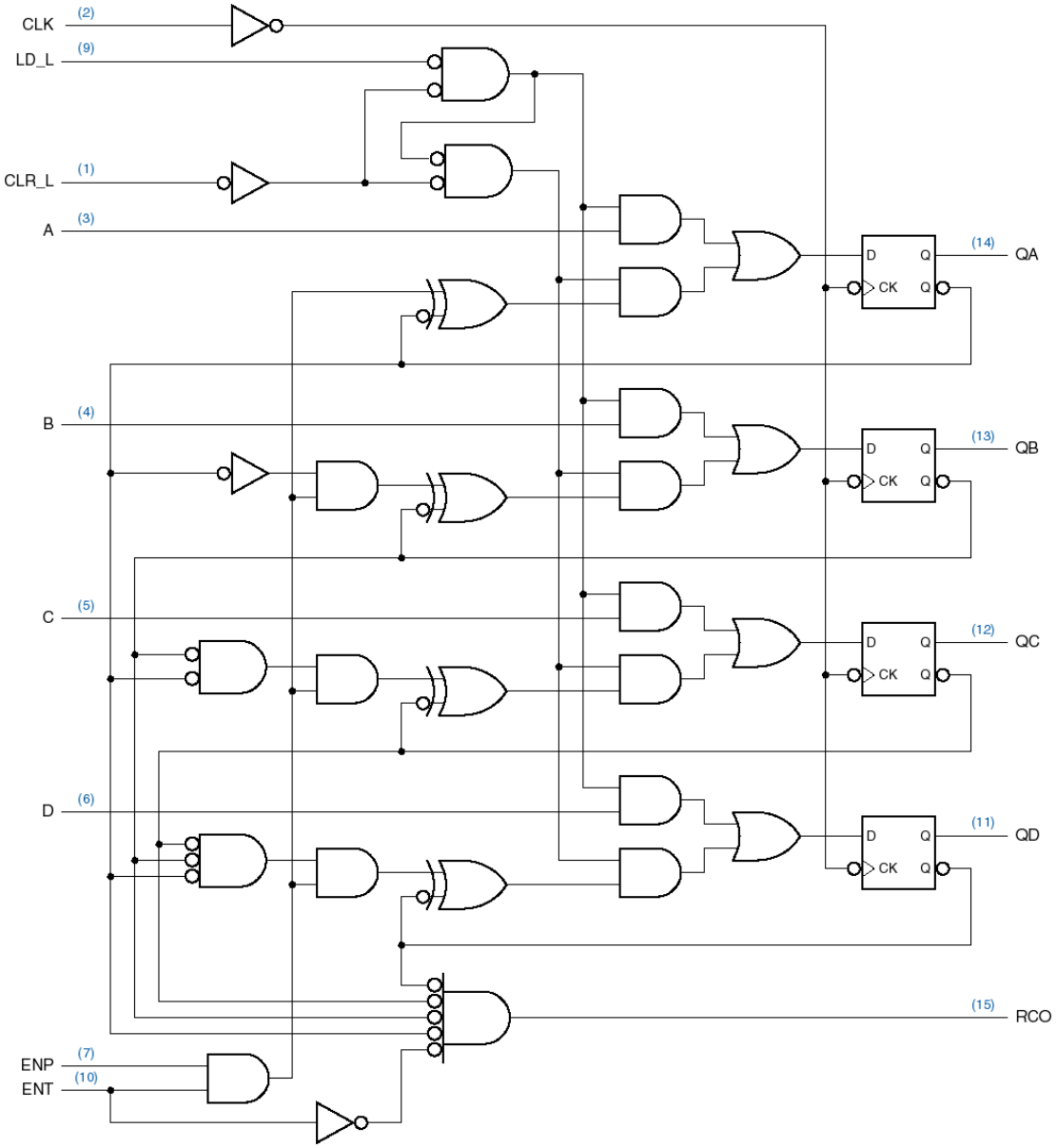


74x163
4-bit Counter

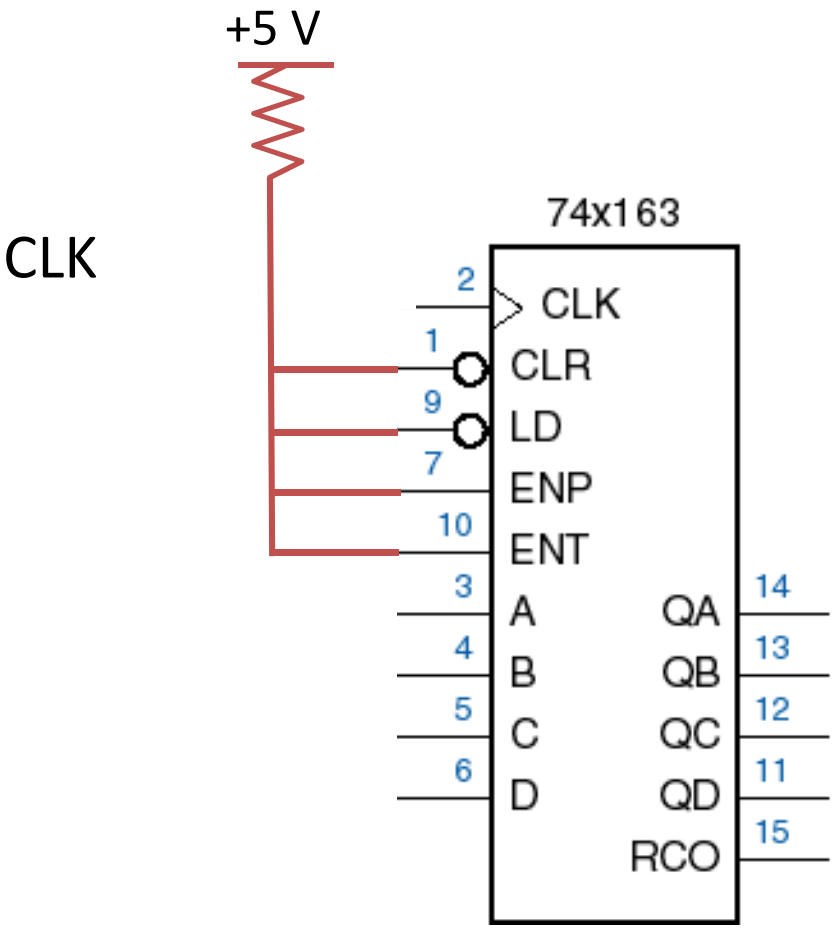


Inputs				Current State				Next State			
CLR_L	LD_L	ENT	ENP	QD	QC	QB	QA	QD*	QC*	QB*	QA*
0	x	x	x	x	x	x	x	0	0	0	0
1	0	x	x	x	x	x	x	D	C	B	A
1	1	0	x	x	x	x	x	QD	QC	QB	QA
1	1	x	0	x	x	x	x	QD	QC	QB	QA
1	1	1	1	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	1	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	1
1	1	1	1	0	0	1	1	0	1	0	0
1	1	1	1	0	1	0	0	0	1	0	1
1	1	1	1	0	1	0	1	0	1	1	0
1	1	1	1	0	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0	1	0	0	1
1	1	1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0

