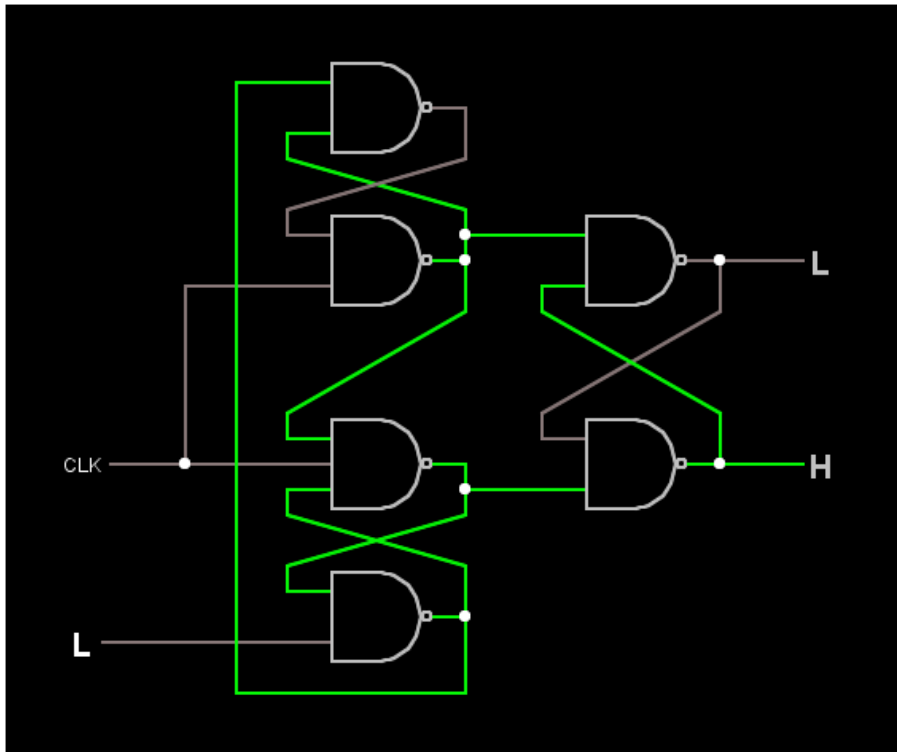
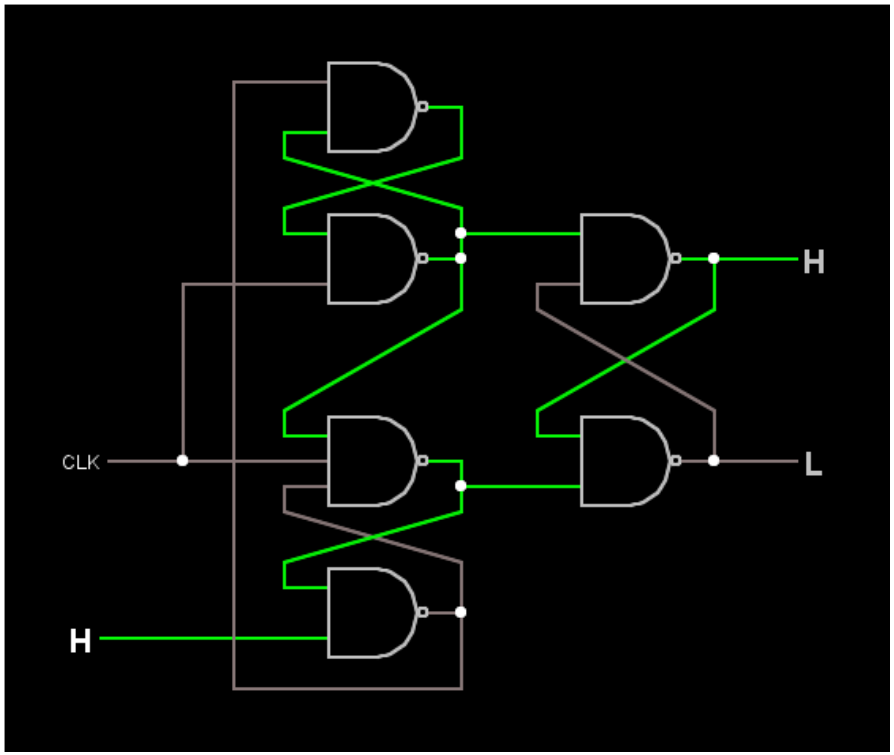
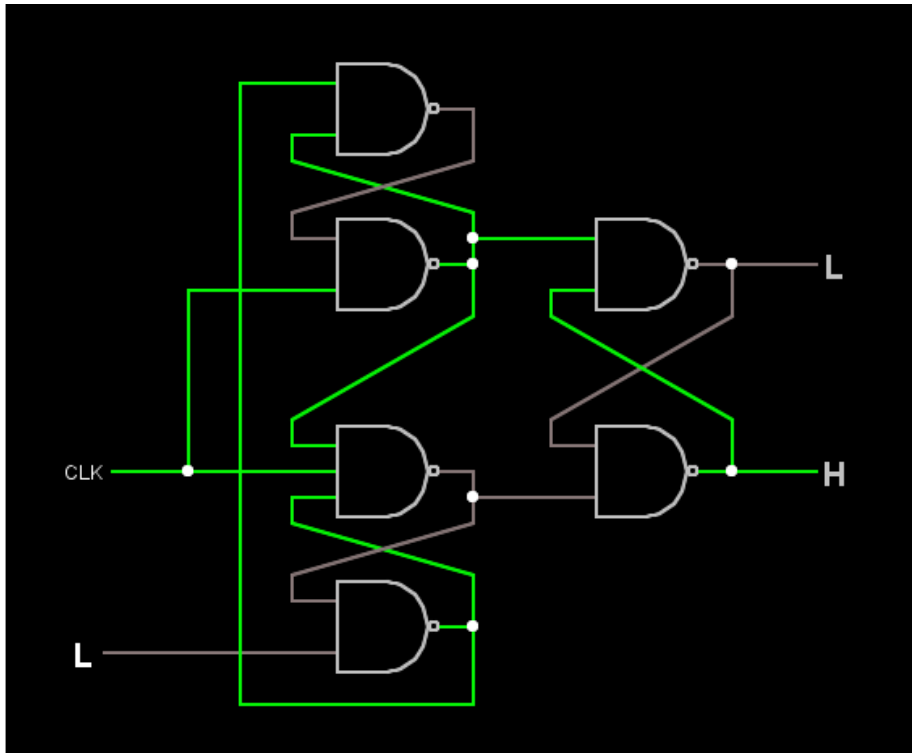
Edge-triggered flip-flop

