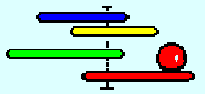
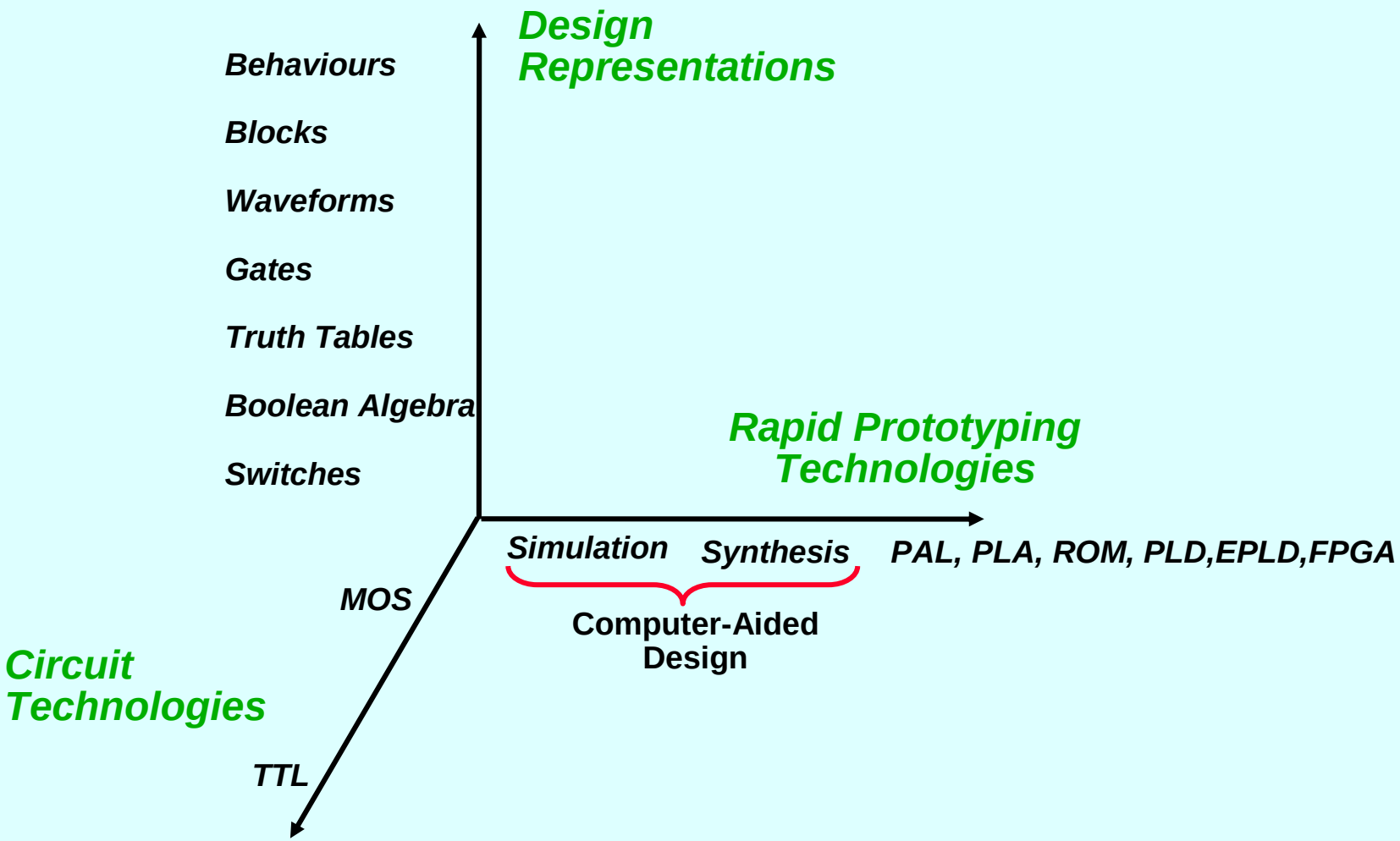
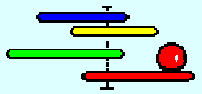


Review for B33DV2-Digital Design

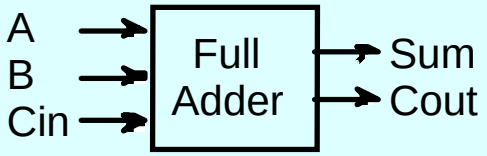
The Elements of Modern Digital Design



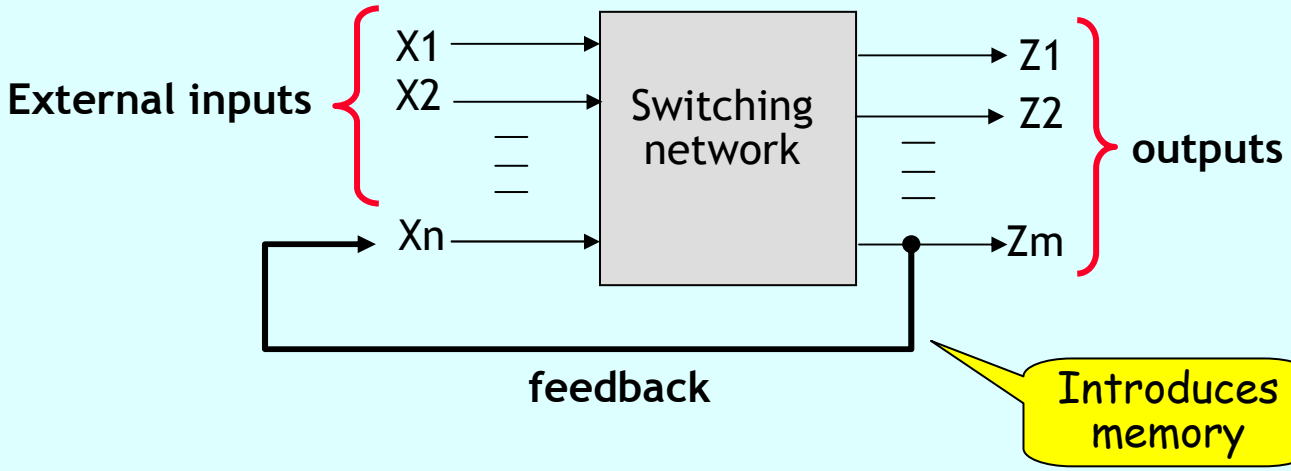
Combinational vs. Sequential Logic



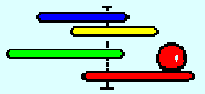
- **Combinational logic**
  - ➔ no feedback among inputs and outputs
  - ➔ outputs are a pure function of the inputs
  - ➔ e.g., full adder circuit:
  - ➔ (*A, B, Carry In*) mapped into (*Sum, Carry Out*)



- **Sequential logic**
  - ➔ Network implemented from switching elements or logic gates. The presence of feedback distinguishes between sequential and combinational networks

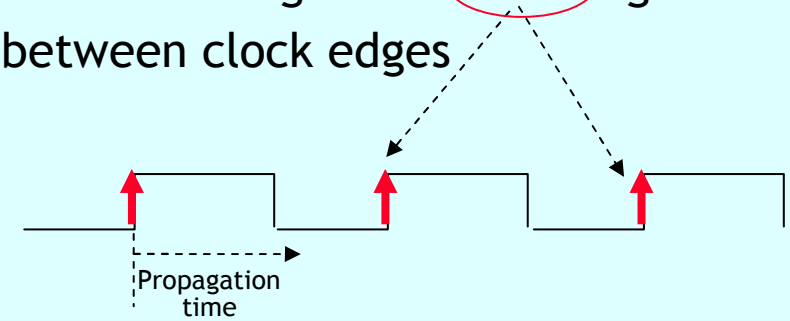


Sequential Systems

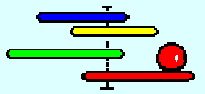


- **Sequential logic**
  - ➔ network typically has only a limited number of unique configurations
    - ➢ these are called **states**
    - ➢ e.g., a simple traffic light controller sequences infinitely through four states
  - ➔ outputs depend on inputs and the entire history of execution!
    - ➢ output and new state is a function of the inputs and the old state
    - ➢ outputs = F(inputs, present_state)
  - ➔ includes components to remember the current state
    - ➢ i.e., the fed back inputs are the state

- **Synchronous systems**
  - ➔ internal state changes synchronized to edges of a **clock** signal
  - ➔ effect of changes propagate between clock edges
    - ➢ as in a standard Intel PC

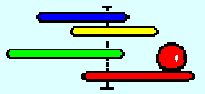


Representations used in Digital Design



- **Boolean algebra**
  - ➔ Mathematical representation of static binary logic
- **Truth tables**
  - ➔ Tabular representation of ALL input/output combinations
- **Minterms,(maxterms)**
  - ➔ Boolean algebra representations of truth tables
- **Logic diagrams**
  - ➔ Schematic diagram of logic system
- **Timing diagrams**
  - ➔ Shows timing relationships between system units
- **Karnaugh maps**
  - ➔ Geometrical representation of truth table data
- **State transition diagrams**
  - ➔ For sequential systems shows relationship between the states of a system
- **ASM charts**
  - ➔ For sequential systems shows sequencing between the states of a system

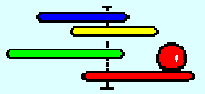
Rules of Boolean Algebra



- 1. Commutative Law
  - ➔ $A \cdot B = B \cdot A$
  - ➔ $A + B = B + A$
- 2. Associate Law
  - ➔ $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
  - ➔ $(A + B) + C = A + (B + C)$
- 3. Distributive Law
  - ➔ $(A + B) \cdot C = (A \cdot C) + (B \cdot C)$
  - ➔ $(A \cdot B) + C = (A + C) \cdot (B + C)$
- 4. Identities
  - ➔ $A + 0 = A$
  - ➔ $A \cdot 1 = A$
- 5.
  - ➔ $A + 1 = 1$
  - ➔ $A \cdot 0 = 0$
- 6.
  - ➔ $A + A = A$
  - ➔ $A \cdot A = A$
- 7.
  - ➔ $A + (A') = 1$
  - ➔ $A \cdot (A') = 0$
- 8. Inverse
  - ➔ $(A')' = A$
- 9. De Morgan's Theorem
  - ➔ $(A+ B)' = (A') \cdot (B')$
  - ➔ $(A \cdot B)' = (A') + (B')$

- Evaluation order
  - ➔ **Parentheses**
  - ➔ **' (NOT)**
  - ➔ *** (AND)**
  - ➔ **+ (OR)**

Truth Table to SOP Form



- **Sum-0f-Products (SOP) form can be written directly from truth table.**

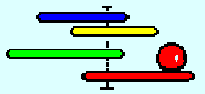
A	B	C	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}\,B\,C$
1	0	0	1	$A\,\bar{B}\,\bar{C}$
1	0	1	1	$A\,\bar{B}\,C$
1	1	0	1	$A\,B\,\bar{C}$
1	1	1	1	$A\,B\,C$

$$F(A,B,C) = \bar{A}\,B\,C + A\,\bar{B}\,\bar{C} + A\,\bar{B}\,C + A\,B\,\bar{C} + A\,B\,C$$

minterms

Note that each term has ALL variables present. If a product term has ALL variables present, it is a MINTERM.

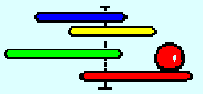
Minterm Notation



- **Each line in a truth table represents a Minterm.**

Row No.	A B C	Minterms
0	0 0 0	$\bar{A}\,\bar{B}\,\bar{C} = m_0$
1	0 0 1	$\bar{A}\,\bar{B}\,C = m_1$
2	0 1 0	$\bar{A}\,B\,\bar{C} = m_2$
3	0 1 1	$\bar{A}\,B\,C = m_3$
4	1 0 0	$A\,\bar{B}\,\bar{C} = m_4$
5	1 0 1	$A\,\bar{B}\,C = m_5$
6	1 1 0	$A\,B\,\bar{C} = m_6$
7	1 1 1	$A\,B\,C = m_7$

Logic Functions: From Expressions to Gates

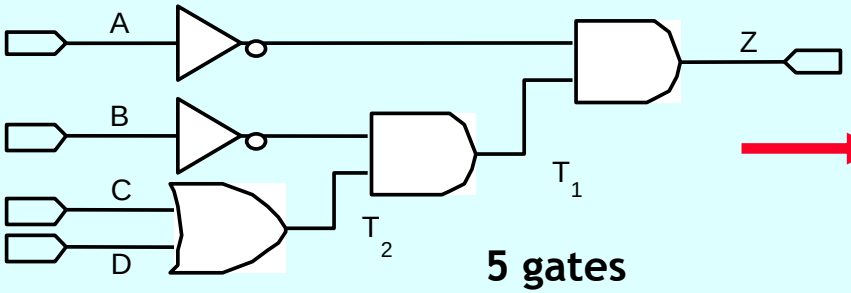


- More than one way to map an expression to gates

$$\text{E.g.,}\quad Z = (\overline{A} \cdot (\overline{B} \cdot (C + D))) = \overline{A} \cdot \overline{B} \cdot (C + D)$$

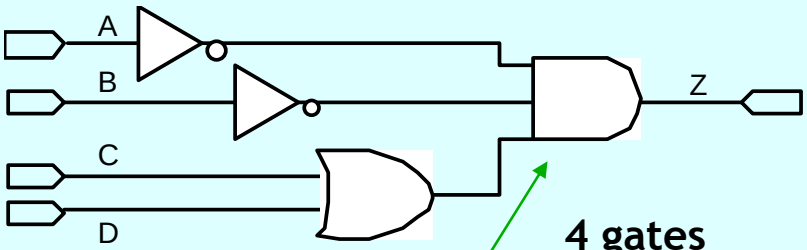
T2 spans $(\overline{B} \cdot (C + D))$; T1 spans $(C + D)$.

Implementation 1



5 gates

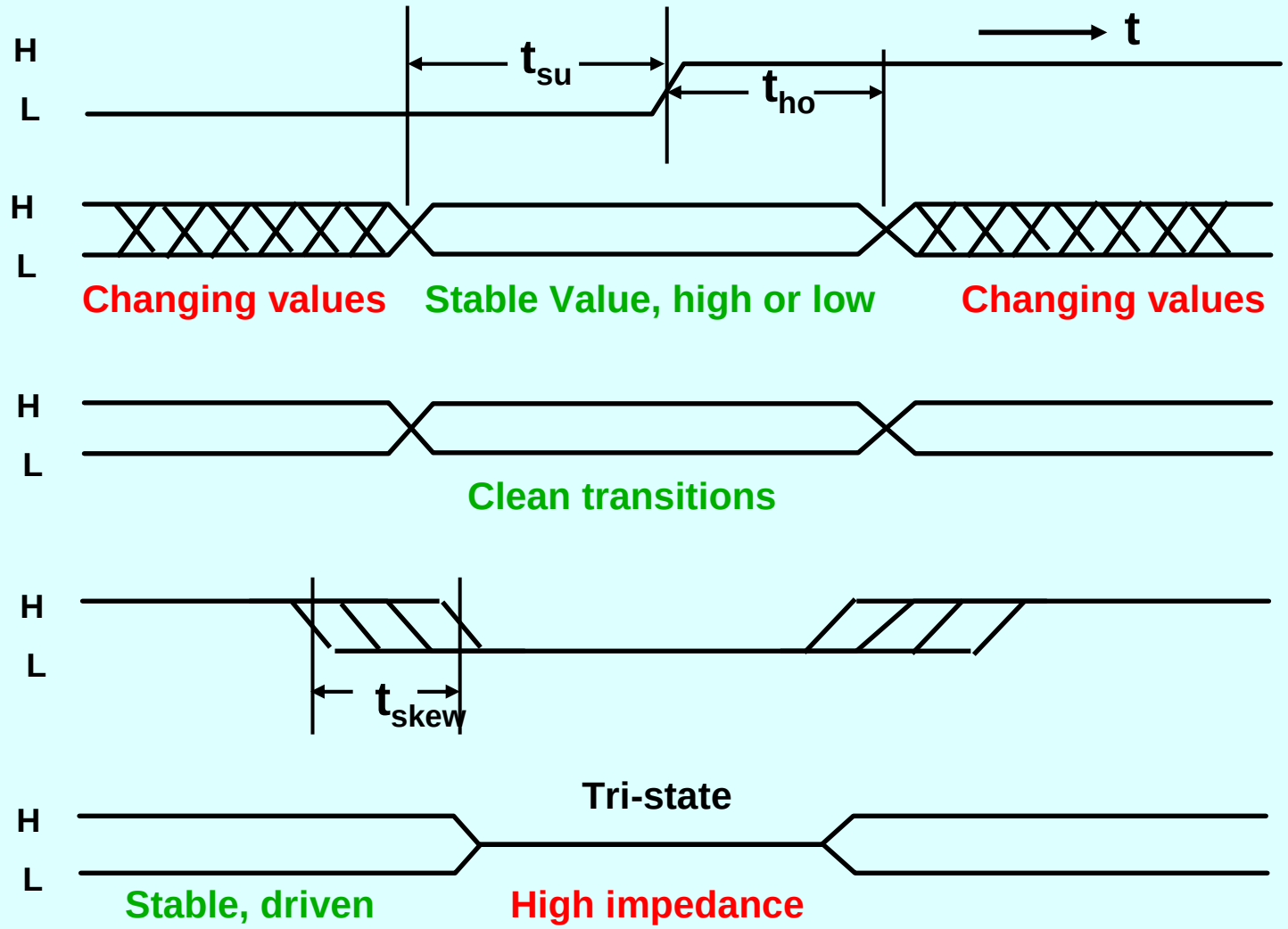
Implementation 2



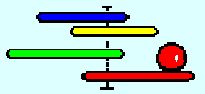
4 gates

use of 3-input gate

Timing Diagram Notation



Karnaugh Map



- **Karnaugh Map Method**
  - ➔ K-map is an alternative method of representing the truth table that helps visualize adjacent terms in up to 6 dimensions