74x163
4-bit Counter

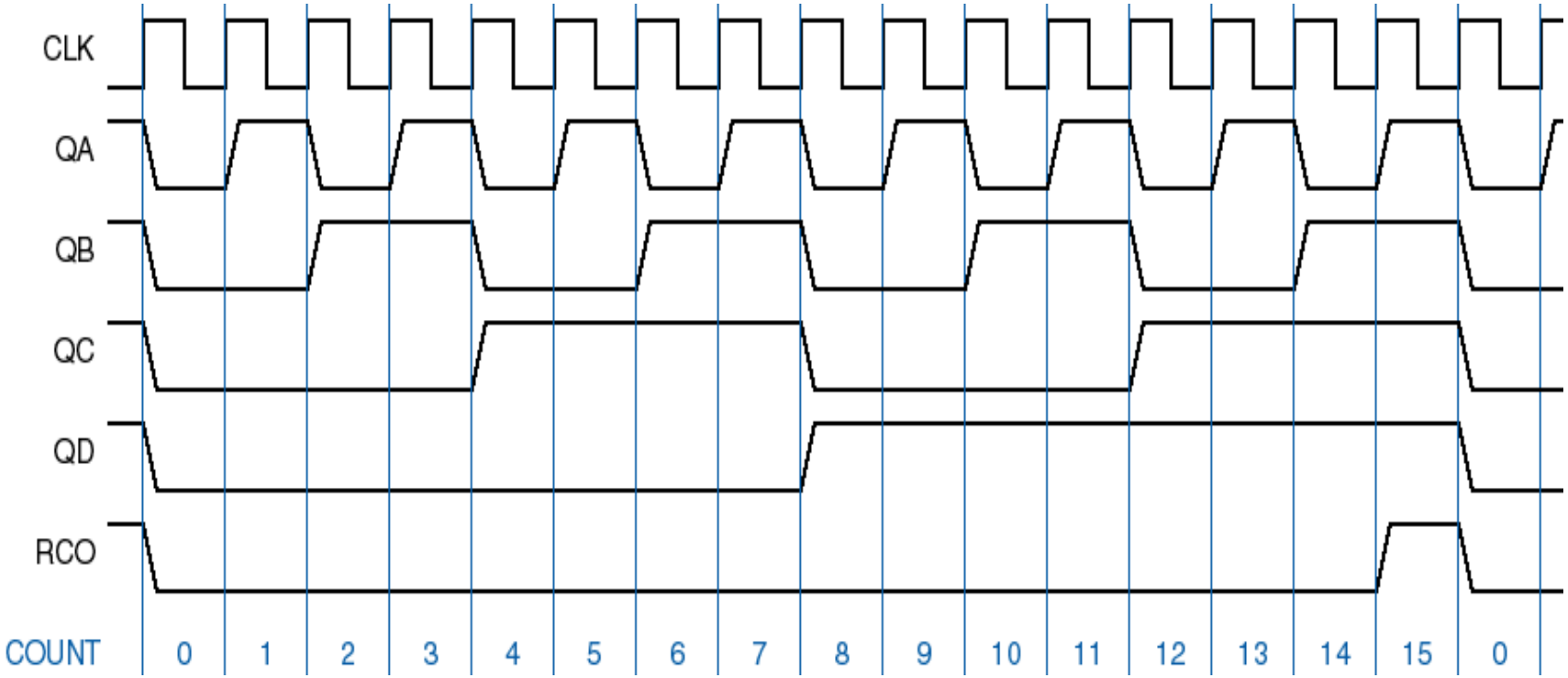


Counter Application-1:
Free Running Modulo-16 Counter



