INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE CENTRATOM TRIESTE

SMR/443 - 3

ICTP - INFN COURSE IN
"BASIC VLSI DESIGN TECHNIQUES"
6 November - 1 December 1989

DIGITAL DESIGN

"Algorithmic State Machines" and "Realizing Logic in Hardware"

A. DE GLORIA
Dipartimento di Ing., Biofisica ed Elettronica
Università di Genova
Via all'Opera Pia 11/A
16145 Genova
Italy

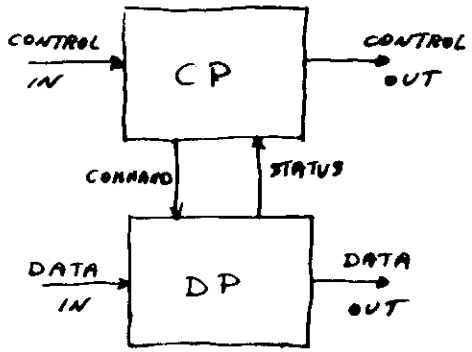
These are preliminary lecture notes, intended only for distribution to participants.

The microarchitecture of any computing unit can be subdivided into 2 parts:

- The control part
- The Data Path.

- The Data Path is composed of hardware structures that perform data transformation and data storage.

- The Control Part performs the control of the flux of the algorithm the computing machine has to accomplish.



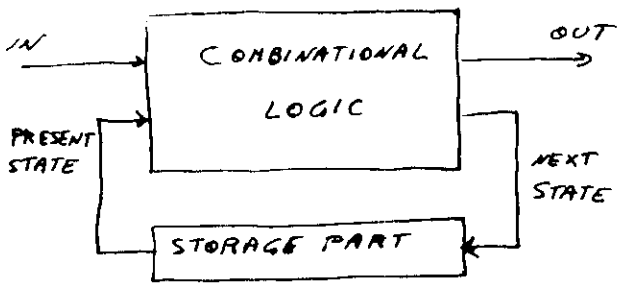
The control part, on the basis of external controls and of the data path status, and of the present machine state, decides what commands have to be send to the data path in order to perform a given algorithm.

The state of the machine implies the knowledge of present and past conditions to determine future behaviour.

At a given instant the control part perform a logic function which, by using appropriate input information, moves at the proper time to the next state.

A control part is then composed of two parts:

- COMBINATIONAL PART
  That performs the logic function that moves from the current state to the next state and determines the proper output signals for the data path.
- STORAGE PART
  That loads the state of the control part.



THE CONTROL PART CAN BE SYNTHESISED BY USING DIFFENT METHODS_ ONE OF THEM IS THE

ALGORITHMIC STATE MACHINES METHOD_

ALGORITHMIC STATE MACHINES (ASM)

SUBDIVIDED THE CONTROL PART INTO TWO TYPES:

SYNCHRONOUS CP (SCP)

ASYNCHRONOUS CP (ACP)

- SCP CHANGE THEIR STATE ACCORDING TO AN EXTERNAL SIGNAL, THE CLOCK. THEY CAN HAVE TWO TYPES OF INPUT SIGNALS: SYNCHRONOUS AND ASYNCHRONOUS_ THE FORMER MANTAIN A CONSTANT RELATION WITH CK; THE LATTER ARE COMPLETELY INDIPENDENT FROM CK_
- IN ACP THE CHANGE OF STATE IS ONLY DUE TO THE CHANGE OF VALUE OF THE INPUT SIGNALS_

The design of an ASM implies the determination of the two equations:

$$X_N \leftarrow f(X_P, IN)$$

where X_N = the next state
X_P = the present state
IN = the input signals

$$OUT \leftarrow g(X_P, IN)$$

where OUT = output signals.

The two equations are derived from the algorithm that describes the behaviour of the machine.

An algorithm can be expressed through two types of statements:

* Transformation statements:
  They specify operations performed by the data path, and represent commands issued by the control part.
  A transformation can be determined by:
  - the state of the machine,
  - both the state and inputs

* Control statements:
  They specify how to control the flux of the algorithm on the basis of the present state and of the input signals.

Example

2 bit up-down synchronous counter
with set and reset.

start:

Sϕ: if reset then out = ϕ, goto Sϕ
else if set the out = 11, goto S3
else if up then out = 1, goto S1
else out = 11, goto S3

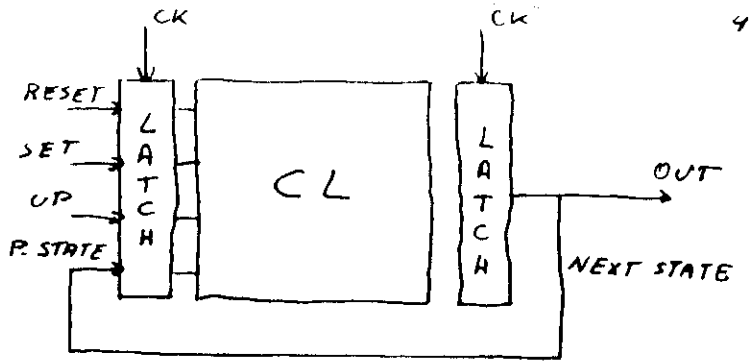
S1: if reset then out = ϕ, goto Sϕ
else if set then out = 1, goto S3
else if up then out = ϕ1, goto S2
else out = ϕ, goto Sϕ.

S2:

S3: - - - -

if ---- decision statement

out = 1 transformation statement.



TRUTH TABLE

RESET	SET	UP	P. STATE	N. STATE
1	–	–	– –	0 0
ϕ	1	–	– –	1 1
ϕ	ϕ	ϕ	$\phi\phi$	1 1
ϕ	ϕ	1	$\phi\phi$	ϕ1
ϕ	ϕ	ϕ	ϕ1	$\phi\phi$
ϕ	ϕ	1	ϕ1	1ϕ
ϕ	ϕ	ϕ	1ϕ	ϕ1
ϕ	ϕ	1	1ϕ	1 1
ϕ	ϕ	ϕ	1 1	1ϕ
ϕ	ϕ	1	1 1	$\phi\phi$

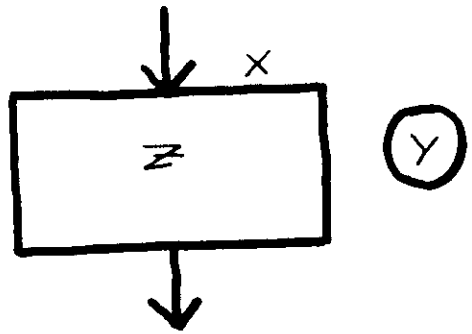
ASM CHARTS

The ASM CHARTS are a graphic method for defining the algorithm the ASM executes. The ASM CHARTS are also very useful in the synthesis of the ASM.

An ASM CHART is composed of three symbols:

- STATE
- DECISION
- CONDITIONAL OUTPUT.

