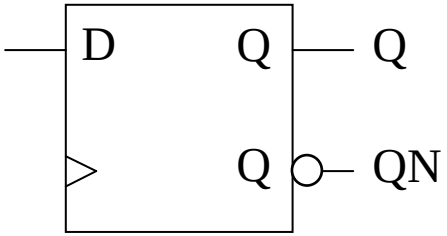
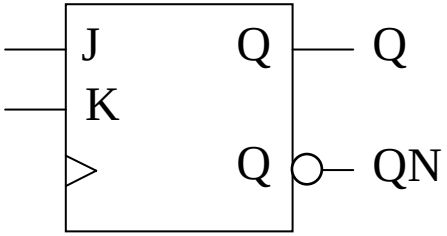
Sequential Circuit Analysis and Timing

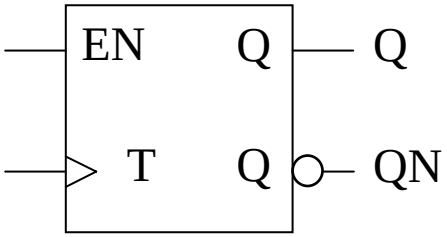
Various Types of FFs



Input D	Next state
0	0
1	1



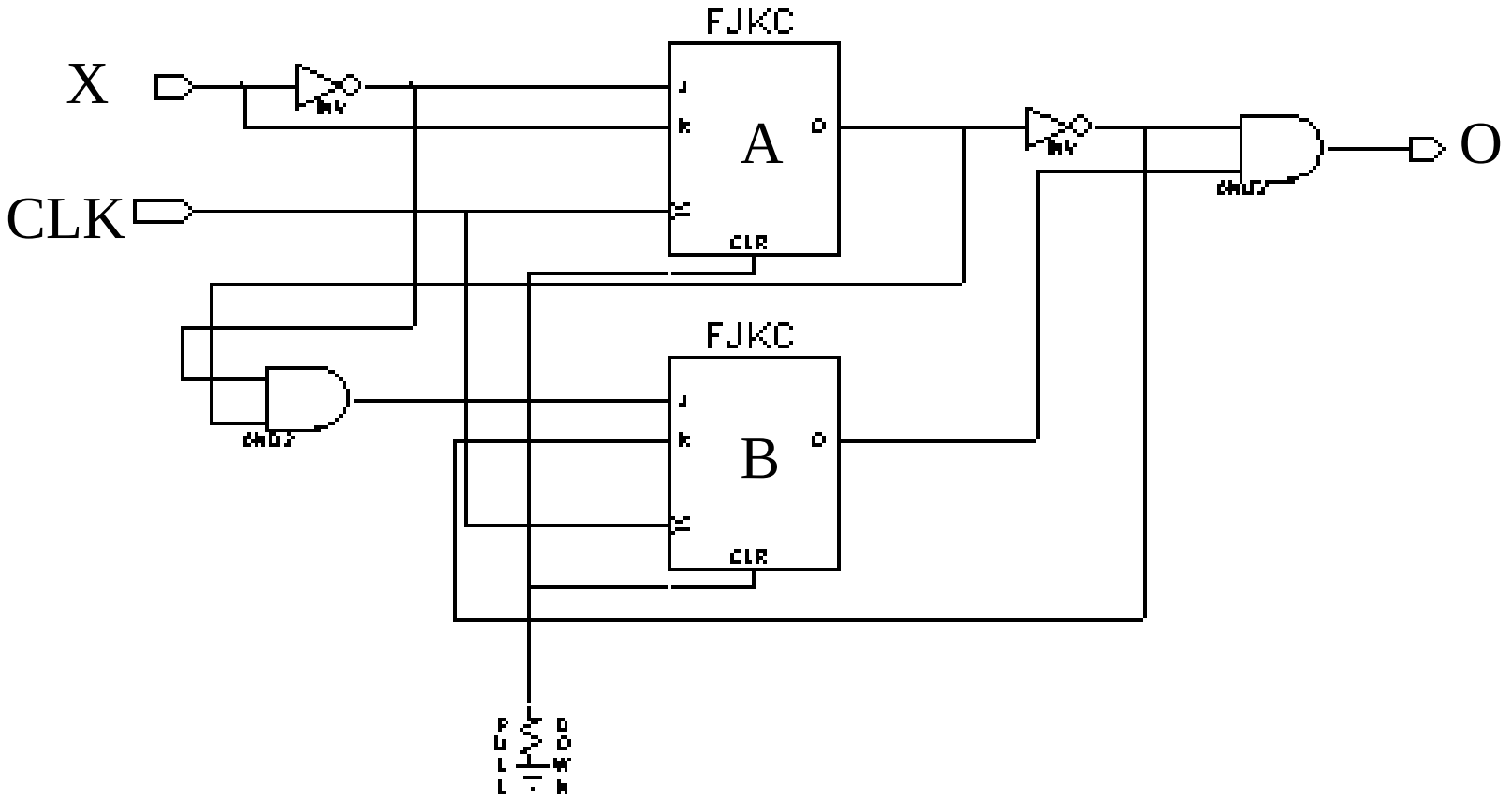
J	K	Next state
0	0	Q
0	1	0
1	0	1
1	1	Q’



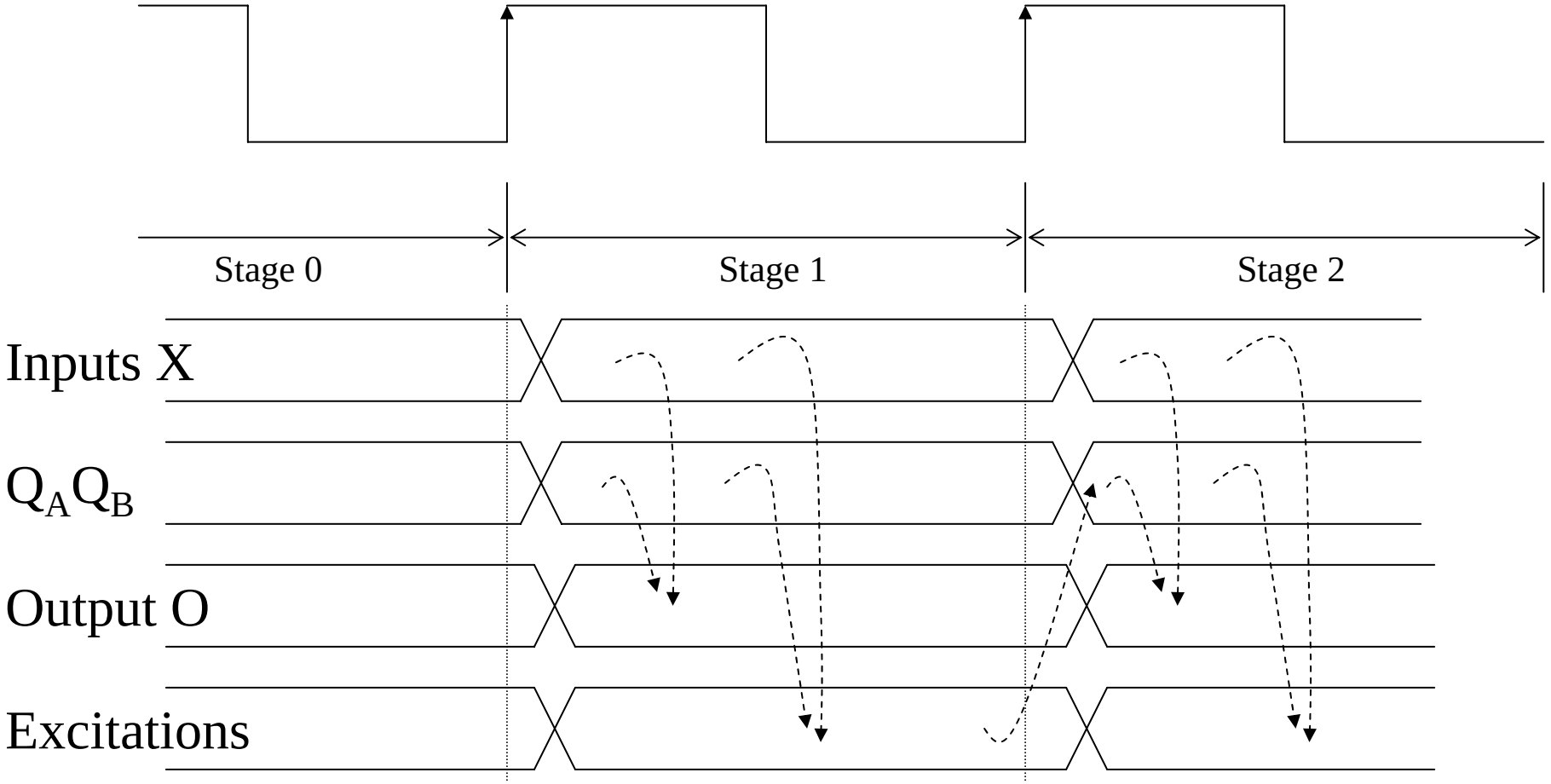
Input EN	Next state
0 at rising edge T	Q
1 at rising edge T	Q’

Analysis of Clocked Synchronous State Machines

- Begin with circuit
- End with state diagram – word description
- 3 step approach



Synchronized Operation With CLK



Step 1: Excitation and Output Equations

- Derive Excitation and Output Equations from the schematic

$$J_A = \overline{X}, K_A = X,$$
$$J_B = Q_A \overline{X}, K_B = \overline{Q_A},$$
$$O = \overline{Q_A} Q_B$$

Step 2: State/Output Table

C.S.		Input	Output
QB	QA	X	O
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Output Table

C.S.		Input	Excitation			
QB	QA	X	JB	KB	JA	KA
0	0	0	0	1	1	0
0	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	0	1
1	0	0	0	1	1	0
1	0	1	0	1	0	1
1	1	0	1	0	1	0
1	1	1	0	0	0	1

Excitation Table

C.S.		Input	N.S	
QB	QA	X	QB	QA
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

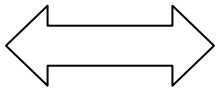
Transition Table

Step 2: State/Output Table

	P.S.		Input	Output	Excitation				N.S.	
	QB	QA	X	O	JB	KB	JA	KA	QB	QA
a	0	0	0	0	0	1	1	0	0	1
	0	0	1	0	0	1	0	1	0	0
b	0	1	0	0	1	0	1	0	1	1
	0	1	1	0	0	0	0	1	0	0
c	1	0	0	1	0	1	1	0	0	1
	1	0	1	1	0	1	0	1	0	0
d	1	1	0	0	1	0	1	0	1	1
	1	1	1	0	0	0	0	1	1	0

Step 2: State/Output Table (Cont.)

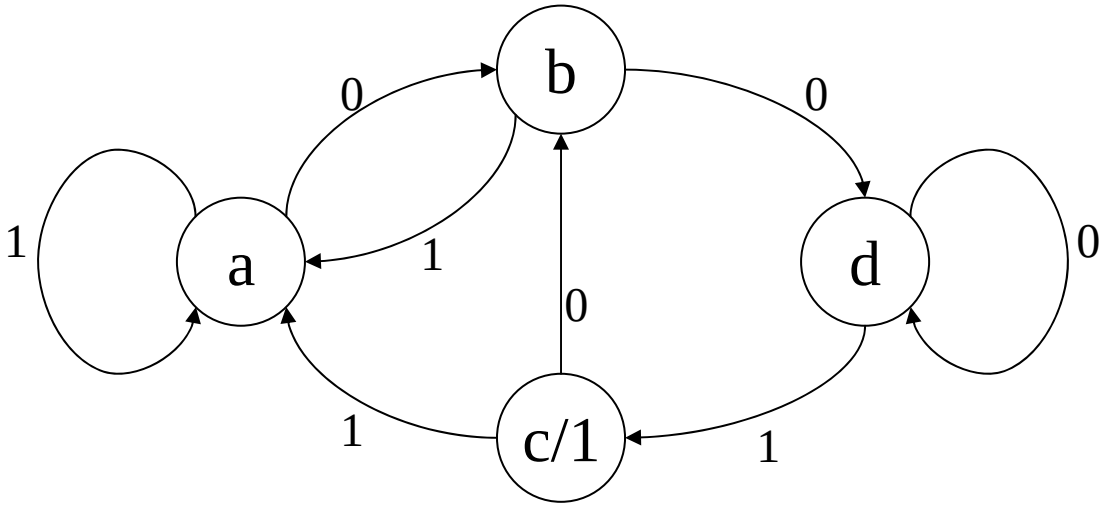
C.S.	X	O	N.S.
a	0	0	b
a	1	0	a
b	0	0	d
b	1	0	a
c	0	1	b
c	1	1	a
d	0	0	d
d	1	0	c



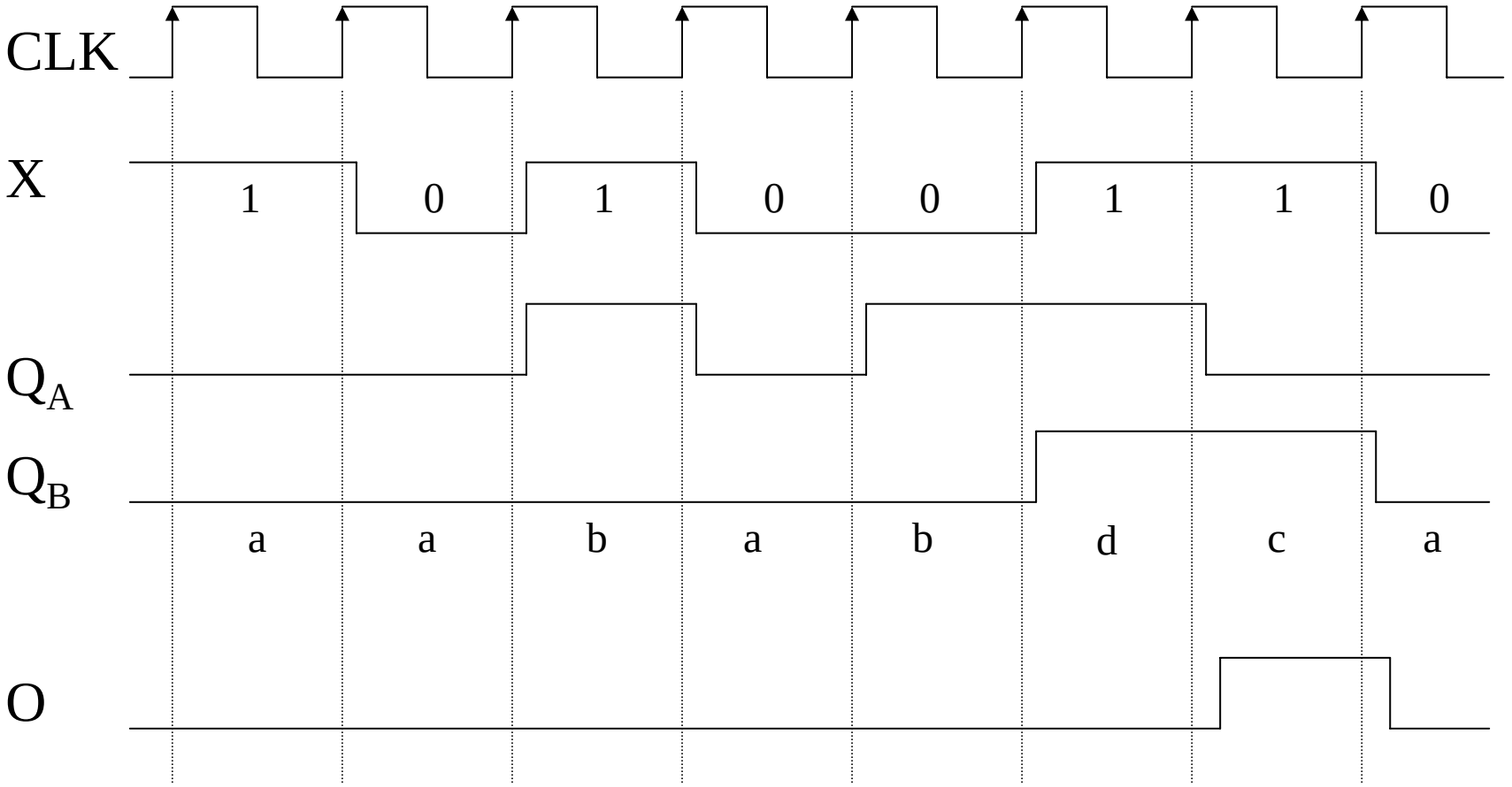
	X	
State	0	1
a	b,0	a,0
b	d,0	a,0
c	b,1	a,1
d	d,0	c,0

Step 3: State Diagram

- Can you tell what this machine is doing?