2-variable K-map

B \ A	0	1
0	0	2
1	1	3

3-variable K-map

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

(A: 11, 10; B: 01, 11)

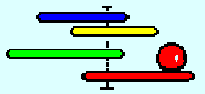
4-variable K-map

C \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

(A: 11, 10; D: 01, 11; B: 01, 11)

- ➔ Numbering Scheme: 00, 01, 11, 10
- ➔ Gray Code -- only a single bit changes one code word to the next

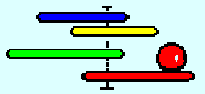
Simplification using K-Maps



- **Grouping blocks of ‘1’**
  - ➔ A group must consist of 16,8,4,2 or 1 cells
  - ➔ Each cell must be horizontally and/or vertically adjacent to cells in the other group
  - ➔ Always include the largest number of ’1’s in a group
  - ➔ Each ’1’ in the map should be included in a group
  - ➔ Groups can overlap
  - ➔ Map edge cells connect in a loop to cells at the opposite edge

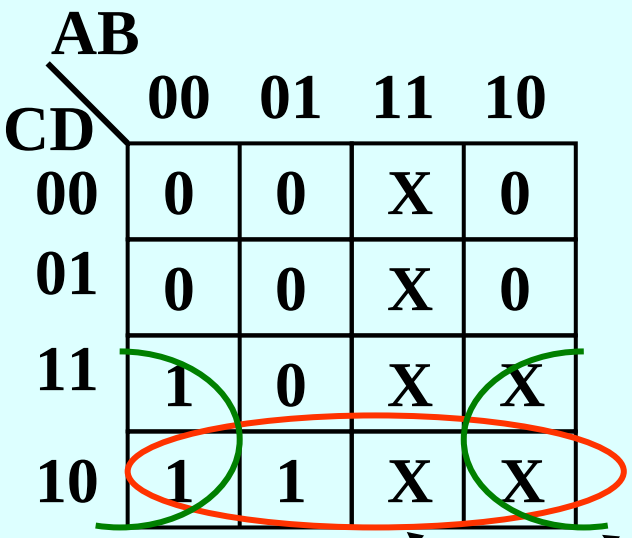
- **Naming groups**
  - ➔ The product description for a group will include ALL variables that are CONSTANT over the group
  - ➔ For example, for a 4-variable map
    - ➢ An 8-cell group is described by a 1-variable product term
    - ➢ An 4-cell group is described by a 2-variable product term
    - ➢ An 2-cell group is described by a 3-variable product term
    - ➢ An 1-cell group is described by a 4-variable product term

Minimal Solution



- A minimal SOP will consist of prime implicants.
- A minimal SOP equation will have all of the essential prime implicants on the map. By definition, these cover a minterm that may not be covered by some other prime implicant.
- The minimal SOP equation may or may not include non-essential prime implicants. It will include non-essential prime implicants if there are ‘1’s remaining that have not been covered by an essential prime implicant.

Don't Cares treated as '0's or '1's

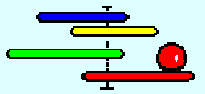


Treat X's as 1's to make larger groupings.

All X's do not have to be covered.

$$F(A,B,C,D) = C\bar{D} + \bar{B}C$$

Variable-Entered Karnaugh Maps (VEM)

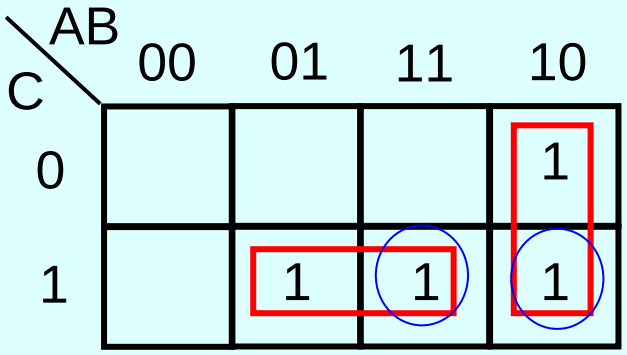


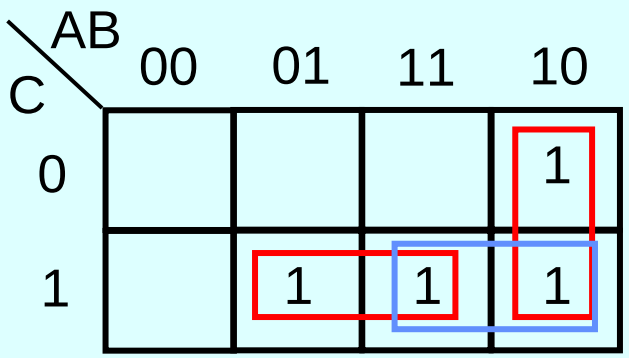
- **VEM Key idea:**
  - ➔ Represent values of function in terms of its variables (called map-entered variables) within Karnaugh map framework
  - ➔ Group like variables in Karnaugh map cells

➔ Minimisation approach

- ➢ Map all 1's
- ➢ Convert 1's to X's (i.e. don't cares)
- ➢ Map SIMILAR components

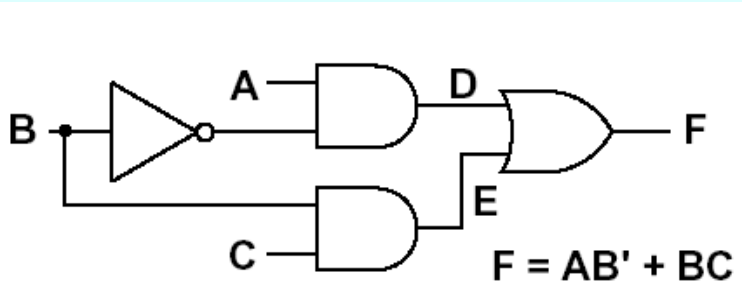
Hazards in Combinational Circuits

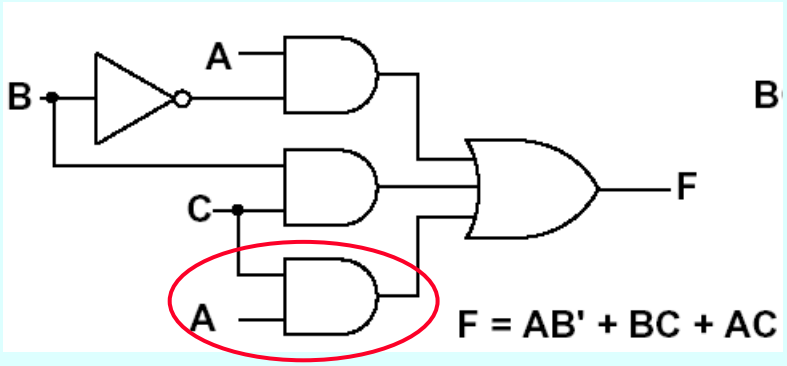




$f = AB' + BC$

$f = AB' + BC + AC$

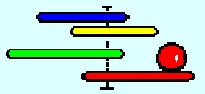




To avoid hazards:

every pair of adjacent 1s should be covered by a 1-group

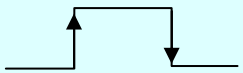
Flip-Flops/ Latches



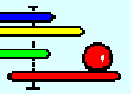
- **Latches and Flip-Flops are devices that can have two internal states (0,1)**

- **The output of a latch or a Flip-Flop (FF) is dependent upon its**
  - ➔ CURRENT STATE
  - ➔ CURRENT INPUTS.

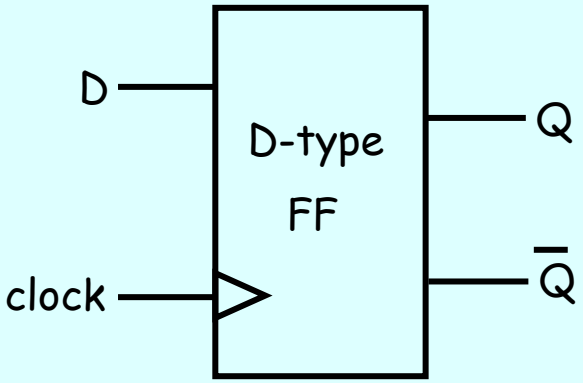
- **Latches and FFs are the simplest examples of sequential systems.**

- **State transitions based on**
  - ➔ levels 
  - ➔ edges 

D-type Flip-Flop



- **Clocked, edge triggered**



present state

next state

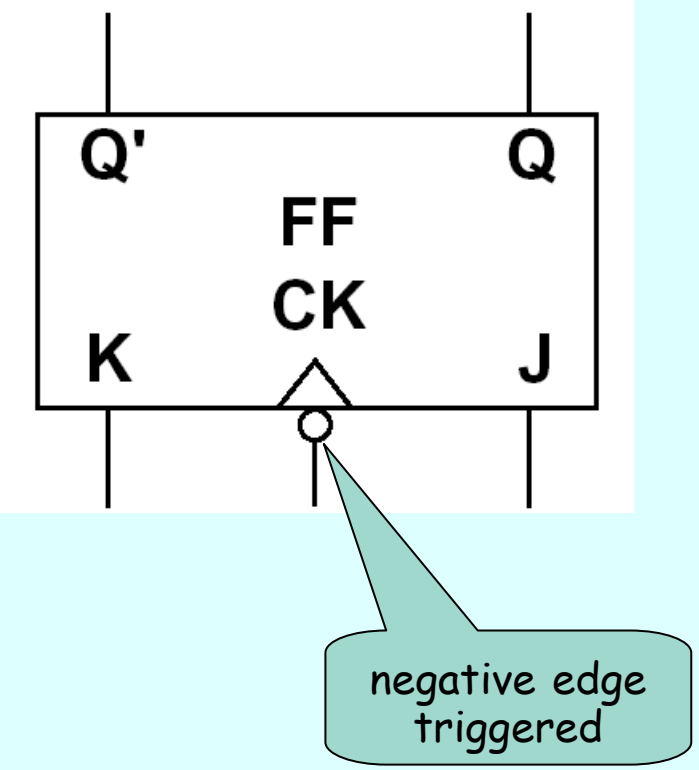
input

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

$Q^+ = D$

The next state in response to the rising edge of the clock is equal to the D input before the rising edge

Clocked JK Flip-Flop



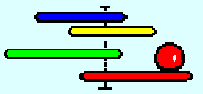
J	K	Q	Q^+	
0	0	0	0	do nothing
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

$$Q^+ = JQ' + K'Q$$

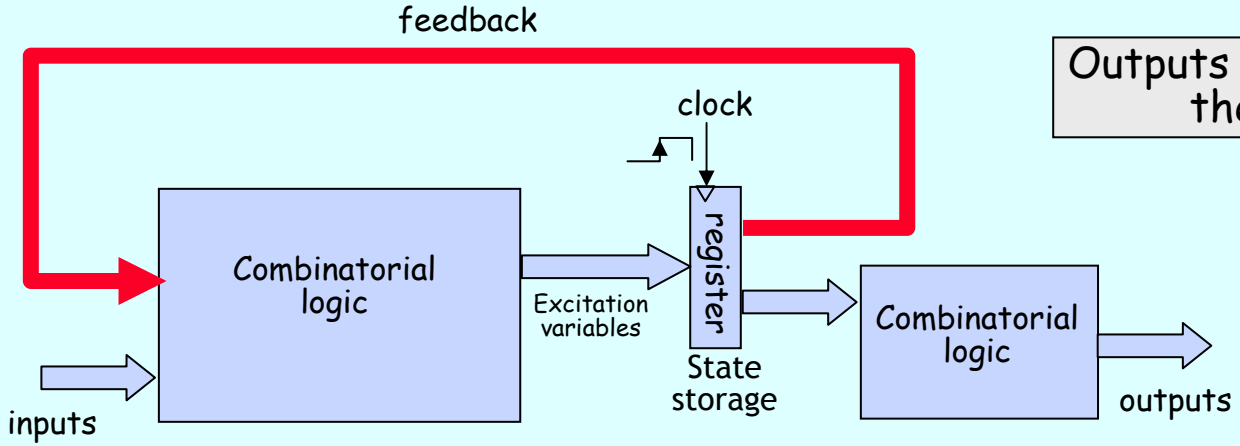
next state

JK = 00 => no state change occurs
JK = 10 => the flip-flop is set to 1, independent of the current state
JK = 01 => the flip-flop is always reset to 0
JK = 11 => the flip-flop changes the state $Q^+ = Q'$ (toggle)

