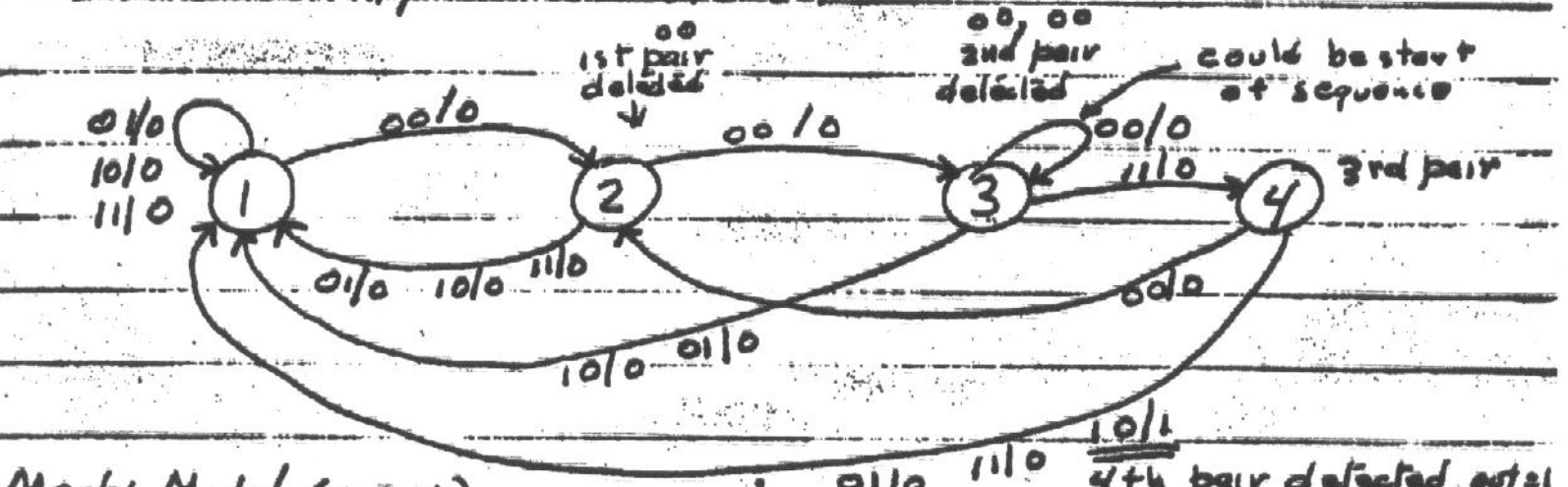
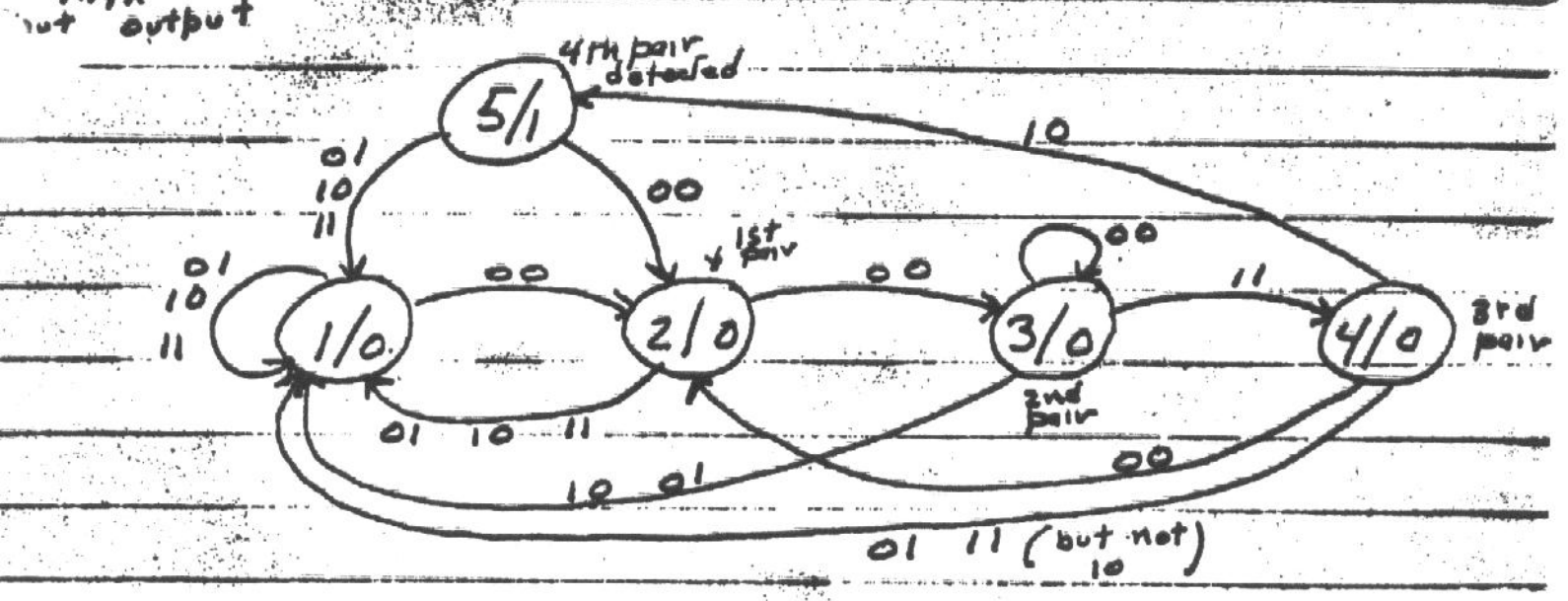
M required to detect occurrence of sequence of pairs
, 00, 11, 10 on the two inputs and to output 1
during final combination of successfully detected sequence,
Assume initially in reset state.



Mealy Model (2 F-F's)



Moore Model (3 F-F's): has one more state indicating desired sequence was observed.

Mealy less states but not necessarily less F-F's.

For Mealy Machine:

State transition table

PS	NS (inputs x_1, x_2)				Output Z			
	00	01	11	10	00	01	11	10
1	2	1	1	1	0	0	0	0
2	3	1	1	1	0	0	0	0
3	3	1	4	1	0	0	0	0
4	2	1	1	1	0	0	0	1

(4 states ⇒ 2 FF's)
(will use D FF's)

For Moore Machine:

State Transition table

PS	NS (inputs x_1, x_2)				Output Z
	00	01	11	10	
1	2	1	1	1	
2	3	1	1	1	
3	3	1	4	1	
4	2	1	1	5	
5	2	1	1	1	1

For Mealy

Make state assignment to get state table:

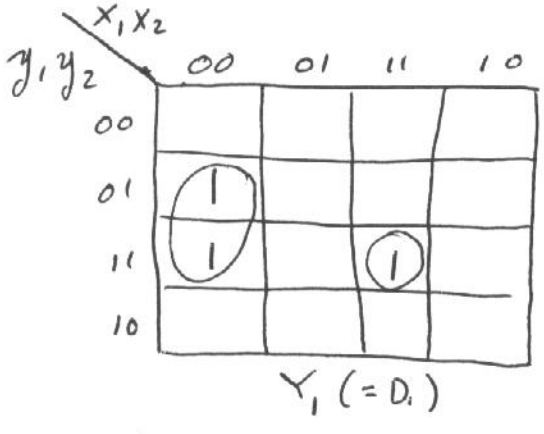
choose: state 1, $y_1 y_2 = 00$ 3 = 11 (not unique choices)