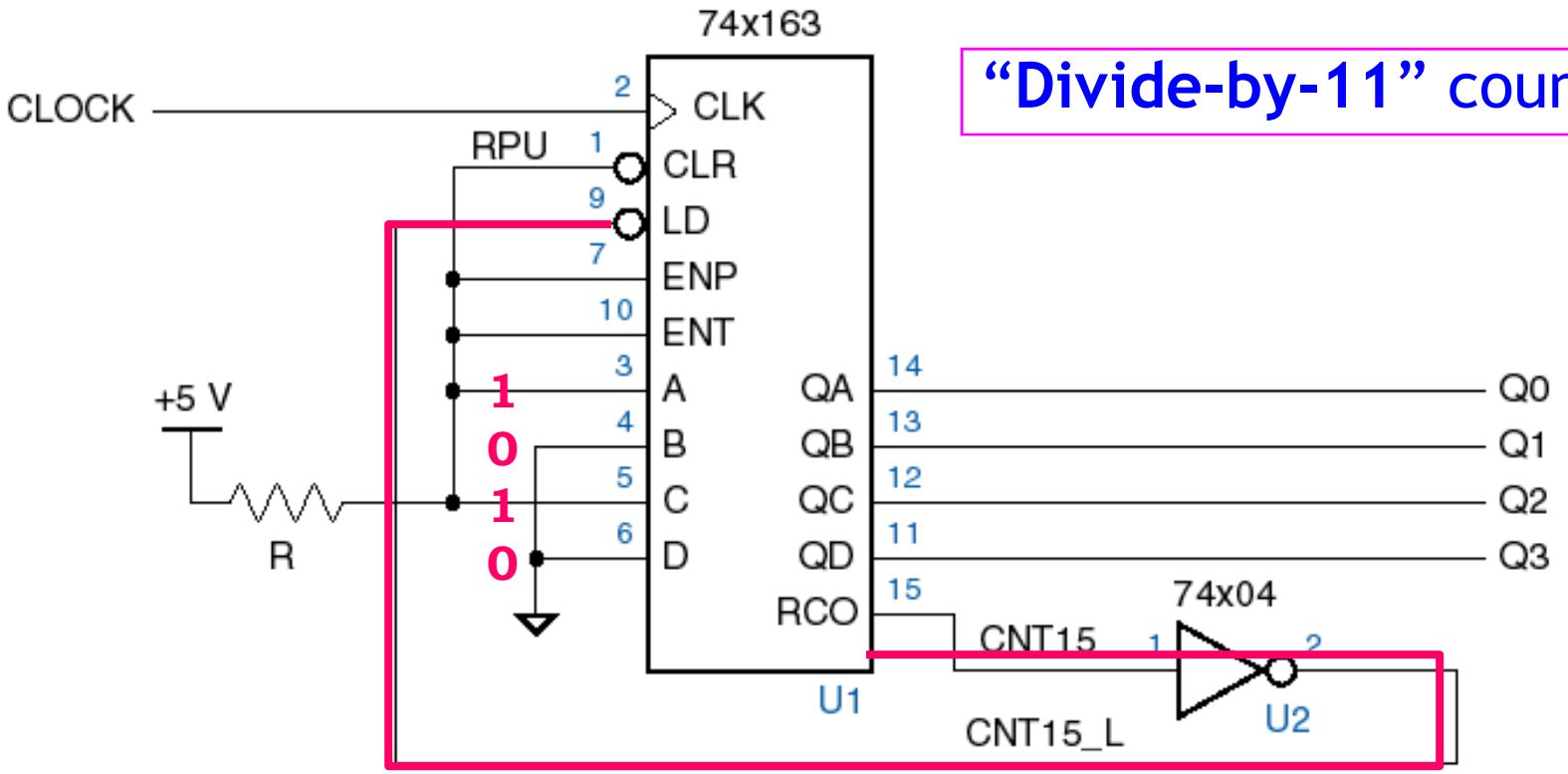
0, 1, 2, 3, 4, …, 15, 0, 1, …

Counter Application-1:



"Divide-by-16" counter

Counter Application-2: Modified Counting Sequence



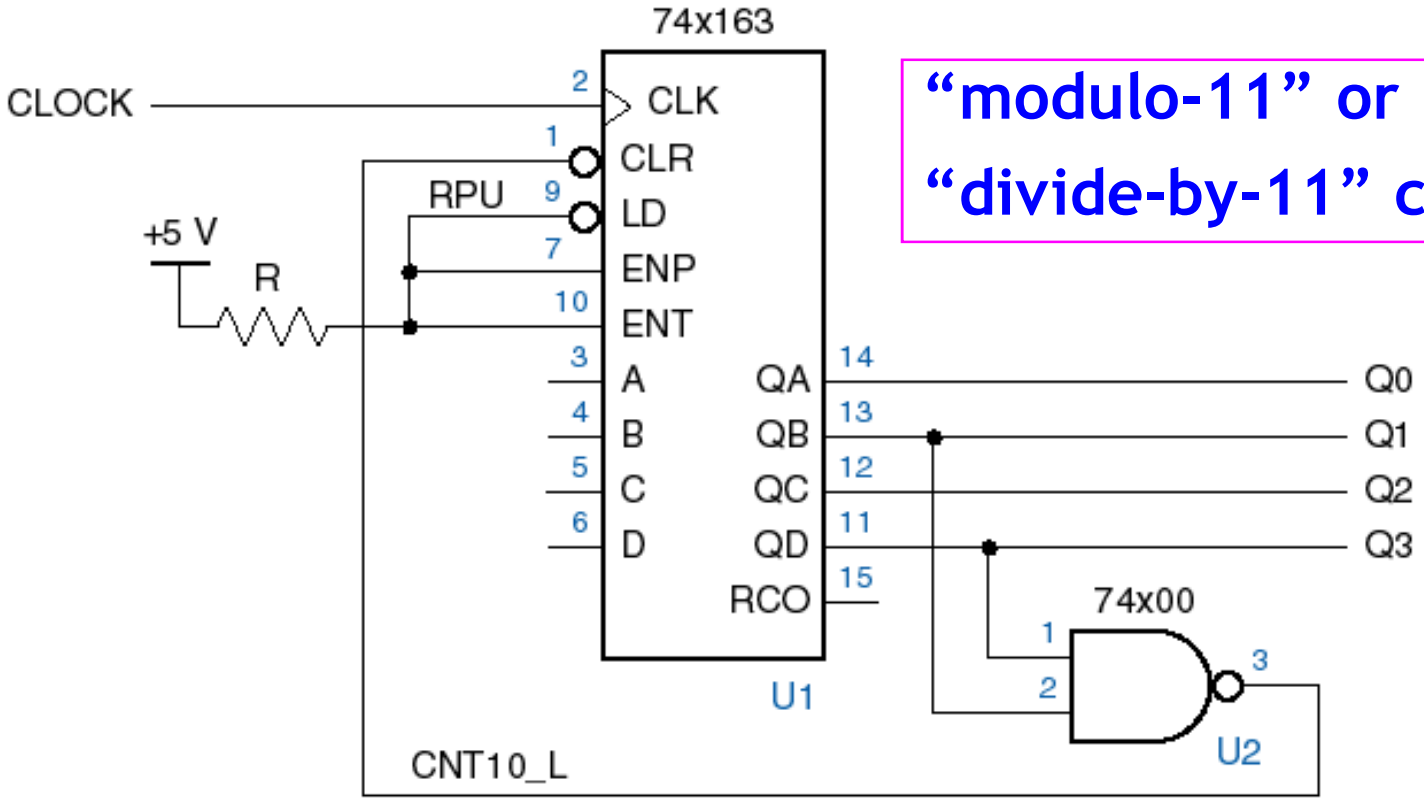
"Divide-by-11" counter

Load 0101 (5) after Count = 15

5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 5, 6, …