Sequential System Concepts

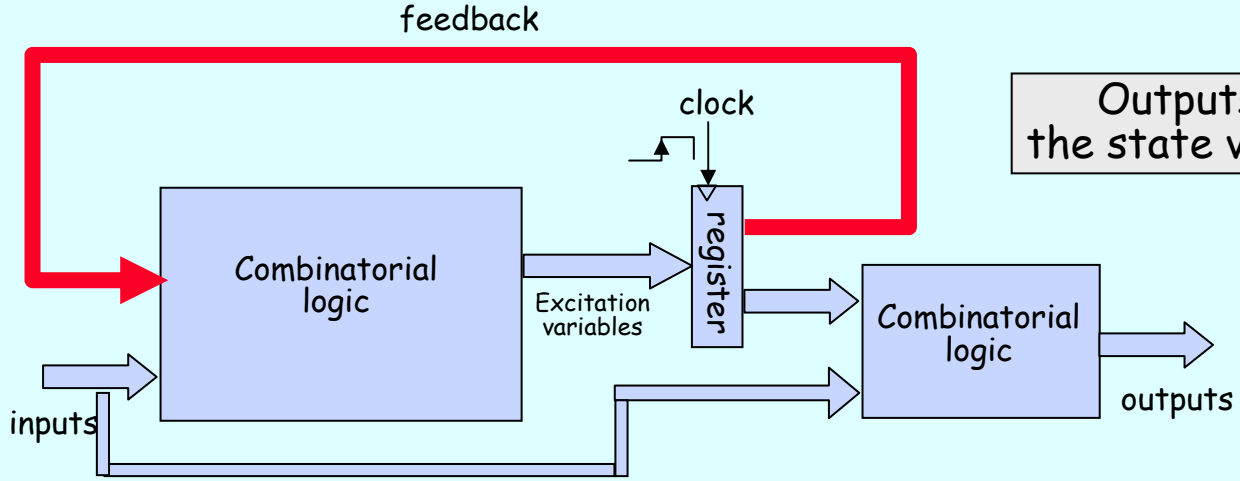


- **Moore machine**



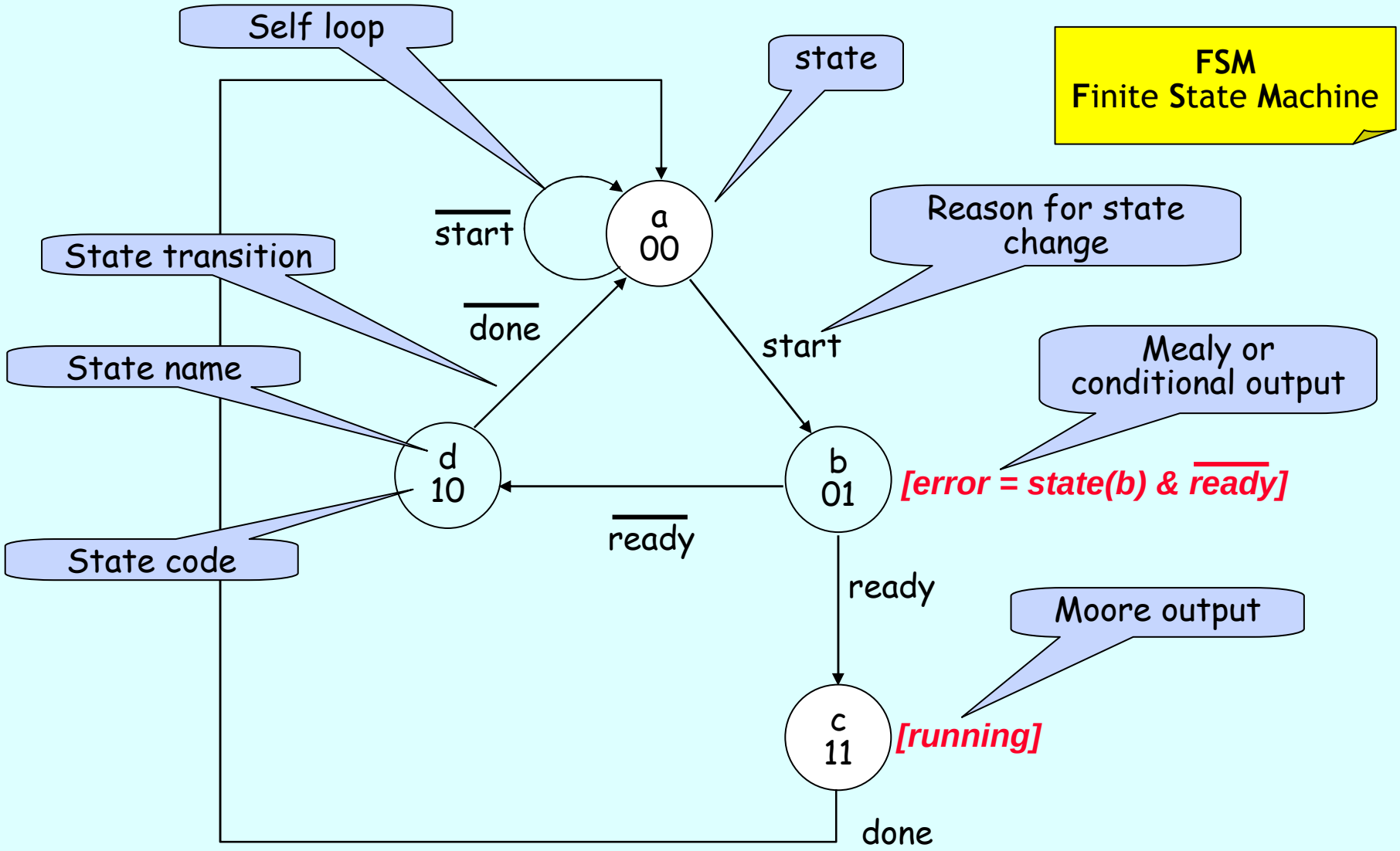
Outputs are only dependent on the state variables

- **Mealy machine**

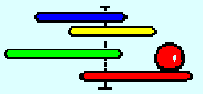


Outputs are dependent on the state variables and the inputs

The state diagram



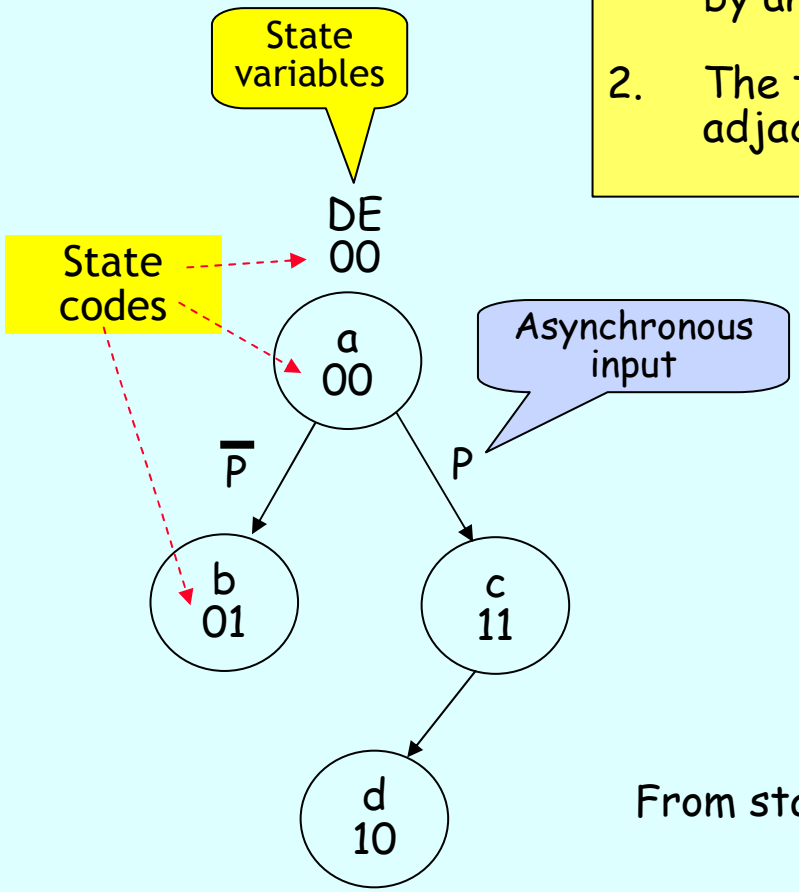
Assignment of state codes

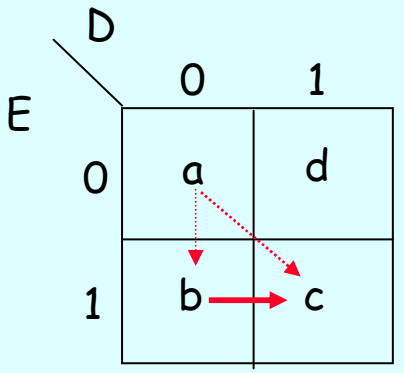


- **The state codes can be allocated in any way, but must adhere to one rule**

Asynchronous variable rule

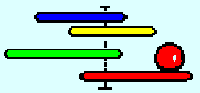
1. There should never be more than 2 states from a state whose branching is controlled by an asynchronous input variable
2. The two next states MUST have logically adjacent states



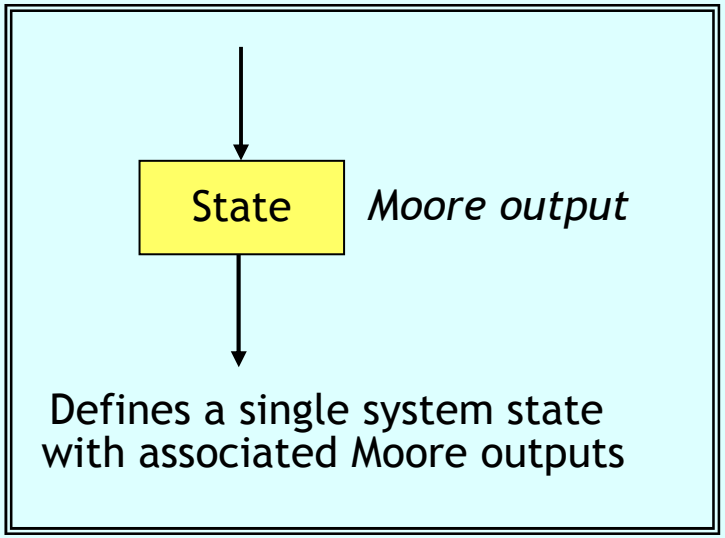


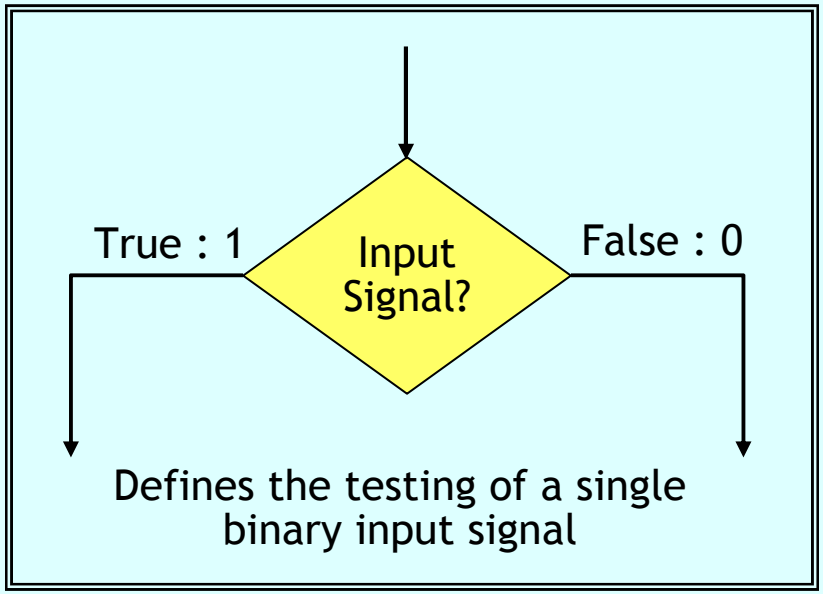
From state a, next states b and c are adjacent.
i.e. only differ by D

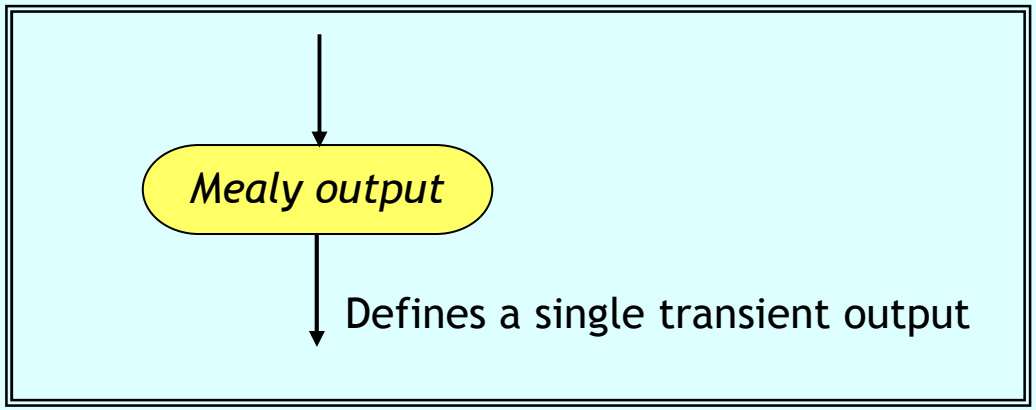
Algorithmic State machine (ASM) chart notation



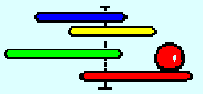
- **Similar to flow charts**



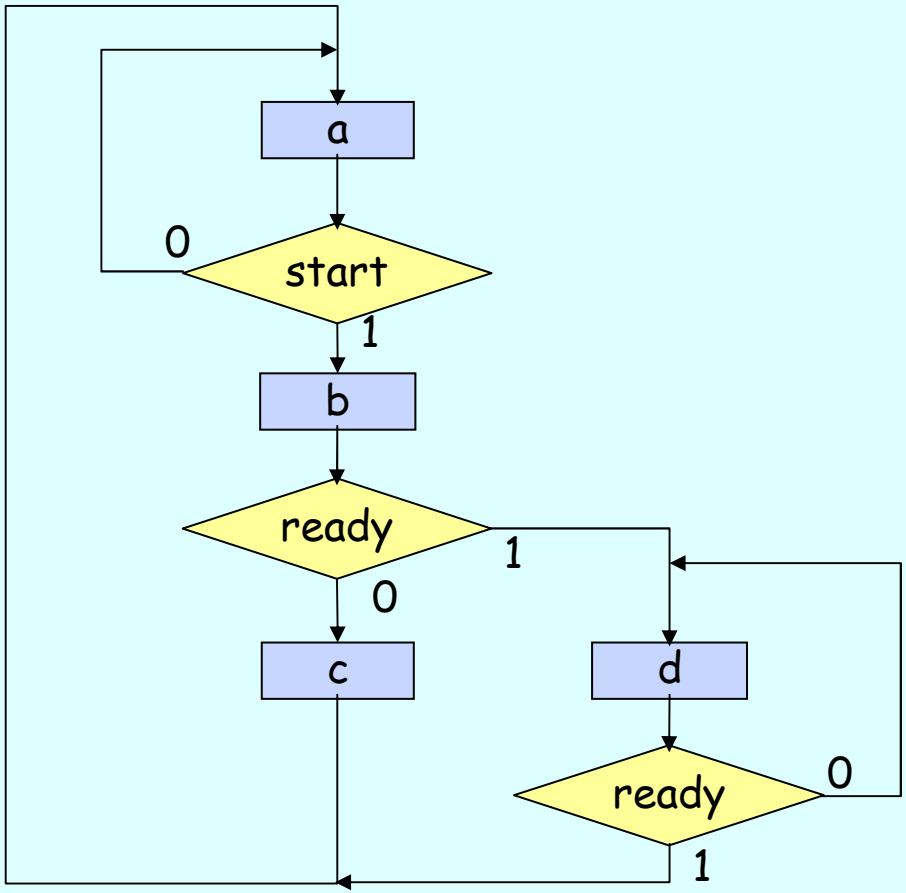


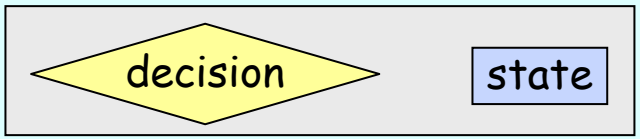


Algorithmic State machine charts

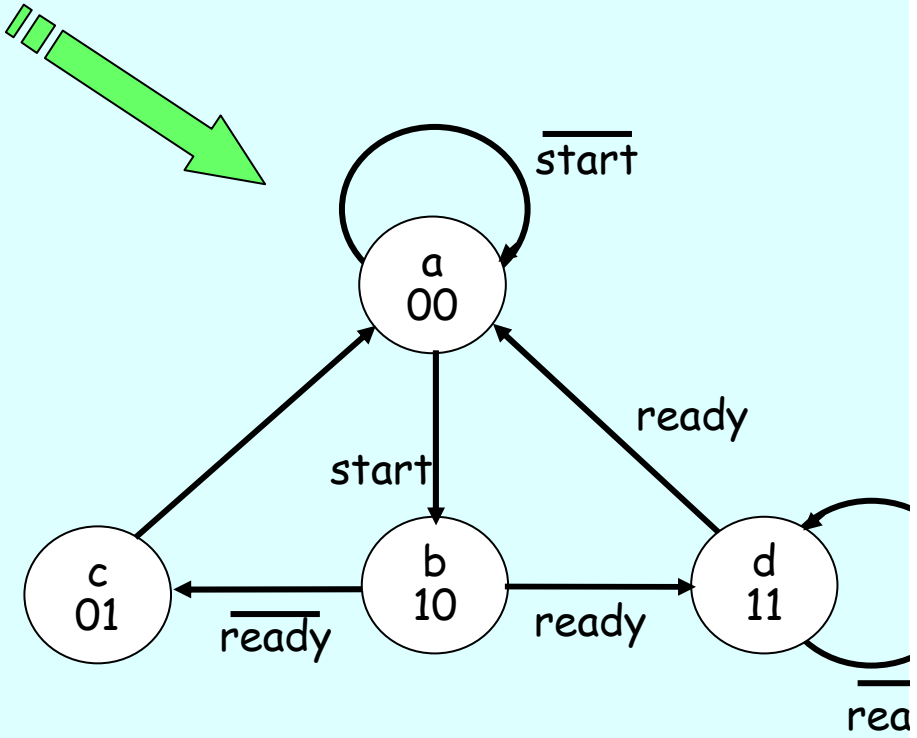


- Based on flow charts

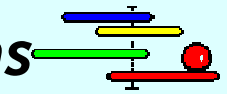




2. Draw state diagram



Excitation table for D-type excitation equations



- **Table of state transitions**

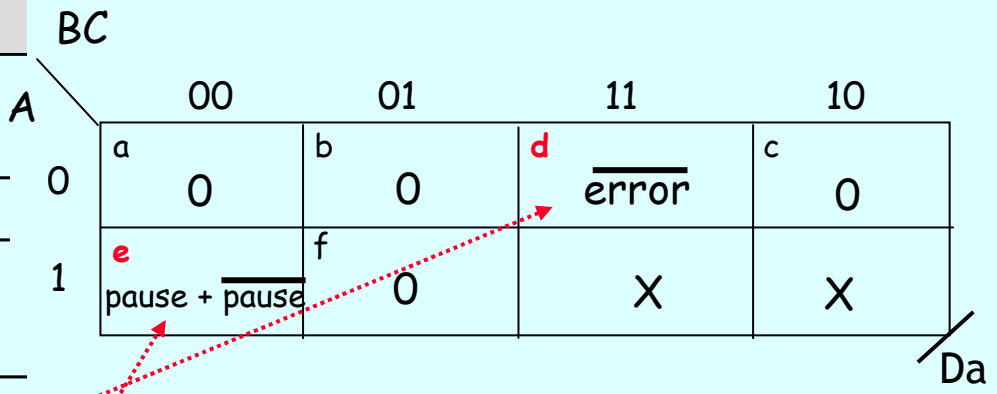
		Code of target state			Boolean on transition		
state	transitions	A	B	C	D_A	D_B	D_C
a	a→a	0	0	0	-	-	-
	a→c	0	1	0	-	start	-
b	b→a	0	0	0	-	-	-
c	c→c	0	1	0	-	wait	-
	c→d	0	1	1	-	$\overline{\text{wait}}$	$\overline{\text{wait}}$
d	d→b	0	0	1	-	-	error
	d→e	1	0	0	$\overline{\text{error}}$	-	-
e	e→e	1	0	0	pause	-	-
	e→f	1	0	1	$\overline{\text{pause}}$	-	$\overline{\text{pause}}$
f	f→d	0	1	1	-	$\overline{\text{done}}$	$\overline{\text{done}}$
	a→a	0	0	0			