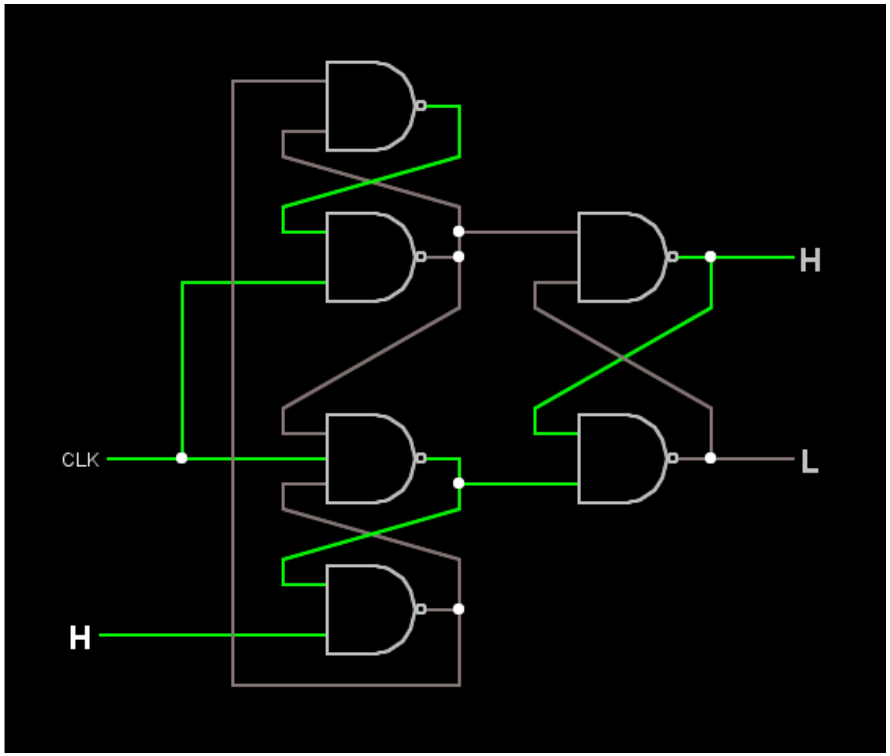




For both H and L inputs **and** CLK=LOW, right latch is kept in memory state

CLK → HI allows data to pass

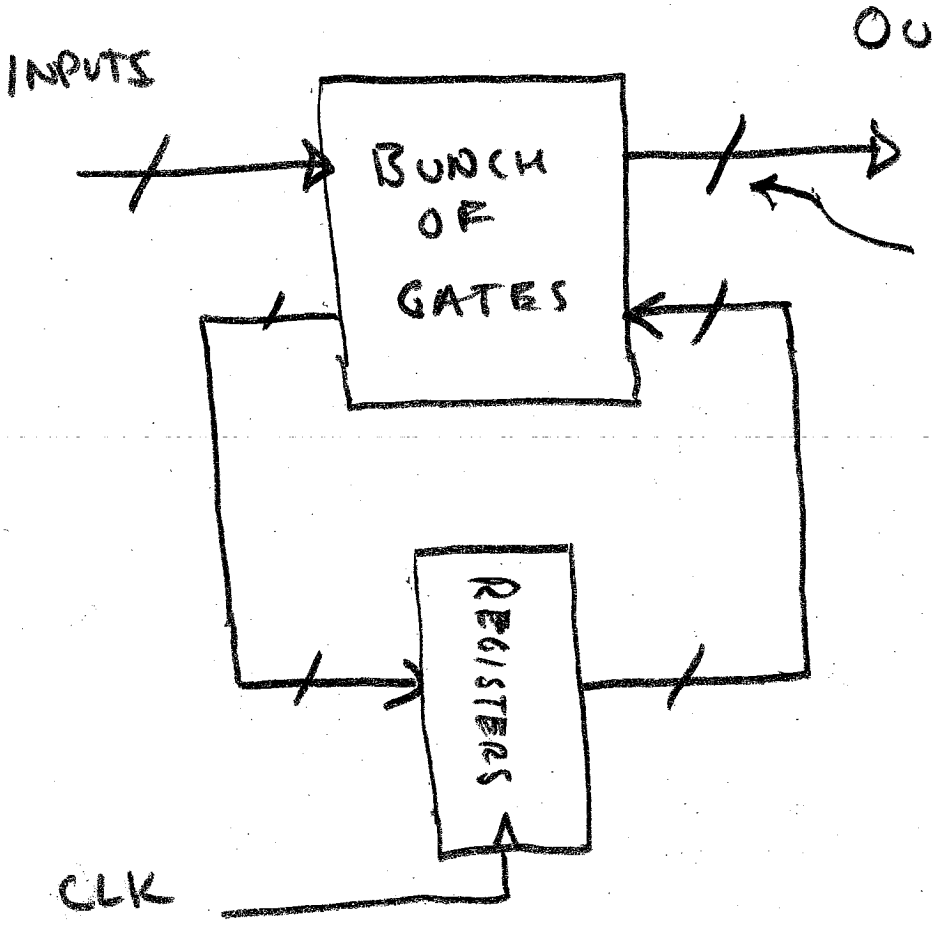




Even if data input (D) changes state after CLK goes HI, it does not propagate.