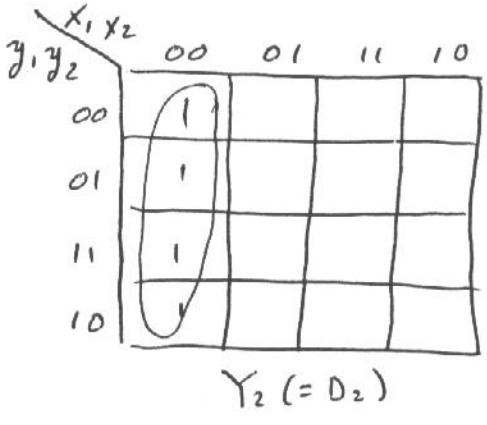
" 2 = 01 4 = 10

	PS	NS (Y_1, Y_2) inputs X_1, X_2				output			
	$y_1 y_2$	00	01	11	10	00	01	11	10
①	00	01	00	00	00				
②	01	11	00	00	00				
③	11	11	00	10	00				
④	10	01	00	00	00				

directly get $Z = X_1 \bar{X}_2 y_1 \bar{y}_2$ (1)

Output (excitation) functions: note using D FF's so that the D inputs are the desired NS's.

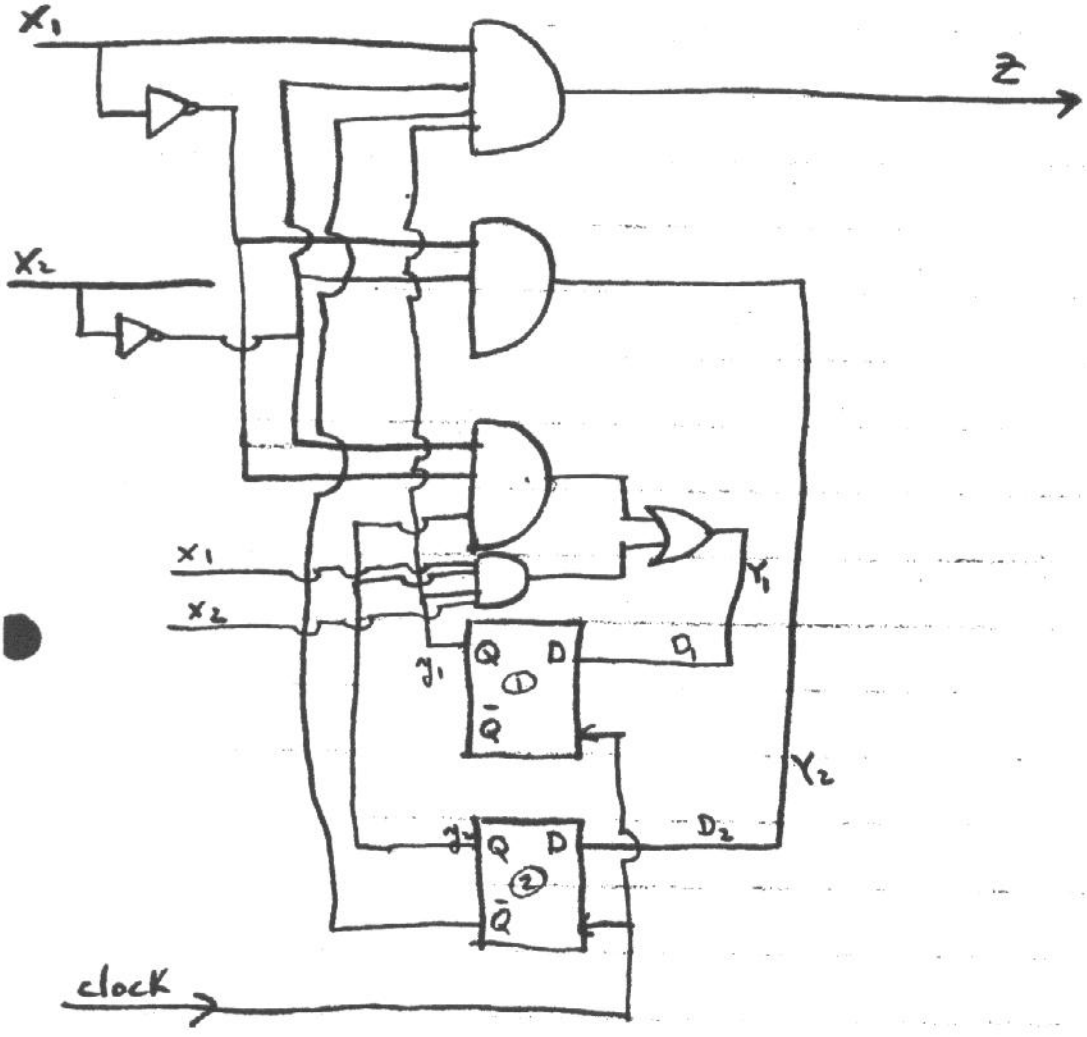




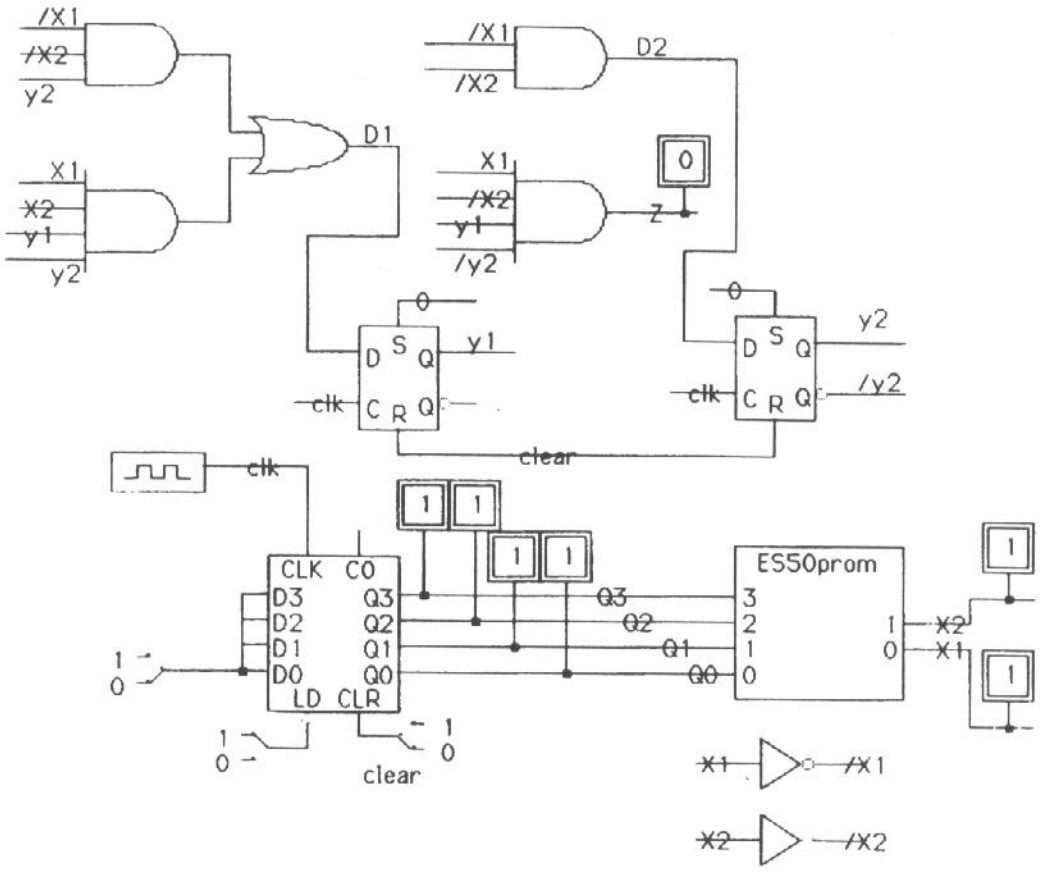
$D_1 = \bar{X}_1 \bar{X}_2 y_2 + X_1 X_2 y_1 y_2$ $D_2 = \bar{X}_1 \bar{X}_2$ Z = above