D-input excitation values for state variable '**A**' D-type flipflop

Excitation table for D-type excitation equations

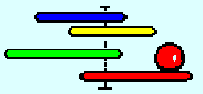
- Transitions table to Karnaugh Map

	Code of target state			Boole
s	A	B	C	D_A
	0	0	0	-
	0	1	0	-
	0	0	0	-
	0	1	0	-
	0	1	1	-
State d	0	0	1	-
	1	0	0	$\overline{error}$
State e	1	0	0	pause
	1	0	1	$\overline{pause}$
	0	1	1	-
	0	0	0	

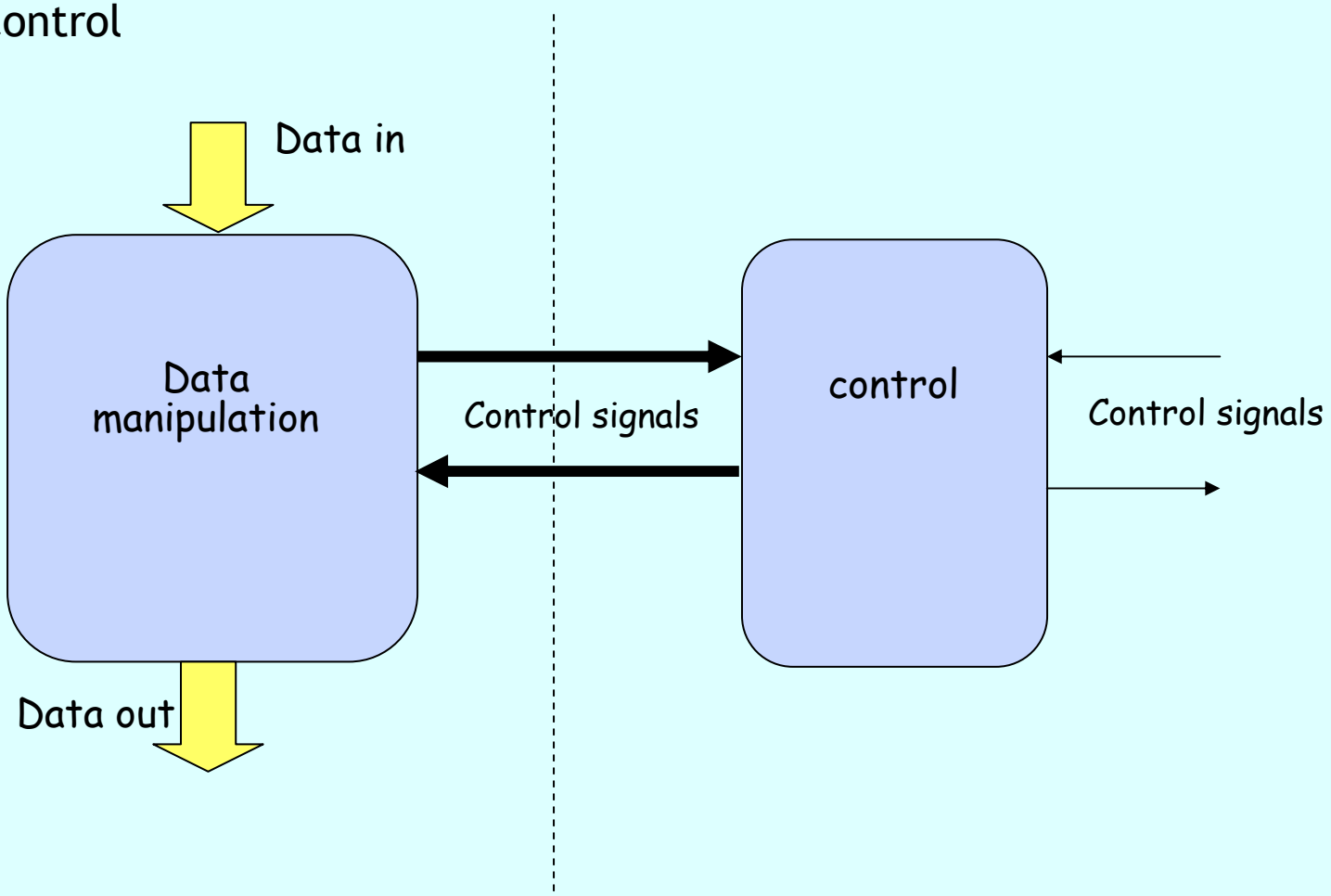


- **Map rules**
  1. Map '1's
  2. Convert '1's to don't cares ('X')
  3. Map LIKE entries

System design

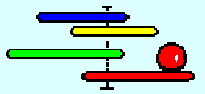


- **Digital system consist of two cooperating units**
  - ➔ Data
  - ➔ Control



  - ➔ E.g. A computer, a graphics card, an ethernet interface, etc

Data manipulation components



- **The components available to the designer include**
  - ➔ Registers
  - ➔ Counters
    - ➢ Up, down
  - ➔ Shift registers
    - ➢ Left, right
  - ➔ Multplexors
    - ➢ Also called data selectors
  - ➔ De-multiplexors
  - ➔ Comparators
  - ➔ Arithmetic units
    - ➢ Adders, subtractors, incrementors, multipliers, ALUs

- **At design time it is important to see these blocks as BLACK BOXES**
  - ➔ Their individual design can be decided at a later stage
  - ➔ Need only define inputs and outputs
    - ➢ Direction
    - ➢ Polarity
    - ➢ Level or edge

Digital Design : Tutorial

1. Minimise the following using Karnaugh maps

a. $F(a,b,c) = \sum m(0,1,3,4,6)$

b. $F(a,b,c,d) = \sum m(2,3,7,9, 11,12,15)$

c. $F(a,b,c,d,e) = \sum m(5,6,7,9,11,17,19,21,25,26,27,30,31)$

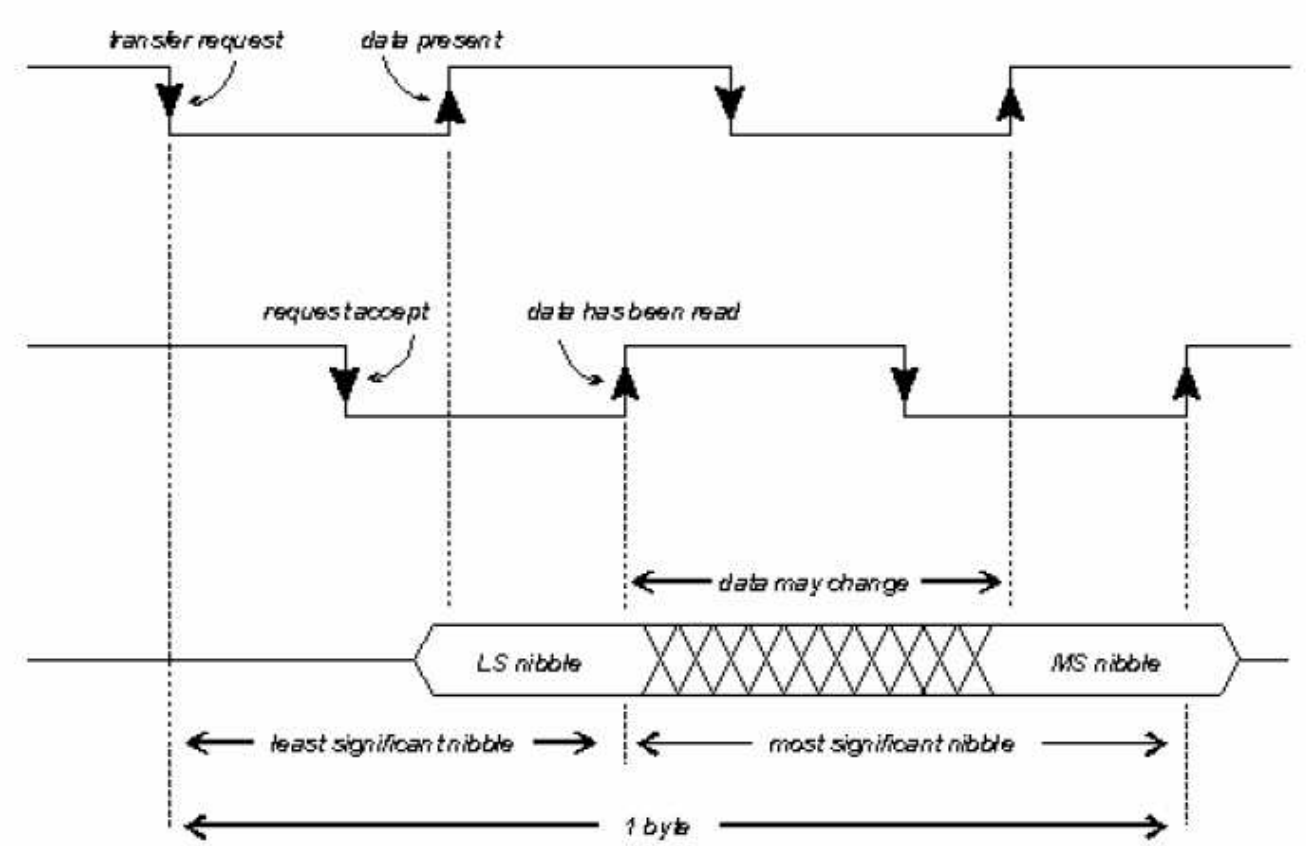
1.

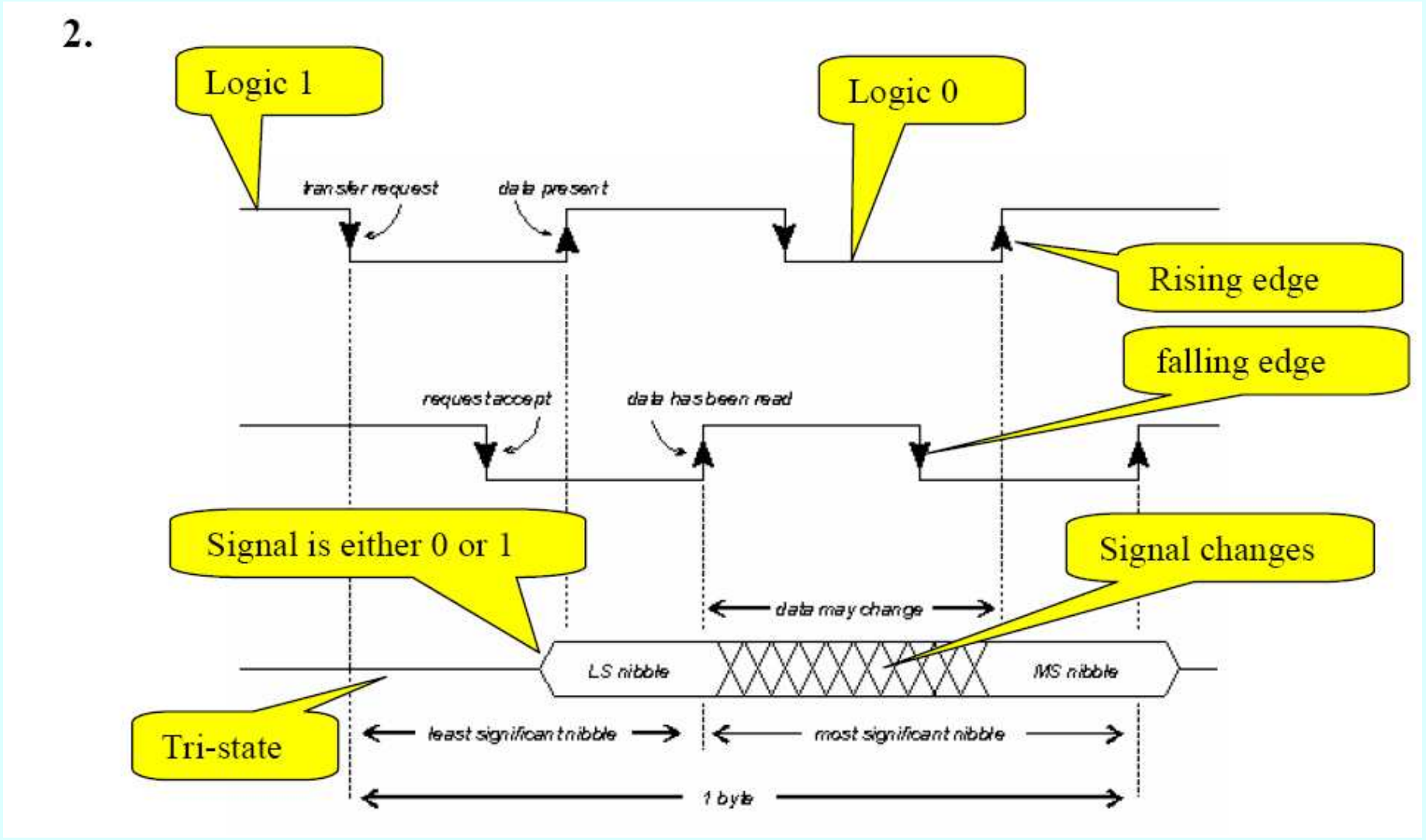
a. $F = \overline{A}\overline{B} + \overline{B}\overline{C} + \overline{A}C + A\overline{C}$

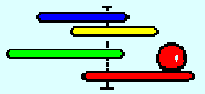
b. $F = CD + \overline{A}\overline{B}C + A\overline{B}D + AB\overline{C}\overline{D}$

c. $F = B\overline{C}E + ABD + \overline{B}C\overline{D}E + \overline{A}\overline{B}CD + A\overline{B}\overline{C}E$

2. Describe the different graphical aspects of this waveform diagram. Ignore the purpose of the diagram.




2.
Logic 1
Logic 0
transfer request
data present
Rising edge
falling edge
request accept
data has been read
Signal is either 0 or 1
Signal changes
data may change
LS nibble
MS nibble
Tri-state
least significant nibble
most significant nibble
1 byte



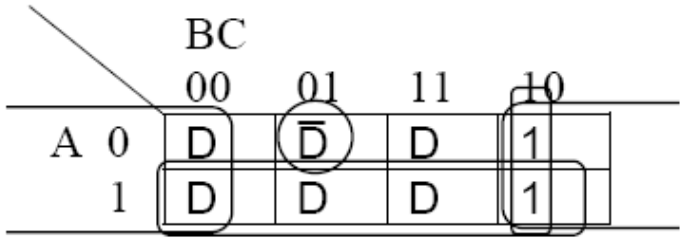
3. Convert the following 4 variable function to a standard Karnaugh map.

$$F(a,b,c,d) = \sum m(1,2,4,5,7,9,11,12,13,15)$$

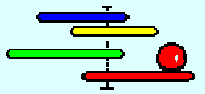
Convert this 4 variable Karnaugh map to a 3 variable map entered Karnaugh map with **d** as the map entered variable.