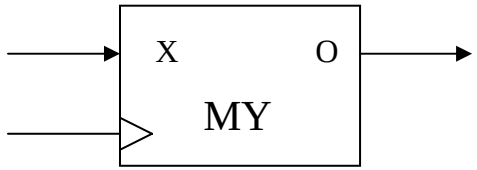


Example

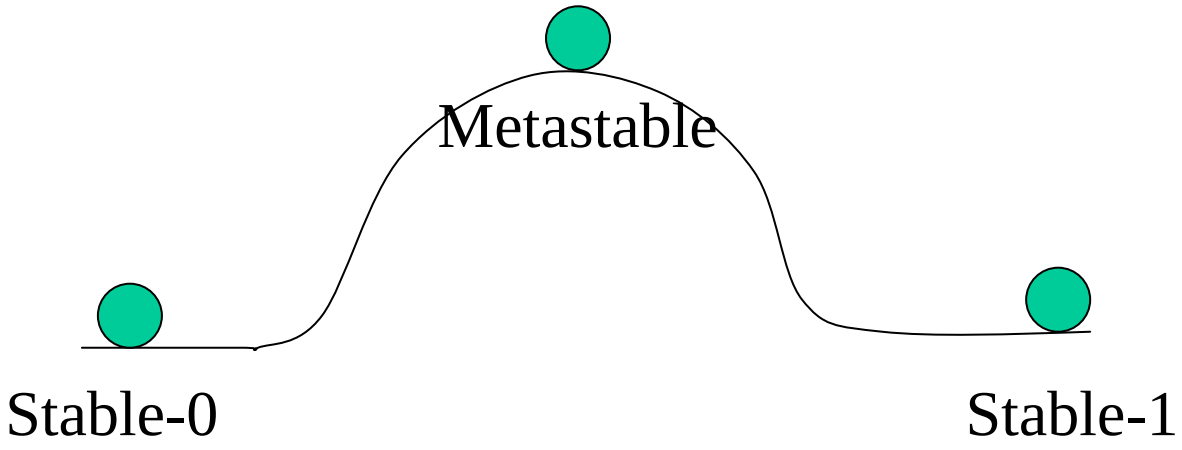


Timing

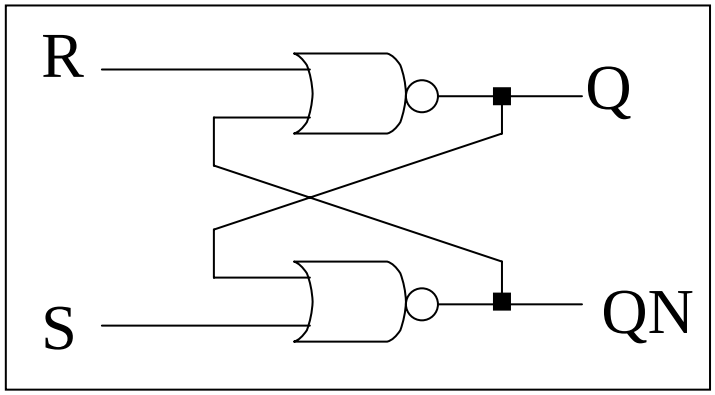
- If this circuit is to work with a larger system, what are the timing requirements? – Timing specification



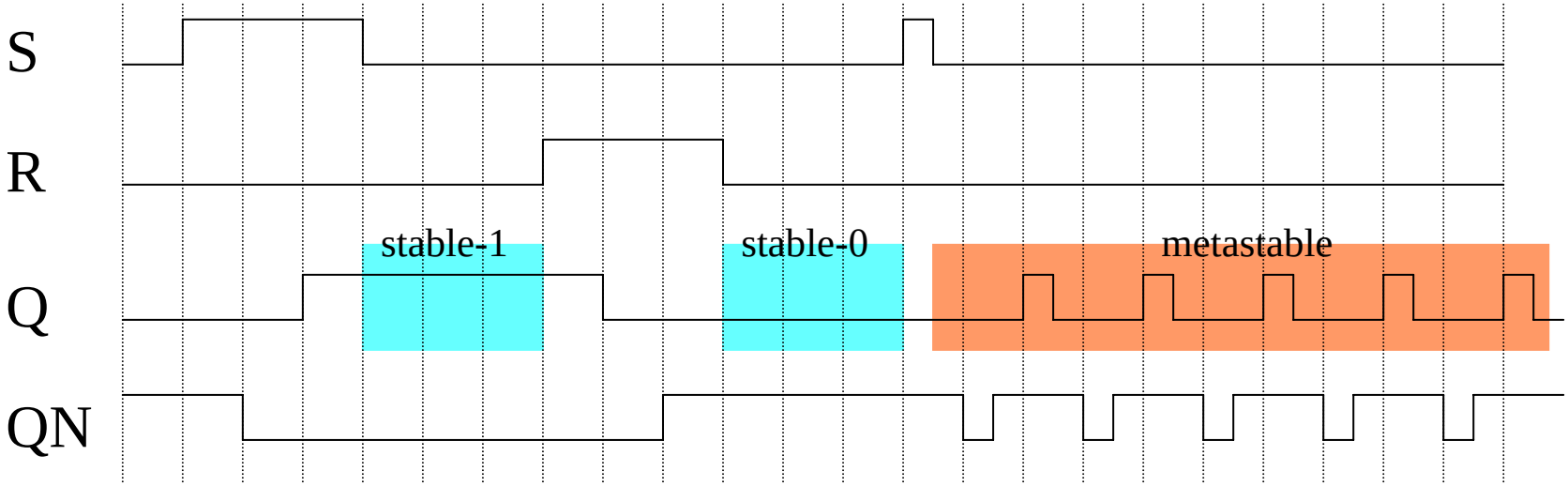
Metastability



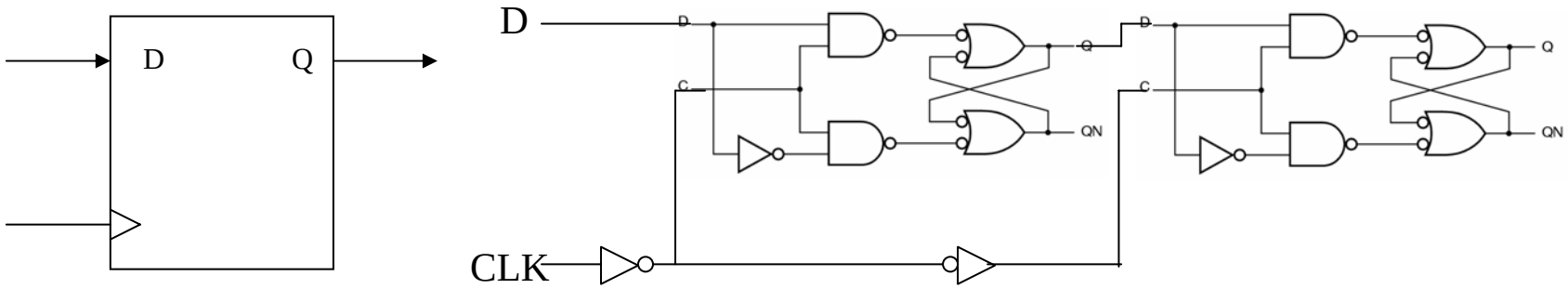
Metastability of a sequential logic

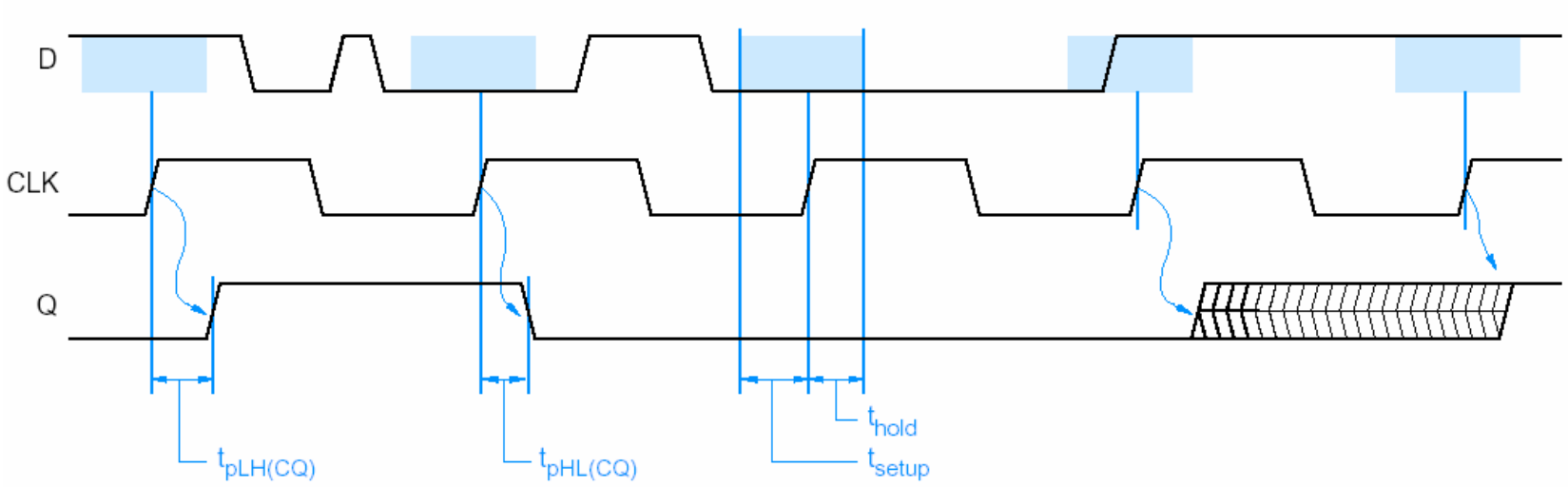


S R	Q	QN
0 0	last Q	last QN
0 1	0	1
1 0	1	0
1 1	0	0



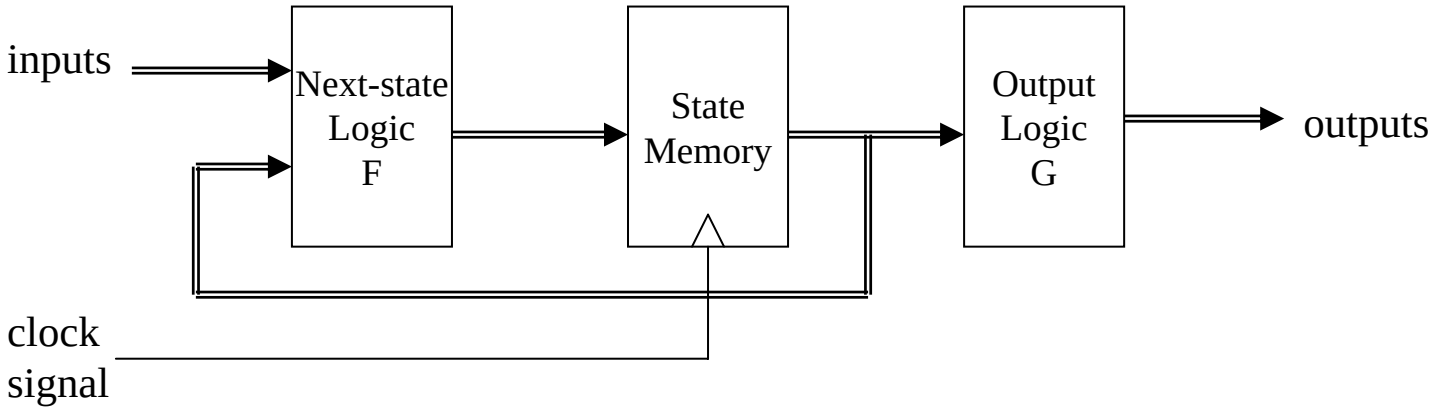
Setup & Hold Times of Sequential Components (e.g. D-ff)



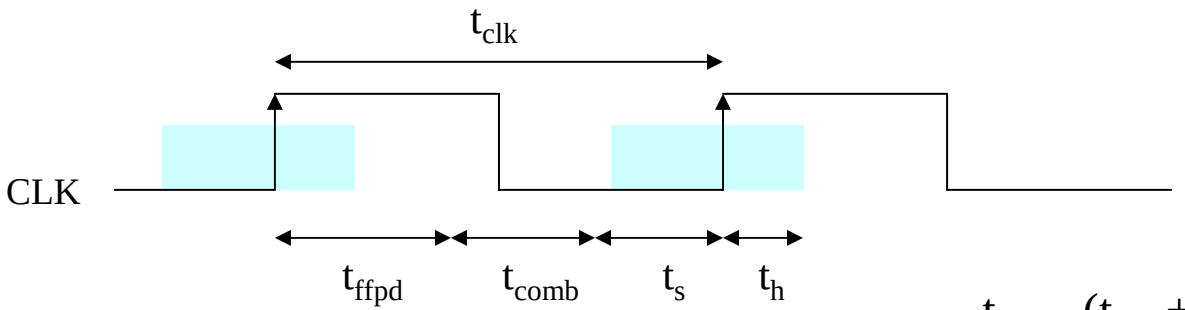


Maximum CLK frequency

- How fast can the circuit work?



- Assume inputs are ready at the right time



$$t_{clk} - (t_{ffpd} + t_{comb}) > t_s$$

Maximum CLK frequency

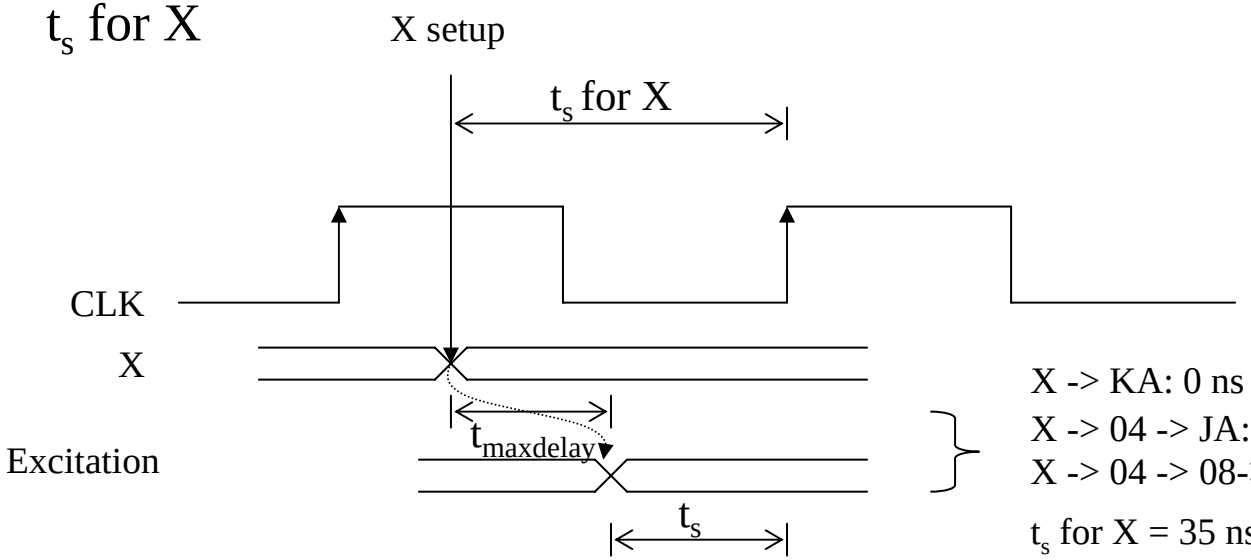
- Timing specs for 74LS parts

	Propagation delay	Setup time	Hold time	Max freq.
74LS04 (Inverter)	t_{pLH} = 15 ns t_{pHL} = 15 ns	N/A		
74LS08 (AND)	t_{pLH} = 15 ns t_{pHL} = 20 ns	N/A		
74LS109 (JK f/f)	t_{pLH} = 25 ns t_{pHL} = 40 ns	t_s = 35ns for H data in t_s = 25ns for L data in	t_h = 5ns	25 Mhz

- Find the worst case delay path
  - Sum up worst case component delay, independent of transition direction H->L, L->H
  - 109 t_p ->08 t_p ->109 t_{setup}: 40 + 20 + 35 = 95 ns
  - 109 t_p ->04 t_p ->109 t_{setup}: 40 + 15 + 35 = 90 ns
- Max f_{clk} = 1/95ns = 10.5 Mhz

Setup and Hold time specifications on X

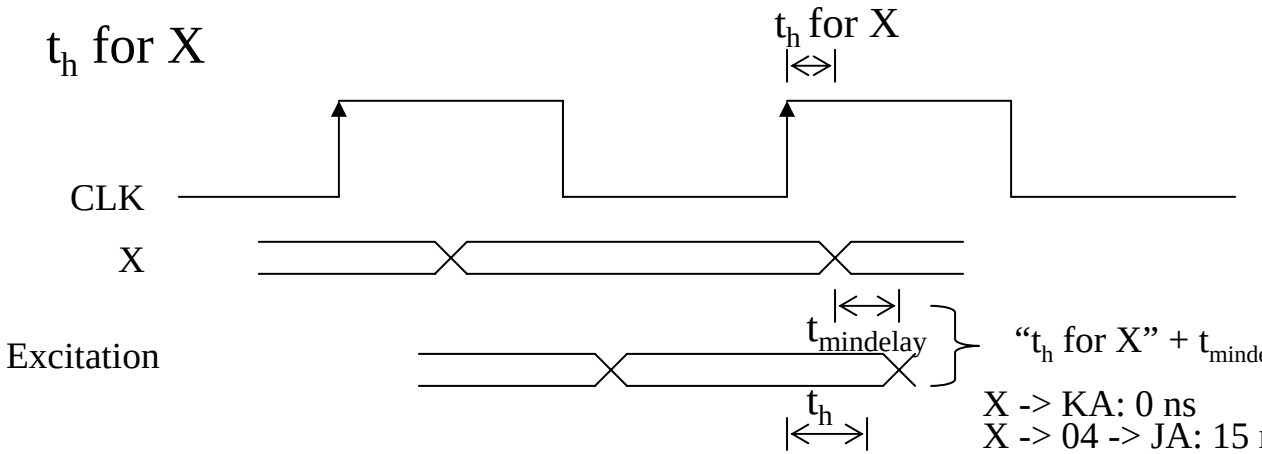
- t_s for X



X -> KA: 0 ns or
X -> 04 -> JA: 15 ns or
X -> 04 -> 08-> JB: 15 ns + 20 ns = 35 ns

t_s for X = 35 ns + 35 ns = 70 ns

- t_h for X



use min value

"t_h for X" + $t_{mindelay}$ > t_h

X -> KA: 0 ns
X -> 04 -> JA: 15 ns
X -> 04 -> 08-> JB: 15 ns + 15 ns = 25 ns

t_h for X = 5 ns

Propagation delay

- X -> O: N/A (Applicable only for Mealy type output)
- CLK -> O:
  - tpLH = max (tpLH ‘109 + tpLH ’08, tpHL ‘109 + tpLH ’04 + tpLH ‘08)
    = max (25 ns + 15 ns, 40 ns + 15ns + 15 ns) = 70 ns
  - tpHL = max (tpHL ‘109 + tpHL ’08, tpLH ‘109 + tpHL ’04 + tpHL ’08)
    = max (40 ns + 20 ns, 25 ns + 15 ns + 20 ns) = 60 ns