Counter Application-2: Modified counting sequence

Another Way

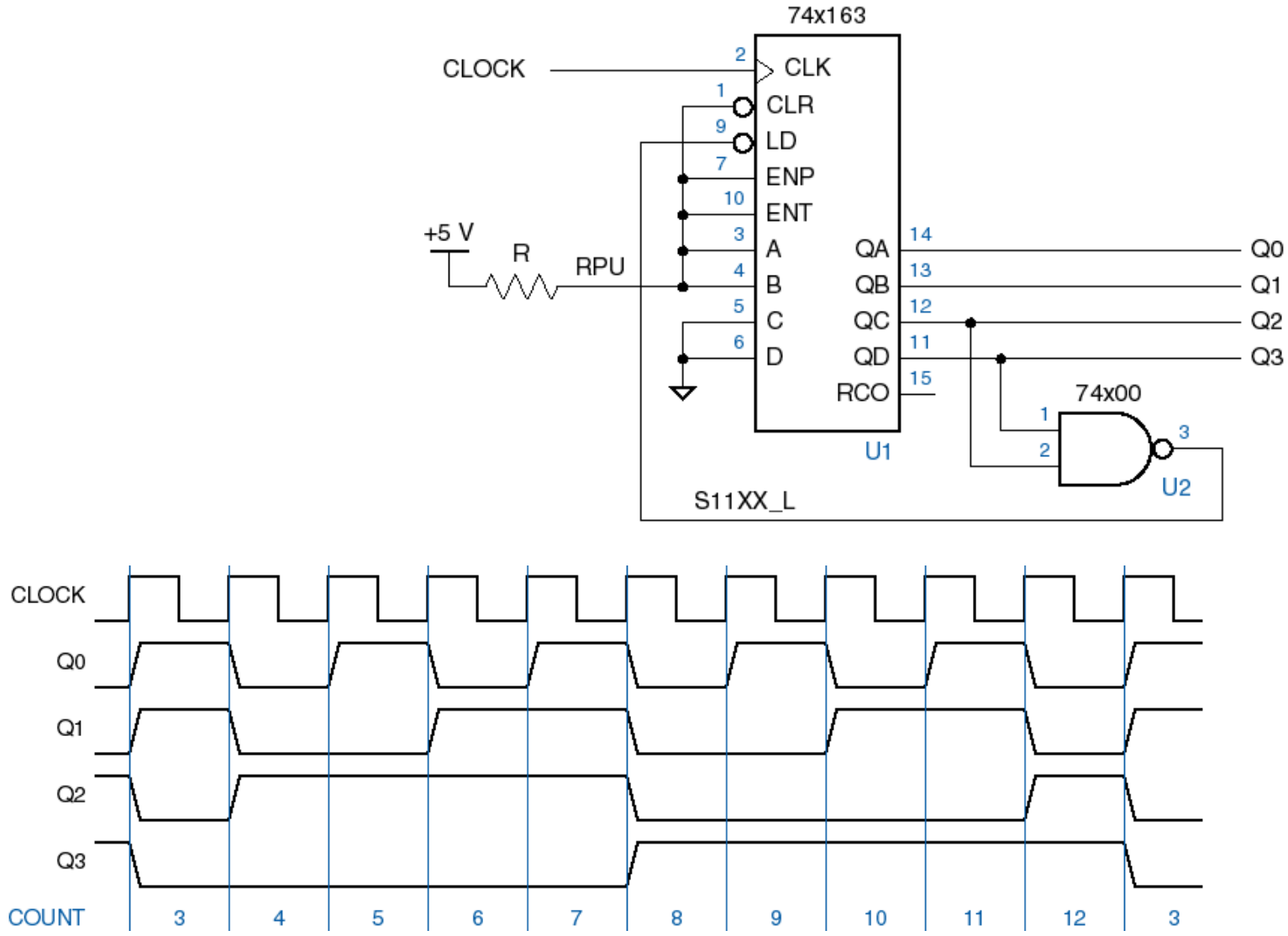


"modulo-11" or
"divide-by-11" counter

Clear after Count = 1010 (10)