Truth table ::

A	B	C	D	Output	Map entry
0	0	0	0	0	D
0	0	0	1	1	
0	0	1	0	1	$\overline{D}$
0	0	1	1	0	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	0	D
0	1	1	1	1	
1	0	0	0	0	D
1	0	0	1	1	
1	0	1	0	0	D
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	0	D
1	1	1	1	1	



$$F = B\overline{C} + AD + BD + \overline{C}D + \overline{A}\overline{B}C\overline{D}$$



4. Answer the following questions

a. Show how a glitch pulse can be created by a race condition setup by a single input signal.

b. Describe the operation of a D-type flip-flop.

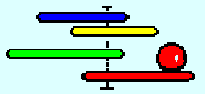
c. Draw the structure of a Moore state machine.

d. Outline the problem of state consistency.

e. State the "asynchronous rule" for state assignment and indicate why it is important when dealing with asynchronous input signals.

4.

These questions are bookwork. Refer to the notes.

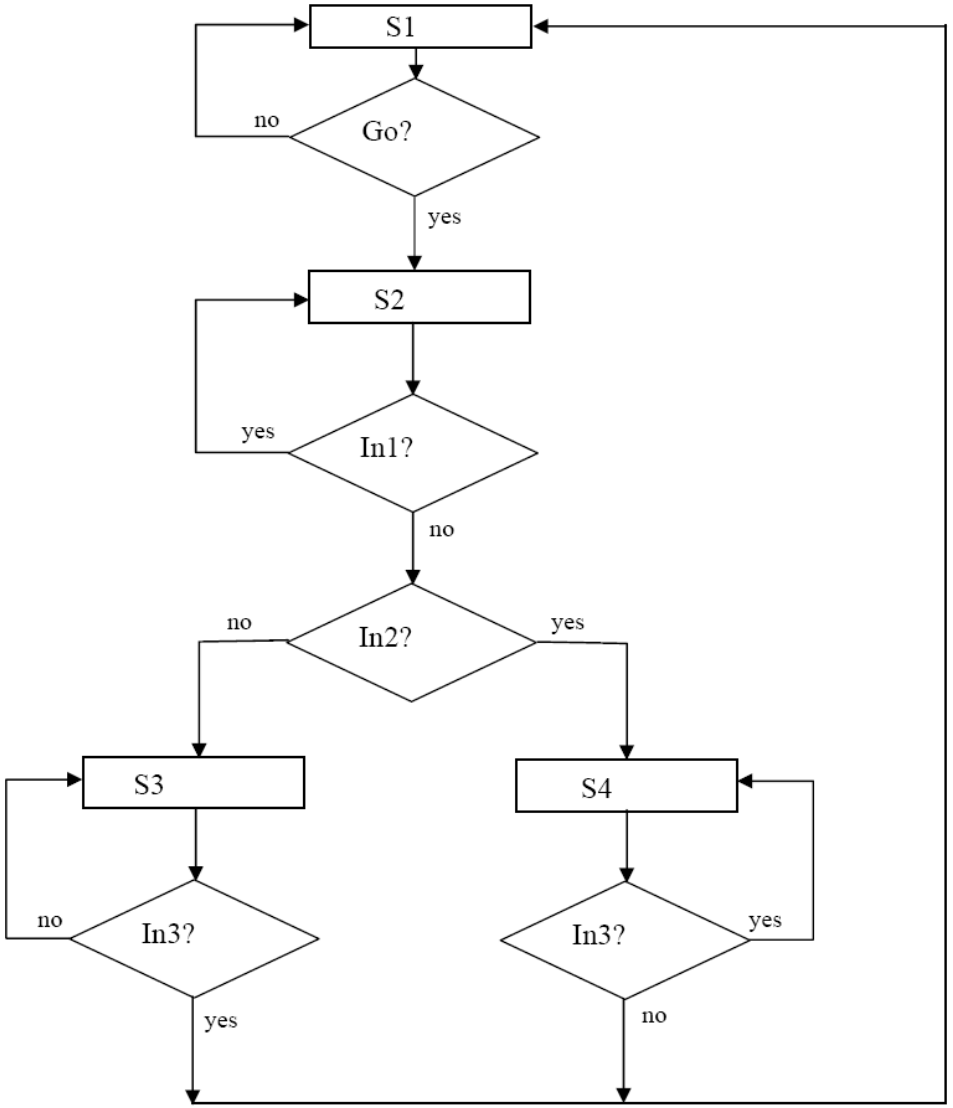


5. What are the rules to produce a mapping from a Variable Entered Karnaugh Map (VEM).

5.

a. map '1's
b. convert '1's to don't cares
c. map squares with identical contents as per a normal karnaugh map.

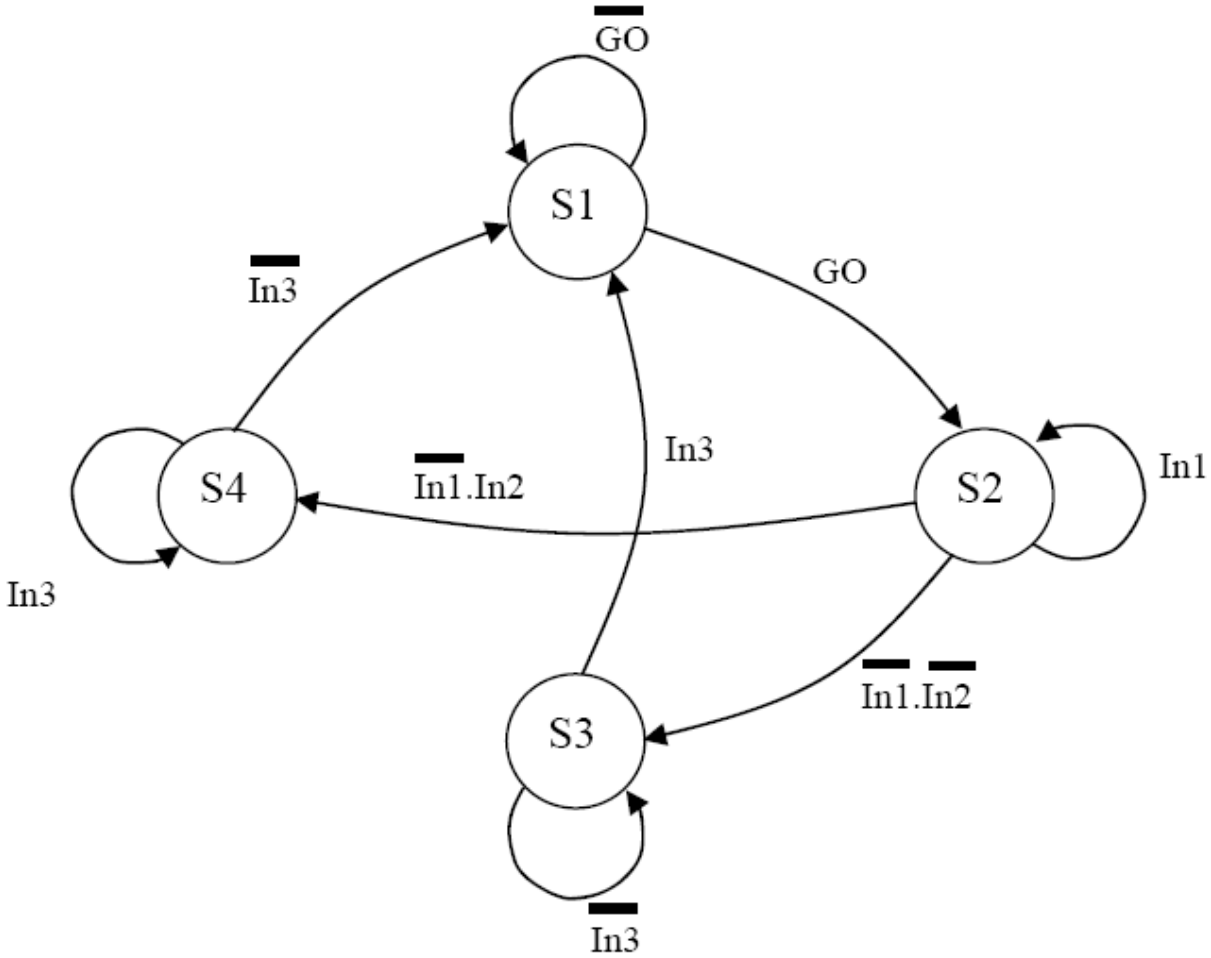
6. Convert the following ASM chart to a state diagram. There are 4 states and 4 in signals



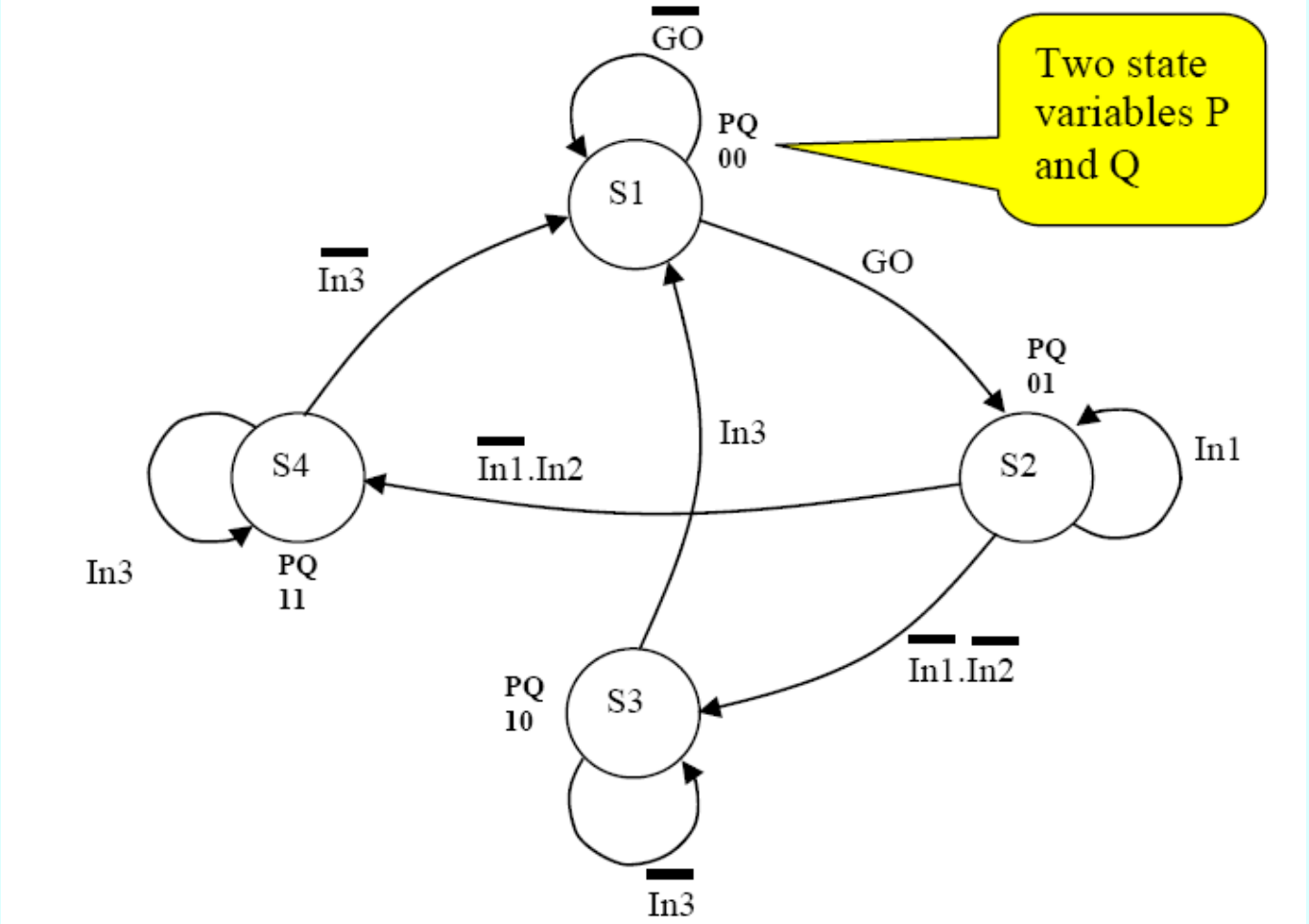
List all transitions ::

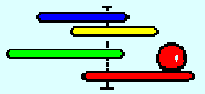
State	Transition	Logic of transition
S1	S1 → S1	$\overline{GO}$
	S1 → S2	GO
S2	S2 → S2	In1
	S2 → S3	$\overline{In1}.\overline{In2}$
	S2 → S4	$\overline{In1}.In2$
S3	S3 → S3	$\overline{In3}$
	S3 → S1	In3
S4	S4 → S4	In3
	S4 → S1	$\overline{In3}$

Resulting state machine.



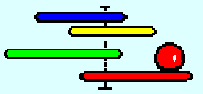
7. Convert the state diagram of question 6 to a set of excitation equations for a D-type implementation of the state machine.





State	Transition	Target state		Dp	Dq
		P	Q		
S1	S1→S1	0	0	--	--
	S1→S2	0	1	--	GO
S2	S2→S2	0	1	--	In1
	S2→S3	1	0	/In1./In2	--
	S2→S4	1	1	/In1.In2	/In1.In2
S3	S3→S3	1	0	/In3	--
	S3→S1	0	0	--	--
S4	S4→S4	1	1	In3	In3
	S4→S1	0	0	--	--

		Q	
		0	1
P	0	S1	S2
	1	S3	S4



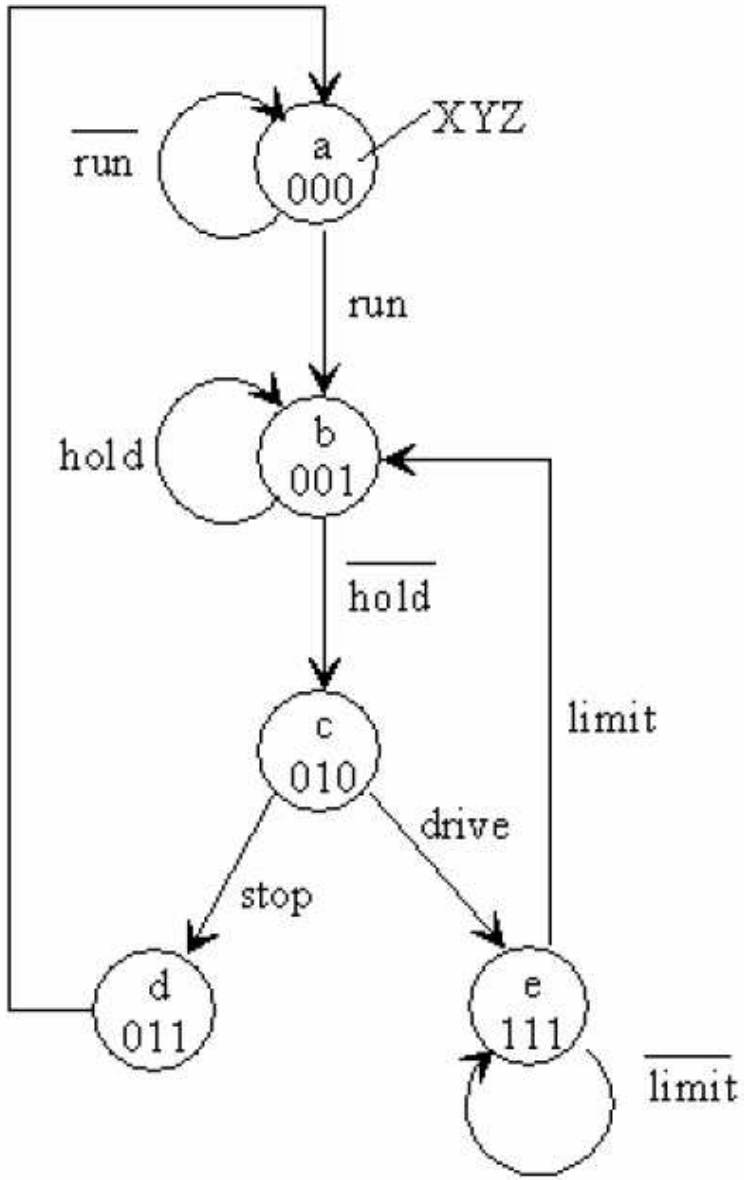
P \ Q	0	1
0	0	/In1./In2 + /In1.In2
1	0	In3

$Dp = P\ \overline{Q}\ (/In1./In2 + /In1.In2) + P\ Q\ In3$

P \ Q	0	1
0	GO	In1. + /In1.In2
1	/In3	In3

$Dq = P\ Q\ GO + \overline{P}\ Q\ (In1 + /In1.In2) + \overline{Q}\ P\ /In3 + P\ Q\ In3$

8. Convert the following state diagram to a set of D-type excitation equations.

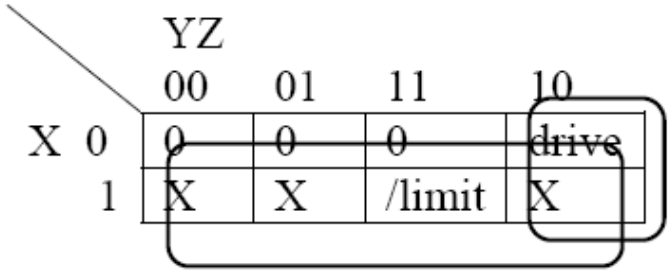


8.