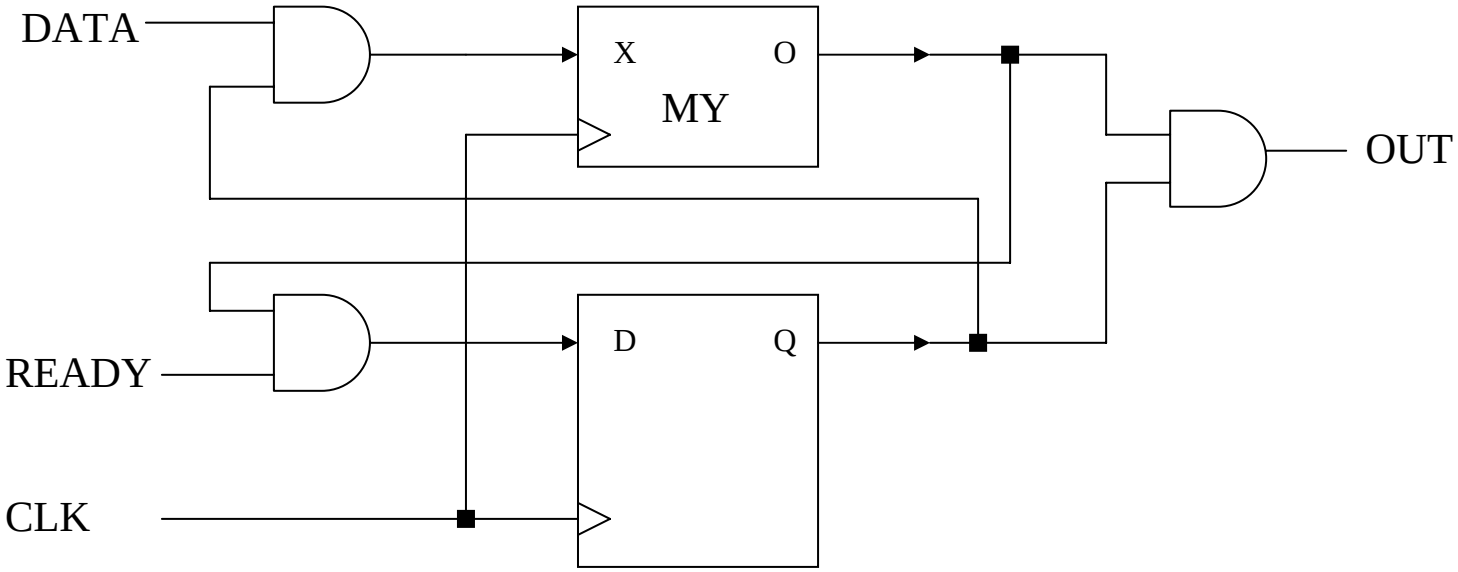
Final timing spec for our circuit

	Propagation delay	Setup time	Hold time	Max freq.
Our circuit	t_{pLH} = 70 ns t_{pHL} = 60 ns	t_s = 70 ns	t_h = 5 ns	10.5 Mhz

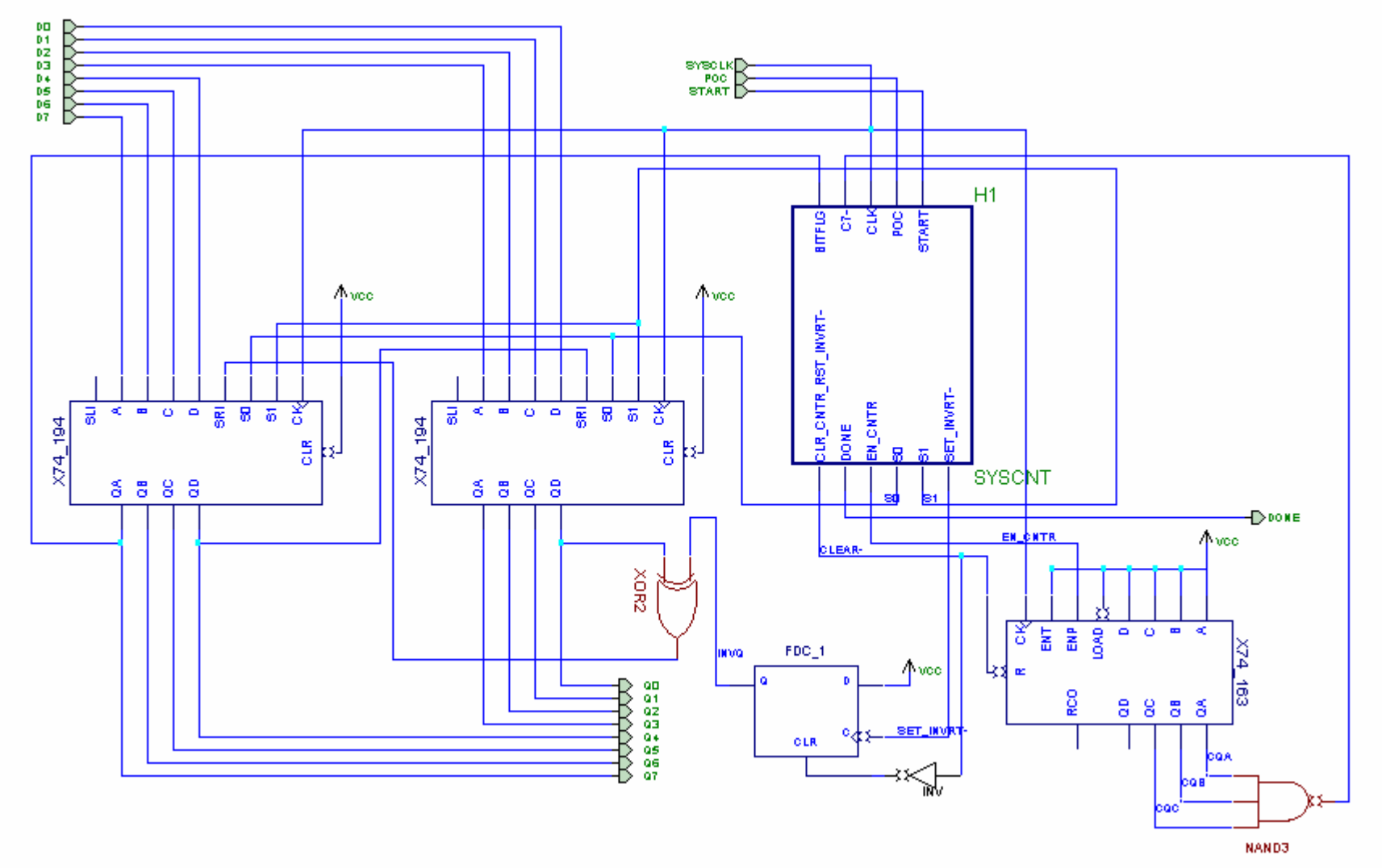
- This spec will be used for analyzing timing of a larger system containing our circuit as a component.

Quiz

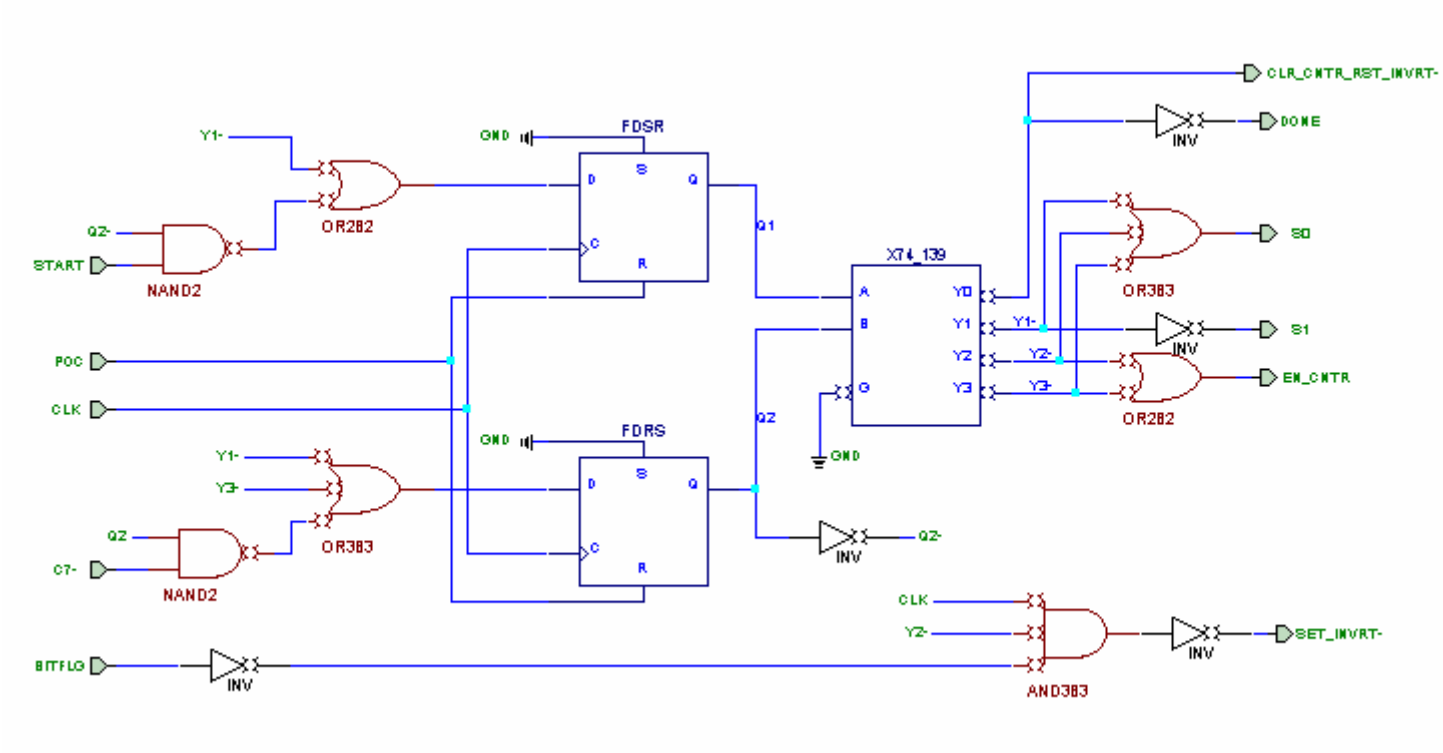
- What is the maximum clock frequency of the following circuit?



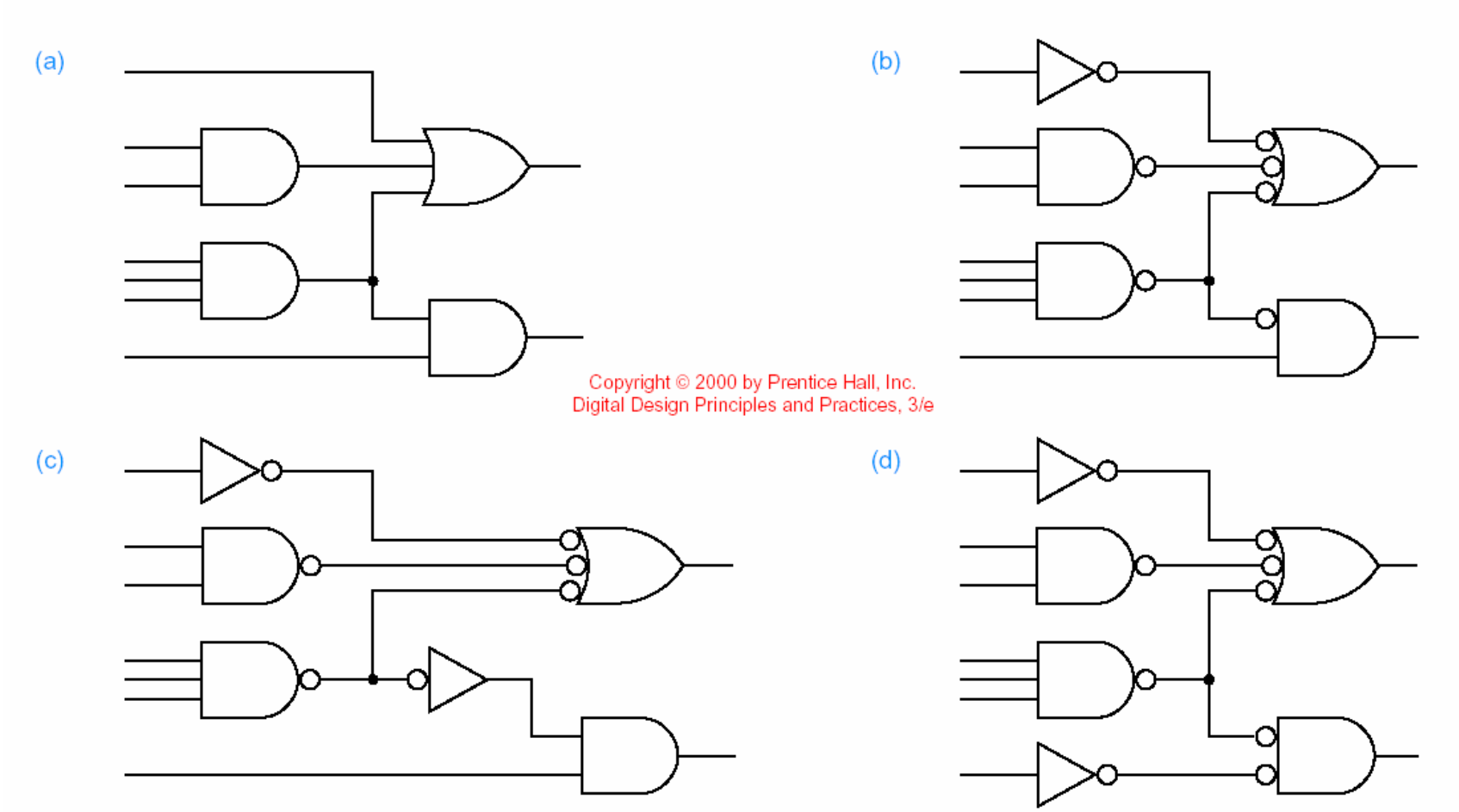
Can you tell what this guy is doing?



SYSCNT

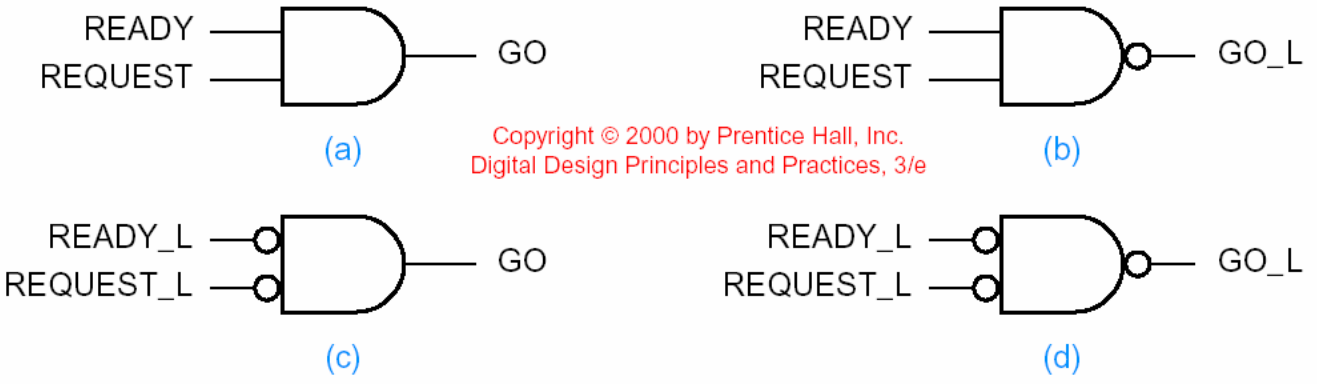

CLR_CNTR_RST_INVRT-
DONE
INV
Y1-
GND
FDSR
OR2B2
Q2-
START
NAND2
D
S
Q
C
R
Q1
X74_139
A
B
G
Y0
Y1
Y2
Y3
OR3B3
S0
S1
EN_CNTR
OR2B2
POC
CLK
GND
FDRS
Y1-
Y3-
OR3B3
Q2
C7-
NAND2
Q2
Q2-
INV
CLK
Y2-
AND3B3
INV
SET_INVRT-
BITFLG
INV

Bubble-to-bubble approach

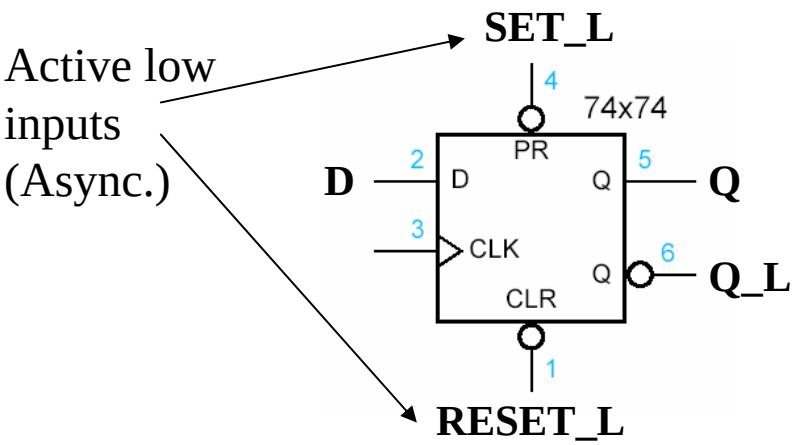


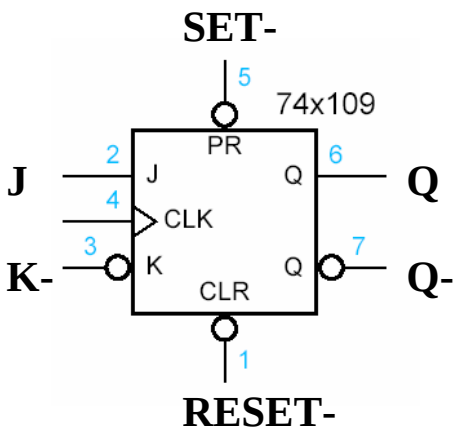
Proper use of bubbles and naming

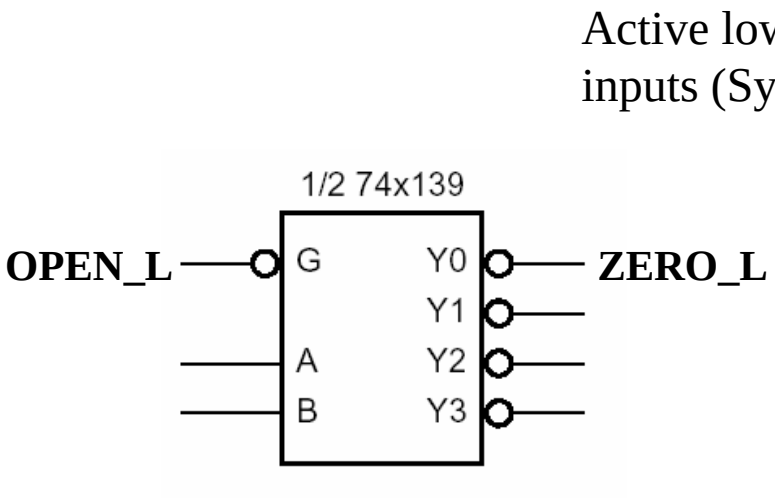
- Name of a signal: Help understanding the circuit like meaningful variable names in C programs (READY, GO, ENABLE, REQUEST, etc)
- Active High or Active Low (to take advantage of gate implementation, e.g., NOR is faster than OR)
- Use the bubble to represent Active Low signal and its name has "_L" or "-" (e.g., READY_L or READY-)