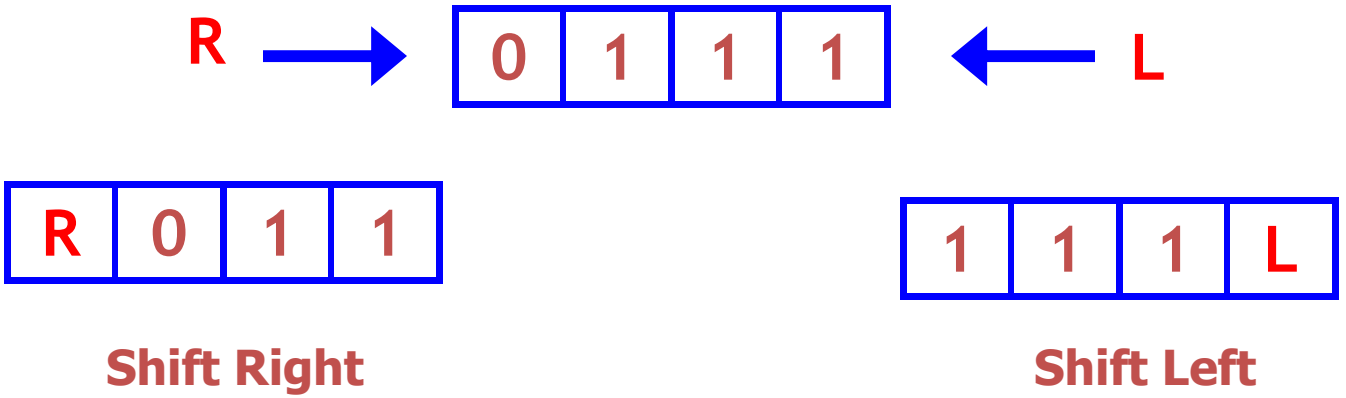
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 1, 2, 3, …

Counter Application-3: Counting from 3 to 12



Shift Registers

Multi-bit register that moves data “sideways” left/right (1 bit/clock)