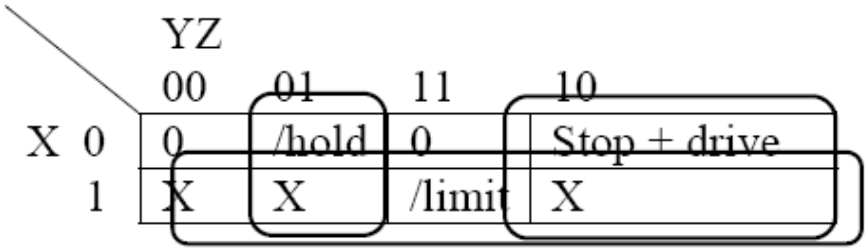
State	Transition	Target state					
		X	Y	Z	Dx	Dy	Dz
a	a→a	0	0	0	--	--	--
	a→b	0	0	1	--	--	/run
b	b→b	0	0	1	--	--	hold
	b→c	0	1	0	--	/hold	--
c	c→d	0	1	1	--	stop	stop
	c→e	1	1	1	drive	drive	drive
d	d→a	0	0	0	--	--	--
e	e→e	1	1	1	/limit	/limit	/limit
	e→b	0	0	1	--	--	limit

State layout

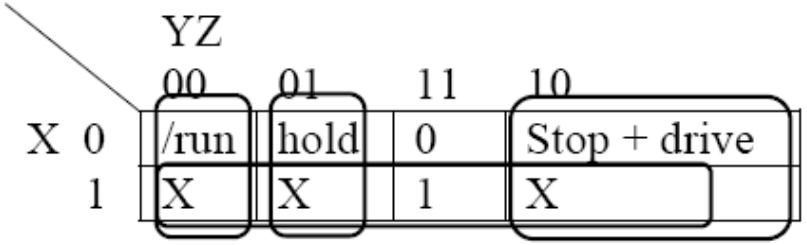
X \ YZ	00	01	11	10
0	a	b	d	c
1	X	X	e	X



$$Dx = Y \overline{Z}\ \text{drive} + X\ \text{/limit}$$

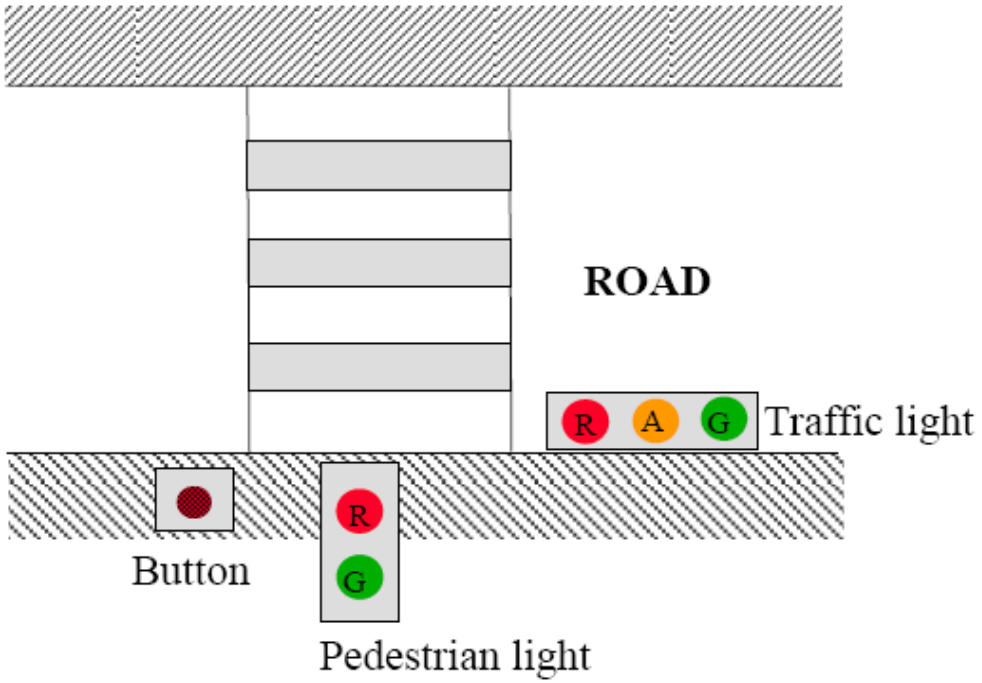


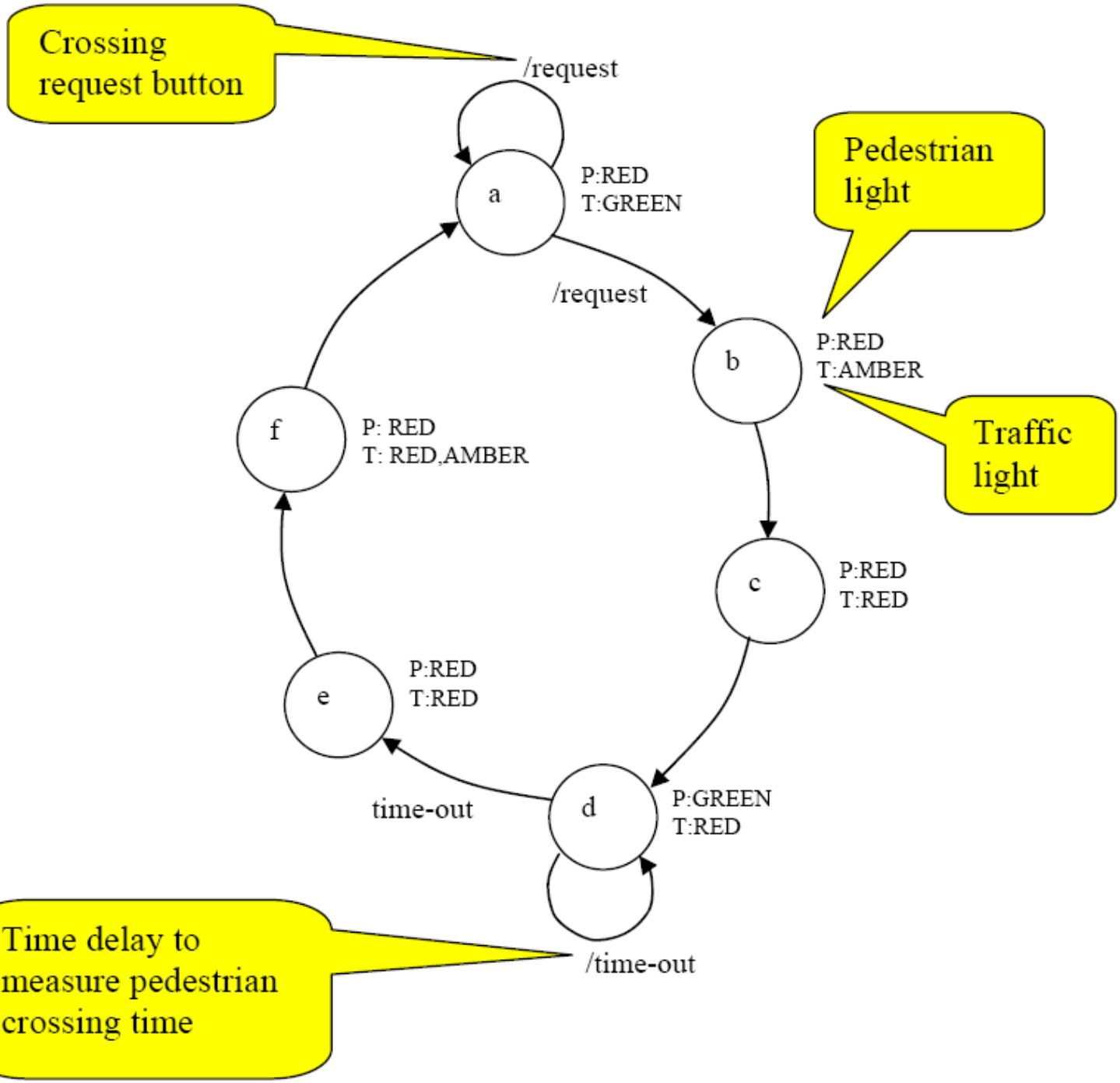
$$Dy = X\ \text{/limit} + \overline{Y}\ Z\ \text{/hold} + Y\ \overline{Z}\ (\text{stop} + \text{drive})$$



$$Dz = X + \overline{Y}\ \overline{Z}\ \text{/run} + \overline{Y}\ Z\ \text{hold} + Y\ \overline{Z}\ (\text{stop} + \text{drive})$$

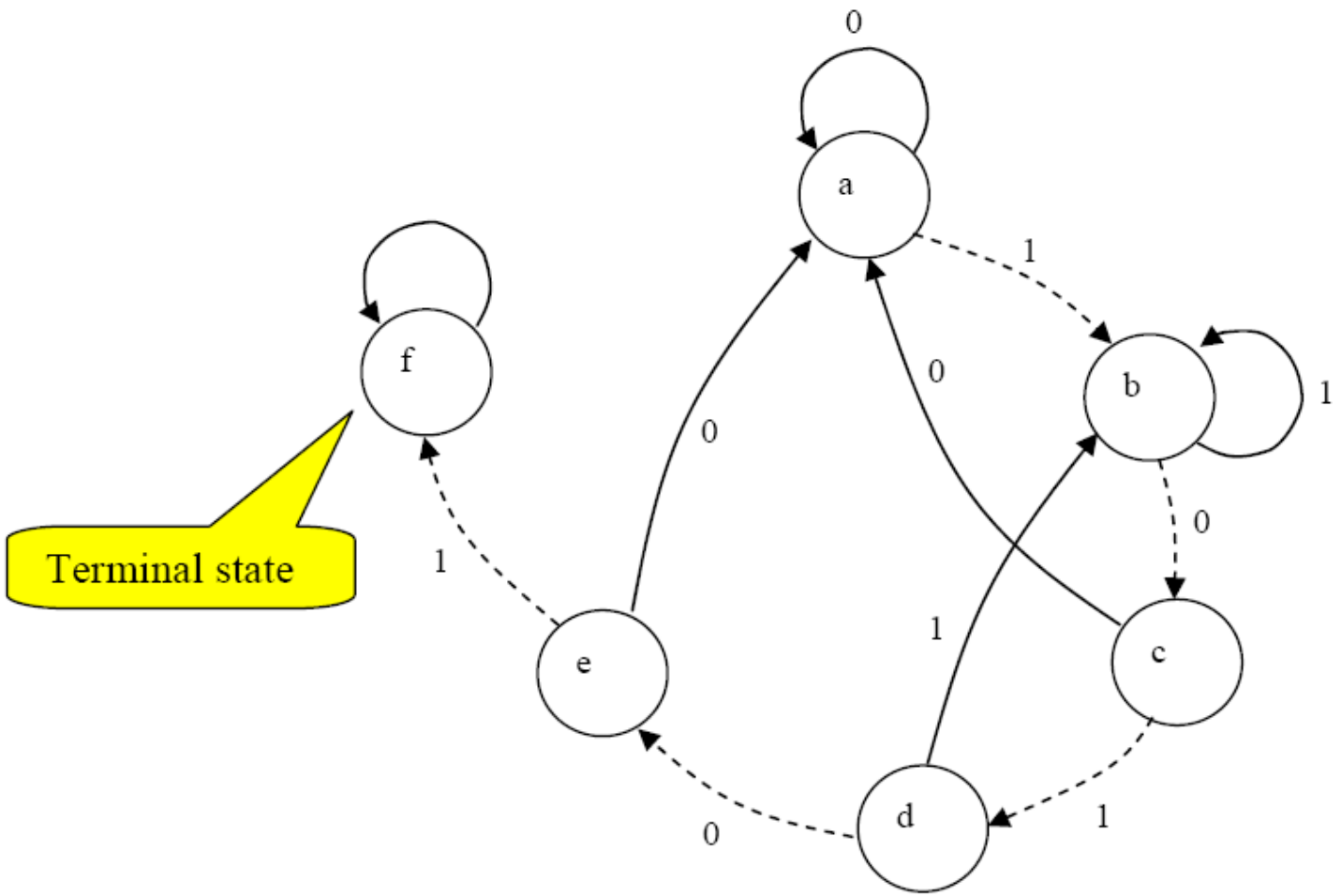
9. Design a state machine to control a simple. “pedestrian crossing”.




Crossing request button
/request
a
P:RED
T:GREEN
Pedestrian light
/request
b
P:RED
T:AMBER
Traffic light
f
P: RED
T: RED,AMBER
c
P:RED
T:RED
e
P:RED
T:RED
time-out
d
P:GREEN
T:RED
Time delay to measure pedestrian crossing time
/time-out

10. Design a state machine to detect the value '10101' in a serial stream of data.

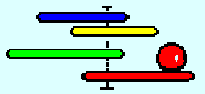
10101 pattern detector state diagram. Dotted line shows main detection route.



11. List the steps requires to solve a state machine problem that starts with a specification and finishes with a D-type flip-flop implementation.

11.

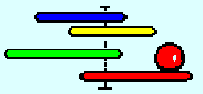
a. ASM chart
b. State diagram
c. Transition table
d. State location karnaugh map
e. Karnaugh map and associated boolean excitation equation for each state variable
f. Karnaugh maps and associated boolean equation for each output



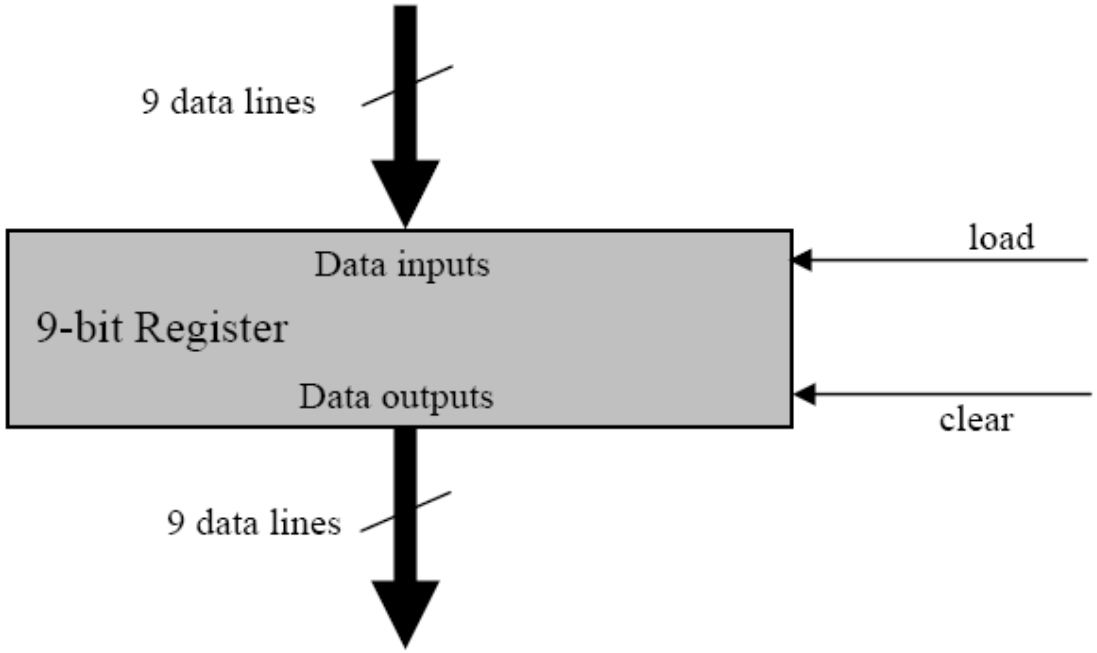
12. List a general set of data components that can be used for digital system design.