COMPUTER FIRST STEPS

REGISTERS + GATES



SLASH MEANS "MULTIPLE"

ON EACH CLOCK EDGE STATE OF REGISTERS CHANGES ACCORDING TO LOGIC OF GATES, PREVIOUS STATE, AND ANY NEW INPUT.

BASIC IDEA OF DIGITAL COMPUTER!!

FUNDAMENTAL NOTION: "STATE MACHINE"

SILLY EXAMPLE (FROM MEH)

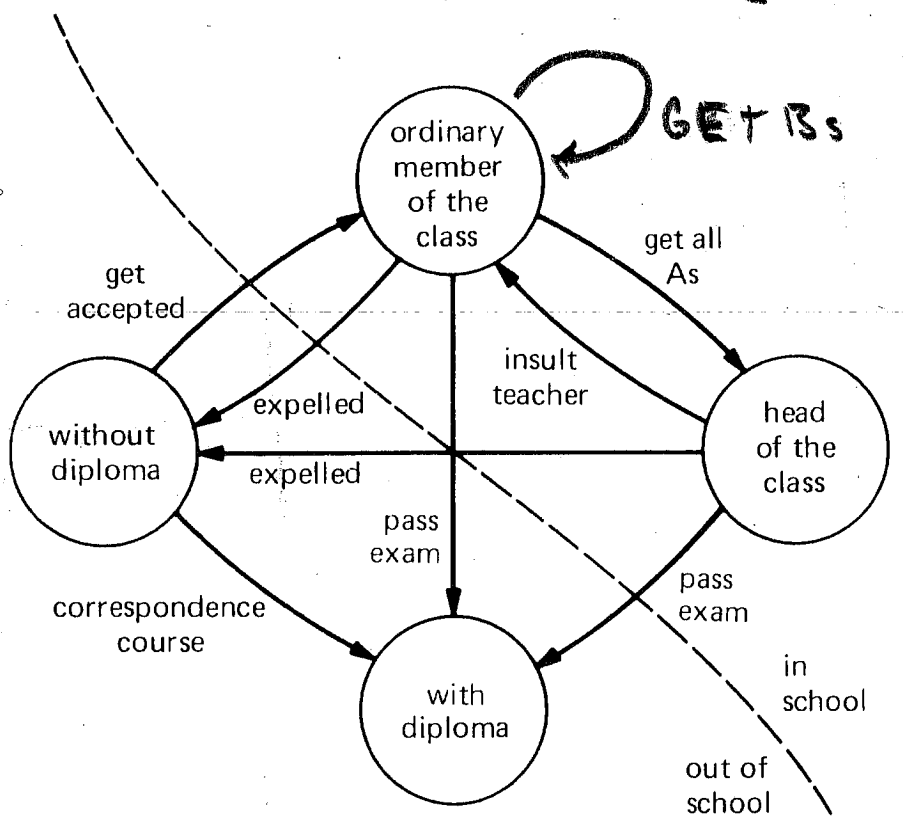


Figure 8.61. State diagram: going to school.

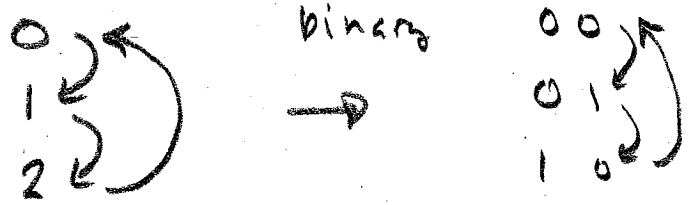
STATES = CIRCLES

CAUSES = ARROWS

EACH STATE REMAINS UNTIL CLOCK (END OF TERM) OR OTHER INTERRUPT (INSULT TEACHER)

STATE DIAGRAM AS DESIGN TOOL

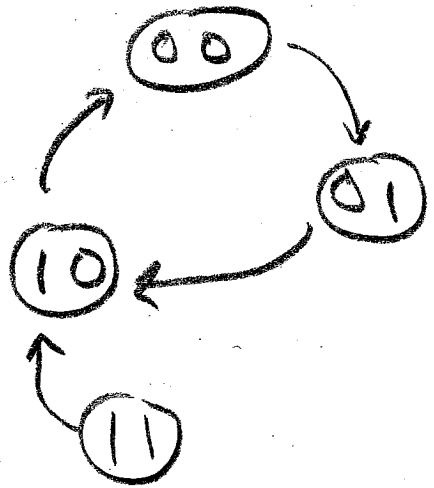
EXAMPLE: "DIVIDE-BY-3" SYNCHRONOUS COUNTER.