STATE SYMBOL



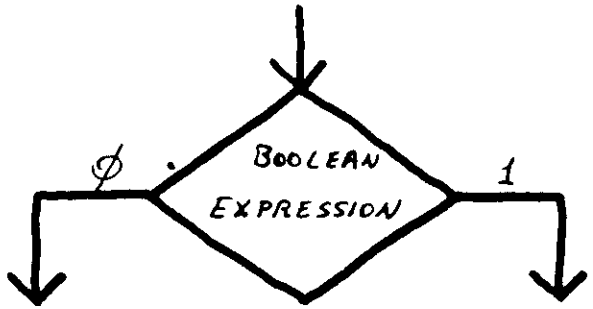
where:

X : is the binary code assigned to the state

Y : is the name of the state

Z : is the list of the active outputs

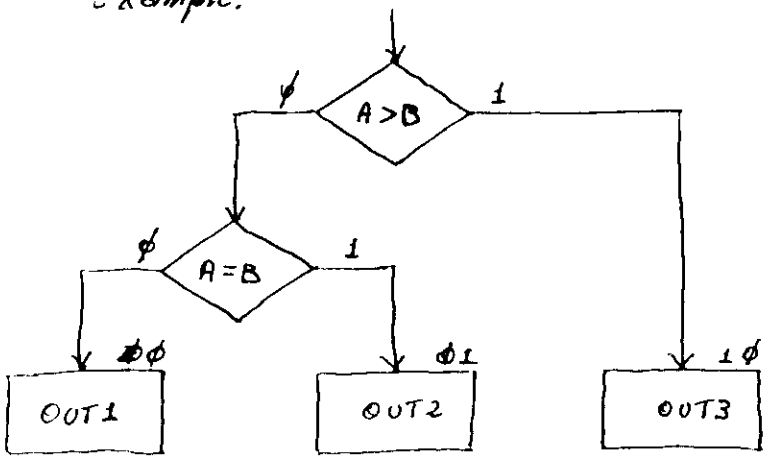
DECISION SYMBOL



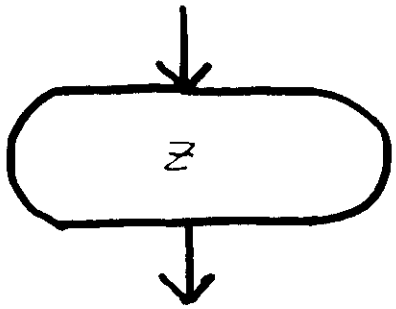
It describes the inputs of the ASM, and has two outputs.

The output is selected according to the value of the boolean expression.

Example:



CONDITIONAL OUTPUT SYMBOL



where :

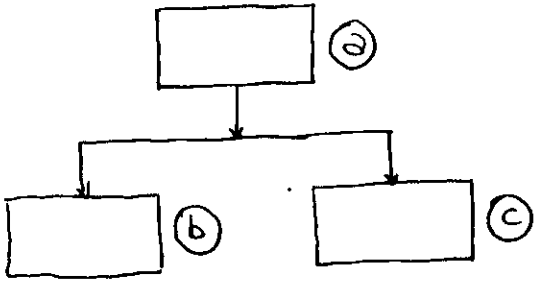
z is the list of the active outputs.

It has been introduced to represent the the direct dependence of the outputs on the inputs.

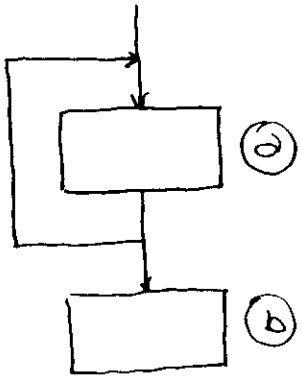
It must be always followed by a state symbol, and must always follow a decision symbol.

SOME COMMON ERRORS OFTEN OCCUR IN THE DESIGN OF ASM CHARTS.

THEY CAN BE:

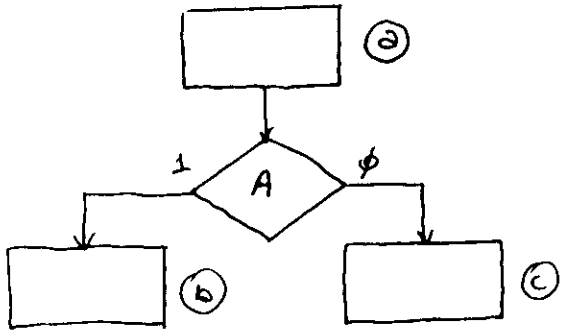


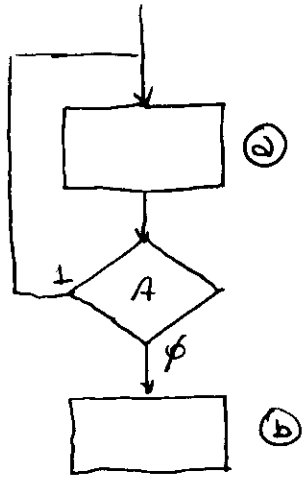
A machine cannot have two states at the same time. No rule is shown to select one state between b and c.



It is not possible to decide what is the next state.

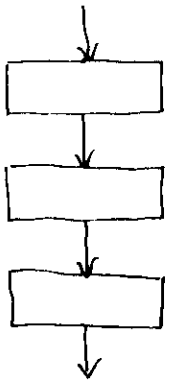
The right configurations are:





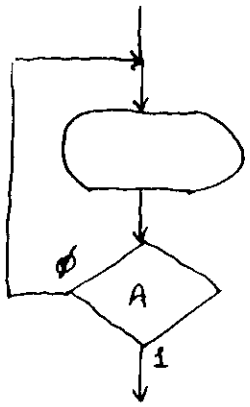
This configuration is called:

WAITING LOOP

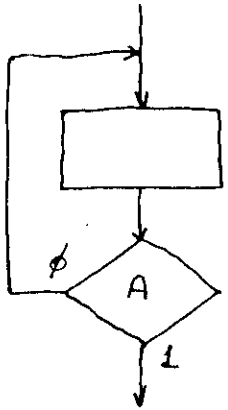


This configuration is valid for synchronous machines only.