12.

a. registers
b. counters
c. shift registers
d. multplexers
e. de-multiplexers
f. arithmetic units
g. comparators

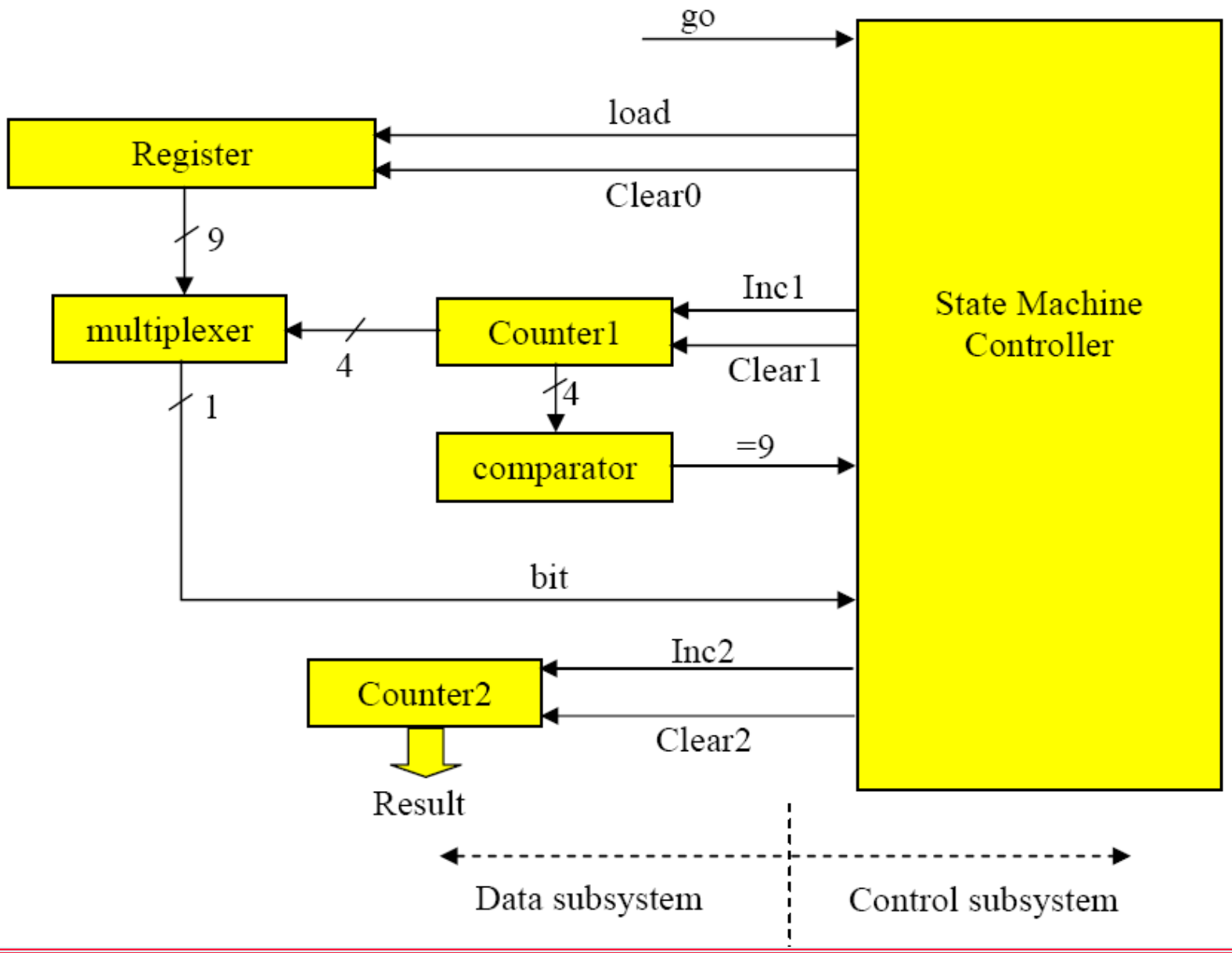


13. Design a state machine based system to count the number of '1's in a register of length 9 bits. The starting point for your design is as follows

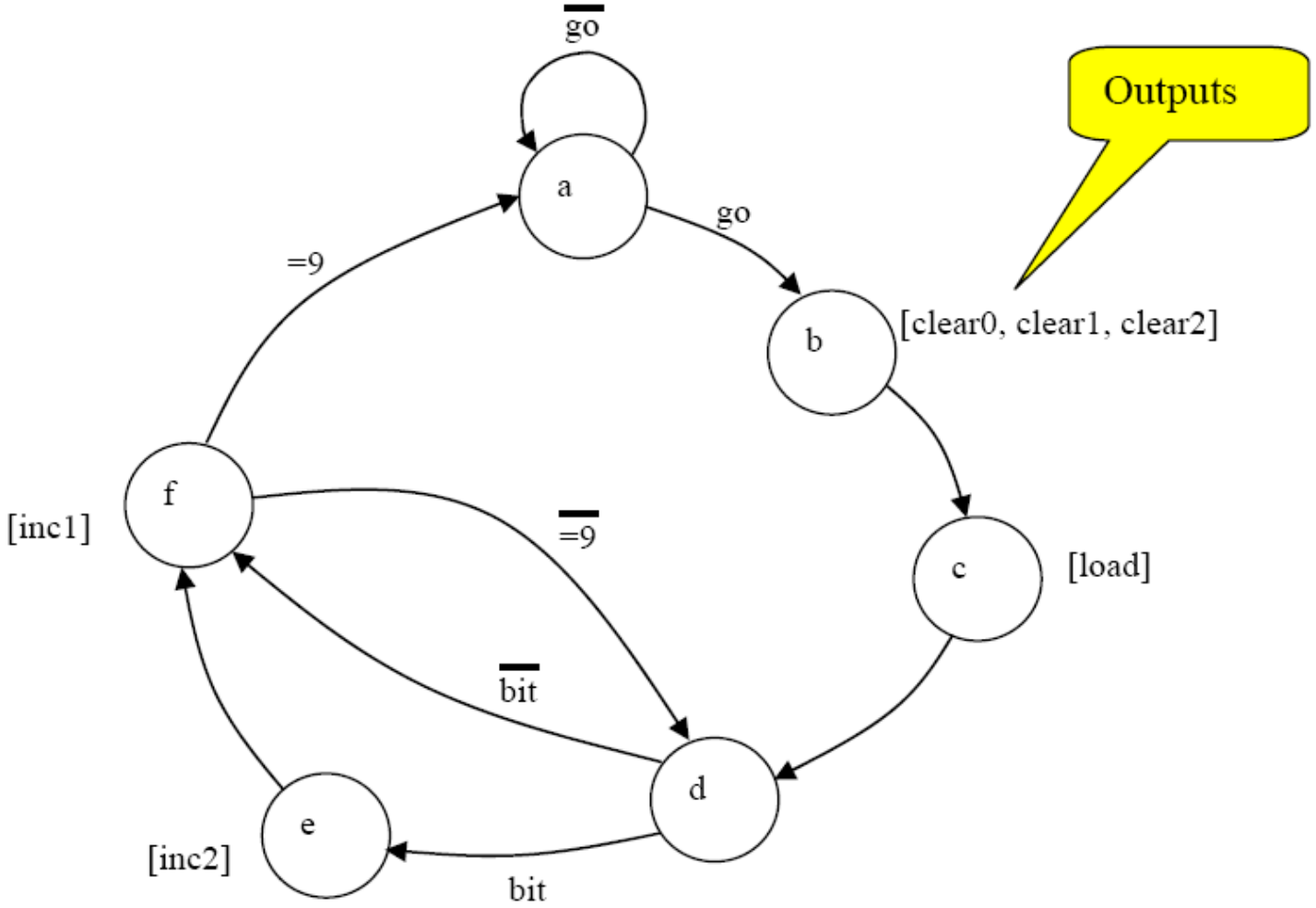


There are two control inputs – load and clear

Block diagram structure



State diagram



14. (Exam question from 2003)

A manufacturer of vacuum cleaners wishes to experiment with robotic cleaners. They plan to design a cheap model that has little intelligence that bumps its way around a room in an almost random manner. It is proposed to build a series of test versions to evaluate the concept.

This initial version has the following hardware features

- A start button
- Motors can drive forward, backwards, turn left, turn right, and stop
- Three bump sensors, front left, front right, and rear provide detection of the environment
- An on-board timer, when triggered, gives a signal after 2 seconds

The operational requirements are

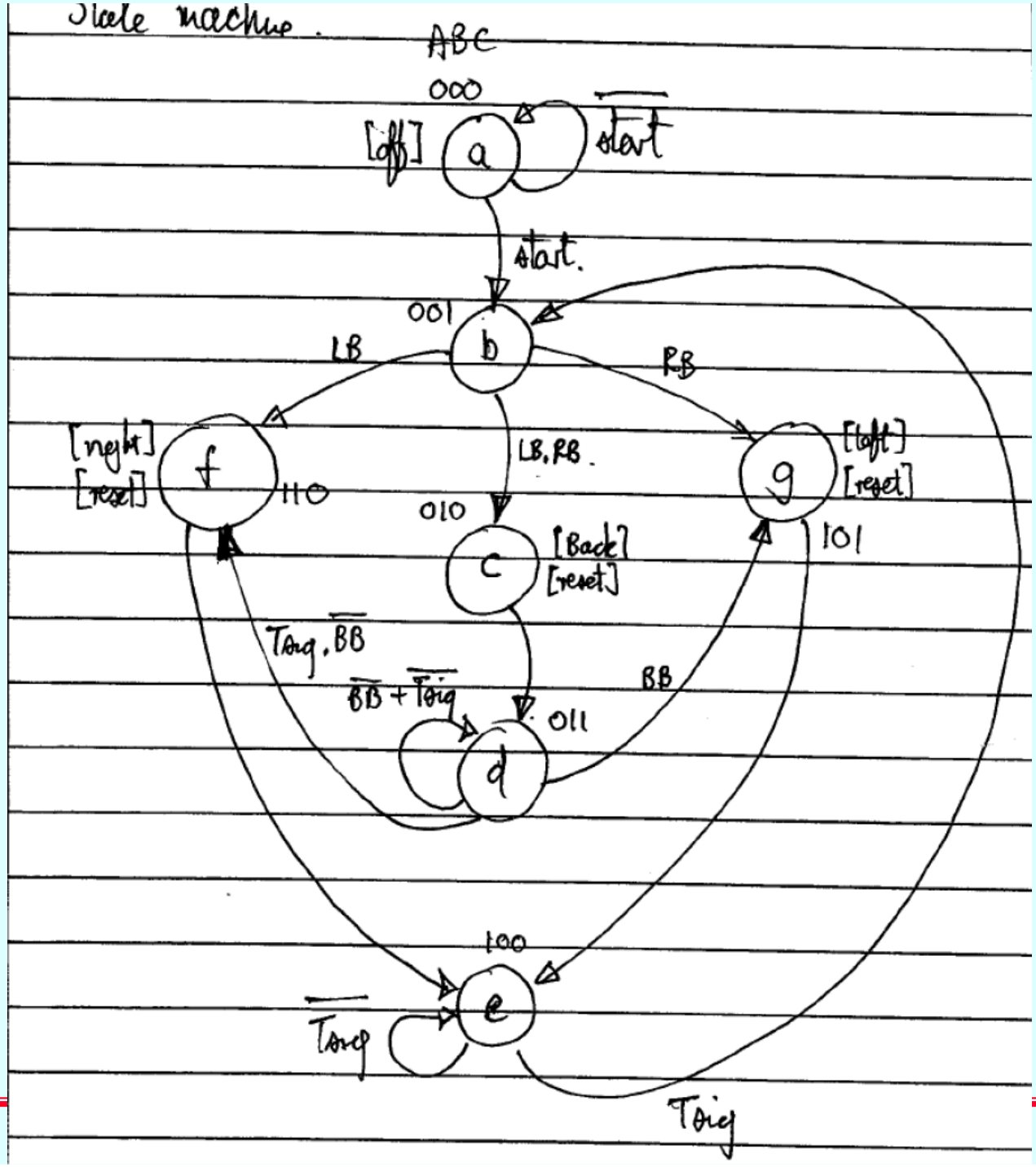
- On start drive forward.
- If left bump activated , turn right for 2 seconds then drive forward.
- If right bump activated, turn left for 2 seconds then drive forward.
- If both left and right bump activated, reverse for 4 seconds, turn right for 2 seconds, then drive forward.
- If rear bump, drive forward for 2 seconds, turn left for 2 seconds, then drive forward.

The control system is to be built as a state diagram.

a) Draw a system block diagram showing the various subsystems and the relevant signals.
b) Create a state diagram for the vehicle controller.
c) From the state diagram derive a set of excitation equations for a D-type flip-flop implementation.
d) Suggest changes to the hardware and state diagram to allow the following modification

- If both left and right bump activated, reverse for 2 seconds, increment a 3-bit counter.
  - If counter is less than 4 turn right for 2 seconds, then drive forward
  - If counter is greater than 4 turn left for 2 seconds, then drive forward

    (note : when counter is at 7 the next count sets count to zero)


Mealy machine.
ABC
000
[off]
a
start
start.
001
b
LB
RB
[right]
[reset]
f
110
LB, RB.
010
c
[Back]
[reset]
[left]
[reset]
g
101
Tarq, BB
BB + Tarq
011
d
BB
100
e
Tarq
Tarq

	A B C	D_A	D_B	D_C	
a → b.	0 0 1	–	–	start	
b → c	0 1 0	–	LB. RB	–	
b → f	1 1 0	LB	LB	–	
b → g	1 0 1	RB	–	RB.	
c → d	0 1 1	–	1	1	
d → d	0 1 1	–	$\overline{BB} + \overline{T_{orig}}$	$\overline{BB} + \overline{T_{orig}}$	
d → f	1 1 0	$T_{orig}.\overline{BB}$	$T_{orig}.\overline{BB}$	–	
d → g	1 0 1	BB	–	BB.	
e → e	1 0 0	$\overline{T_{orig}}$	–	–	
e → b	0 0 1	–	–	T_{orig}	
f → e.	1 0 0	1	–	–	
g → e	1 0 0	1	–	–	

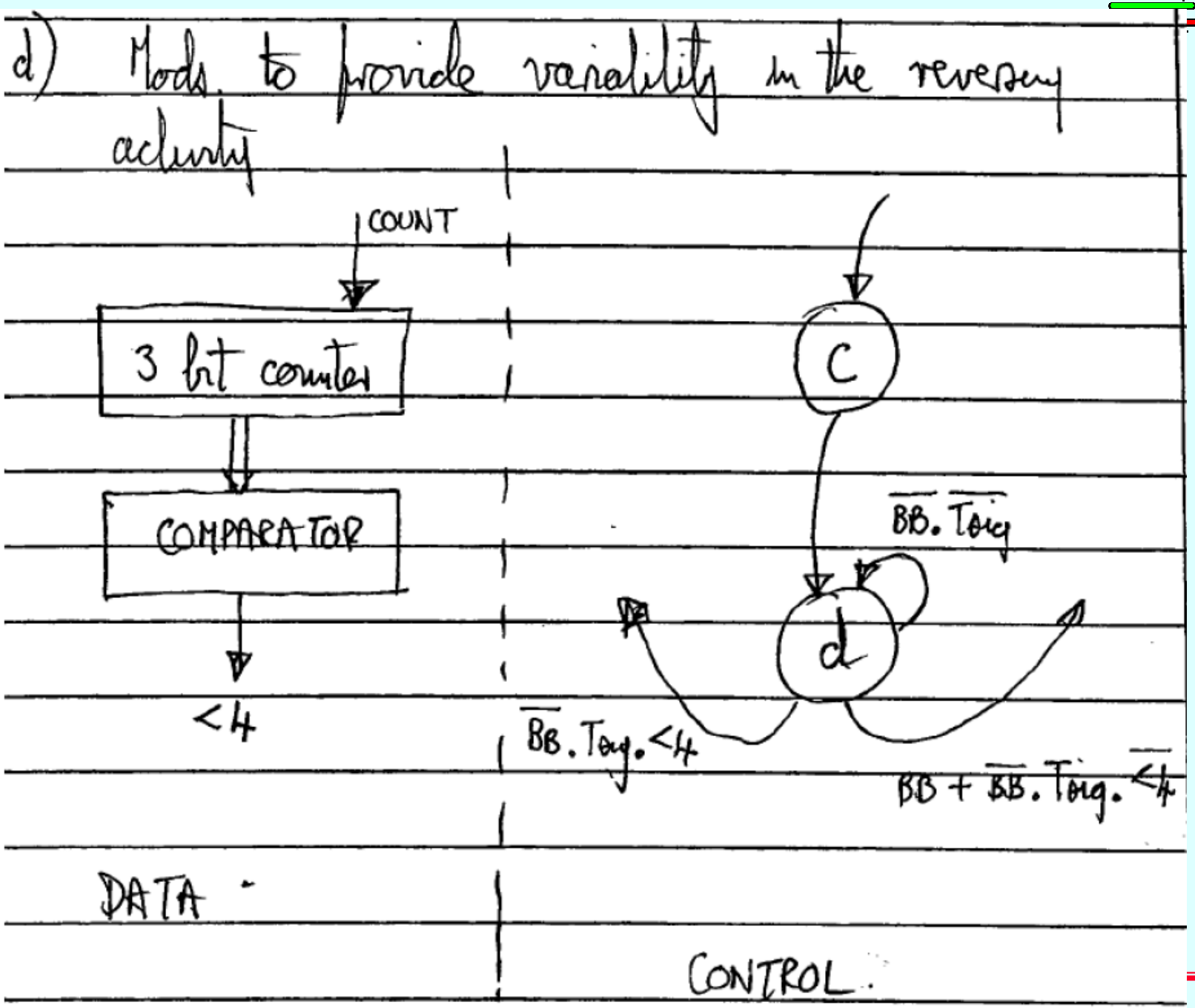
Design

$$D_A = A.B + B.C + A.C + A.\overline{T_{arg}} + C.(LB + RB)$$

$$\therefore D_B = \bar{A}B\bar{C} + \bar{A}\bar{B}C.LB + \bar{A}BC(\overline{BB} + T_{arg})$$

