14:332:231
DIGITAL LOGIC DESIGN

Ivan Marsic, Rutgers University
Electrical & Computer Engineering
Fall 2013

Lecture #20: Sequential Logic Design Practices ; Counters

Sequential Circuit – Timing Diagram

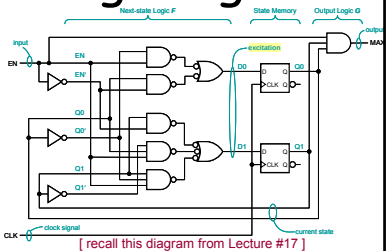
- To function correctly, we must have:

$$t_{clk} - t_{ffpd(max)} - t_{comb(max)} - t_{setup} > 0$$

setup-time margin

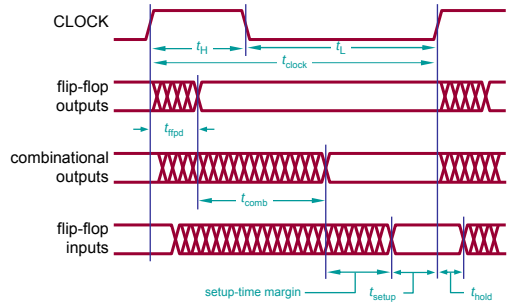
$$t_{ffpd(min)} - t_{comb(min)} - t_{hold} > 0$$

hold-time margin



[recall this diagram from Lecture #17]

XXXX ← Crosshatching indicates when change can happen:



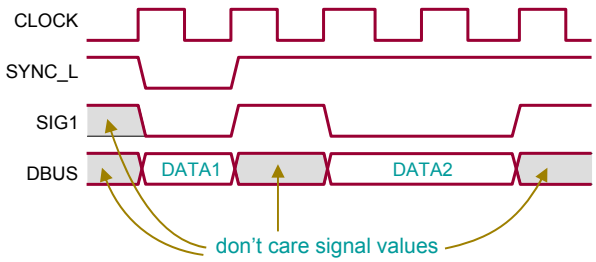
state of flip-flops t_{ffpd} →

excitation logic t_{comb} →

next state input needs ≥ t_{setup} →

Sequential Circuit – Functional Timing

- We have to find the *longest delays* for each part
- *Typical* & *longest delays* are given for the circuit by the manufacturer
  - For some latches & flip-flops, the timing parameters are in the book, Table 8-1 on pages 684–685
- Te diagram shows only the *functional behavior*
  - qualitative relationships, not actual delay quantities:

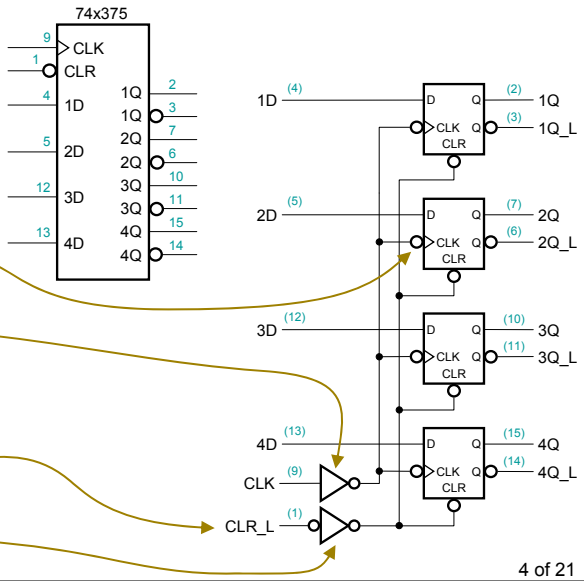


- Propagation, setup and hold times are not shown

Multibit Registers and Latches

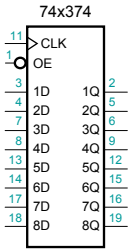
Wakerly, 4th Ed., Section 8.2.5, page 691

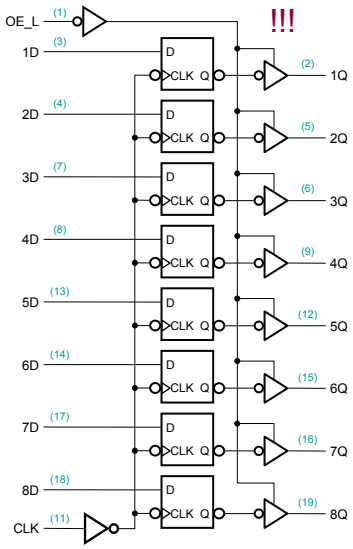
- **74x175**
  4-bit register
- **Register** = collection of ≥2 D flip-flops with a common clock input
- Note negative edge triggered flip-flops
- But the *external* CLK has an inverter → positive-edge triggered w.r.t. external CLK input pin
- Asynchronous CLR_L
- CLK & CLR_L buffered before fanning out