In asynchronous machine it produces a continuos state change



WRONG

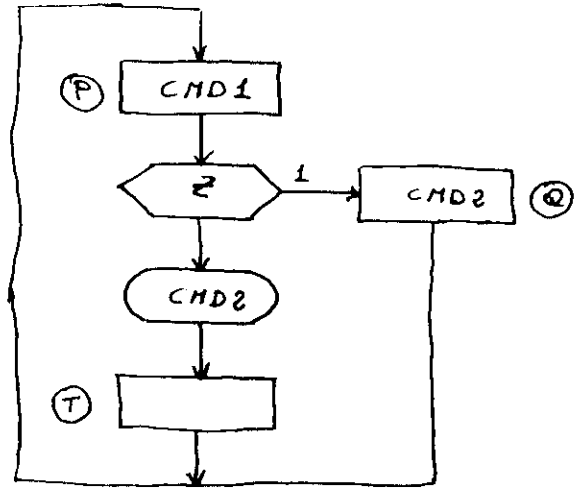


RIGHT

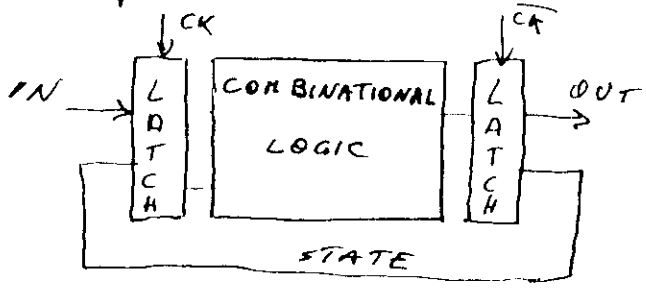
WAITING LOOPS WITH CONDITIONAL OUTPUTS MUST BE AVOIDED.

SYNTHESIS FROM AN ASM CHART

Consider the following ASM CHART:



The synthesis process allows to design the function performed by the combinational logic of an ASM.



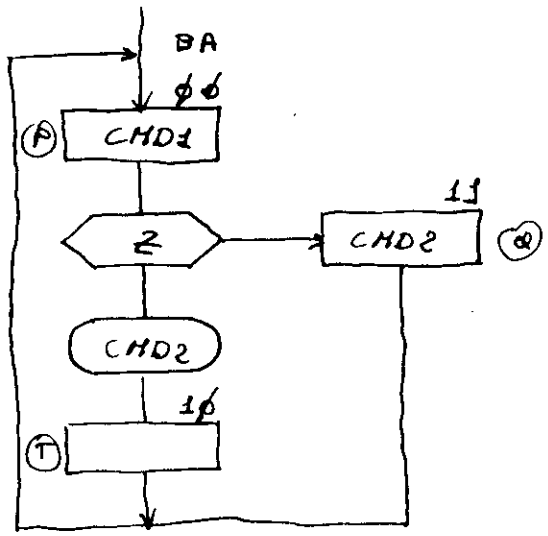
- The <u>first step</u> is to perform the

state assignment

of binary state variable values to states.

- In synchronous state machines with synchronous inputs the assignment can be arbitrary.

We choose to have two state variables (A, B) with the following state assignment.



The <u>second step</u> is to design the truth table of the combinational logic.

INPUT		OUTPUT		
z	BA	BA	CMD1	CMD2
φ	φφ	1φ	1	1
1	φφ	11	1	φ
—	1φ	φφ	φ	φ
—	11	φφ	φ	1

$B = \overline{B} \cdot \overline{A} \cdot \overline{z} + \overline{B} \cdot \overline{A} \cdot z = \overline{B} \cdot \overline{A}$

$A = \overline{B} \cdot \overline{A} \cdot z$

$CMD1 = \overline{z} \cdot \overline{A} \cdot \overline{B} + B \cdot A$

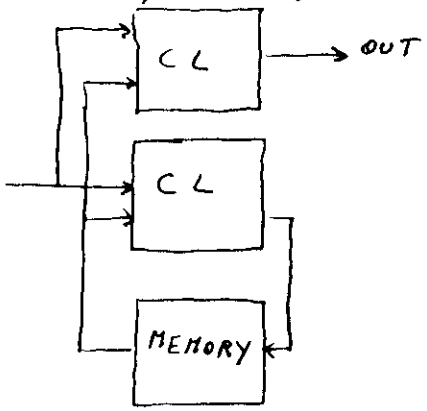
$CMD2 = \overline{B} \cdot \overline{A}$

The KARNAUGH MAPS CAN BE USED TO MINIMIZE LOGIC EQUATIONS

CLASSIFICATION OF THE ALGORITHMIC STATE MACHINES

The ASM's have been sub-divided into 5 classes

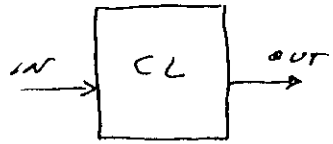
The schematic diagram of an ASM is the following:



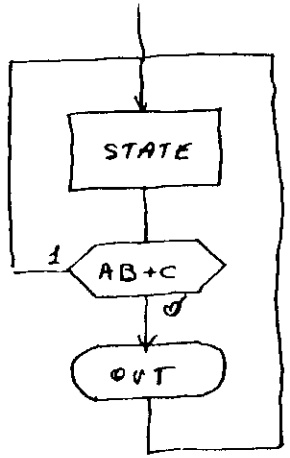
This schematic includes the all 5 classes and it represents the more general ASM.

CLASS ϕ MACHINE:

The class ϕ machine is composed of a combinational circuit:



It has only one state. The output are always of conditional type.



CLASS 1 MACHINE

the next state does not depend on the current one and the outputs do not depend on inputs.

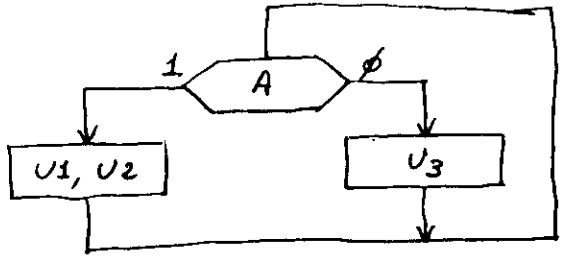
A clan 1 machine is described by the following equations:

$$X \leftarrow f(IN)$$

$$OUT \leftarrow g(x)$$

An ASM CHART for a CLASS ϕ1 machine does not contain any conditional outputs.

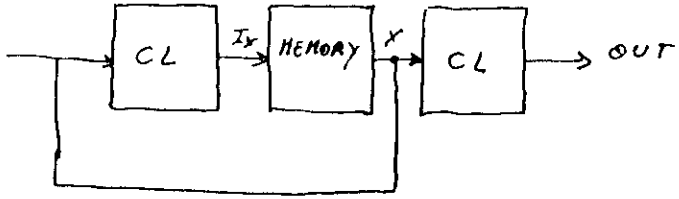
Example:



Truth table

INPUTS A	NEXT STATE x	NEXT STATE y	OUTPUTS $U_1 U_2 U_3$
ϕ	1	1	$\phi\phi 1$
1	ϕ	ϕ	11ϕ

CLASS 2 MACHINE



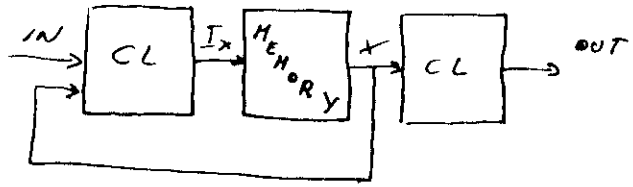
The class-2 machines do not have any input, the outputs only depend on the state.

$$OUT = g(x)$$
$$x = f(x)$$

These machines execute sequences of states, an example is given by the synchronous counter.