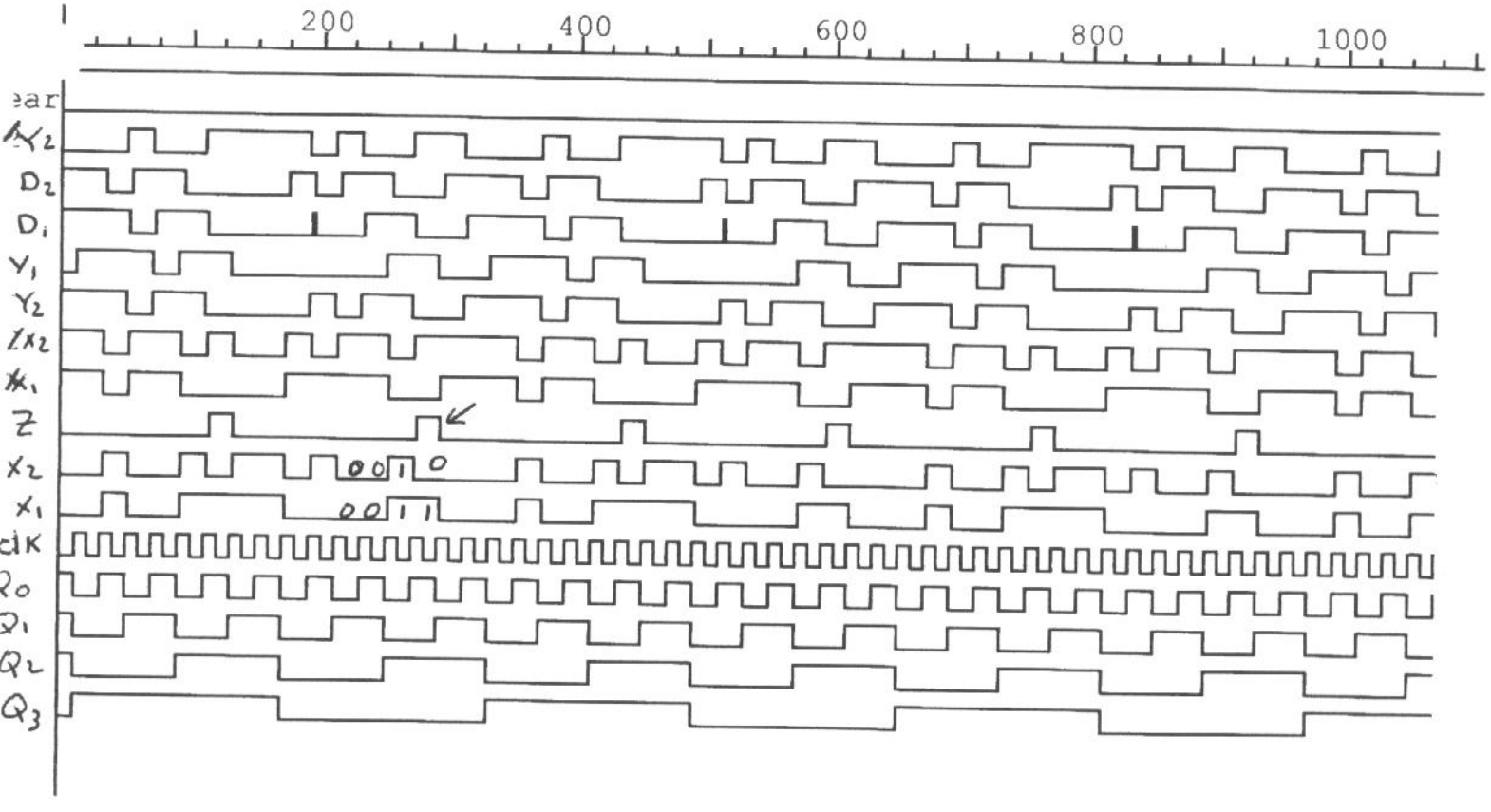
Now have recipe for the circuit:

I the final circuit is:



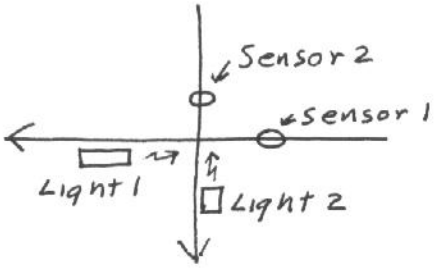
Mealy Machine to detect sequence of pairs X1,X2= 0,0 0,0 1,1 1,0 Output=Z

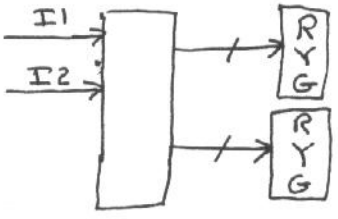




Traffic controller M:

Rural intersection of two one-way roads N→S, E→W.





Light 1 controls east-west traffic
" 2 " north-south "

Sensor 1 indicates car(s) waiting at or approaching intersection from E-W direction so
I1 = 0 ⇒ no cars in E-W direction
I1 = 1 ⇒ one or more cars waiting or traveling in E-W direction

Sensor 2 operates similar to #1 but in N-S direction.

Each light has Red, Yellow, Green signals. Only 4 combinations of the six lights control traffic. Green on allows flow in one direction while Red on stops flow in other direction. When Green about to change to Red, Yellow turned on briefly.