8-bit (octal) Register: 74x374

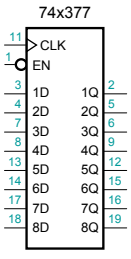
- 74x374
  - Eight positive-edge-triggered D flip-flops
  - 3-state output buffer drives active-high output
  - Common active-low OE_L (output enable) input

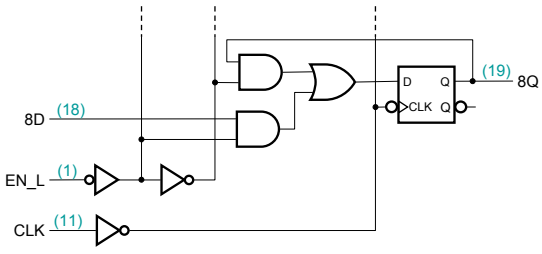




8-bit (octal) Register: 74x377

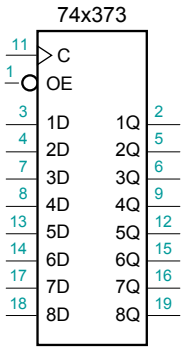
- 74x377
  - Similar to 74x374, but does not have 3-state output
  - Clock enable input EN_L





Octal Latch

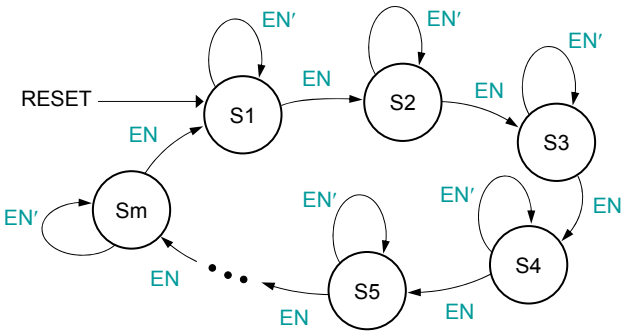
- 74x373 D *latches*
- Output enable when C is asserted *and* OE_L commands the three-state output (look up 74x374 above)



- Register vs. latch, what's the difference?
  - Register: edge-triggered behavior
  - Latch: output follows input when C is asserted

Counters

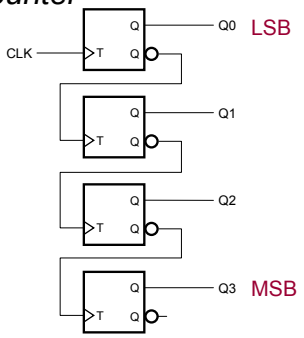
- **Counter**: Any sequential circuit for which the state diagram is a single cycle



- A counter with m-states is called a modulo-*m*-counter, or sometimes a divide-by-*m* counter
- Counters can count up, count down, or count through other fixed sequences

Ripple Counter

- T flip-flops are used in simple counters
- The simplest solution is the *ripple counter*
- 4-bit binary ripple counter:
  - It doesn't have EN!
  - Clock is connected to flip-flop clock input on the LSB bit flip-flop
  - For all other bits, a flip-flop output is connected to the clock input → circuit is *not* truly synchronous
  - Output change is delayed more for each bit toward the MSB bit
  - Resurgent because of low power consumption

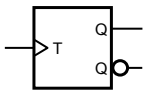


time = n · t_{TQ}

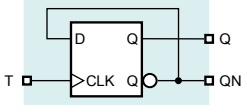
- Synchronous counters are faster …
  - The operation of all flip-flops is synchronized by a common clock

[Recall Lecture #16] T (toggle) Flip-flop

- Changes state at every clock tick
  - Signal on the Q output has frequency = ½ T
- Used in counters and frequency dividers
- Can be constructed from D flip-flops



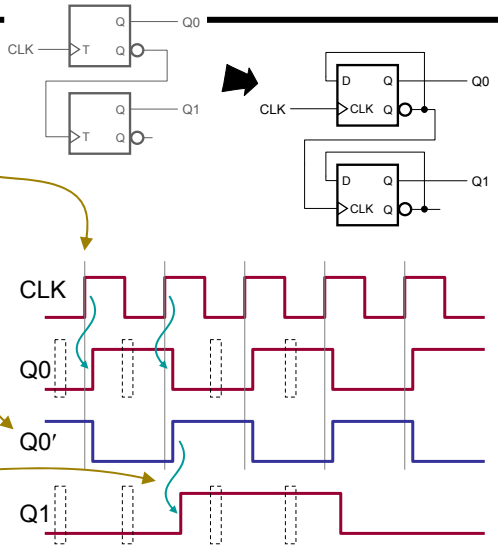
D flip-flop:



Next-state equation: $Q^* = Q'$

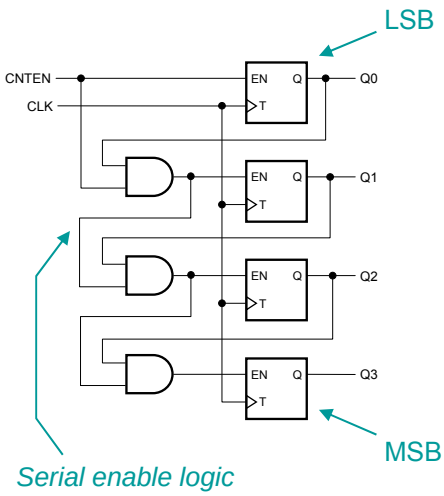
How Ripple Counter Works

- When there is a positive edge on the clock input of Q0, Q0 complements (toggles)
- The clock input for flip-flop Q1 is the complemented output of the first flip-flop, Q0′
- When flip Q0 changes from “1” to “0”, there is a positive edge on the clock input of Q1 causing Q1 to complement



Synchronous Serial Counter

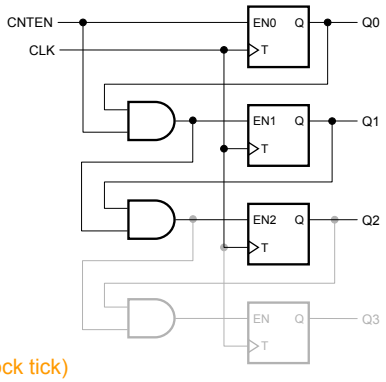
- Called so b/c combined enable signals propagate serially from LSB to MSB
- Master count-enable CNTEN
  - T flip-flop toggles if-and-only-if CNTEN=1 *and* all lower-order counter bits are “1”
  - Fixed amount of logic per bit
- If CLK period too short, the counter doesn’t count
  - Not enough time for a change in LSB to propagate to MSB



Synchronous Serial Counter

State table for binary counter:

Current State			Next State			Flip-Flop Inputs		
Q2	Q1	Q0	Q2*	Q1*	Q0*	EN2	EN1	EN0
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



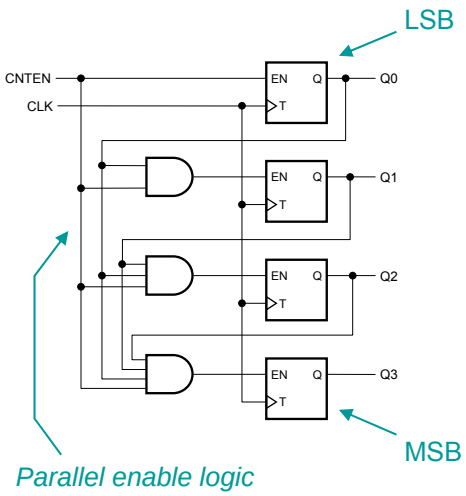
Q0 toggles always (every clock tick)

Q1 toggles every time Q0 = 1

Q2 toggles every time Q1 and Q0 are both 1

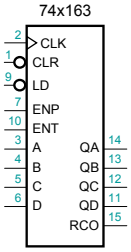
Synchronous Parallel Counter

- Eliminates the problem w/ Synchronous Serial Counter
  - Replace AND carry chain with ANDs in parallel
    - Each EN input driven w/ a dedicated AND gate—just a single level of logic
- Advantages:
  - Reduces path delays
  - Called “parallel gating”
  - Like carry lookahead
- The fastest binary counter



74x163 MSI 4-bit Counter

- Most popular MSI counter
- Synchronous CLR_L & LD_L ("Load")
- Synchronous parallel count enable ENP & ENT
- ENT acts also on RCO ("ripple carry out")
  - Indicates a carry from MSB



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

Logic Diagram for 74x163

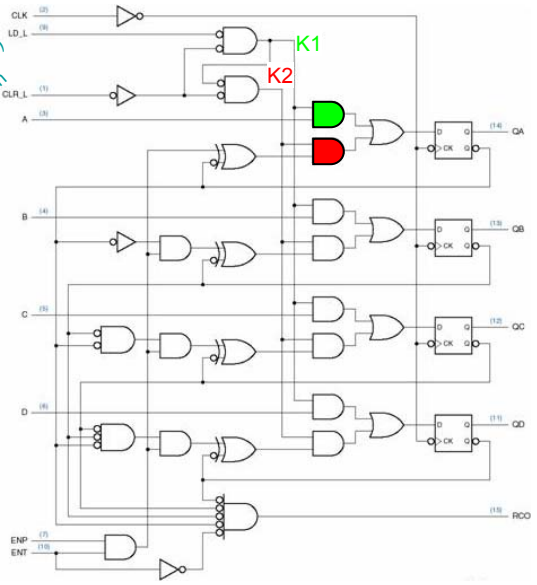
- D flip-flops, to have easier *synchronous* clear and load than T flip-flop
  Q*=EN⊕Q