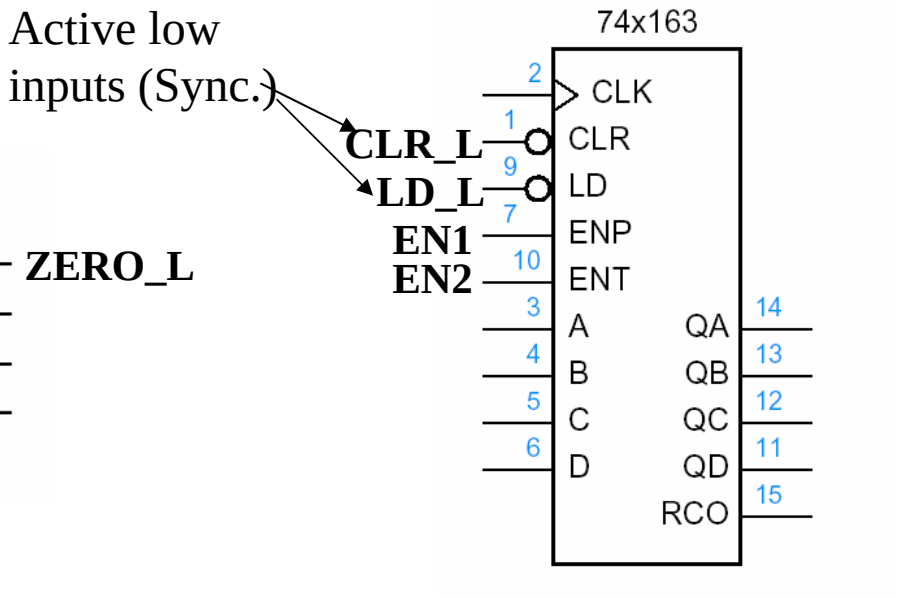


Examples


Active low
inputs
(Async.)
SET_L
74x74
PR
D
CLK
CLR
Q
Q_L
RESET_L


SET-
74x109
PR
J
CLK
K
CLR
Q
Q-
RESET-


1/2 74x139
OPEN_L
G
A
B
Y0
Y1
Y2
Y3
ZERO_L


Active low
inputs (Sync.)
74x163
CLR_L
LD_L
EN1
EN2
CLK
CLR
LD
ENP
ENT
A
B
C
D
QA
QB
QC
QD
RCO

MSI Chips
(used in our 2's complement machine)

Read: 8.4, 8.5, 6.4

(3rd Edition 8.4, 8.5, 5.4)

74LS163

- 4-bit, synchronous, parallel load, binary counter

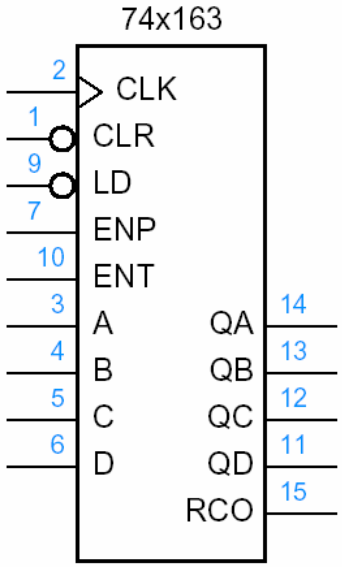
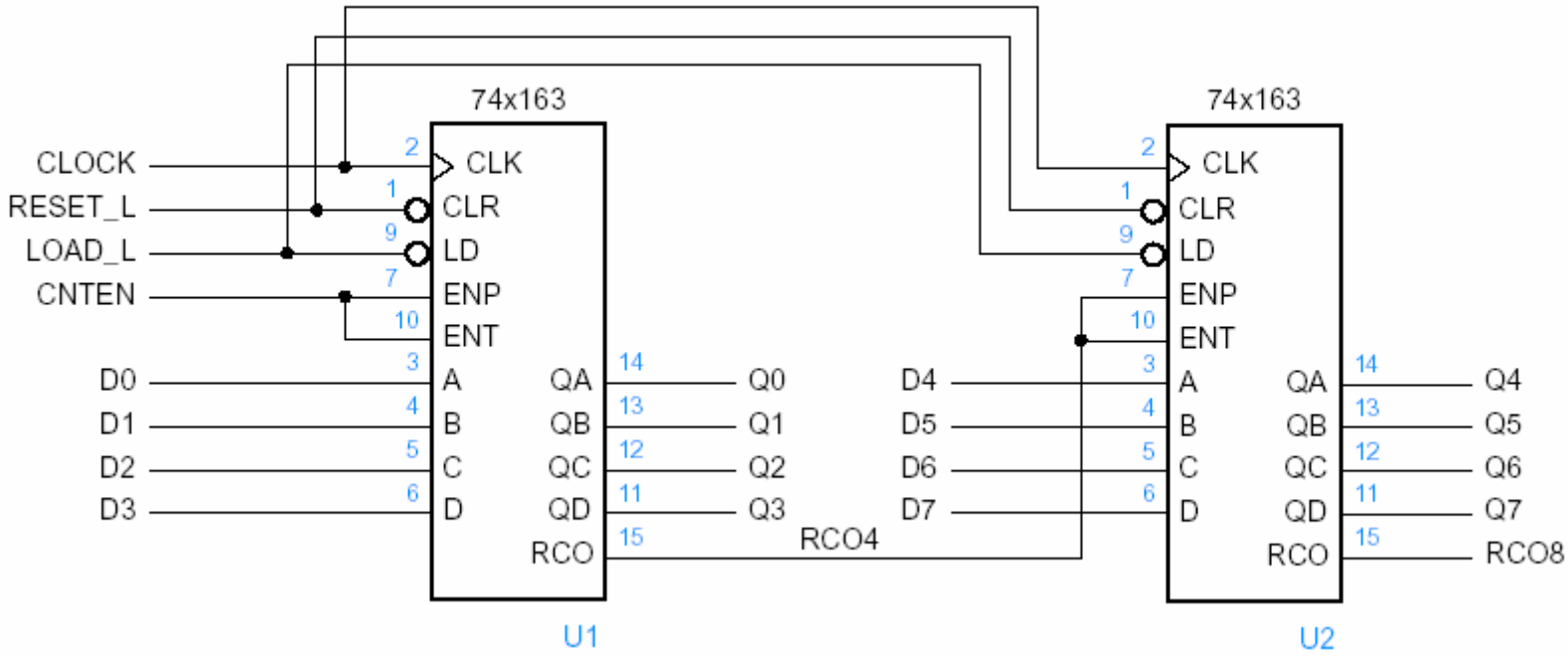


Table 8-11 State table for a 74x163 4-bit binary counter.

Inputs				Current State				Next State				RCO
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	0
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	1

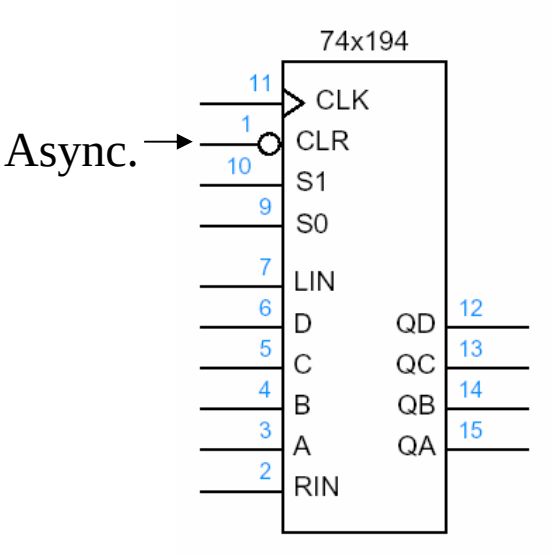
8 bit counter using 74LS163 ?

- Cascading using RCO



74LS194

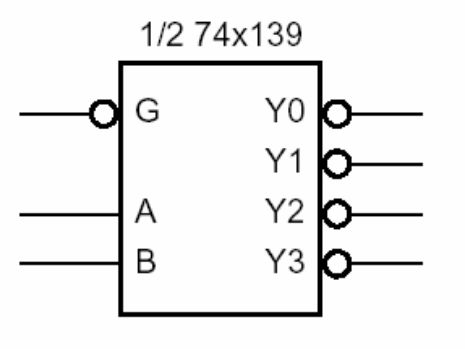
- 4-bit, parallel in, parallel out, bi-directional shift register



Function	Inputs S1	Inputs S0	Next state QA*	Next state QB*	Next state QC*	Next state QD*
Hold	0	0	QA	QB	QC	QD
Shift right	0	1	RIN	QA	QB	QC
Shift left	1	0	QB	QC	QD	LIN
Load	1	1	A	B	C	D

74LS139

- Dual 2-to-4 Decoder



Inputs			*Outputs*			
G_L	B	A	Y3_L	Y2_L	Y1_L	Y0_L
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

74LS138

- 3 Enables, 3-to-8 Decoder

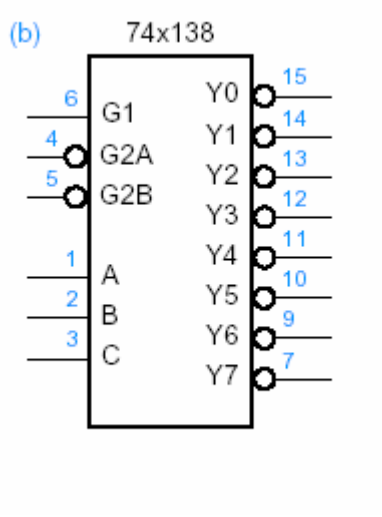
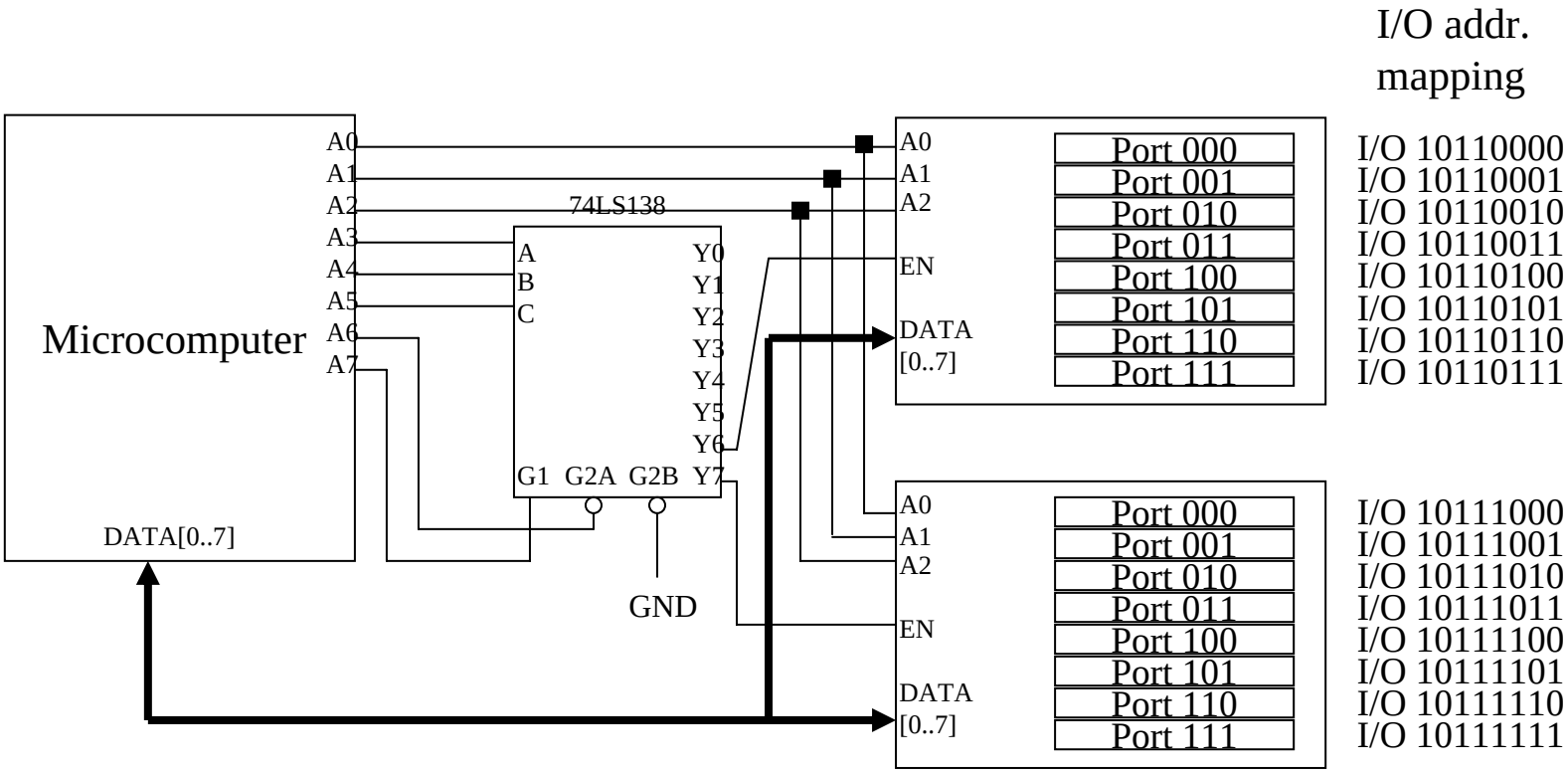


Table 5-7 Truth table for a 74x138 3-to-8 decoder.

Inputs						*Outputs*							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

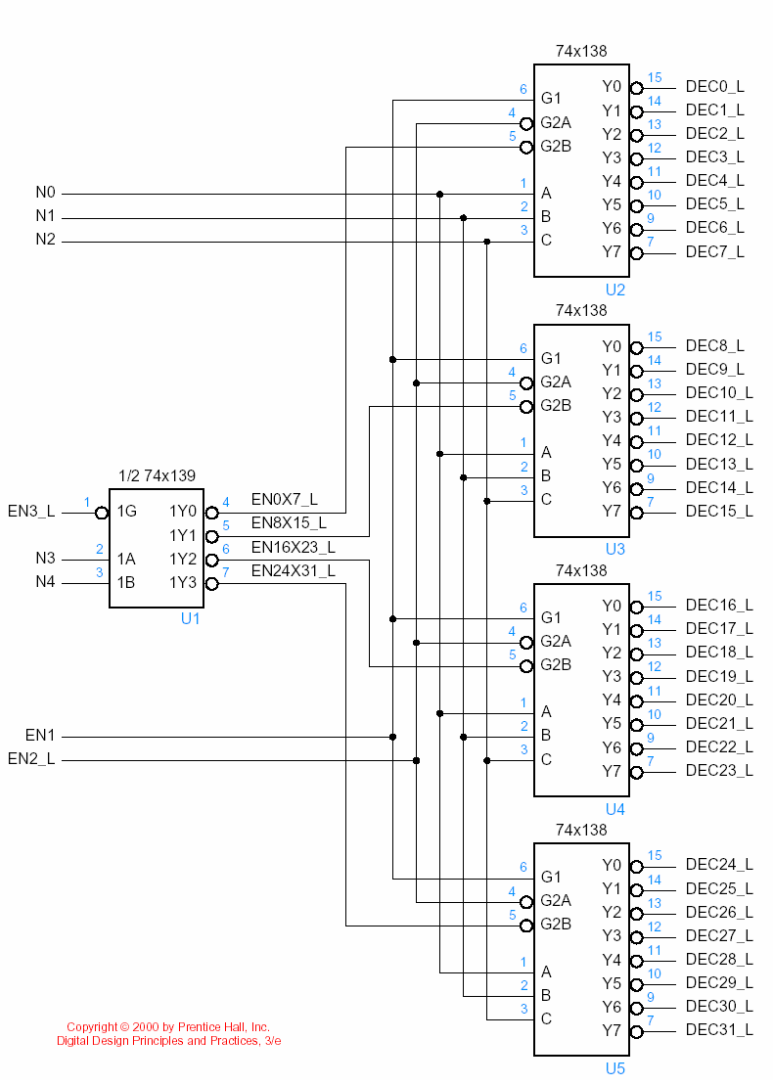
Applications of Decoder

- Address decoder
  - In microcomputers, an I/O address is 8 bits so that there are 256 unique device addresses.
  - How to make 16 I/O ports of two I/O chips (8 ports of each) to the following I/O mapped addresses?



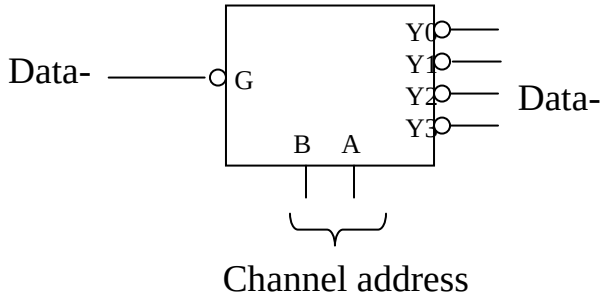
Applications of Decoder

- Cascading
  - Cascade small decoders for longer bits decoding
  - How to make 5-to-32 decoder (with 3 enables EN1, EN2-, EN3-) using 74LS138 and 74LS139?