List for reference the 4 combinations (ie the 4 states)

State	Light 1	Light 2	Function
S1	Green	Red	Traffic flows E→W
S2	Yellow	Red	Lights changing (time set by clock)
S3	Red	Green	Traffic flow N→S
S4	Red	Yellow	Lights changing

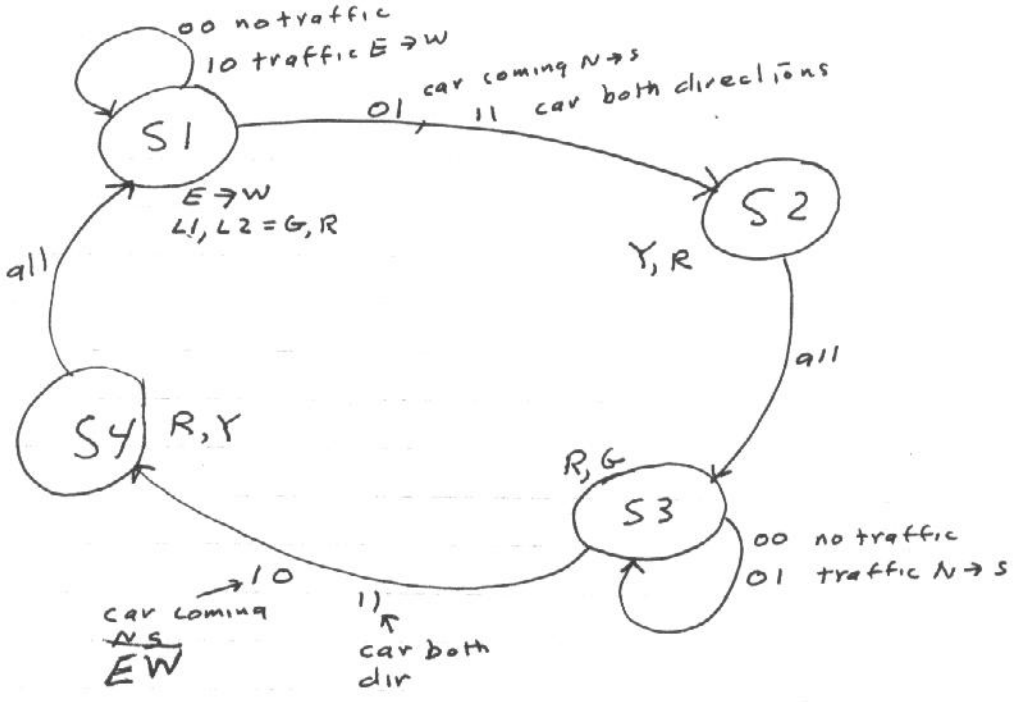
If in S1 (traffic E→W) remain there as long as I1I2 = 00 (no traffic) or I1I2 = 10 (traffic in E-W direction). If I1I2 = 01 (car(s) from N-S or I1I2 = 11 (cars coming both directions) then S1→S2. S2 then →S3 regardless of sensors.

If in S3 (traffic N→S) remain there as long as I1I2 = 00 (no traffic) or I1I2 = 01 (traffic in N→S direction). If I1I2 = 10 (cars from E→W) or I1I2 = 11 (cars coming both directions) then S3 → S4. S4 then S1 regardless of sensors.

Summarize:

EW I_1	NS I_2	
0	0	no traffic
1	0	traffic E→W or car approaching
0	1	traffic N→S or car approaching
1	1	traffic in or approaching both intersections

Next state depends on current state and inputs. Depending on inputs may stay in S1 or S3 indefinitely. States S2, S4 change S2→S3, S4→S1 for any input combinations.
Come back later for signal lights.



state flow graph

present state	next state for I_1, I_2 = 00	01	10	11
S1	S1	S2	S1	S2
S2	S3	S3	S3	S3
S3	S3	S3	S4	S4
S4	S1	S1	S1	S1

outputs
R1, Y1, G1, R2, Y2, G2
(leave for now)

# of F-F's: 4 states ⇒ 2 F-F's with no unused states

type of F-F's: for now stick to D Types

State assignments: need 4 assignments for $q_1 q_0$. No unique choice:

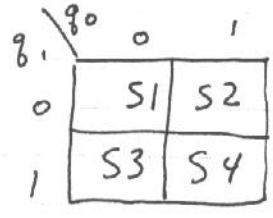
first assignment for S1 — 4 possibilities
next " for S2 — 3 " (one already used)
" " " S3 — 2 " (two " ")
" " " S4 — 1 " (three " ")

# of possible assignments = 4! = 24 — not all really different in terms of final topology - e.g. can swap the 2 FF's since only the inputs for the F-F's would be changed

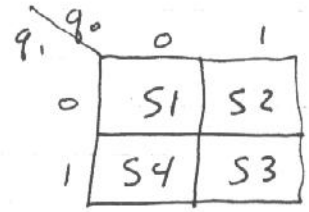
Turns out only three different assignments result in different circuits. They are:

	① $q_1 q_0$	② $q_1 q_0$	③ $q_1 q_0$
S1	0 0	0 0	0 0
S2	0 1	0 1	1 1
S3	1 0	1 1	1 0
S4	1 1	1 0	0 1

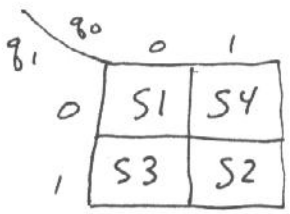
can be depicted as



①



②



③

State assignments:

(See Hill + Peterson - Intro to Sw. Th. + Log. Des 277)

1. The # of possible state assignments, S, for N states realized with M state variables is:

$$S = \frac{(2^M)!}{(2^M - N)!}$$

for 4 states ($N = 4$) and 2 state variables ($M = 2$)

get $S = \frac{(2^2)!}{(2^2-4)!} = \frac{4!}{1} = 24$ $\quad (0! = 1)$

2. The # of <u>non-equivalent</u> state assignments is:

$$S_{\substack{non \\ equ}} = \frac{(2^M - 1)!}{(2^M - N)!\, M!}$$ which here is $\frac{(2^2-1)!}{(2^2-4)!(2)!} = 3$

Note # increases rapidly:

	No. of states (N)	# state var (M)	# Non-equ. states Snon
(1 FF)	2	1	1
(2 FF)	3	2	3
	4	2	3
(3 FF) up to 8 states	5	3	140
	6	3	420
	7	3	840
	8	3	840
(4 FF) up to 16 states	9	4	$10.81 \cdot 10^6$
	10	4	$75.67 \cdot 10^6$
	16	4	$54.48 \cdot 10^9$
	20	5	$143.14 \cdot 10^{21}$

For 4 states A B C D and two state variables the possible are

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	$q_1 q_0$	$q_1 q_0$	$q_1 q_0$	$q_1 q_0$	$q_1 q_0$	$q_1 q_0$	$q_1 q_0$	$q_1 q_0$
A	00	00	00	00	00	00	01	01
B	01	01	10	10	11	11	10	10
C	10	11	01	11	01	10	11	00
D	11	10	11	01	10	01	00	11

	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
A	01	01	01	01	10	10	10	10
B	11	11	00	00	11	11	00	00
C	10	00	10	11	00	01	11	01
D	00	10	11	10	01	00	01	11

	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)
A	10	10	11	11	11	11	11	11
B	01	01	00	00	01	01	10	10
C	11	00	01	10	00	10	00	01
D	00	11	10	01	10	00	01	00