An ASM CHART for this class does not contain any decision symbol.

CLASS 3 MACHINE

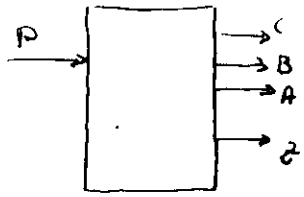


A state depend on both the previous state and the inputs.
It is described by the following equations:

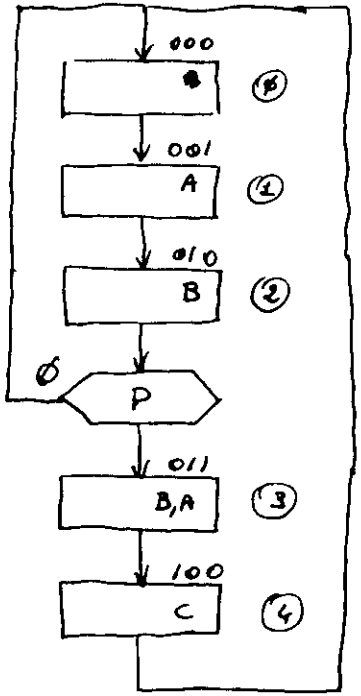
$$x \leftarrow f(x, IN)$$
$$OUT \leftarrow g(x)$$

Example:

Design a synchronous counter with an input P and an output z.
If P is true the counter counts up to five,
otherwise the counter counts up to 3.
z signals when the counter reaches the value "ϕ_4".



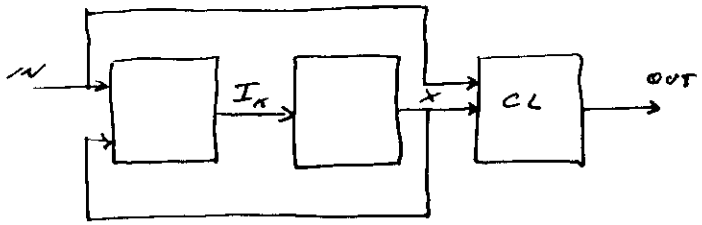
A, B, C = state variables



Truth table

INPUT	P.STATE	N.STATE	OUT
P	CBA	CBA	Z
—	000	001	1
—	001	010	0
0	010	000	0
1	010	011	0
—	011	100	0
—	100	000	0

CLASS-4 MACHINE

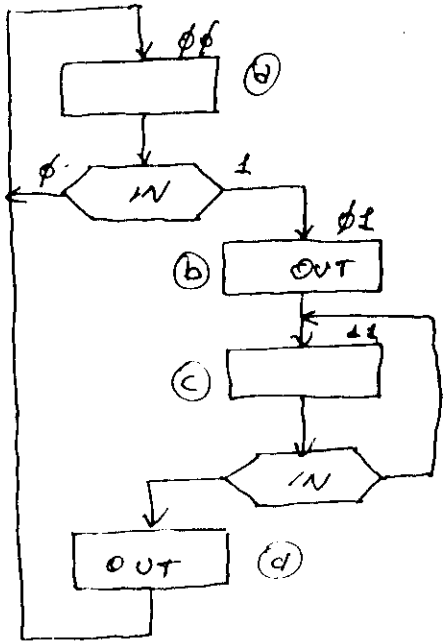


A class-4 machine CHART can contain conditional outputs.

Design an ASM which with the input of the form:

outputs the followings waveform:

We can use a SYNCHRONOUS ASM with clock signal whose period is very short with respect to the input signal period.



INPUTS	PRESENT STATE	NEXT STATE	OUTPUTS
IN	A B	A B	OUT
ϕ	00	00	0
1	00	01	0
–	01	10	1
1	10	10	0
ϕ	10	11	0
–	11	00	1

state assignment

A \ B	0	1
ϕ	a	b
1	d	c

FLIP-FLOP

SET-RESET

It is the most simple asynchronous ASM and it is also the basic component to build an asynchronous ASM.

It has two inputs: R, S

and two outputs: Q, $\overline{Q}$

The R input signal causes the output Q set to 1,

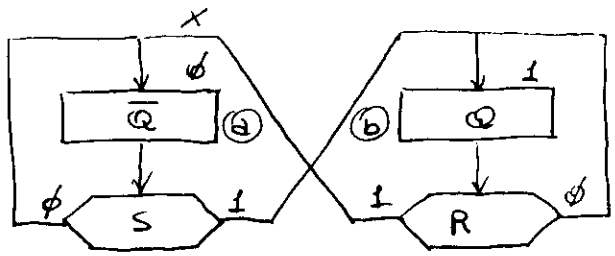
The S input signal causes the output Q reset to ϕ.

When both R and S are ϕ the output is unchanged.

It is not possible to have R and S with value 1

R	S	Q_n $\overline{Q}_n$
ϕ	ϕ	Q_{n-1} $\overline{Q}_{n-1}$
ϕ	1	1 ϕ
1	ϕ	ϕ 1
1	1	Not defined

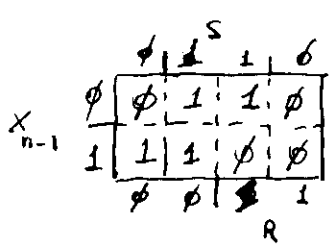
The ASM CHART of the SR-FLIP FLOP:

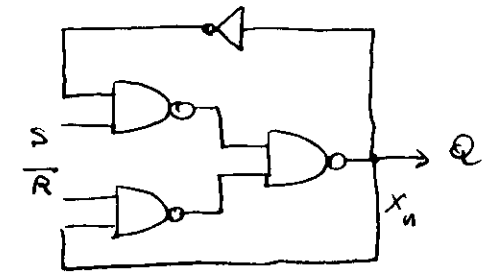


It can be seen that if both S and R are "1, the machine oscillates between the two states.

Truth table:

INPUTS		P.STATE	N. STATE	OUTPUTS	
R	S	X_{n-1}	X_n	Q	$\overline{Q}$
-	ϕ	ϕ	ϕ	ϕ	1
-	1	ϕ	1	ϕ	1
ϕ	-	1	1	1	ϕ
1	-	1	ϕ	1	ϕ



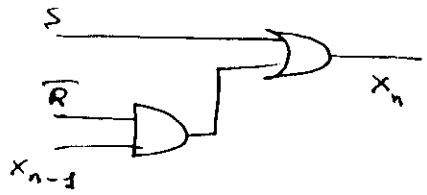


$$\overline{X_n} = S \cdot \overline{X}_{n-1} + \overline{R} \cdot X_{n-1}$$

But S and R cannot be "1" at the same time, then:

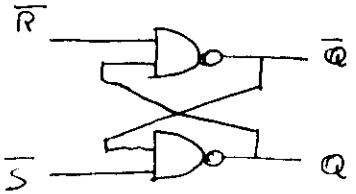
X_{n-1} \ S	0	1	-	ϕ
1	1	1	-	ϕ
R				

$$X_n = S + \overline{R} \cdot X_{n-1}$$



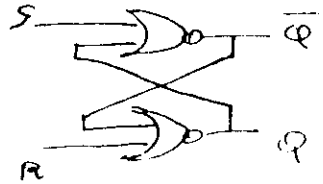
NAND SYNTHESIS

$$X_n = S + \overline{R} \cdot X_{n-1} = \overline{\overline{S} \cdot \overline{\overline{R} \cdot X_{n-1}}}$$



NOR SYNTHESIS

$$X_n = S + (\overline{R} \cdot X_{n-1}) = \overline{\overline{R + \overline{X}_{n-1}} + S}$$



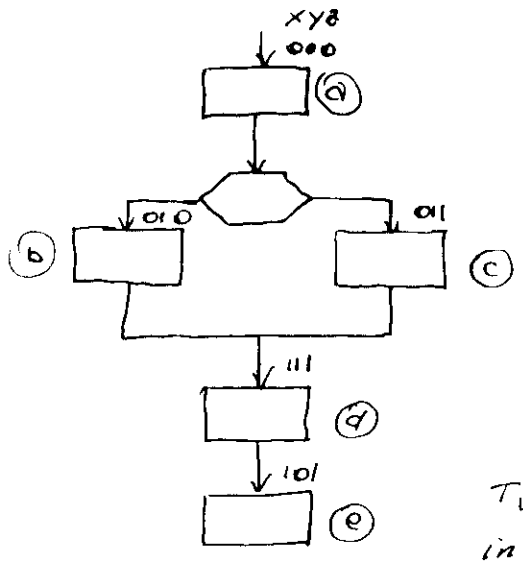
STATE ASSIGNMENT

There are two general criteria that can be used to assign states in an ASM CHART.

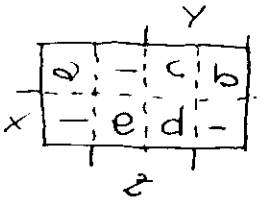
1) <u>Minimization of the number of variable changes</u>

These criterium use the <u>Hamming distance</u>. It is defined Hamming distance between two states the number of variables which change moving from one state to the other.

Consider the following ASM CHART:



Ex: a: 000, c: 011
Hamming distance a − c = 2



Two contiguous states in the map have Hamming distance = 1

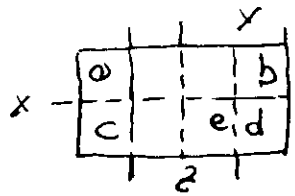
The criteria consist in evaluating the Hamming distance for all the possible state transitions:

TRANSITION	DISTANCE	TRANSITION	DISTANCE
a - b	1	c - d	1
a - c	2	d - e	1
b - d	2	e - a	2

The sum of the distances is 9.

The optimum assignment is that where the sum is minimum.

In our case it is possible to have a better assignment



In fact:

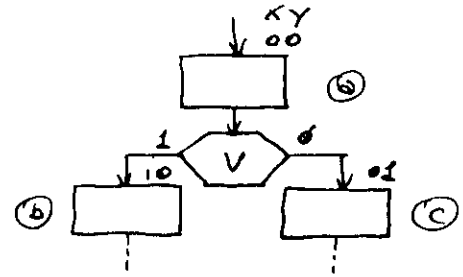
TRANSITION	DISTANCE	TRANSITION	DISTANCE
a - b	1	c - d	1
a - c	1	d - e	1
b - d	1	e - a	3

The sum is 8.

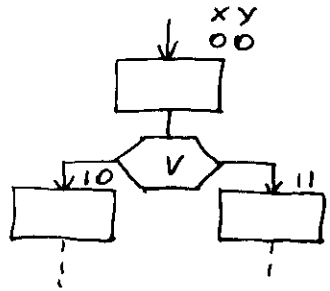
2) Reduction of the dependency on input variables

Ex:

The following chart:



can be transformed in:

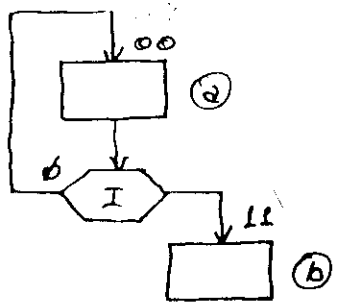


where X does not depend on V.

In an ASM chart with many decision blocks it is convenient to limit the dependency of the state variables on input variables.

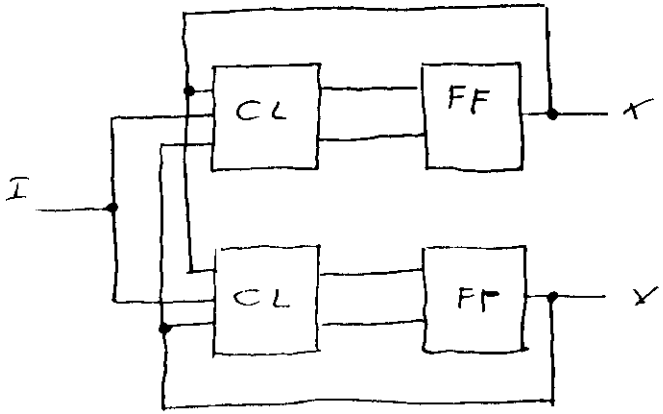
STATE ASSIGNMENT IN ASYNCHRONOUS ASM

Consider the following chart:

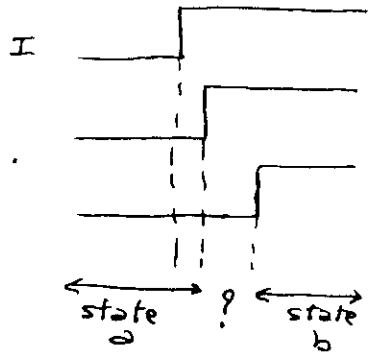


which is part of an ASM CHART with 4 states.

The asynchronous ASM will be composed of two SR flip-flops and two combinational logics.



When I has a transition from 0 to 1, the flip-flops have to set to 1. But we cannot expect that x and y reach the 1 at the same time.

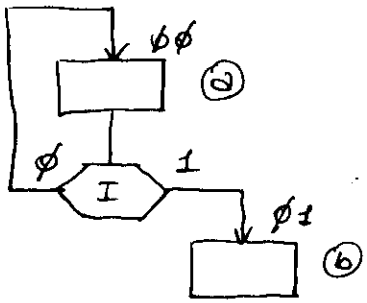


We have 3 different situations:

1) (a) → (?) → (b): From state (a) the machine goes to state (?) and then to state (b).
   If in the ASM the (?) exists we can have:
2) (a) → (?) : The machine assumes the (?) state as a stable state.
3) (a) → (?) → (??): The state of the machine is unpredictable.