NEED 2 BITS

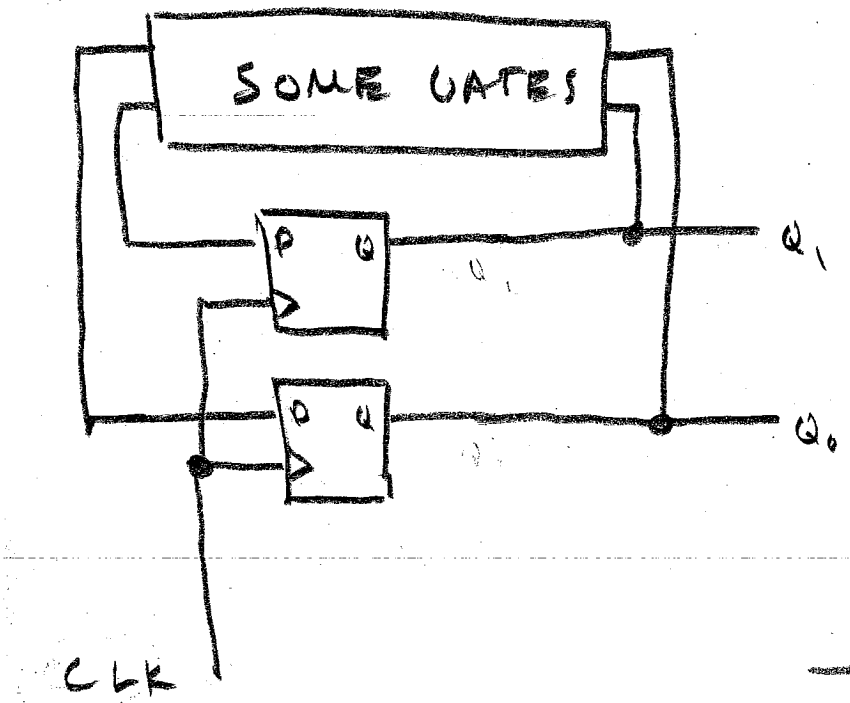
BUT 2 bits ⇒ 4 STATES

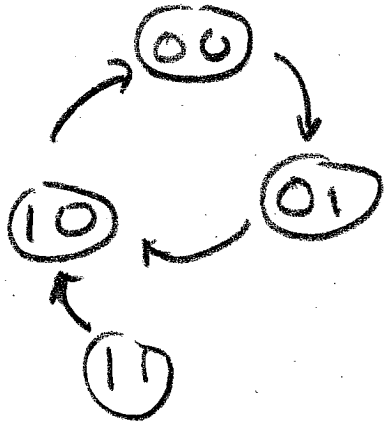
LET 11 STATE GO TO 10 STATE



STATE DIAGRAM FOR ÷3 COUNTER.

÷ 3 CONT.





COMBINATIONAL PROBLEM:

GIVEN POSSIBLE OUTPUTS

WANT GATES TO PRESENT

NEXT STATE TO INPUTS

Qs are INPUTS / Ds are OUTPUTS

Q_1	Q_0	D_1	D_0
0	0	0	1
0	1	1	0
1	0	0	0
1	1	1	0

÷ 3 CONT

K-MAPS (FOR PRACTICE)