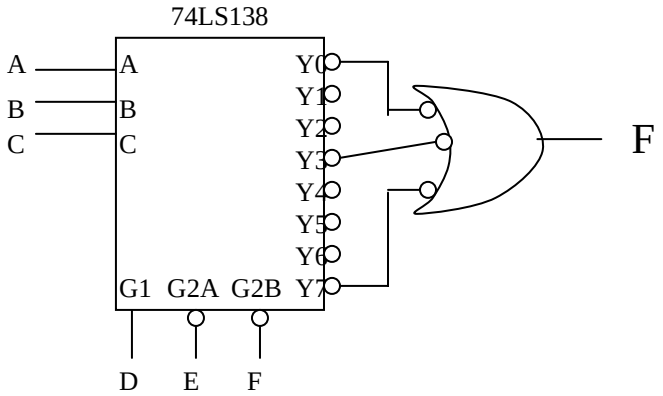


Applications of Decoder

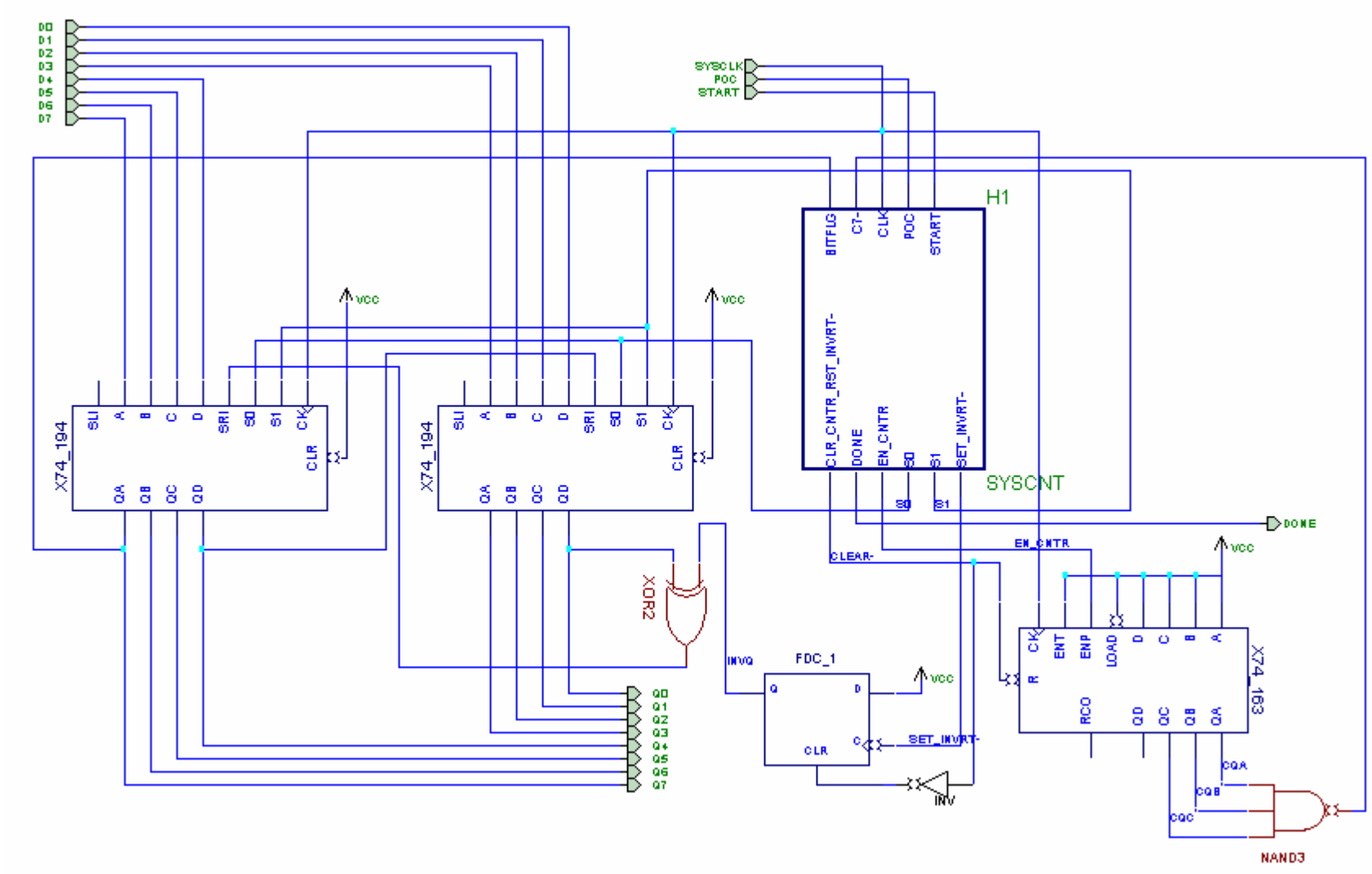
- Use as a Demultiplexer



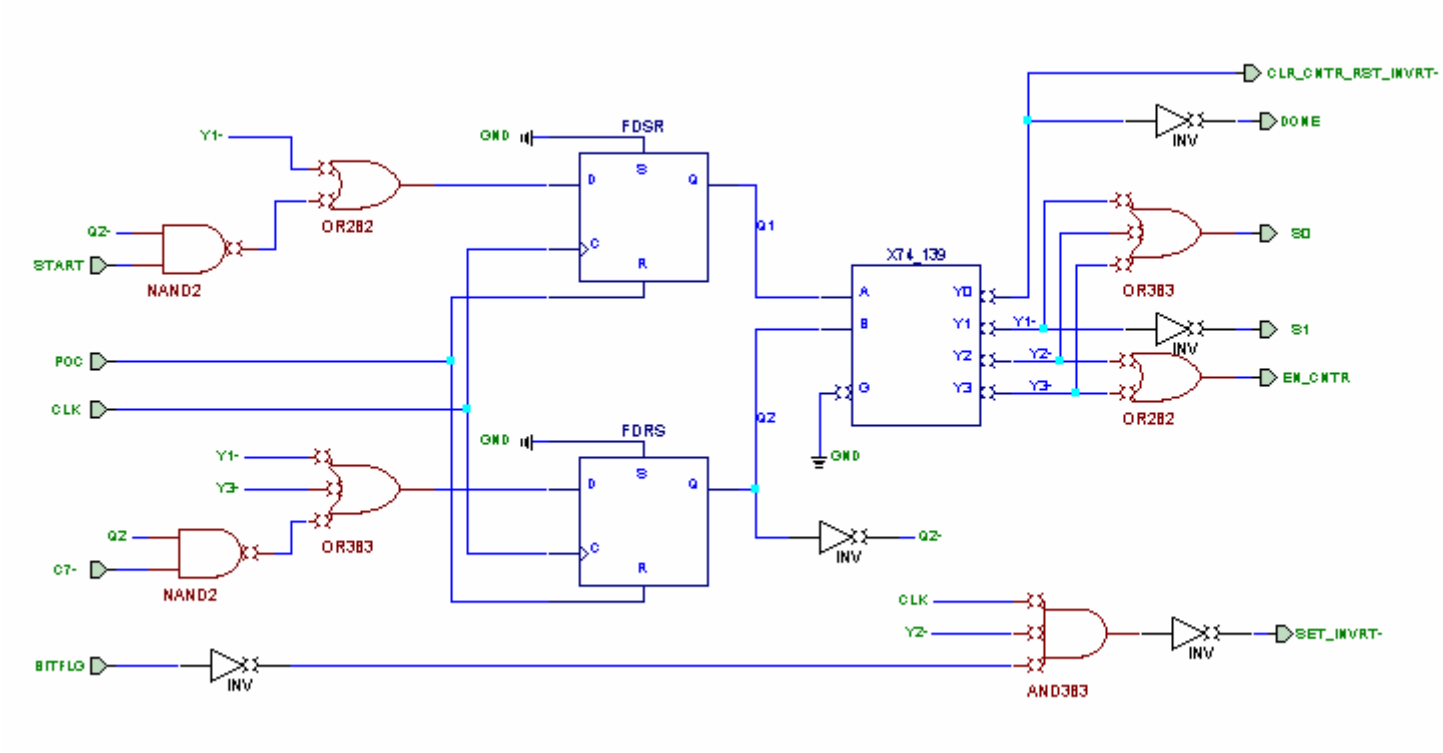
- Use in combinational logic design
  - Use a 74LS138 to implement $F = D\overline{E}\overline{F}(AB + \overline{A}\overline{B}\overline{C})$



C \ BA	00	01	11	10
0	1	0	1	0
1	0	0	1	0


D0
D1
D2
D3
D4
D5
D6
D7
SYSCLK
POC
START
H1
BITFLG
CT-
CLK
POC
START
CLR_CNTR_RST_INVRT-
DONE
EN_CNTR
S0
S1
SET_INVRT-
SYSCNT
X74_194
SLI
A
B
C
D
SRI
S0
S1
CK
CLR
QA
QB
QC
QD
VCC
XOR2
DONE
CLEAR-
EN_CNTR
INVQ
FDC_1
Q
D
C
CLR
SET_INVRT-
INV
X74_163
CK
R
ENT
ENP
LOAD
RCO
CQA
CQB
CQC
NAND3

SYSCNT


CLR_CNTR_RST_INVRT-
DONE
INV
Y1-
GND
FDSR
OR2B2
Q2-
START
NAND2
D
S
Q
C
R
Q1
X74_139
A
B
G
Y0
Y1
Y2
Y3
OR3B3
S0
S1
EN_CNTR
OR2B2
POC
CLK
GND
FDRS
Y1-
Y3-
OR3B3
Q2
C7-
NAND2
Q2
Q2-
INV
CLK
Y2-
AND3B3
INV
SET_INVRT-
BITFLG
INV

Hints

- Sequential Two's complement machine
  - Analyze a machine that takes the 2's complement of an 8-bit number
    - 8 bits in, START → 8 bits out, DONE
  - More realistic example that uses MSI chips
  - For PLDs, FPGAs design, we usually use functional blocks (LBB – Logic Building Block) equivalent to the counters, shift registers, decoders, etc

General Architecture and Operation

- Example: 01001010 → 10110110 (2's complement of A = 2^n –A)
  - 01001010 → 11111111+1 – 01001010 = 10110101 + 1 = 10110110
  - Write down bits from right until a 1 is encountered. Complements all bits there after
- General Operation Flow
  - Load 8 bits into 2 × 74194 (4 bit shift right/left register)
  - Do a circular shift on the data, inverting bits as necessary
  - Finally, the 2's complement data will appear at the output after 8 shift operations

	Parallel Data Out $Q_7Q_6Q_5Q_4Q_3Q_2Q_1Q_0$	Invert InvertQ
	- - - - - - - -	0
	0 1 0 0 1 0 1 **0**	0
1	**0** 0 1 0 0 1 0 1	0
2	1 **0** 0 1 0 0 1 0	1
3	1 1 **0** 0 1 0 0 1	1
4	0 1 1 **0** 0 1 0 0	1
5	1 0 1 1 **0** 0 1 0	1
6	1 1 0 1 1 **0** 0 1	1
7	0 1 1 0 1 1 **0** 0	1
8	1 0 1 1 0 1 1 **0**	1

General Architecture and Operation

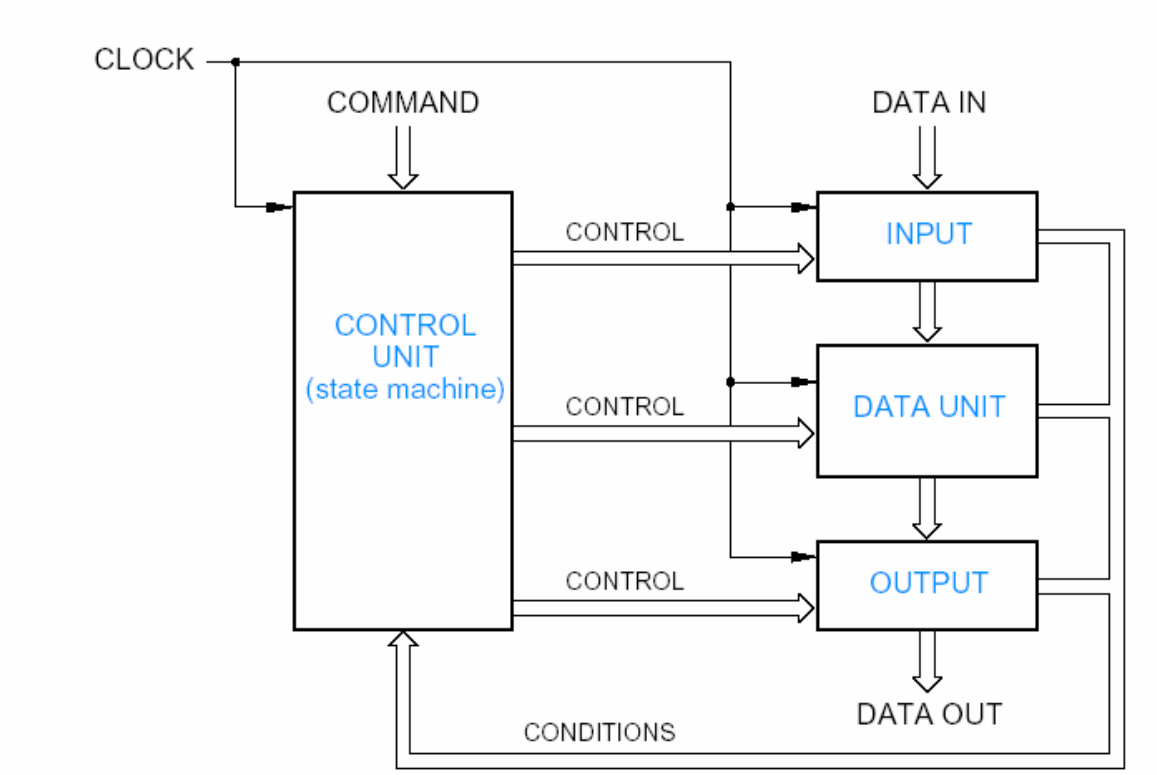
- 74LS194 (4 bit shift register) is used for loading & shifting 8 bit data
- We use D f/f (with asynchronous clear) to remember from when inverting is necessary
- We use 74LS163 (a synchronous 4-bit counter) to count 8 shifts
- System controller control the overall operation
  - The system controller determines when data should be loaded, shifted or held by controlling S1 and S0
  - The system controller also looks at BITFLG so as to know when to set the INVERT D f/f
  - The system controller also clears 74LS163 at the beginning, increments it each time a bit is shifted, and detects when 8 bits have been shifted.
  - Finally, the system controller asserts DONE signal

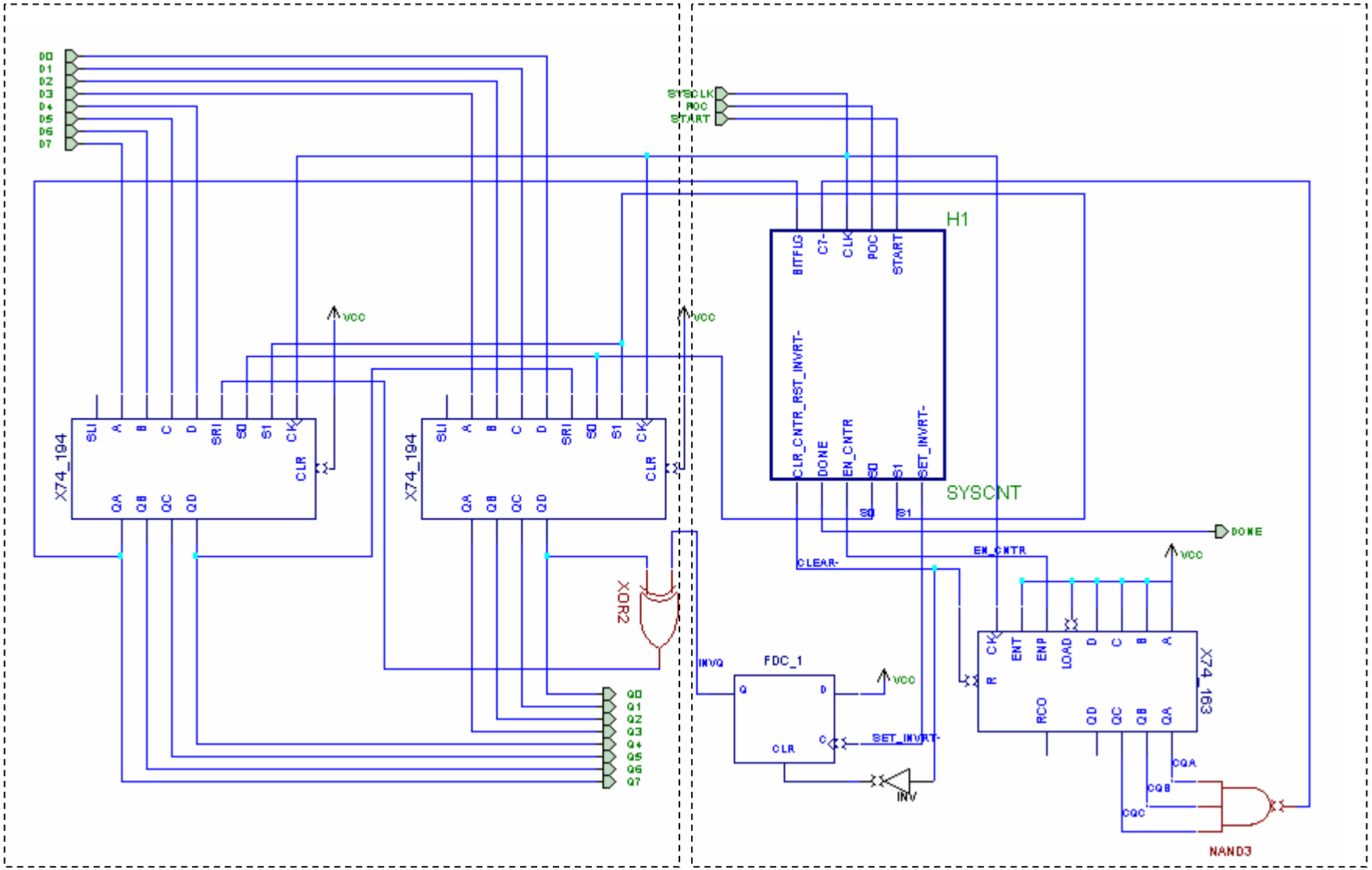
Much larger system analysis

- Analysis of the structure
  - More than a few f/fs in circuit – not practical to treat as a single state machine
  - Try directly applying the 3-step approach
    - How many f/fs?
      - Shift reg – 8, Counter – 4, INVERT –1, System Controller –2
      - 15 f/fs => 2^{15} states
- Then, 3 step analysis only on system controller

Synchronous System Structure

- Generally 2 Parts: Data Unit & Control Unit
  - Data unit: process data (store, route, combine)
  - Control unit: starting and stopping actions, test conditions, decide what to do next
  - Only control unit – designed as state machine