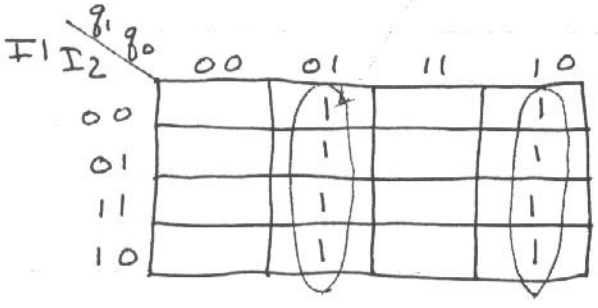
Not all are really different: two assignments are considered equivalent if they produce the same results in implementing the logic - This happens when one assignment is the complement of another e.g. (1) is comp of (24) (2) is comp of (23) etc. Also have equivalence when columns are interchanged e.g (1) and (3)

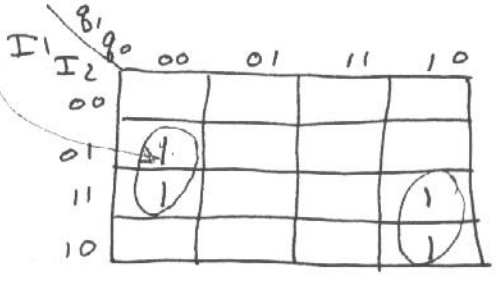
Design using assignment #1 (and D FF's)

q_1 \ q_0	0	1
0	S1	S2
1	S3	S4

	present state	next state for $I_1 I_2$ =			
		0 0	0 1	1 0	1 1
	$q_1\ q_0$	$Q_1\ Q_0$	$Q_1\ Q_0$	$Q_1\ Q_0$	$Q_1\ Q_0$
(S1)	0 0	0 0	0 1	0 0	0 1
(S2)	0 1	1 0	1 0	1 0	1 0
(S3)	1 0	1 0	1 0	1 1	1 1
(S4)	1 1	0 0	0 0	0 0	0 0

etc etc





Q_1 (= D1)

$D1 = \bar{q}_1 q_0 + q_1 \bar{q}_0$

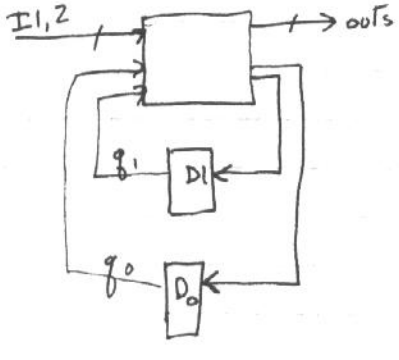
$= q_1 \oplus q_0$

Q_0 (= D_0)

$D_0 = I_2 \bar{q}_1 \bar{q}_0 + I_1 q_1 \bar{q}_0$

$= \bar{q}_0 [I_2 \bar{q}_1 + I_1 q_1]$

reminder



gates needed: e.g.

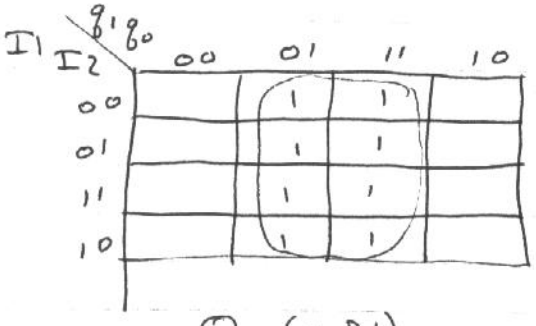
1) 1 EXOR or 2-2in AND + 1-OR

2) 2-3in AND + 1-OR
or 3-2in AND + 1-OR

Design using assignment #2 (and D FF's)

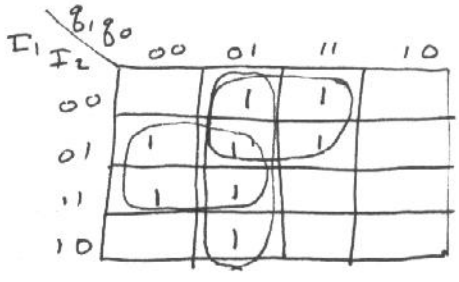
q_1 \ q_0	0	1
0	S1	S2
1	S4	S3

present state	$q_1\ q_0$	next state for $I_1 I_2$ = 00	01	10	11
		$Q_1\ Q_0$	$Q_1\ Q_0$	$Q_1\ Q_0$	$Q_1\ Q_0$
(S1)	0 0	0 0	0 1	0 0	0 1
(S2)	0 1	1 1	1 1	1 1	1 1
(S3)	1 1	1 1	1 1	1 0	1 0
(S4)	1 0	0 0	0 0	0 0	0 0



Q_1 (= D1)

$D_1 = q_0$



Q_0 (= D_0)

$$D_0 = \bar{I}_1 q_0 + I_2 \bar{q}_1 + \bar{q}_1 q_0$$
$$= q_0(\bar{q}_1 + \bar{I}_1) + I_2 \bar{q}_1$$
$$= q_0(\overline{I_1 \cdot q_1}) + I_2 \bar{q}_1$$

gates needed: e.g.

1) 3 - 2 in AND plus 2 OR + one inv (?)

or

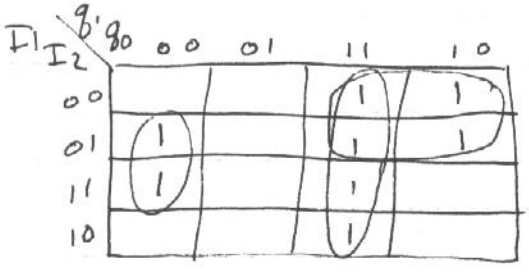
2) 2 - 2 in AND + 2 OR + one inv

or 3) one - 2 in NAND + 2 - 2 in AND + 1 OR

Design using assignment #3 (and DFF's)

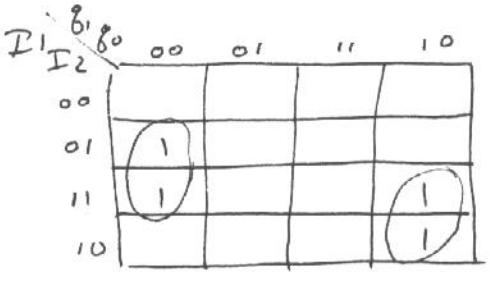
q_1 \ q_0	0	1
0	S1	S4
1	S3	S2

	present state	next state for $I_1 I_2$ =			
		0 0	0 1	1 0	1 1
	$q_1 q_0$	$Q_1 Q_0$	$Q_1 Q_0$	$Q_1 Q_0$	$Q_1 Q_0$
(S1)	0 0	0 0	1 1	0 0	1 1
(S2)	1 1	1 0	1 0	1 0	1 0
(S3)	1 0	1 0	1 0	0 1	0 1
(S4)	0 1	0 0	0 0	0 0	0 0



Q_1 (= D_1)

$D_1 = I_2 \bar{q}_1 \bar{q}_0 + q_1 q_0 + \bar{I}_1 q_1$



Q_0 (= D_0)

$D_0 = I_2 \bar{q}_1 \bar{q}_0 + I_1 q_1 \bar{q}_0$

gates needed, e.g.

1) one 3-in AND (share both D_1, D_0)
   two 2-in AND + 2 OR + inv(?)