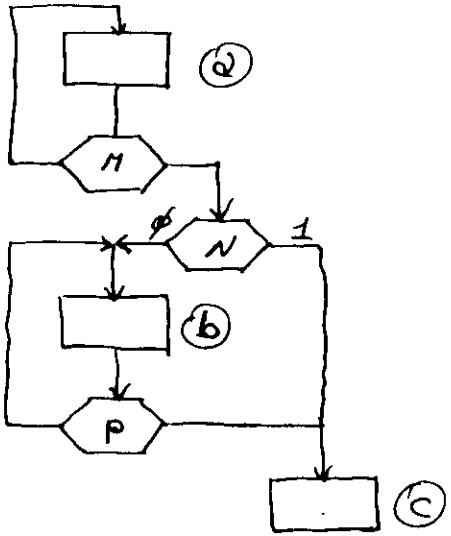
To avoid the situations shown above it is necessary that the distances among the states are always 1.

in this way a state transition causes the change of only one variable and spurious states cannot reached

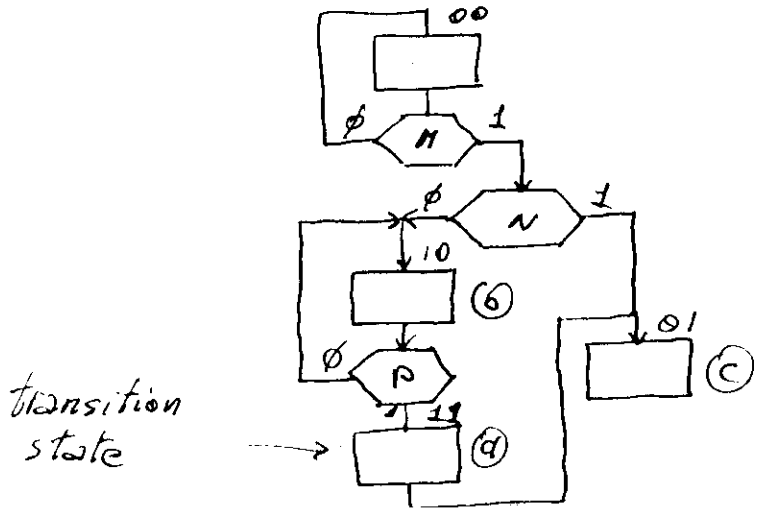


It is not always possible to have the distance equal to 1.

Consider the following chart:



It is necessary to add a transition state to insure that the distance between state is 1.

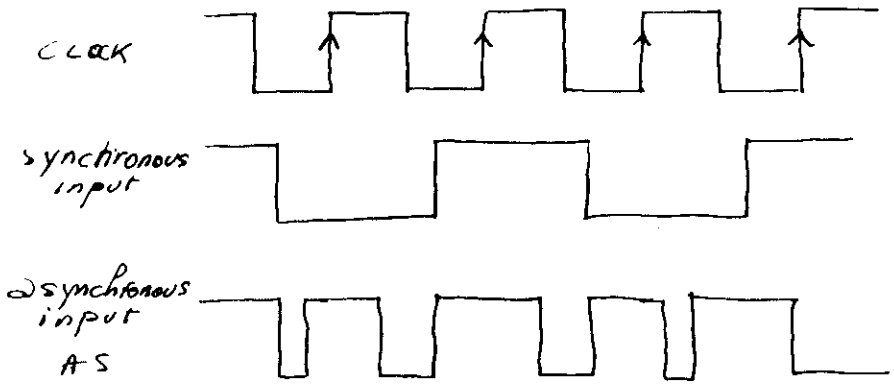


An alternative to the addition of transition states is the introduction of new state variables.
In this way larger maps are obtained and is easier to have distance equal 1.

STATE ASSIGNMENT IN SYNCHRONOUS ASM

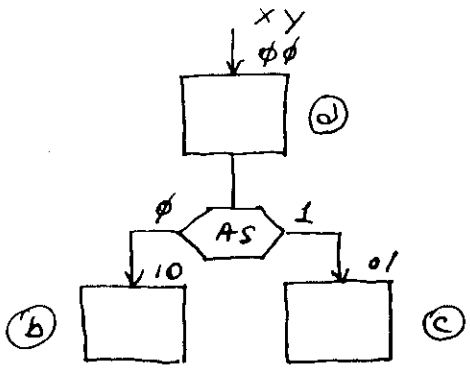
TRANSITION RACES:

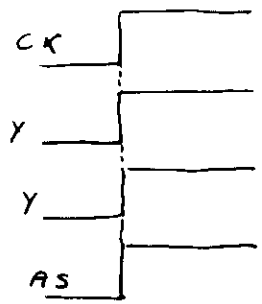
SUPPOSE to have an ASM active on the rising edge of the clock. We know that synchronous inputs have a fixed relation with the clock, while asynchronous inputs are random.



We have problems with asynchronous inputs when they change at the same time of the clock.

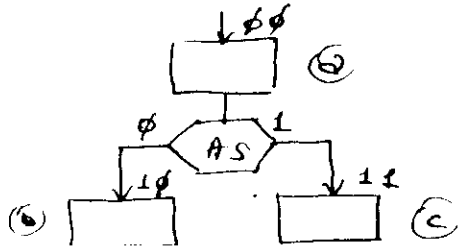
Consider the following ASM chart:





If AS changes on the rising edge of CK we don't know how it will be interpreted (if AS = φ, or AS = 1).
Therefore from state (a) the machine can indifferently move to state (b) or to state (c).
But it is also possible that, for the delays of the flip-flops, the machine reaches the states φφ or 11.

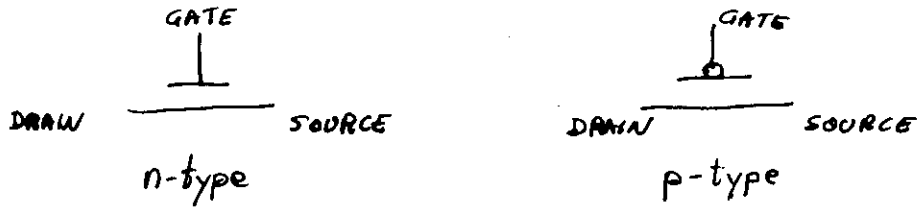
To insure that the machine reaches the states b or c it is necessary that the two states have distance equal to 1.



REALIZING LOGIC IN HARDWARE

CMOS technology provides two types of transistors:

n-type transistor
p-type transistor

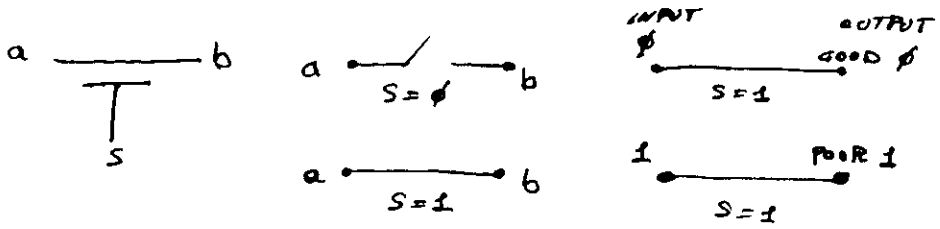


The gate controls the passage of current between the drain and source.
Simplifying this to the extreme allows the MOS transistor to be viewed as simple on/off switches.