D_0

Q_1 \ Q_0	0	1
0	1	0
1	0	0

D_1

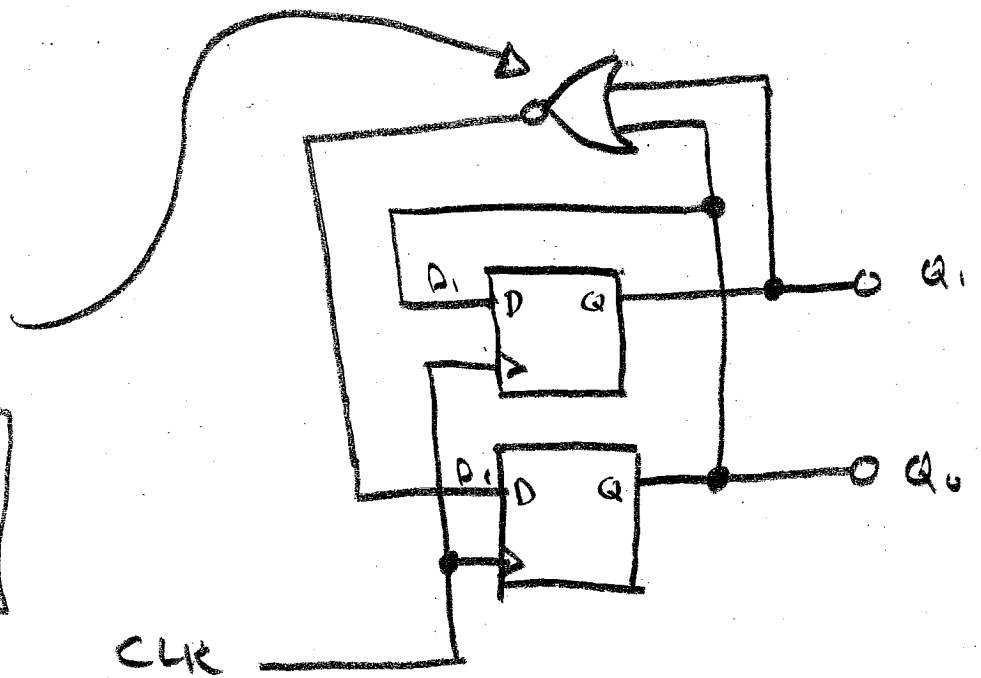
Q_1 \ Q_0	0	1
0	0	1
1	0	1

Q_0

So $D_0 = \overline{Q_0}\,\overline{Q_1}$ (NOR) $= \overline{(Q_0 + Q_1)}$

$D_1 = Q_0$

ONLY NEED ONE GATE



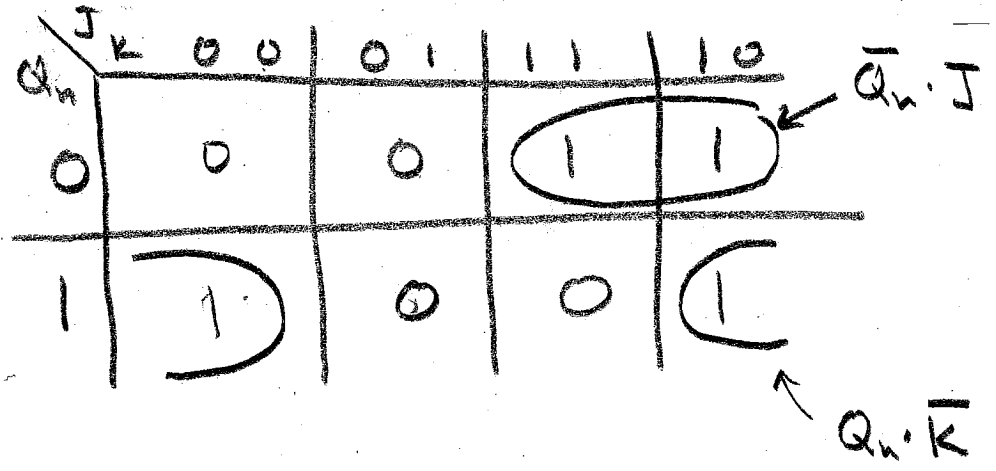
EXERCISE: DRAW TIMING DIAGRAM

A BIT MORE COMPLICATED? J-K FLOP FROM D-FLOP.

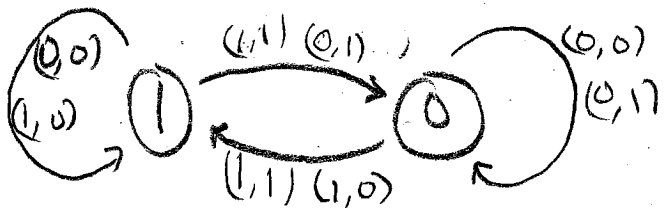
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

→ EXPAND TRUTH TABLE →

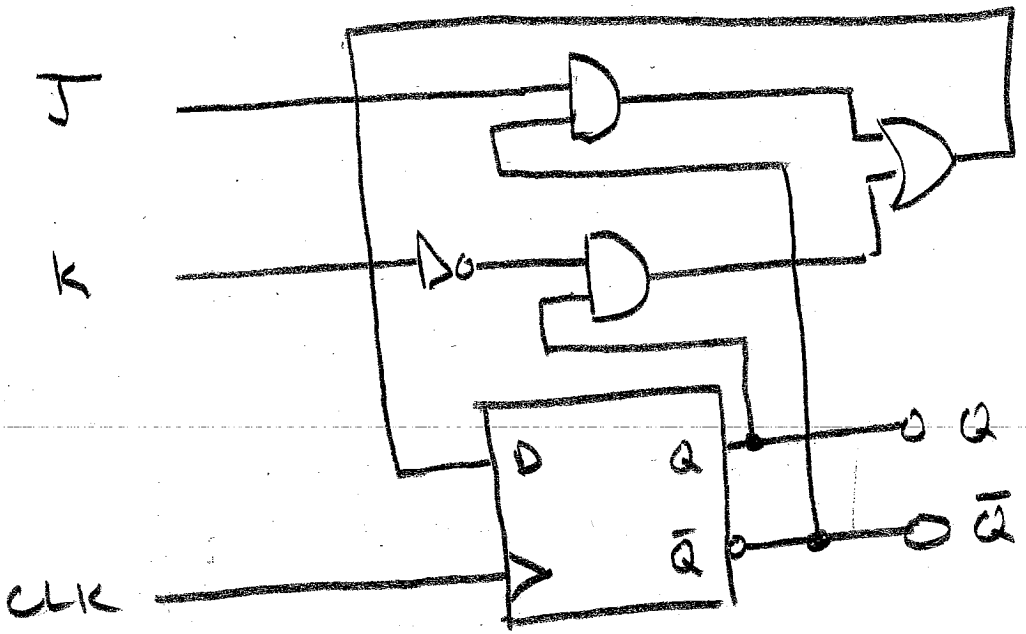
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$Q_{n+1} = \overline{Q_n} \cdot J + Q_n \cdot \overline{K}$$