2) one 3-in AND + one OR

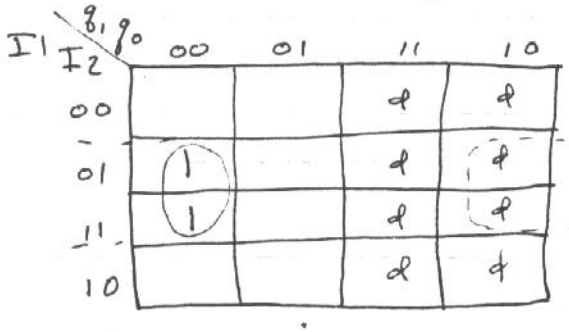
Design using assignment #3 (and J-K FF's)

q_1 \ q_0	0	1
0	S1	S4
1	S3	S2

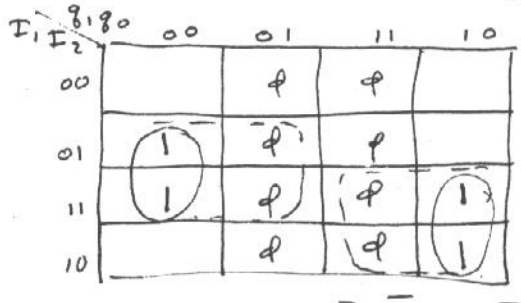
need excitation table

Q_n	Q_{n+1}	J_n	K_n
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

	Present State q_1 q_0	Inputs I_1 I_2	Next State Q_1 Q_0	required excitations J_1 K_1	J_0 K_0
(S1)	0 0	0 0	0 0 S1	0 ϕ	0 ϕ
	0 0	0 1	1 1 S2	1 ϕ	1 ϕ
	0 0	1 0	0 0 S1	0 ϕ	0 ϕ
	0 0	1 1	1 1 S2	1 ϕ	1 ϕ
(S2)	1 1	0 0	1 0 S3	ϕ 0	ϕ 1
	1 1	0 1	1 0 S3	ϕ 0	ϕ 1
	1 1	1 0	1 0 S3	ϕ 0	ϕ 1
	1 1	1 1	1 0 S3	ϕ 0	ϕ 1
(S3)	1 0	0 0	1 0 S3	ϕ 0	0 ϕ
	1 0	0 1	1 0 S3	ϕ 0	0 ϕ
	1 0	1 0	0 1 S4	ϕ 1	1 ϕ
	1 0	1 1	0 1 S4	ϕ 1	1 ϕ
(S4)	0 1	0 0	0 0 S1	0 ϕ	ϕ 1
	0 1	0 1	0 0 S1	0 ϕ	ϕ 1
	0 1	1 0	0 0 S1	0 ϕ	ϕ 1
	0 1	1 1	0 0 S1	0 ϕ	ϕ 1

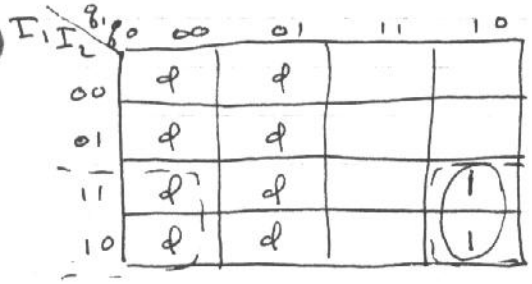


$J_1 = I_2 \bar{q}_1 \bar{q}_0$
or $= I_2 \bar{q}_0$



J_0

$J_0 = I_2 \bar{q}_1 \bar{q}_0 + I_1 q_1 \bar{q}_0$
(note available)
or $= I_2 \bar{q}_1 + I_1 q_1$



$K_1 = I_1 q_1 \bar{q}_0$
or $= I_1 \bar{q}_0$

by inspection $K_0 = 1$

Gates needed
2-3 in AND plus 1 OR

Outputs: have ignored so far -
Original statement specified what lights are on for each condition. Repeat here

State	Lights on	
S1	G1	R2
S2	Y1	R2
S3	R1	G2
S4	R1	Y2

More than one way to design:

1. Conventional

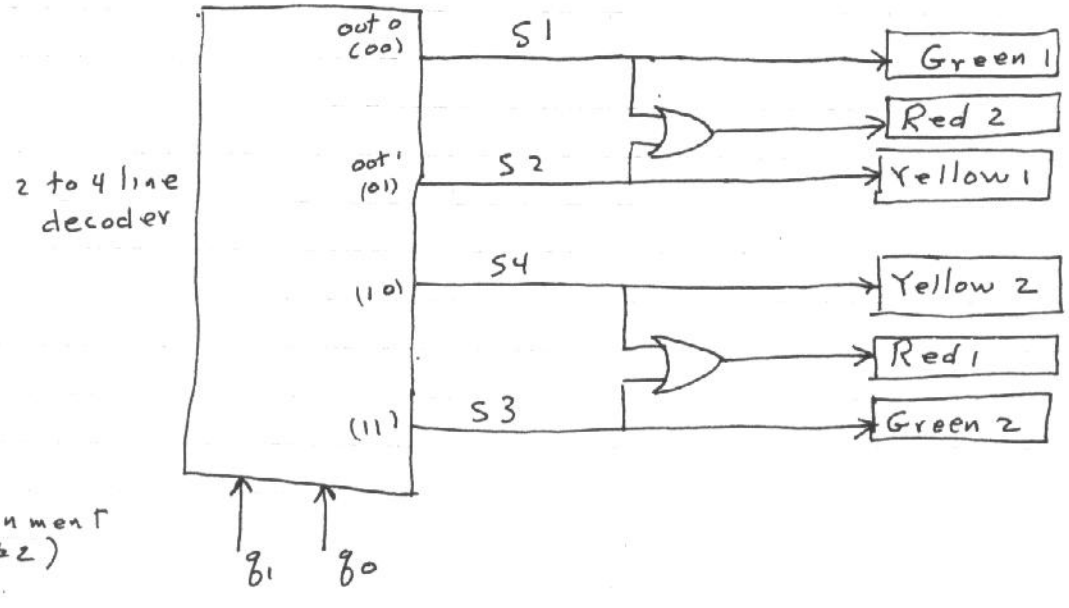
		q_1	q_0	G1	Y1	R1	G2	Y2	R2
S1	depends on state assign. →	-	-	1					1
S2		-	-		1				1
S3		-	-			1	1		
S4		-	-			1		1	

use SSI, MUX, ROM etc for each of the lights

2. State decoder

Use the $q_1 q_0$ values which define the states S1 → S4 as address lines on a decoder - selected line goes high and then combine as per table above - to wit:

$G1 = S1$
$G2 = S3$
$Y1 = S2$
$Y2 = S4$
$R1 = S3 + S4$
$R2 = S1 + S2$



(assignment #2)

"One hot" design :- One FF/state. Easy debuggin/maintenance but wasteful of FF's and must pay attention to unused states.