In the following we will assume that a "1" is a high voltage normally set to 5 volts, and called POWER or V_{DD}.
The symbol "φ" will be assumed to be a low voltage, that is normally set to φ volts and called GROUND or V_{SS}.

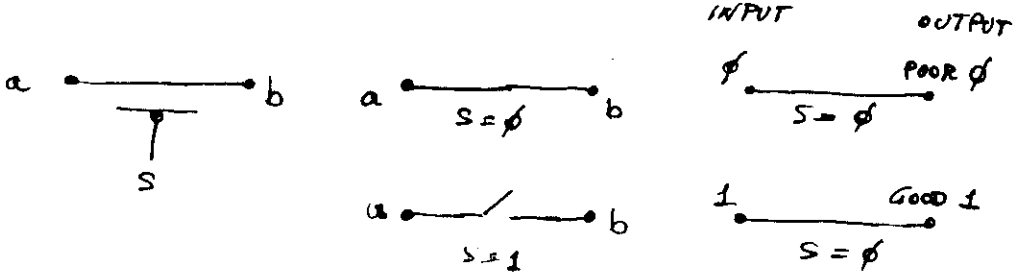
N-type transistor

- The N-switch is closed or ON when drain and source are connected. This occurs when there is a 1 on the gate
- The N-switch is open or OFF when drain and source are disconnected, this occurs when there is a ϕ on the gate.
- * The N-switch is a perfect switch when a zero is passed
- The N-switch is an imperfect switch when a one is passed - In doing this the 1 voltage level is reduced a little.

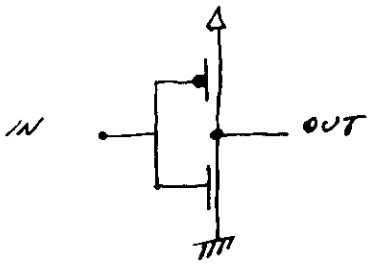


P-type transistor

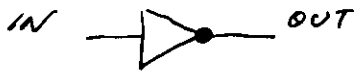
- The P-switch is closed or ON when there is a ϕ on the gate.
- * The P-switch is open or OFF when there is a 1 on the gate.
- * A P-switch is a perfect switch for passing 1 signals.
- * A P-switch is a imperfect when passing ϕ signals.



THE INVERTER

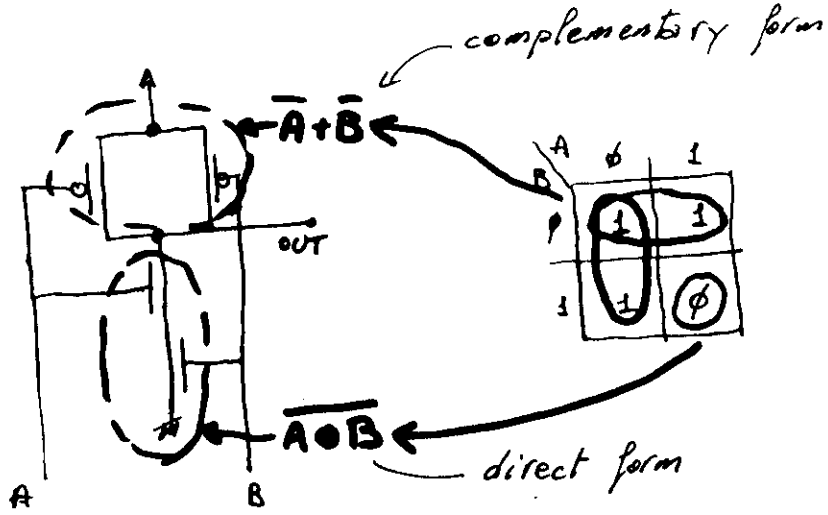


TRANSISTOR SCHEMATIC



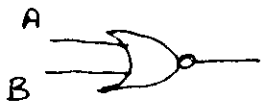
LOGIC SYMBOL

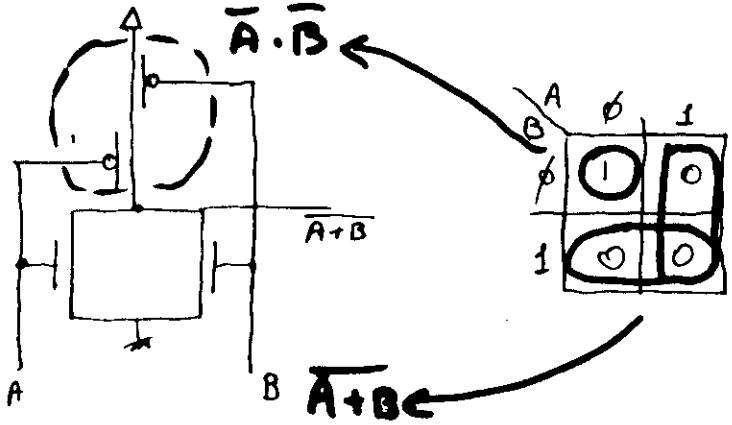
NAND GATE



To have perfect switching:
The p-structure is used for passing 1's
The n-structure is used for passing ϕ's

NOR GATE

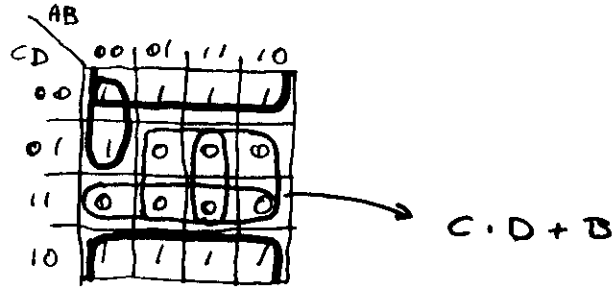




COMPOUND GATES

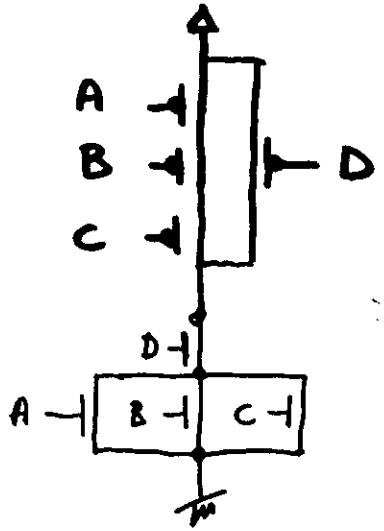
A compound gates is derived by using a combination of series and parallel switch structures.