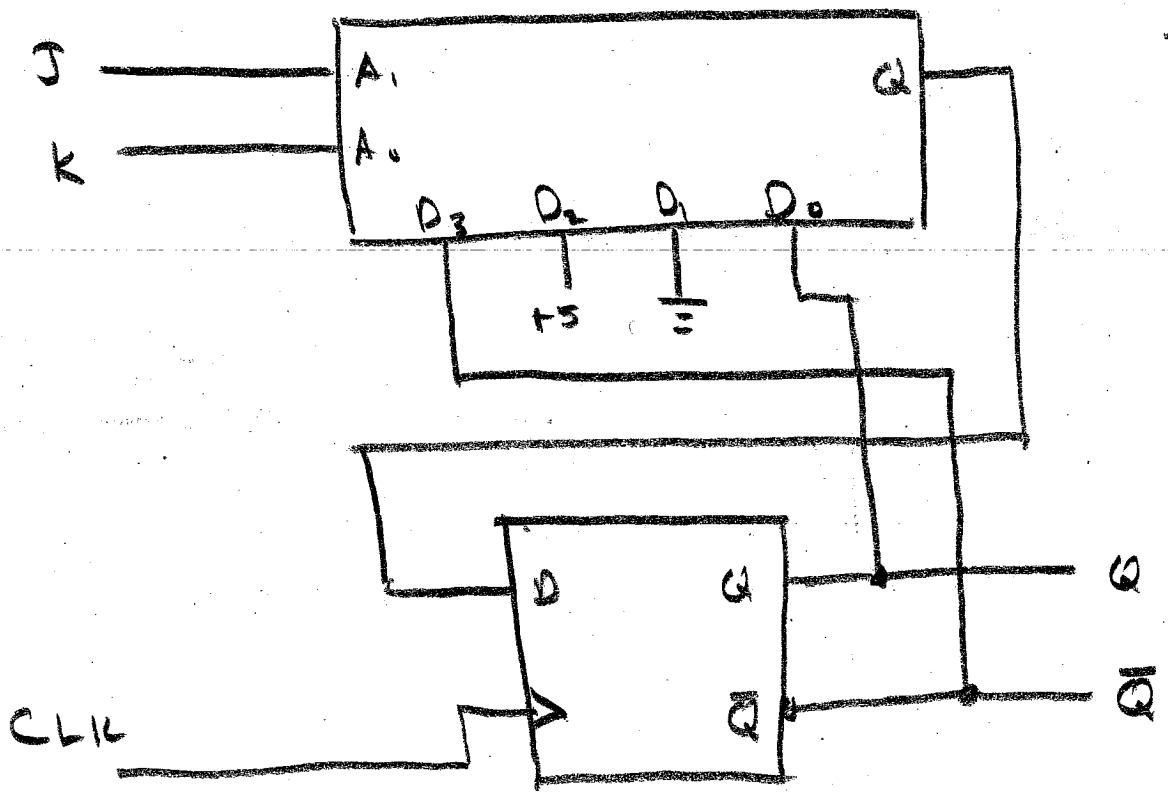


NOTICE: BECAUSE D-FLOPS HAVE BOTH Q, $\bar{Q}$ OUTPUTS, WE CAN USE TO ADVANTAGE



EXERCISE: MAKE WITH NANDS ONLY

HERE'S THE SAME THING USING 4-INPUT MUX



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

COUNTER CHIPS (NO ONE BUILDS COUNTERS FROM SCRATCH)

BCD - "BINARY-CODED DECIMAL"

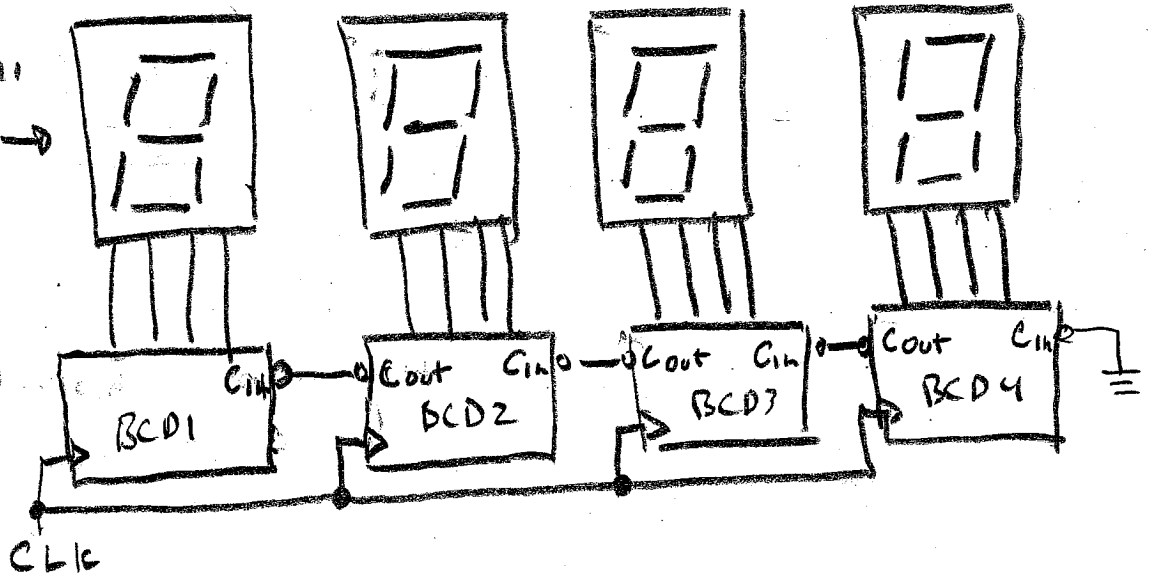
(ONLY COMPUTER GEEKS READ HEX, REST OF US READ BASE 10)

BCD → EACH DIGIT IS 0 - 9

0000 ... 1001 → CARRY

IDEA:

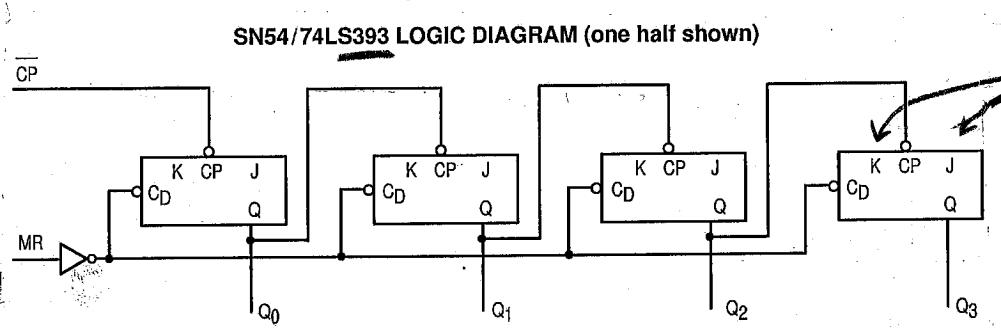
"7-SEGMENT" DISPLAYS →



LS 390 (BCD) vs LS 393 (HEX)