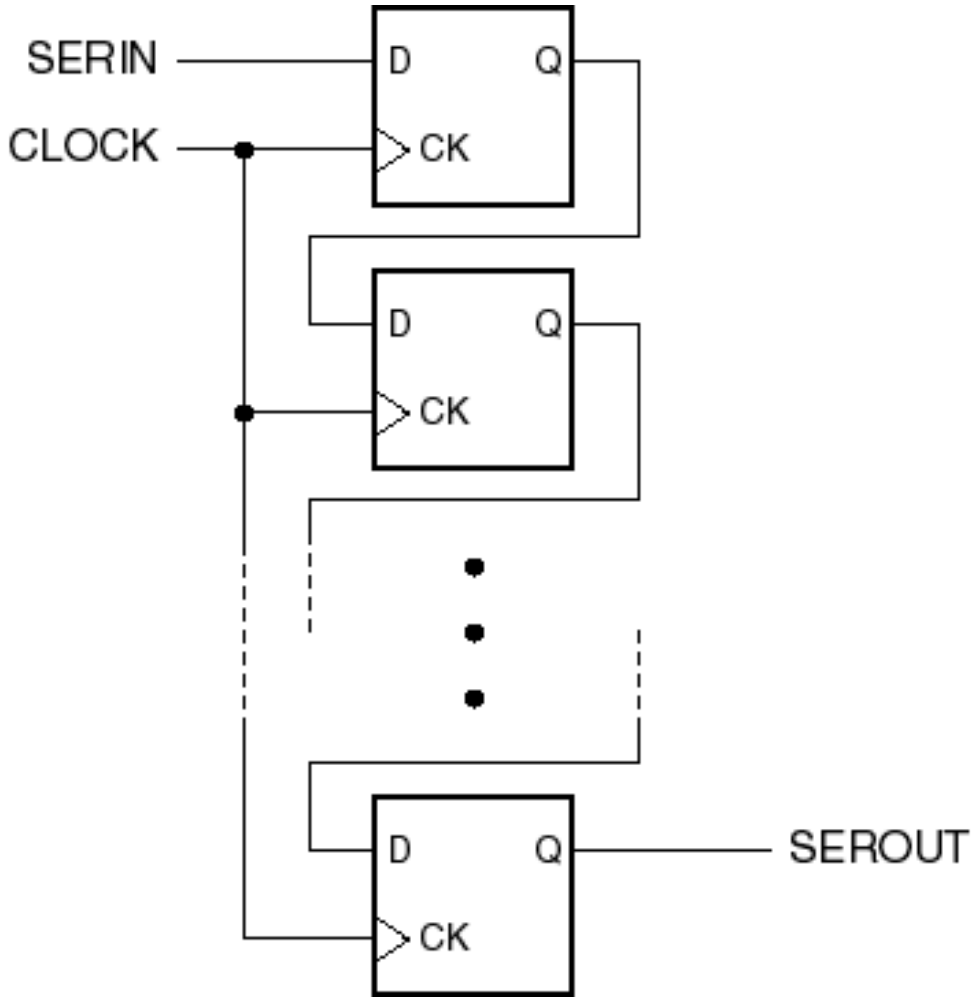


To rearrange bits or **Multiply/Divide by 2**

Shift Registers: Serial-in, Serial-out

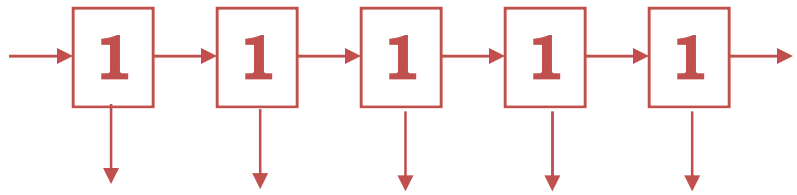
For handling serial data, e.g.

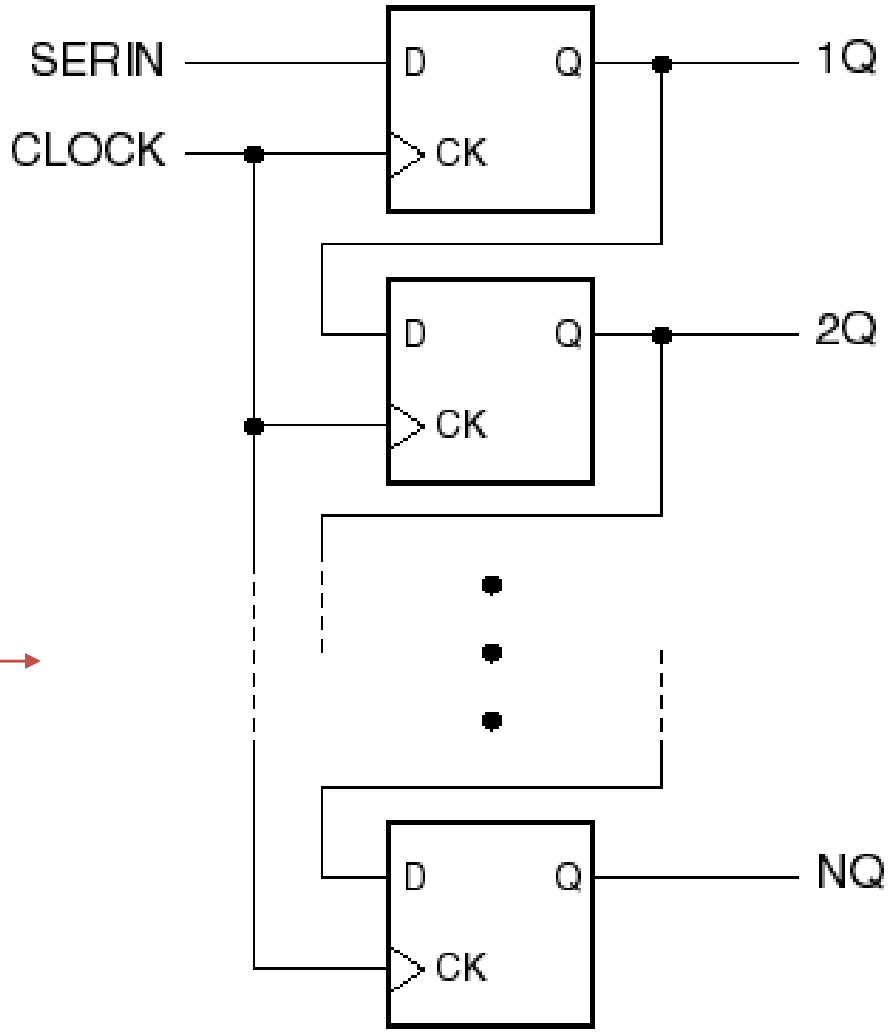
- RS-232 and
- modem transmission,
- Ethernet links,
- SONET,
- etc.