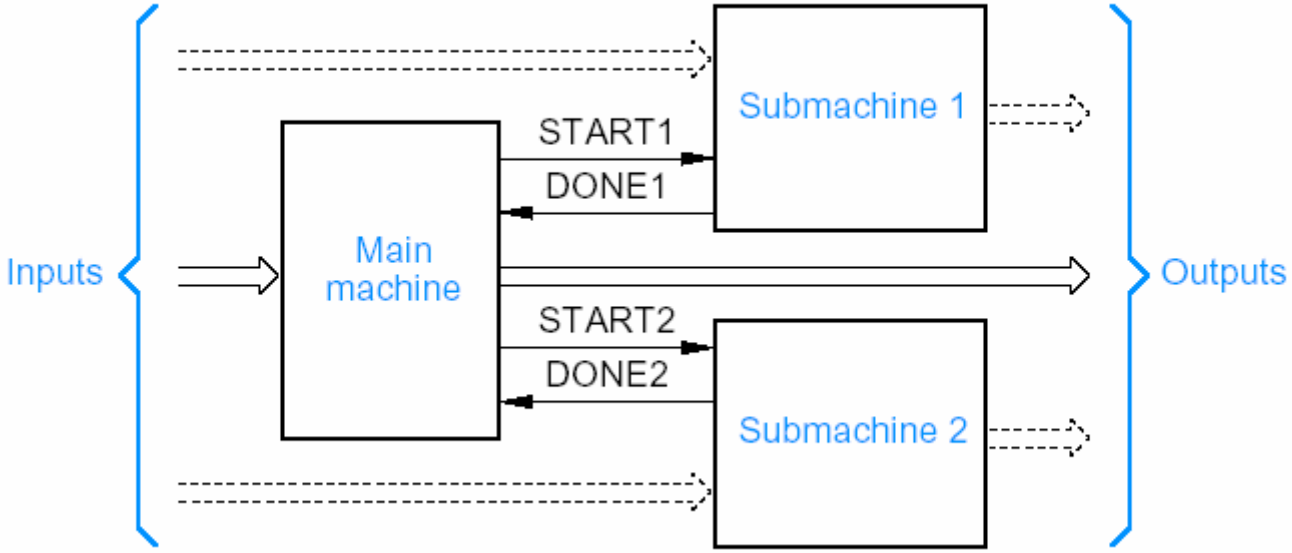


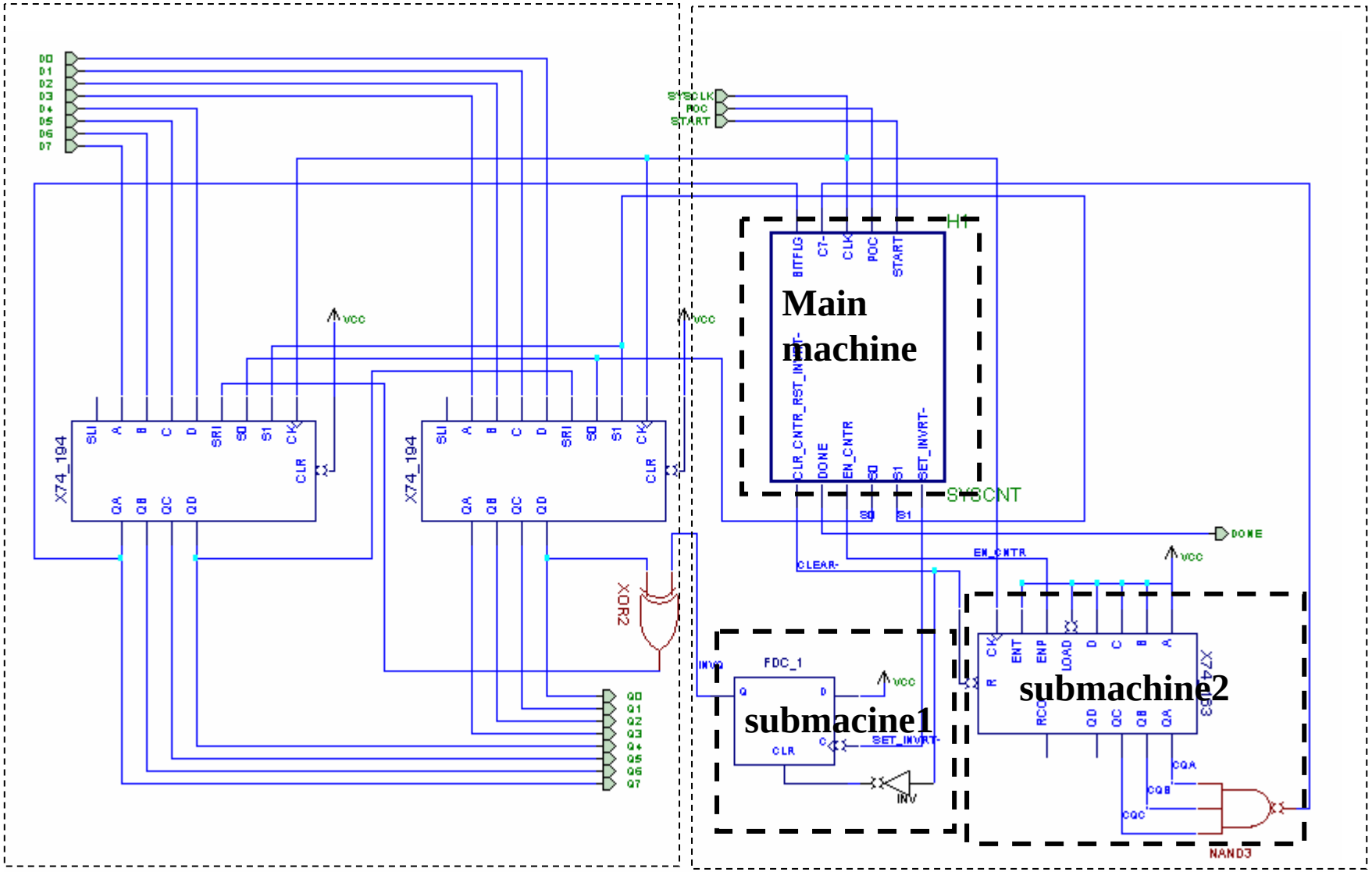
Data unit

Control unit (State Machine)

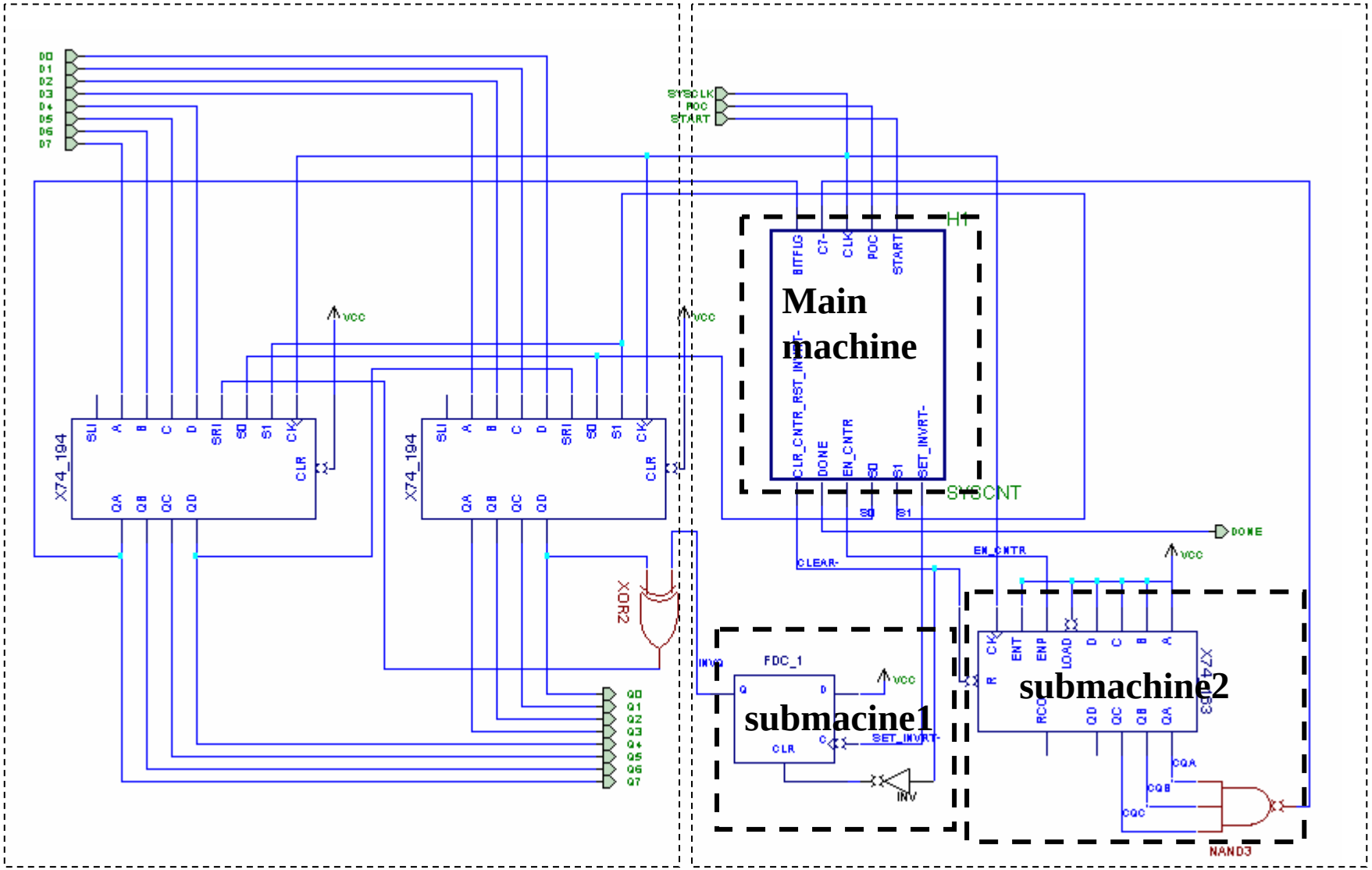
Decomposing State Machines

- The control unit may be further partitioned
  - Main machine – system controller
  - Sub machines – counter, INVERT D f/f



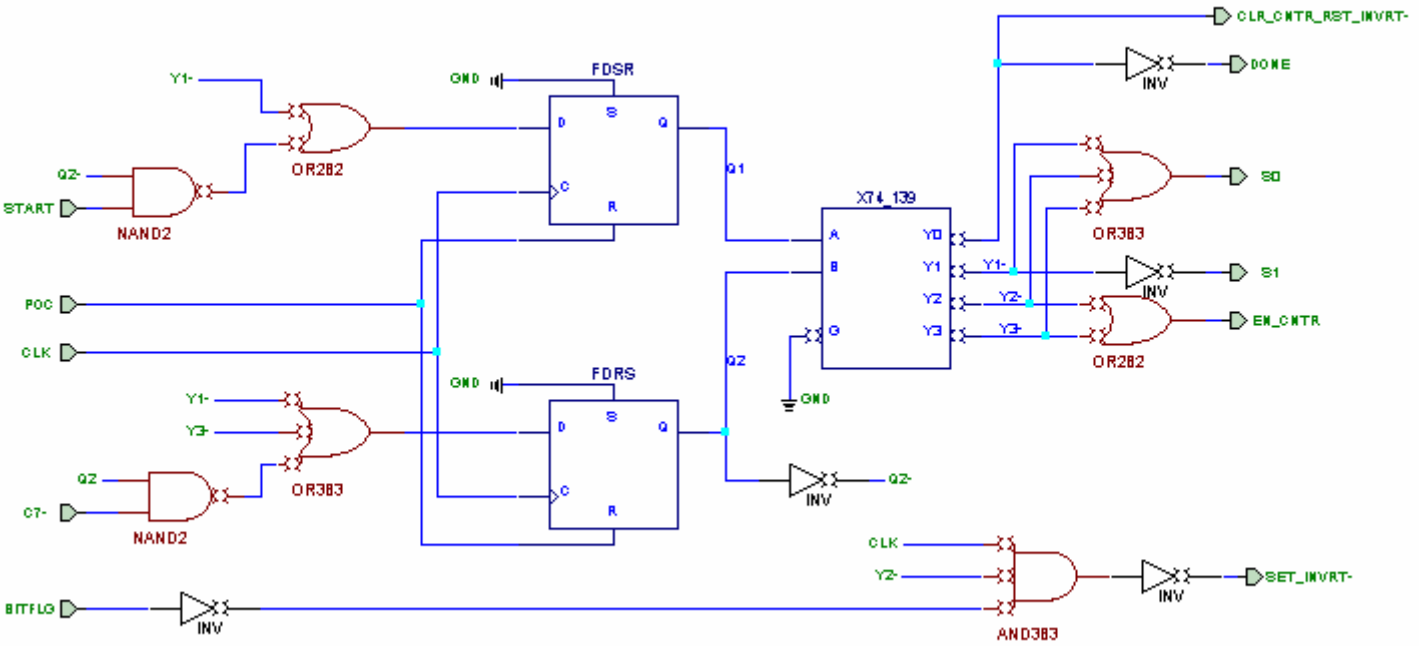

Main
machine
submacine1
submachine2
Data unit
Control unit (State Machine)

Do a 3 step analysis only on system controller



Data unit

Control unit (State Machine)


CLR_CNTR_RST_INVRT-
DONE
INV
S0
OR3B3
S1
INV
EN_CNTR
OR2B2
Y1-
Y2-
Y3-
GND
X74_139
A
B
G
Y0
Y1
Y2
Y3
Q1
Q2
FDSR
FDRS
D
S
C
R
Q
Y1-
Q2-
START
NAND2
OR2B2
POC
CLK
Y1-
Y3-
Q2
C7-
NAND2
OR3B3
Q2-
INV
CLK
Y2-
AND3B3
SET_INVRT-
INV
BITFLG
INV

Step 1: Excitation and Output Eqs.

- Inputs?
  - External inputs (4): CLK, START, BITFLAG, C7 (ignore POC for simplification)
  - P.S. (2): Q1, Q2
- Outputs?
  - External outputs (7): CLR_CNTR, RST_INVRT, S0, S1, ENCNTR, SET_INVRT, DONE
  - N.S. (2): = Excitations D1, D2

$$D_2 = Y_1 + Y_3 + Q_2\overline{C}_7 = Q_1 + Q_2\overline{C}_7$$

$$D_1 = Y_1 + \overline{Q}_2 START = \overline{Q}_2 Q_1 + \overline{Q}_2 START$$

$$ENCNTR = Y_3 + Y_2 = Q_2$$

$$S_1 = Y_1 = \overline{Q}_2 Q_1$$

$$S_0 = Y_3 + Y_2 + Y_1 = Q_2 + Q_1$$

$$DONE = Y_0 = \overline{Q}_2\overline{Q}_1$$

$$CLR\_CNTR = \overline{Q}_2\overline{Q}_1$$

$$RST\_INVRT = \overline{Q}_2\overline{Q}_1$$

$$SET\_INVRT = \overline{CLK \cdot Y} \cdot BITFLAG = \overline{CLK \cdot Q_2\overline{Q}_1} \cdot BITFLAG$$

Why falling edge of CLK?
Can we remove CLK from SI equation? → No
1. avoid glitch on SI when transit to Y2
2. hold time on RIN (not likely the problem)

Step 2: State/Output Table

- How many rows and columns?

P.S.		Inputs				Outputs							N.S.	
Q2	Q1	CLK	START	BFLAG	C7	EC	CC	S1	S0	RI	SI	DONE	Q2(=D2)	Q1(=D1)

64 rows

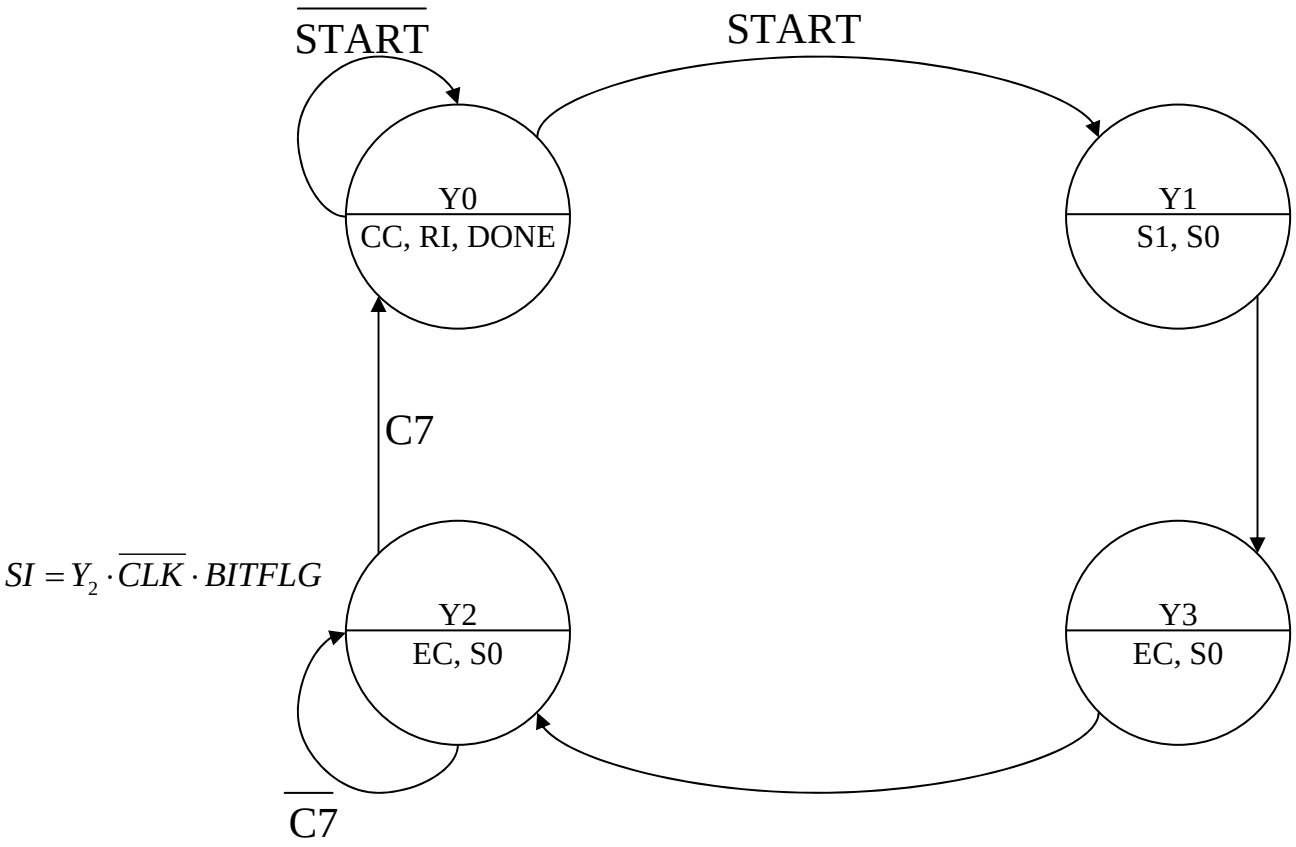
Step 2: State/Output Table

- Variable entered table

	P.S.		Outputs							N.S.	
	Q2	Q1	EC	CC	S1	S0	RI	SI	DONE	Q2	Q1
Y0	0	0	0	1	0	0	1	0	1	0	ST
Y1	0	1	0	0	1	1	0	0	0	1	1
Y3	1	1	1	0	0	1	0	0	0	1	0
Y2	1	0	1	0	0	1	0	↑	0	$\overline{C_7}$	0