CLOCK

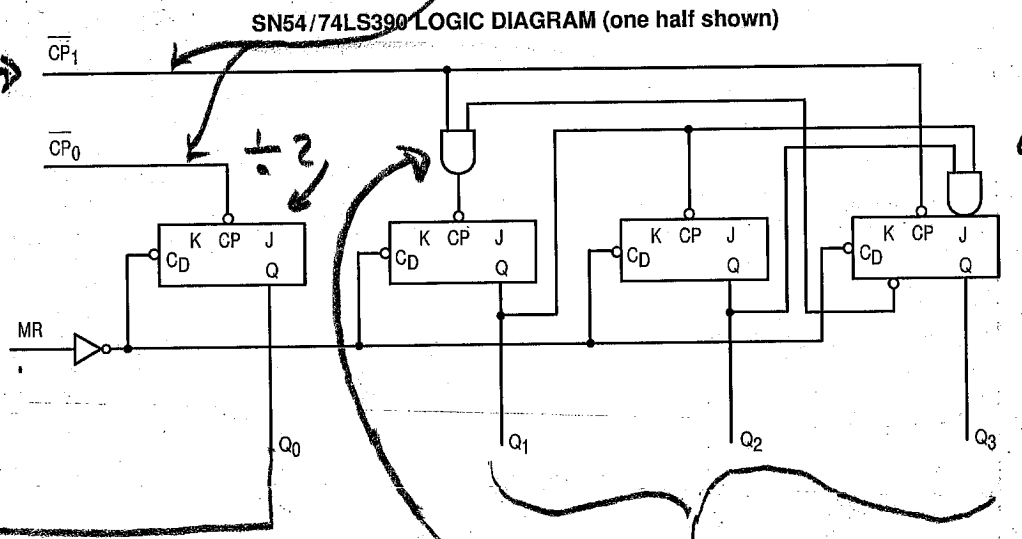
RESET LINE



J, k in toggle (H1) STATE (LS floats H1)

'393 HEX COUNTER – RIPPLE OR SYNCHRONOUS?

2 CLOCKS



For ÷10

J = 0, k = 1 ⇒ $Q_3 = 0$ ⇒ CP1 EN

÷5

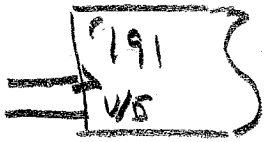
Q_3	Q_2	Q_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

'390 BCD COUNTER: ÷2 + ÷5

COUNTER CHIPS, CONTINUED

FEATURES

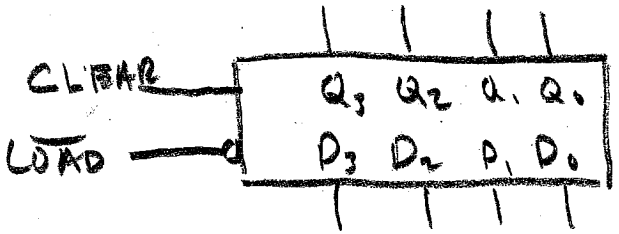
- SIZE - 4, 8, 12, 16 → 24 BITS
  (SOME BITS INTERNAL)
- TYPE - RIPPLE OR SYNCHRONOUS
  '390 '160
- UP/DOWN SELECTION?
  '191, '193



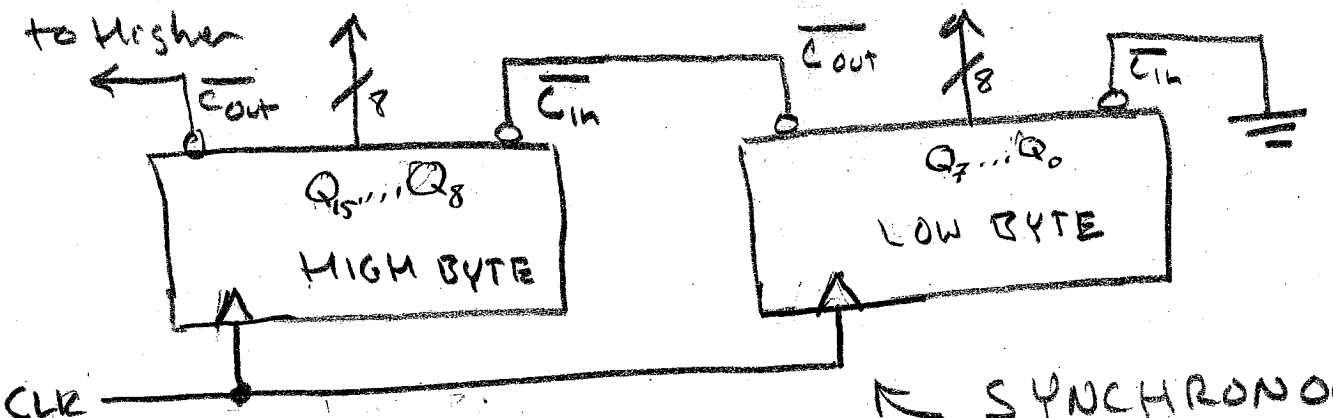
- LOAD & CLEAR

  LOAD - SETS COUNTER TO PREDEFINED START VALUE

  CLEAR - SETS COUNTER TO ZERO



COUNTERS DESIGNED TO EXPAND



← SYNCHRONOUS CLOCKING

C_{IN} = ENABLE - when asserted (here LO)
ALLOWS CLOCK TO ADVANCE

C_{OUT} = ASSERTED WHEN ALL BITS ARE 1
ABOUT TO ROLLOVER

...00|11111111 → CLK ...01|00000000

LOW BYTE

ADVANCE BY ONE

LOW BYTE

⇒ C_{OUT} ON

C_{OUT} OFF (NOW)