Each FF on indicates a particular state and only one FF on at a time. No state assignment issue involved. Generally use D type FF's.

state table :

present state	next state S^+ for I1I2 = 00	01	10	11
S1	S1	S2	S1	S2
S2	S3	S3	S3	S3
S3	S3	S3	S4	S4
S4	S1	S1	S1	S1

these are available now as indiv. FF outputs

note e.g. that S3 becomes next state if either the present state is S3 and inputs I1I2 are 00 or 01, or if the present state is S2 — hence can write

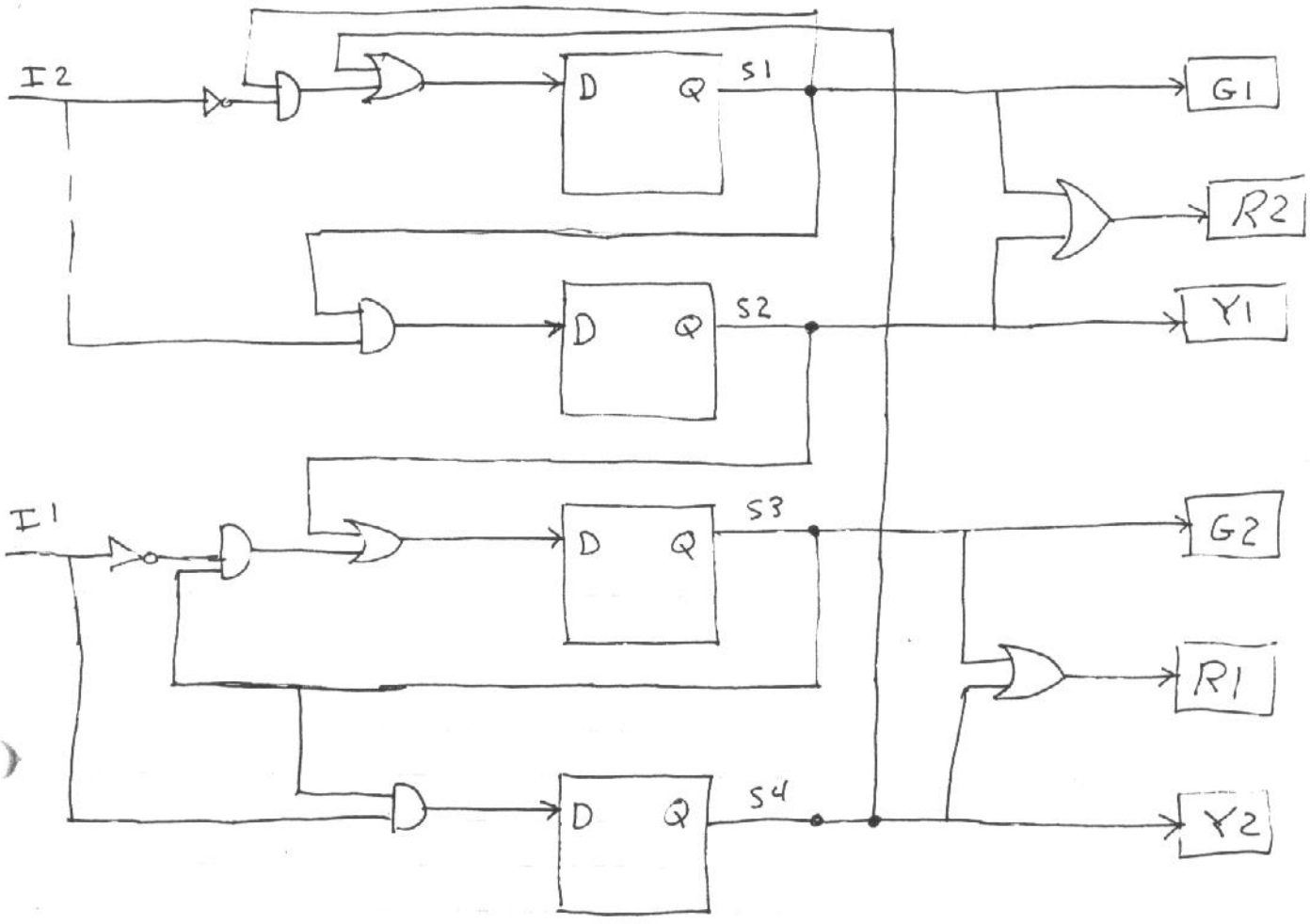
$$S1^+ = S1\cdot(\bar{I}_1\bar{I}_2) + S1\cdot(I_1\bar{I}_2) + S4$$
$$= S1[\bar{I}_2(\bar{I}_1+I_1)] + S4 = S1\cdot\bar{I}_2 + S4$$

$$S2^+ = S1\cdot(\bar{I}_1 I_2 + I_1 I_2) = S1\cdot I_2$$

$$S3^+ = S2 + S3\cdot(\bar{I}_1\bar{I}_2 + \bar{I}_1 I_2) = S2 + S3\cdot\bar{I}_1$$

$$S4^+ = S3\cdot(I_1\bar{I}_2 + I_1 I_2) = S3\cdot I1$$

which are recipes for ckt:

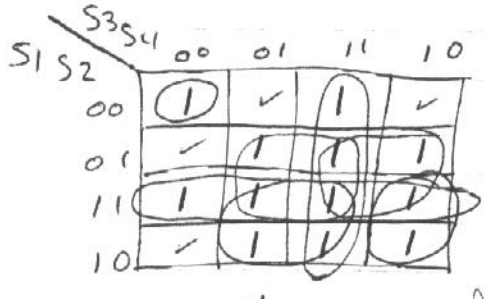


one hot design

Unused states: only 4 of the possible 16 states are used here & obviously should ensure the others are taken care of f.

⇒ Design a "unused state" detector whose output would be used in conjunction with asy. preset/clear to go to a legal state,

eg S1 required gate G1 and R2



✓ = legal state, ie only one "1" in S1S2S3S4

$$U = \bar{S}_1 \bar{S}_2 \bar{S}_3 \bar{S}_4 + S_2 S_3 + S_1 S_4 + S_3 S_4 + S_1 S_3 + S_2 S_4 + S_1 S_2$$

with U triggering preset/clear no unused state would appear for more than a brief time

SM's using PROM's: if have access to PROM burner can give rapid design. Basically takes the place of the logic forming blocks

e.g. Assume a state table like:

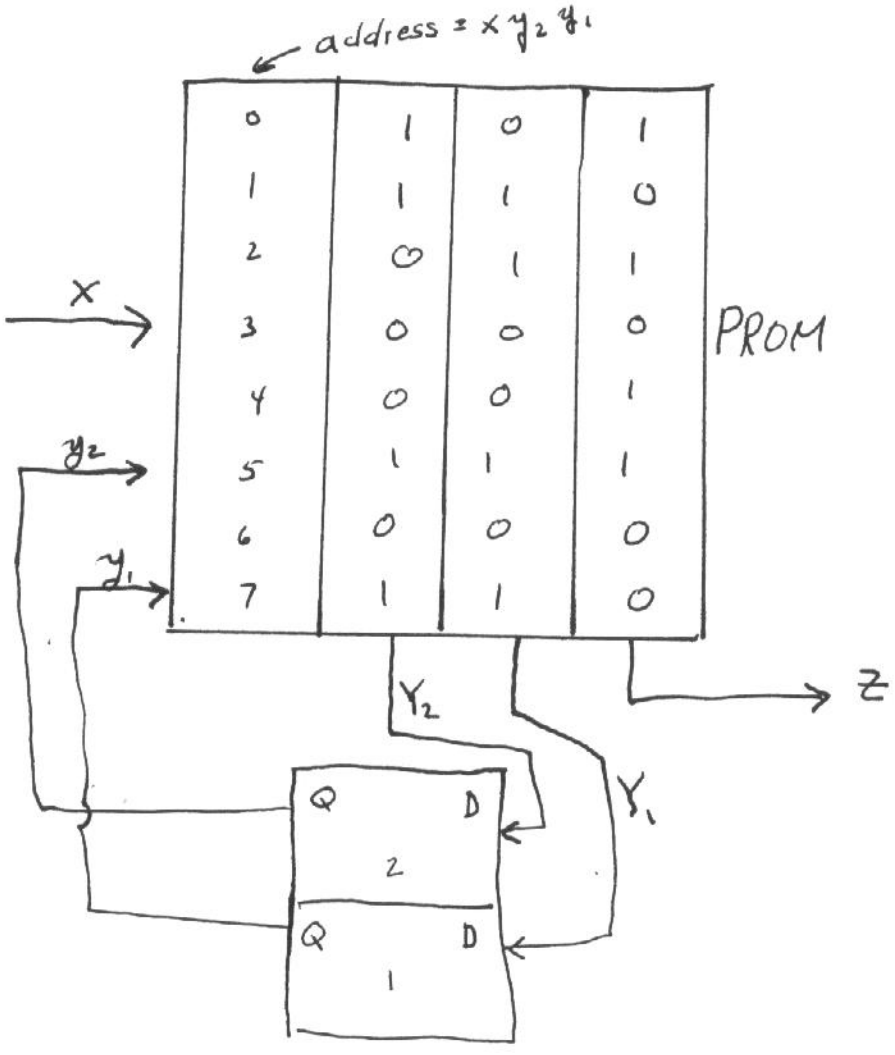
PS	NS/out = z	
	x = 0	x = 1
$y_2\ y_1$	$Y_2\ Y_1$	$Y_2\ Y_1$
0 0	1 0/1	0 0/1
0 1	1 1/0	1 1/1
1 0	0 1/1	0 0/0
1 1	0 0/0	1 1/0

so 3 variables in logic blocks: $y_2\ y_1\ x$

Arrange as PROM addresses/contents

	x y_2 y_1 (PROM address)			Y_2 Y_1 z (contents)		
(0)	0	0	0	1	0	1
(1)	0	0	1	1	1	0
(2)	0	1	0	0	1	1
(3)	0	1	1	0	0	0
(4)	1	0	0	0	0	1
(5)	1	0	1	1	1	1
(6)	1	1	0	0	0	0
(7)	1	1	1	1	1	0

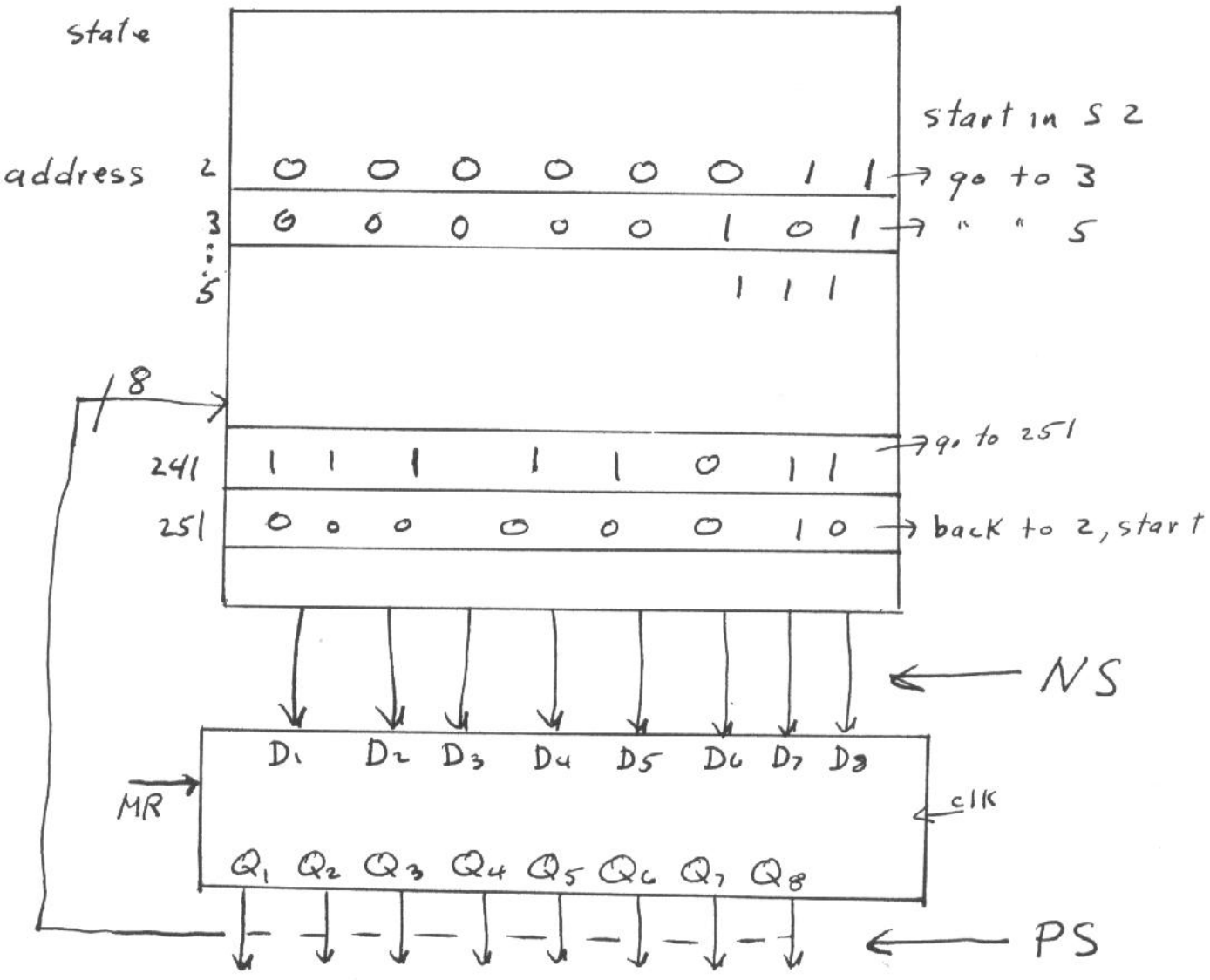
And ckt is then:



PROM size here is 8 words of 3 bits each (quite small).

Ex 2. Build SM to output sequences equal to the prime #'s less than 256. (8 bits)

Primes are: 0, 2, 3, 5, 7, 11, 13, 17, 19
23, 29, 31, 37 41, 43, 47, 53
- - - - - - - 241, 251 (257, 263 next)



(74 273 = 8 D edge FF's used as holding register)

Only certain locations in the 256 x 8 PROM are used - those containing the prime #'s. All unused locations loaded with 2_{10} so that on power up, if not in 2, will go to 2 on clock tick and sequence will start.

0, 2, 3, 5, 7 ···· 241, 251