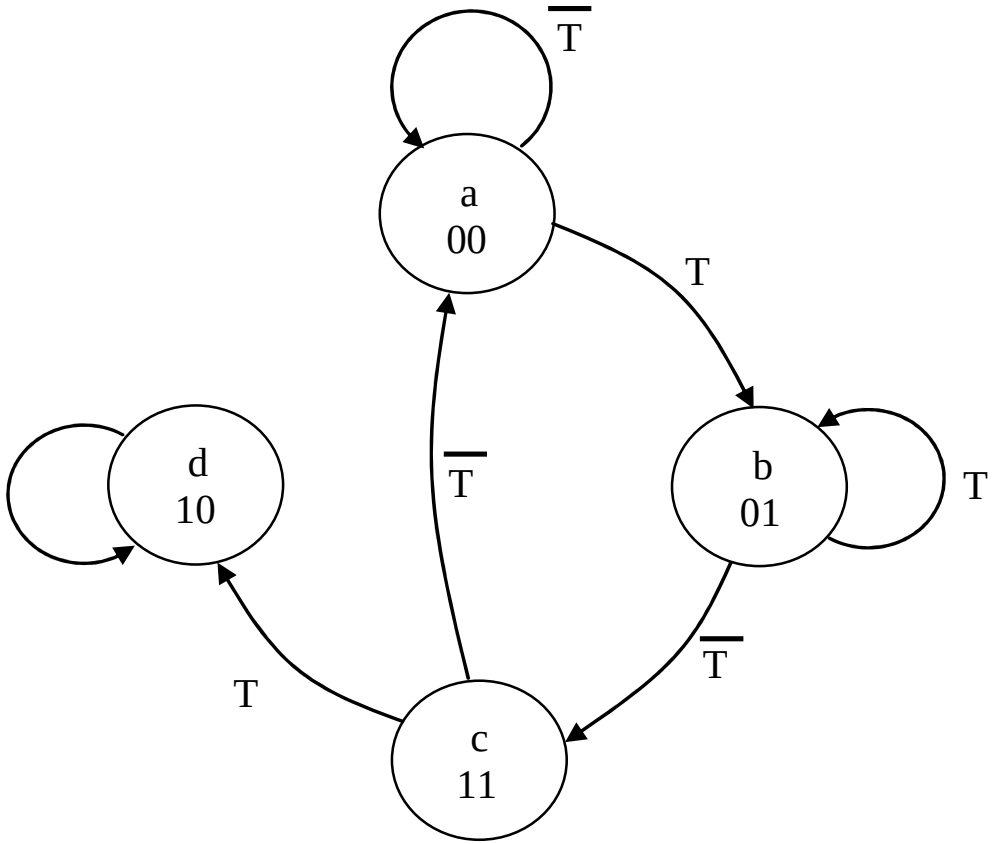
$$D_C = \bar{A}B + \bar{A}\bar{C}\,start + \bar{A}C.RB + A\bar{B}\bar{C}.T_{arg}$$

d) Mods. to provide variability in the reverse activity



2006 Q1

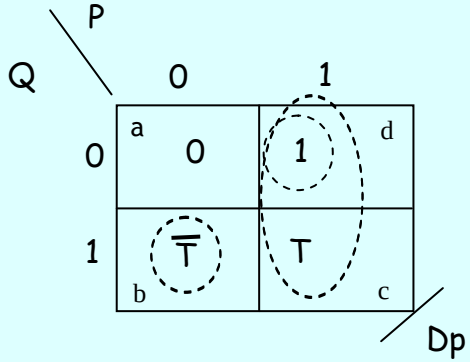
Convert the following state diagram into a set of excitation equations for a D-type implementation. T is a single input variable and each of the 4 states (a to d) has been given an appropriate binary coding.



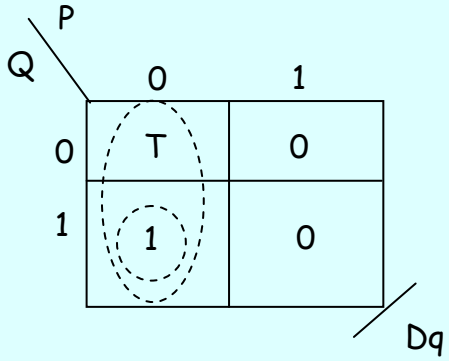
State transition table

State	Transitions	Code of target state		Boolean on transition	
		P	Q	Dp	Dq
a	a→a	0	0	-	-
	a→b	0	1	-	T
b	b→b	0	1	$\overline{-}$	$\overline{T}$
	b→c	1	1	T	T
c	c→a	0	0	-	-
	c→d	1	0	T	-
d	d→d	1	0	1	-

Karnaugh maps



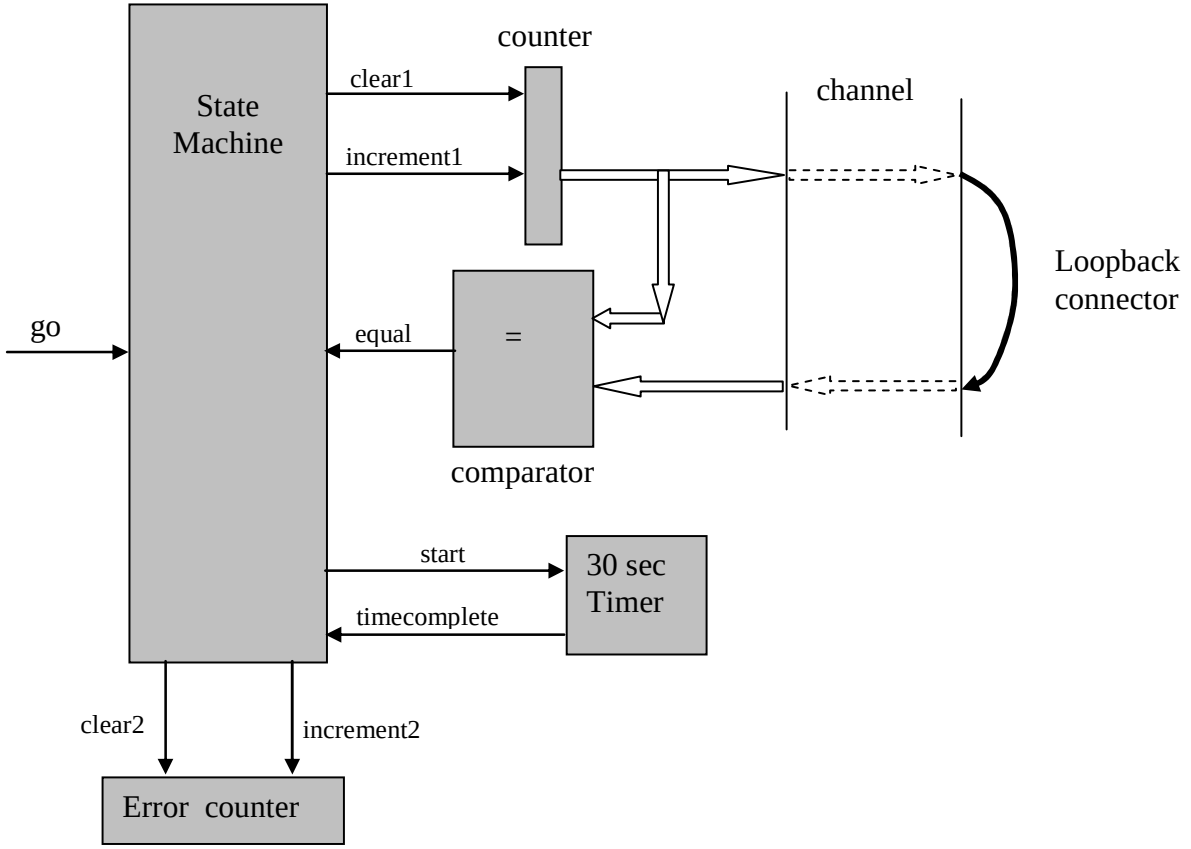
$Dp = P\overline{Q} + PT + \overline{P}Q\overline{T}$



$Dq = \overline{P}Q + \overline{P}T$

2006 Q2

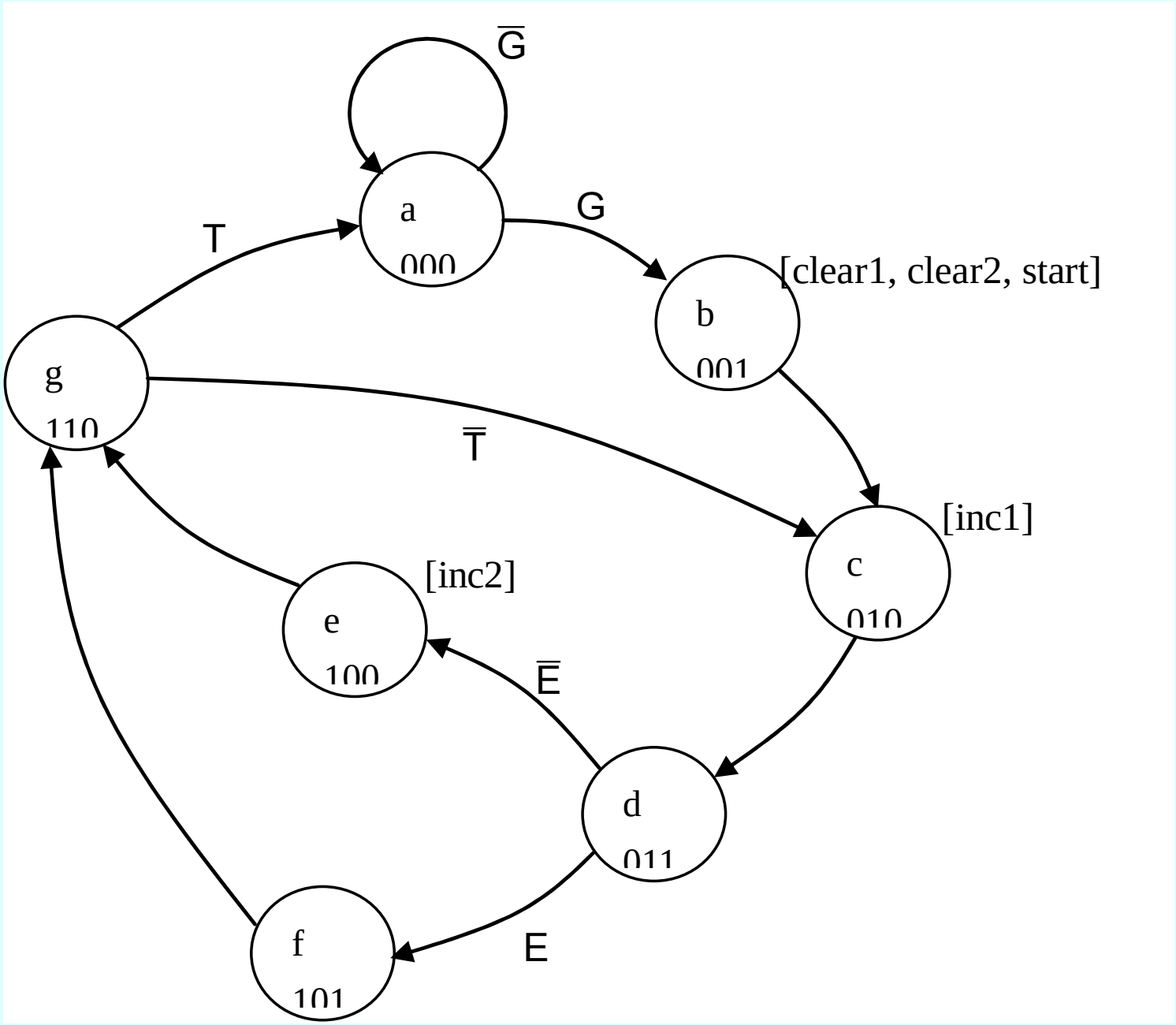
A long communications channel needs to be tested for potential problems. The technique used is to send a sequence of values through the channel and route them back through a loopback connection. The simplest sequence is to send the numbers 0, 1, 2, 3, …etc. Each sent value is compared to the received value. Any difference will cause an error counter to be incremented. A test session is to be run for 30 seconds. A timer unit will provide this feature. During a test session, a large number of values will be sent through the channel and compared for errors. The block diagram is as follows



20

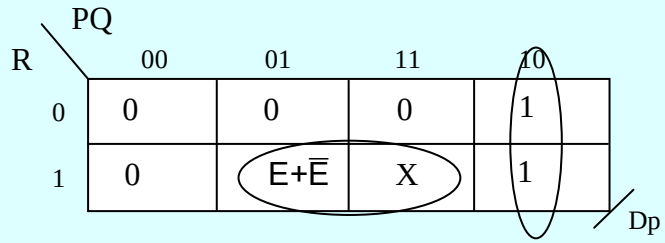
Generate a state transition diagram for this system. Using a Moore machine representation, show the outputs as part of the state machine.

Hence produce a set of excitation equations for a D-type flip-flop implementation.

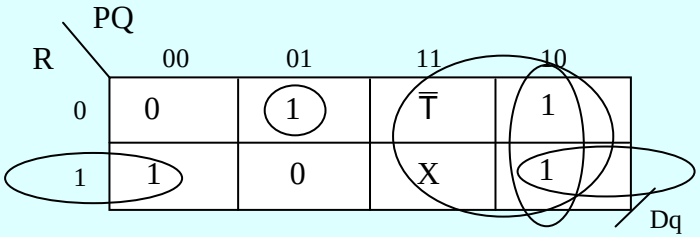

$\overline{G}$
a
000
G
[clear1, clear2, start]
b
001
T
g
110
$\overline{T}$
[inc1]
c
010
[inc2]
e
100
$\overline{E}$
d
011
f
101
E

Transition table

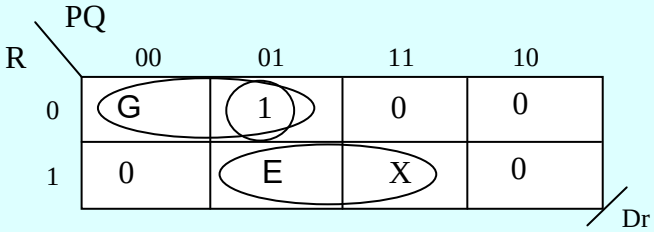
	Target state					
	P	Q	R	Dp	Dq	Dr
a→a	0	0	0	-	-	-
a→b	0	0	1	-	-	G
b→c	0	1	0	-	1	-
c→d	0	1	1	-	1	1
d→e	1	0	0	$\bar{E}$	-	-
d→f	1	0	1	E	-	E
e→g	1	1	0	1	1	-
f→g	1	1	0	1	1	-
g→c	0	1	0	-	$\bar{T}$	-
g→a	0	0	0	-	-	-



$Dp = P\overline{Q} + QR$



$Dp = \overline{P}Q\overline{R} + P\overline{Q} + \overline{Q}R + P\overline{T}$



$Dr = \overline{P}Q\overline{R} + \overline{P}\overline{R}G + QRE$