COUNTER RESET/CLEAR OPERATION

RESET or CLEAR CAN BE

- **SYNCHRONOUS** — CLEAR IS ASSERTED, BUT ACTION **WAITS** FOR CLOCK EDGE
- "JAM" (ASYNCH.) - CLEAR HAPPENS **IMMEDIATELY**

SYNCH. TYPES

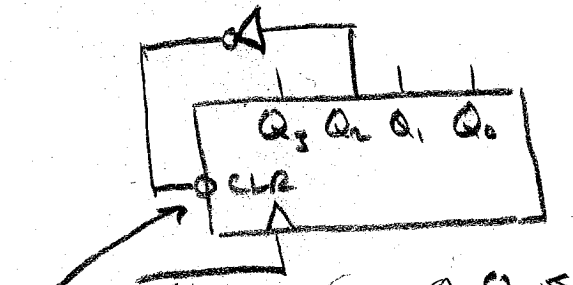
'163 (4 bit)
'269 (8 bit)

BOTH: '569

JAM TYPES

'161 (4 bit)
'393 (8 bit

EXAMPLE — DIVIDE BY 5

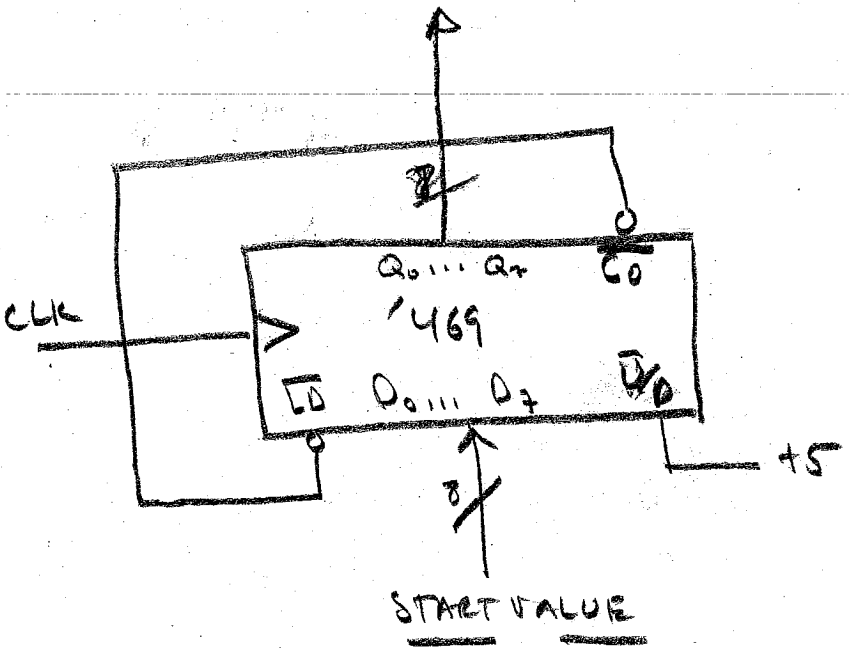


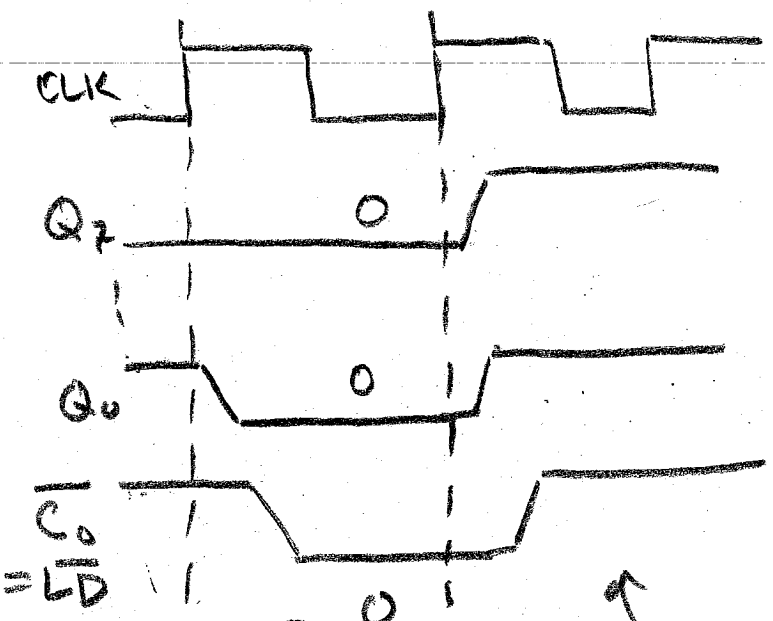
RESET AFTER $Q_4 = 1$

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

LOAD OPTION ALLOWS COUNTERS TO START AT NUMBERS OTHER THAN ZERO. (AGAIN, SYNCH & JAM TYPES)

EXAMPLE: MODULO-N COUNTER





COUNTER COUNTS DOWN ($\overline{C_0}$ asserted at 00_{16})