pass data input (A,B,C,D)
do counting function

CLR_L	LD_L	K1	K2
1	1	0	1
1	0	1	0
0	1	0	0
0	0	0	0

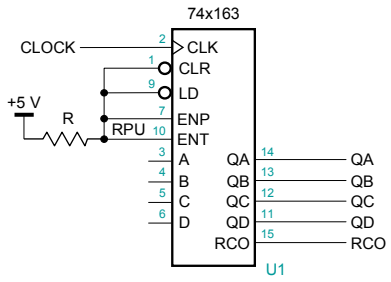
+++

- ENT, ENP are "1" to count on bits QA to QD
- RCO ("ripple carry out"), when ENT is asserted



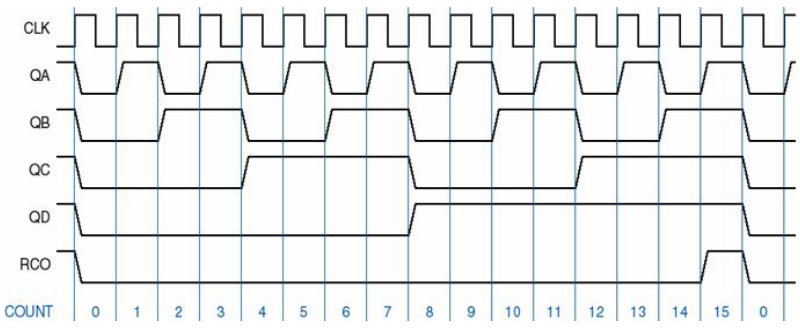
Free-running Counter Operation

- "*Free-running*" == enable inputs enabled continuously
- Count if ENP and ENT both asserted and
  - Load if LD_L is asserted [=0] (overrides counting)
  - Clear if CLR_L is asserted [=0] (overrides loading and counting)
- All operations take place on rising CLK edge
- RCO is asserted if ENT is asserted and count = 15



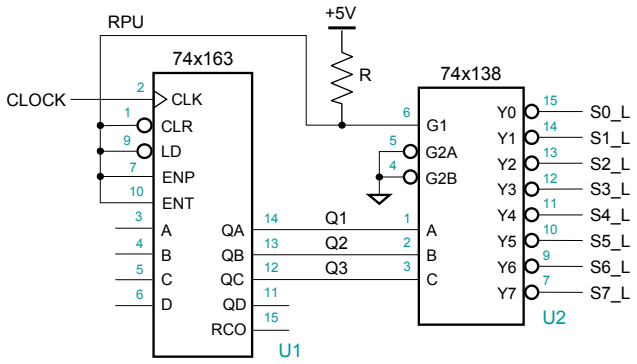
Free-running 4-bit 74x163 Counter

- Timing diagram for a free-running "divide-by-16" counter
- QD is the MSB and QA is the LSB
- From QA on, each signal has ½ frequency of preceding one
- ➔ can be used as *divide-by-2, -4, -8, or -16* counter



Decoding Binary-counter States

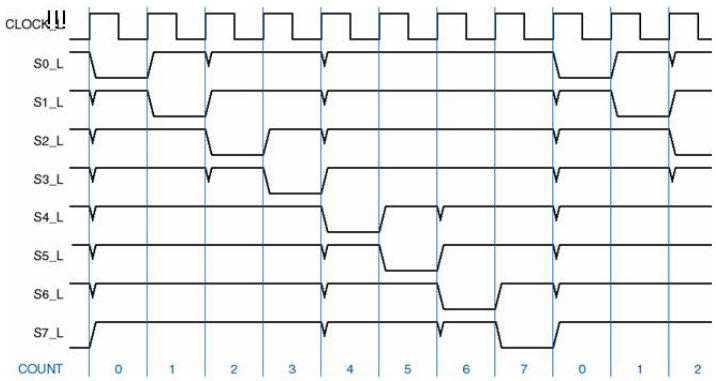
- A modulo-8 counter and decoder combined for a set of 1-out-of-*m*-coded signals, each representing a counter state
- Used for controlling a set of devices, based on counter state → each output enables different device



- but it has limitations due to a *function hazard* …

Modulo-8 Counter/Decoder Timing Diagram

- Glitches on state transitions in 74x138 (*function hazard*)
  - The outputs of the counter do not change at exactly the same time
  - Signal paths in the decoder 74x138 have different delays



Glitch-free Outputs

- To achieve the same function of a mod-8 binary counter and decoder, but remove glitches on outputs:
  - Connect decoder 74x138 outputs to a register (74x374) that samples the enable-decoded outputs on the next clock tick
  - 74x374 is an 8-bit register with OE_L three-state output
- Registered outputs delayed by one clock tick
- Alternative solution: use an 8-bit “ring counter”