Serial-to-Parallel Conversion

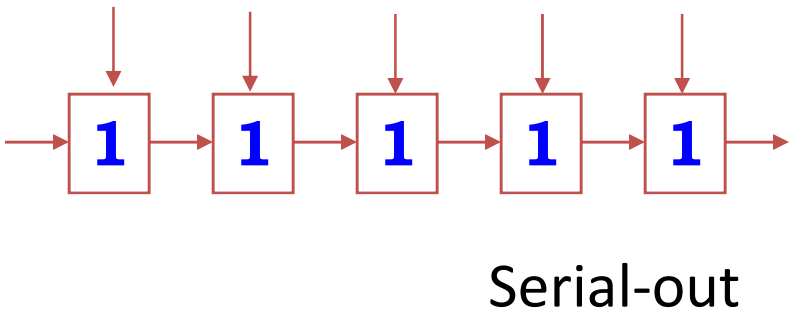
Serial-In, Parallel-Out shift register

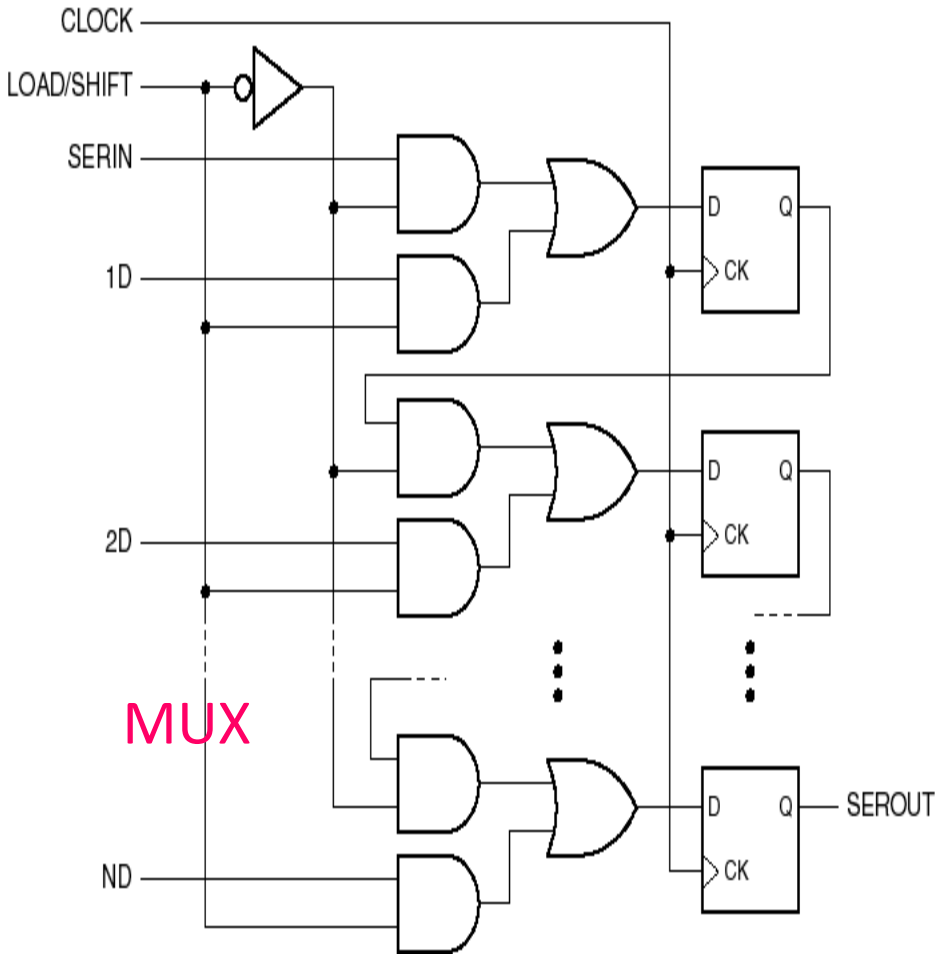




Parallel-to-Serial Conversion

Parallel-In, Serial-Out shift register

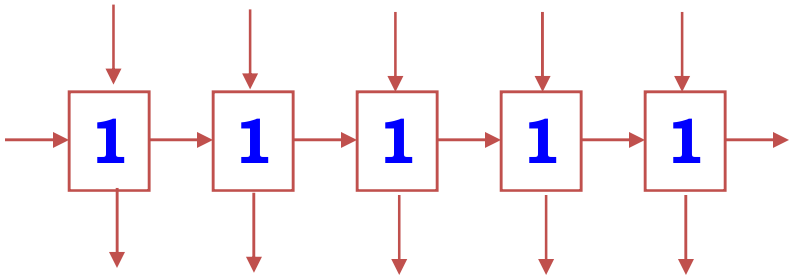




Do both: Parallel-in, Parallel-out Shift Register

Parallel-In, Parallel-Out shift register

Parallel-in



Parallel-out