$\overline{CLK} \cdot BITFLG$

Step 3: State Diagram

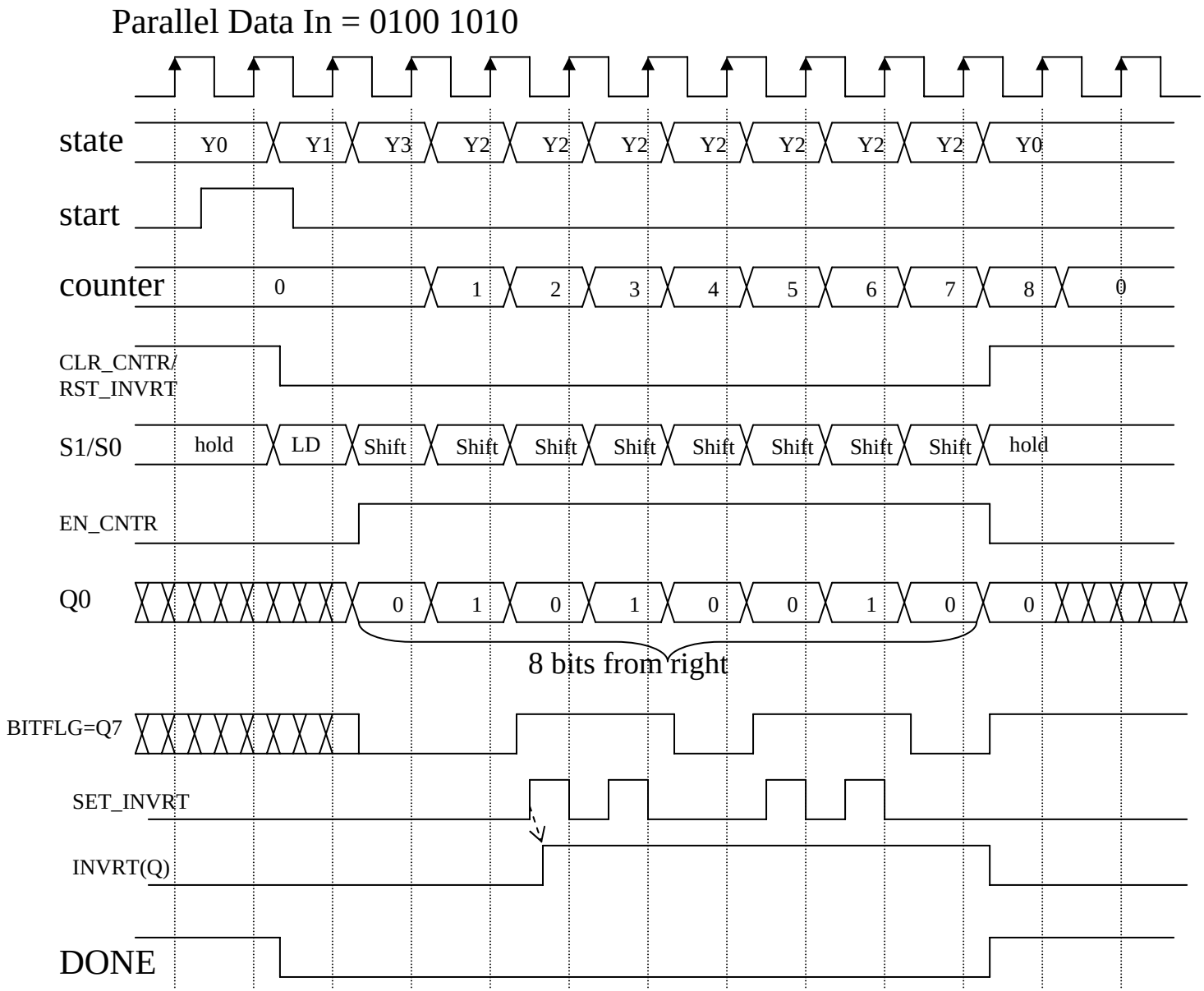


Quiz: Why we need Y3? Can we merge it with Y2?

Example

	Shift register	State	Counter	Invert	
START →	XXXXXX	Y0	-	-	
	XXXXXX	Y1	0	0	
	11001010	Y3	0	0	
	01100101	Y2	1	0	
	10110010	Y2	2	0->1	
	11011001	Y2	3	1	
	01101100	Y2	4	1	
	10110110	Y2	5	1	
	11011011	Y2	6	1	
	01101101	Y2	7	1	
	00110110	Y0	8	1	→ DONE
	00110110	Y0	0	0	

Sample Timing Diagram



Timing Analysis

- Timing specs. for the parts we have used

Chip	tpLH(ns)	tpHL(ns)
LS00, LS04, LS10, LS27	15	15
LS86	30	22
LS139 A,B -> Y LS139 G -> Y	29 24	38 32

LS74	tpLH	tpHL	ts	th
CLR, CLK, PR->Q	25	40		
D			20	5
fmax = 25 Mhz				

LS163	tpLH	tpHL	ts	th
CLK->Q	24	27		
CLK->RCO	35	35	20	5
ENT->RCO	14	14		
CLR->Q		28		
A,B,C,D,ENP,ENT, LD			20	0
fmax = 25 Mhz				

LS194	tpLH	tpHL	ts	th
CLR->Q		35		
CLK->Q	26	30		
S1, S0			30	
L,R,A,B,C,D			20	
All				0
fmax = 25 Mhz				

Maximum CLK frequency

- We must satisfy setup time for all f/f inputs (we will consider only D2, S0, RIN as examples)

D2 setup	Path1: CLK→Q2(LS74) + B→Y3(LS139) + Y3→D2(LS10) + D2_setup = 40+38+15+20=113ns
	Path2: CLK→Q2(LS74) + LS00 + LS10 + D2_setup = 40+15+15+20=90ns
	Path3: CLK→CNTR_Q(LS163) + CNTR_Q→C7(LS10) + LS00 + LS10 + D2_setup = 27+15+15+15+20=92ns
S0 setup	Path1: CLK→Q2,Q1(LS74) + A,B→Y(LS139) + Y→S0(LS10) + S0_setup = 40+38+15+30=123ns
RIN (=SRI: Shift Right Input) of Left 'x194' setup	Path1: CLK→Q0(LS194) + LS86 + RIN_setup = 30+30+20=80ns
	Path2:CLK(falling edge)→SI (LS27,LS04) + SI→INVQ(LS74) + LS86 + RIN_setup = 15+15+25(40?)+30+20=105(120?)ns ➔ ½ tclk > 105(120?)ns ➔ 210(240?)ns

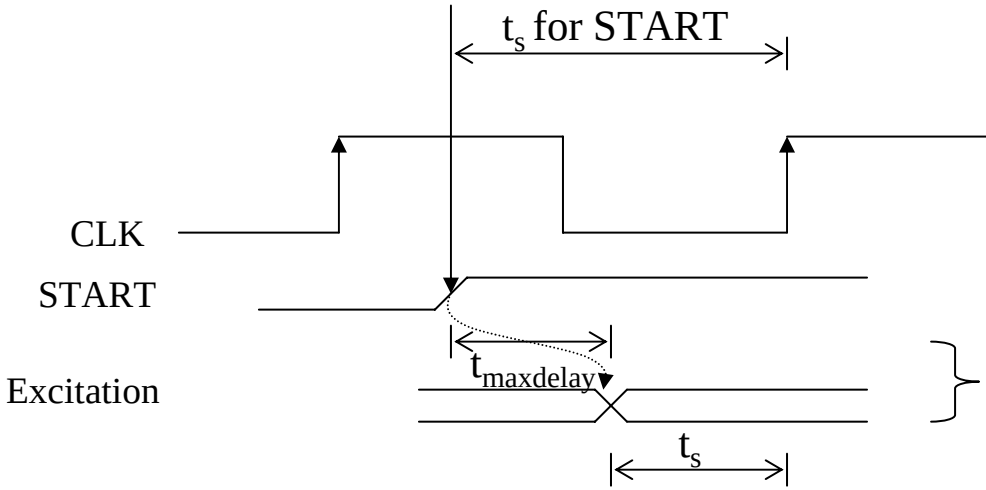
Max clk frequency = 1/210ns = 4.8Mhz

If we use the pure maximum value approach,

Max clk frequency = 1/240ns = 4.2Mhz

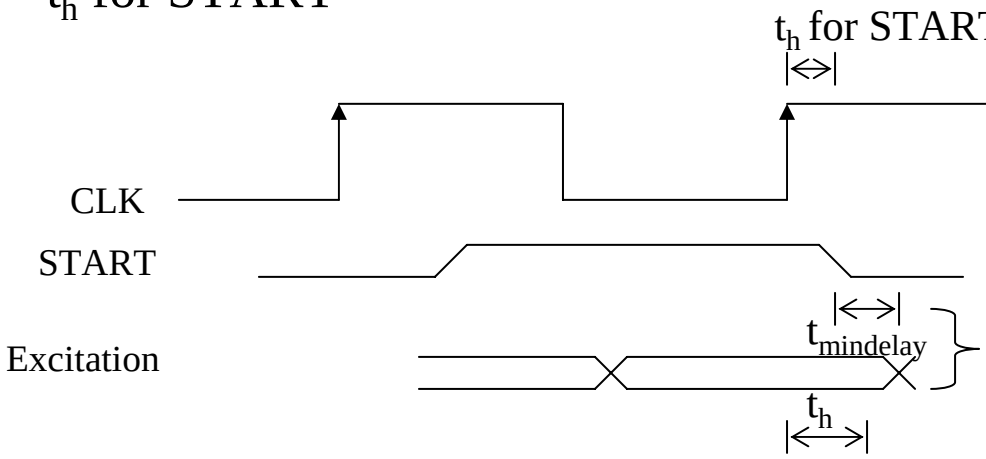
Setup and Hold time specifications on START

- t_s for START



START -> D1: LS00+LS00
$t_{maxdelay}$=15ns+15ns
START setup = 30 ns + 20 ns = 50 ns

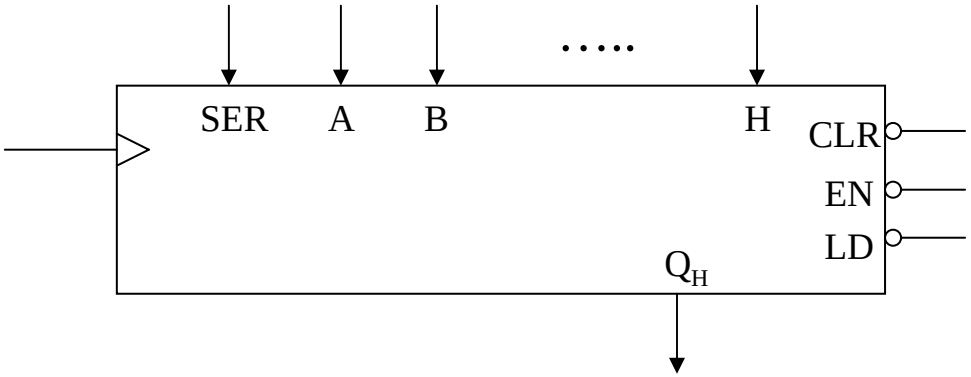
- t_h for START



"t_h for START" + $t_{mindelay}$ > t_h
$t_{mindelay}$=15ns+15ns
t_h = 5 ns -> don't have to hold START

Problem Statement

- Design 74x166 – 8 bit parallel-in, serial-out shift register with enable



CLR – Asynchronous

EN – when asserted -> according to LD

otherwise -> hold

LD – when asserted -> LD

otherwise -> Shift

Step 1: State/Output Table

- A state table (or diagram) isn't very helpful since LD can take you from any state to any other -> messy!
  - Inputs: A-H 8
    SER 1
    CLR 1
    EN 1
    LD 1
    State variables: Qs 8

2^{20} rows!

Alternatives

- CLR asynchronous – not needed
- EN – take care in special way
  - initially assume always asserted
- Think 2 bits rather than 8 bits and then generalize
- Variables: SER, A, B, LD, Q_A, Q_B → still 2^6=64 rows! → "Variable Entered Table"

Variable-Entered Table

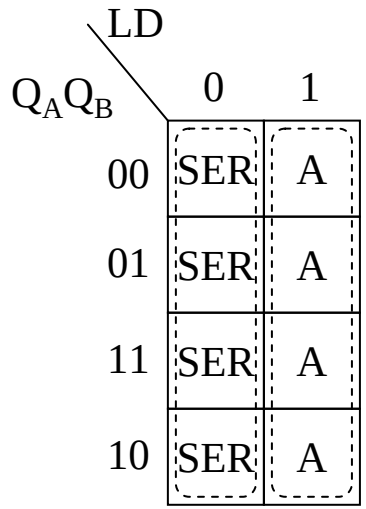
Q_A	Q_B	LD	Q_A	Q_B
0	0	0	SER	0
0	0	1	A	B
0	1	0	SER	0
0	1	1	A	B
1	0	0	SER	1
1	0	1	A	B
1	1	0	SER	1
1	1	1	A	B

Steps 2-6

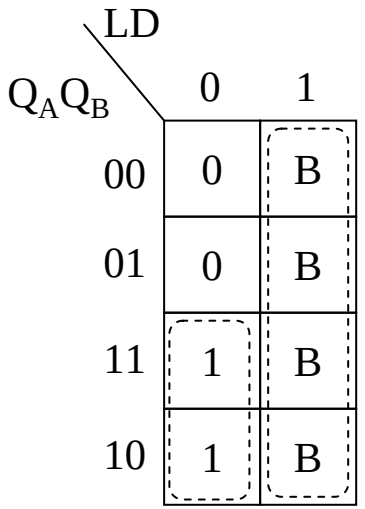
- Step 2: state minimization – not relevant
- Step 3: state assignment – not relevant
- Step 4: Transit/output table – we already have
- Step 5: Choose f/f – D f/f
- Step 6: Excitation table – same as transit table

Steps 7-8: Excitation/Output Eqs. (Variable Entered Map)

D_A



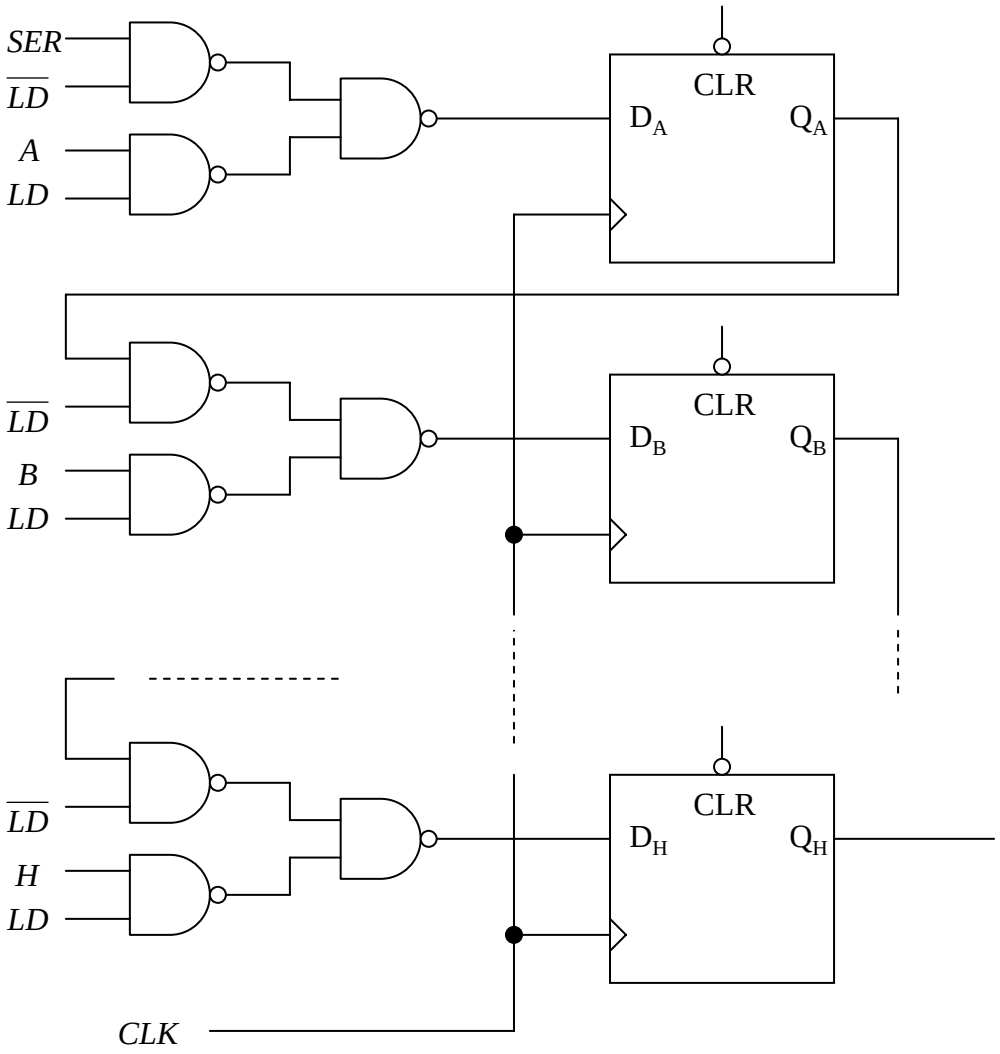
D_B



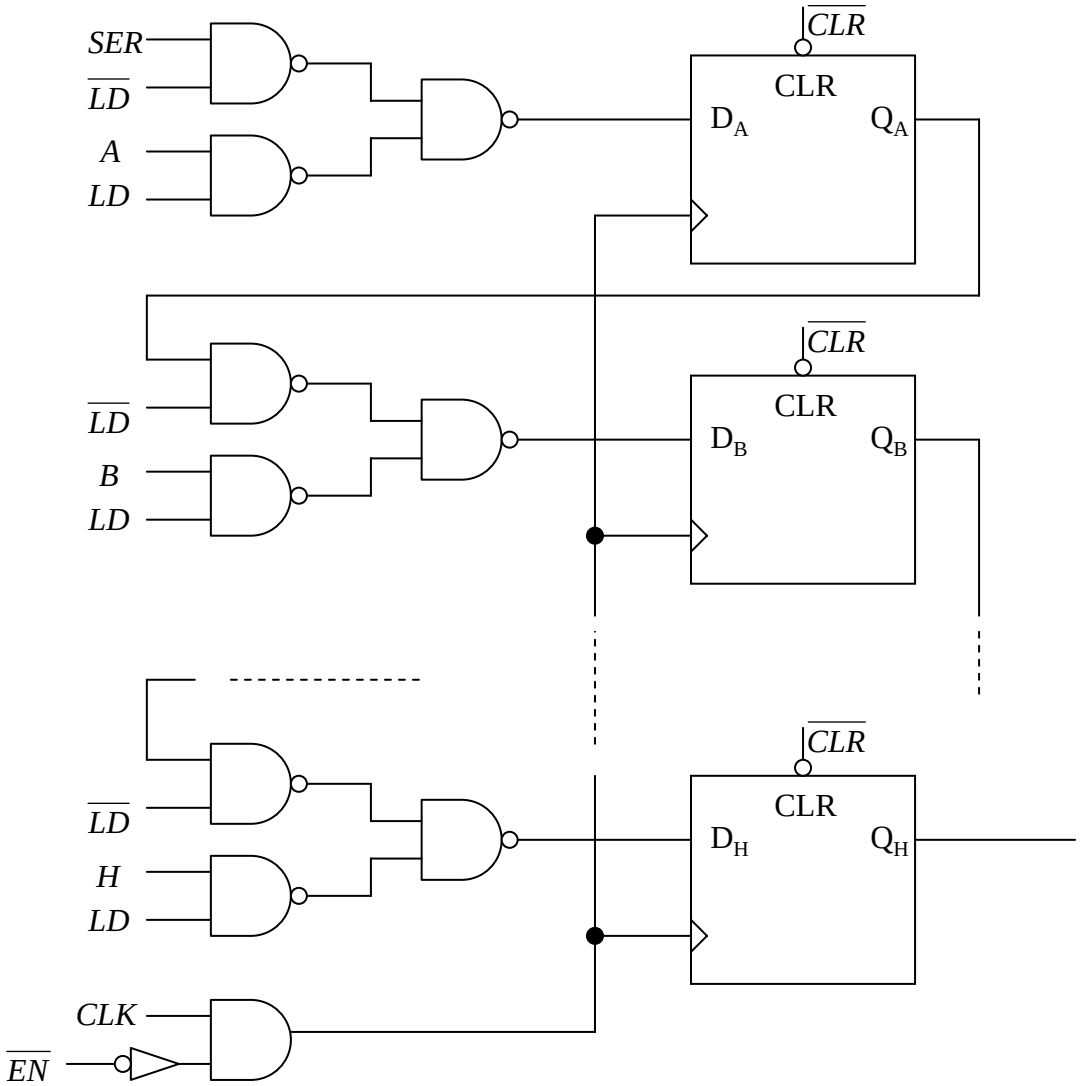
$$D_A = \overline{LD} \cdot SER + LD \cdot A$$