Examples: $F = \overline{((A+B+C)\cdot D}$

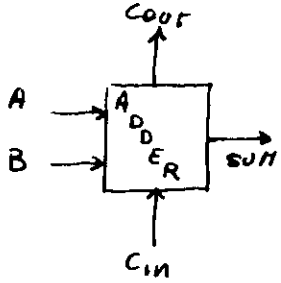


$C\cdot D + B\cdot D + A\cdot D = D\cdot(A+B+C)$

$\bar{D}+\bar{A}\cdot\bar{B}\cdot\bar{C}$



THE ADDER

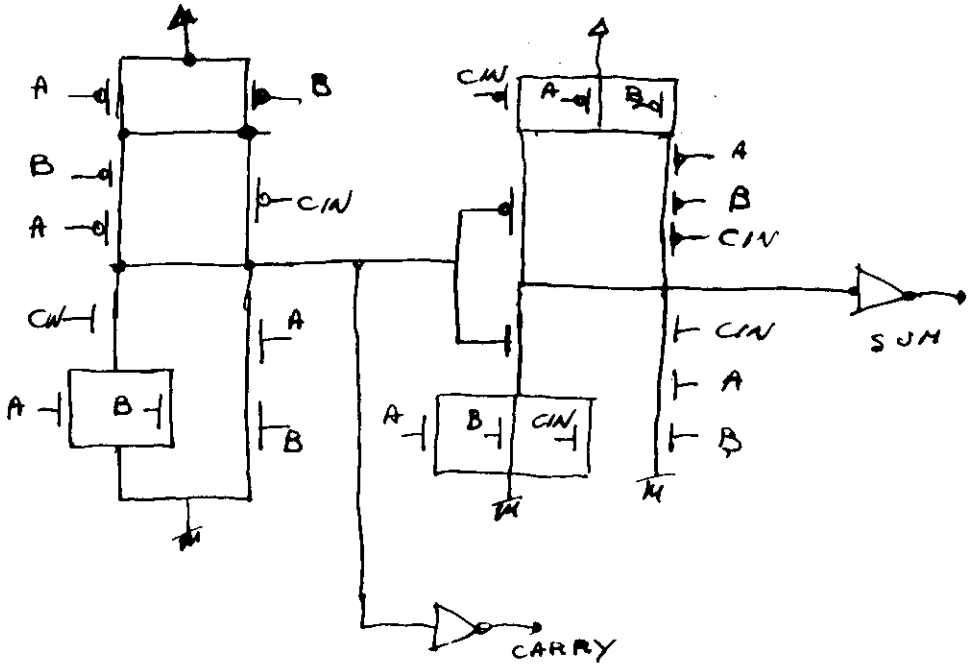


A	B	C_{in}	SUM	COUT
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

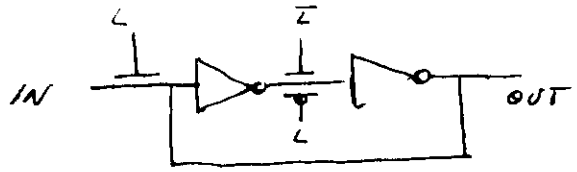
$COUT = AB + C(A+B)$

$SUM = ABC + A\overline{B}\overline{C} + \overline{A}BC + \overline{A}B\overline{C} =$

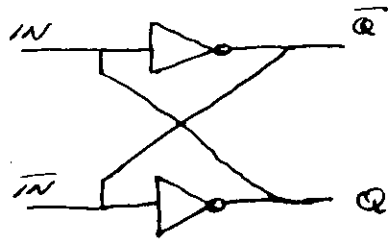
$\overline{COUT} \cdot (A+B+CIN) + A \cdot (B+CIN)$



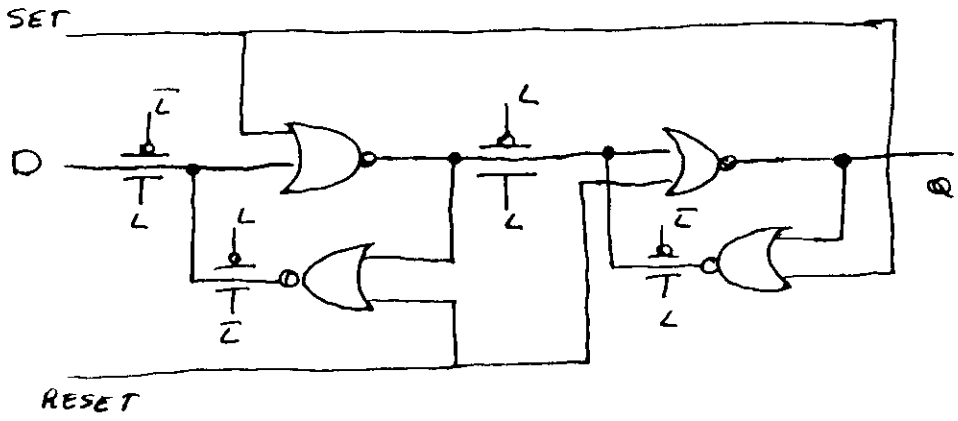
LATCH REGISTER



PSEUDO-STATIC LATCH



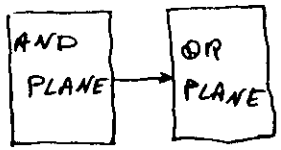
static latch



STATIC D flip-flop with set and reset

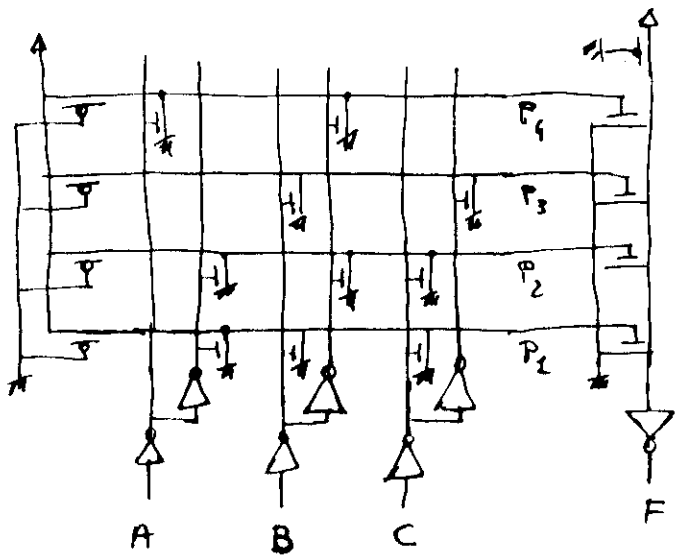
PROGRAMMABLE LOGIC ARRAY
(AN INTRODUCTION)

A PLA provides a regular structure for implementing combinatorial and sequential logic functions.
A typical PLA uses an AND-OR structure shown below:



The basis for a PLA is sum of products form of representation of logic expressions.
The electrical design of a CMOS PLA depends on the style of PLA. A straightforward implementation is the NOR-NOR form.

PSEUDO-nMOS NOR gate



$$F = \bar{A}BC + \bar{A}B\bar{C} + B\bar{C} + A\bar{B}$$
$$P_1 \quad P_2 \quad P_3 \quad P_4$$

Simply by designing transistors it is possible to program the structure to implement any logic function.