$$D_B = \overline{LD} \cdot Q_A + LD \cdot B$$

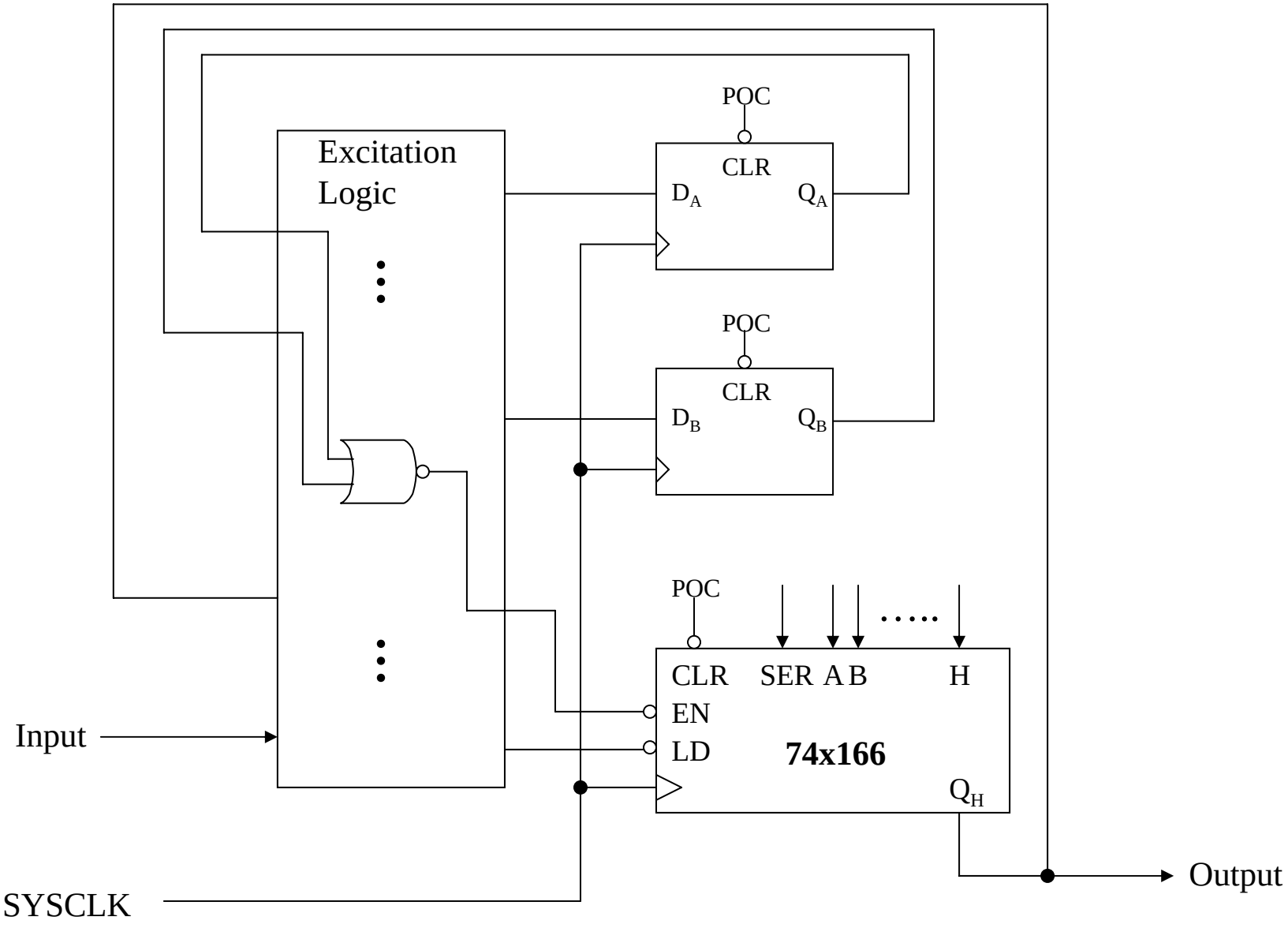
Steps 9: Logic Diagram



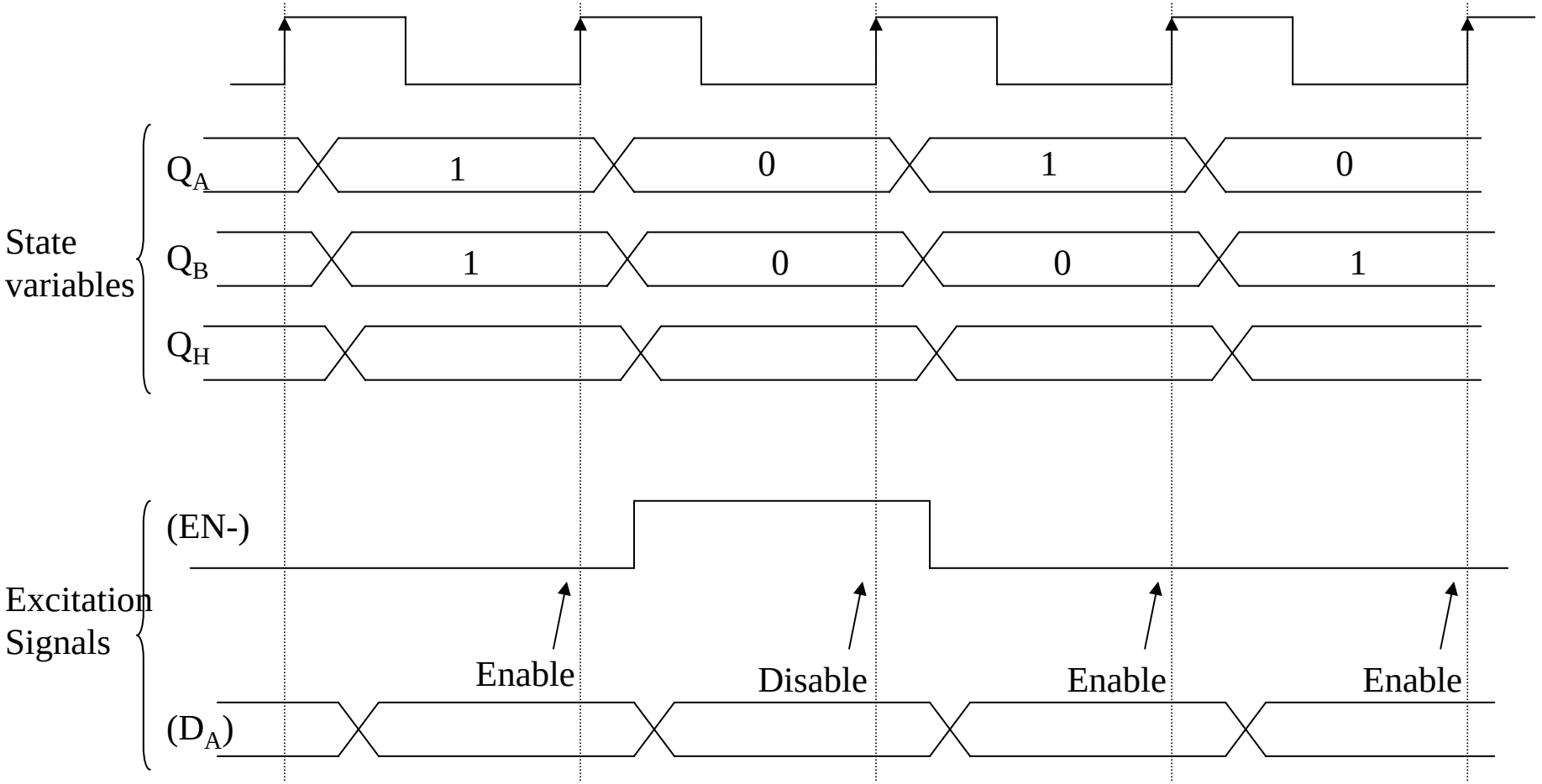
Steps 9: Logic Diagram



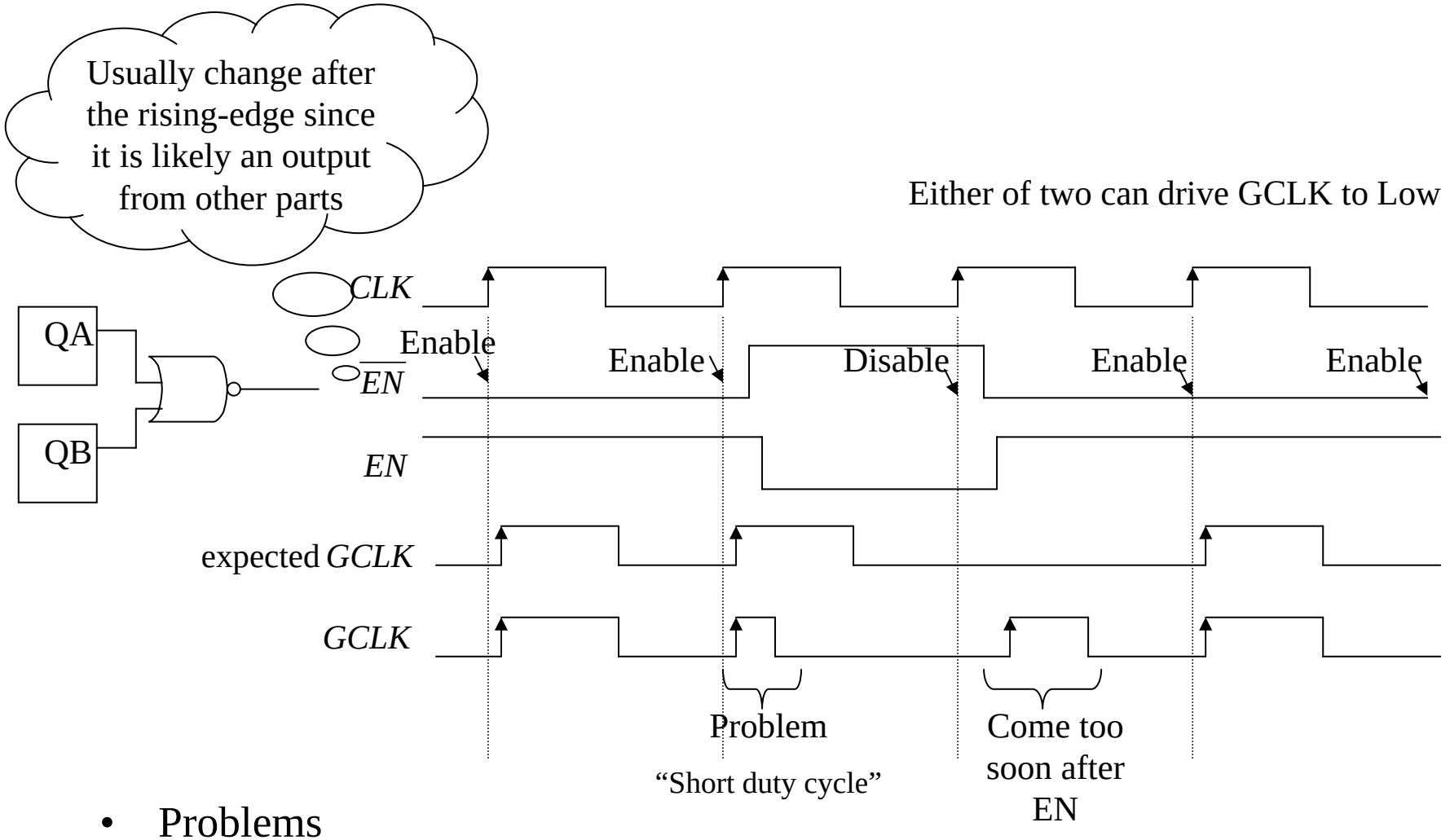
Remind (General Sequential Circuit Structure)



Remind (General Sequential Circuit Timing)



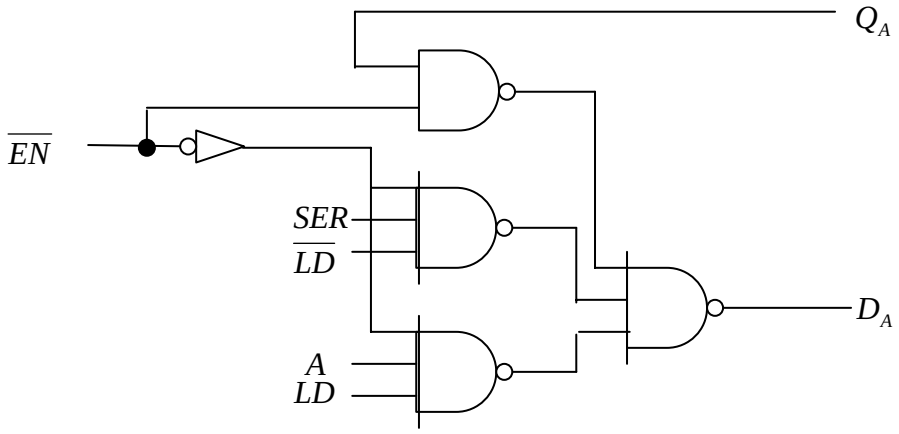
Gating the CLK



- Problems
  - Short GCLK pulse – dependent on delay
  - Comes too soon (or late) after asserting enable

Different approach to holding

- Delay in CLK line is not good design practice
  - Puts in CLK skew
  - All f/f CLKs don't triggered at same time → eliminates the good points to use synchronous design (we can ignore a lot of difficult timing issues)
- Desired: "Synchronous function-enable input"
  - EN should sampled along with data at CLK edge

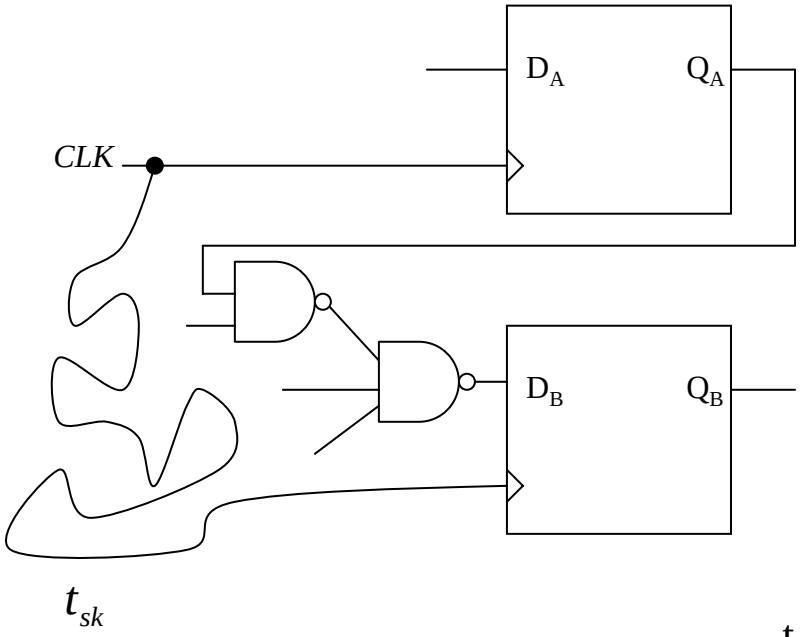


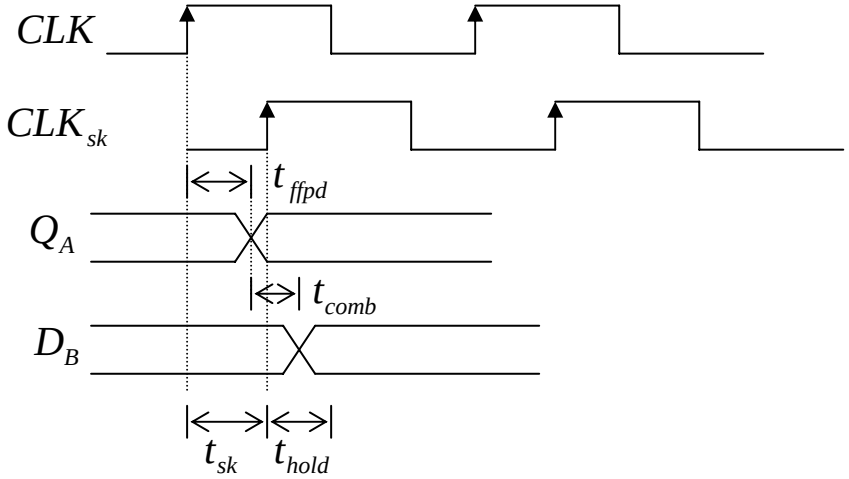
Clock Skew

- Clock skew: difference between arrival times of CLK at different devices
- Caused by
  - Gating in the CLK line
  - Delay along long lines (1ns/ft – speed of light) – CAD serial routing

Clock Skew

- Problem
  - If t_{sk} too long – CLK edge get to B f/f after D_B changes => Wrong operations
  - In general – hold time on D_B can be violated





$$t_{ffpd}(\min) + t_{comb}(\min) > t_{sk} + t_h$$

t_{sk} max?: use timing specs

x74 (25ns) + x00 (9ns) + x10 (9ns) – x74hold (5ns) = 38ns