PE Refresher Course

Digital Systems and Computers

Joanne Degroat

degroat.1@osu.edu

ece.osu.edu/~degroat

The Basics

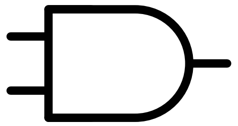
- Basic Switching Algebra
  - Truth Tables of Basic Functions AND and OR

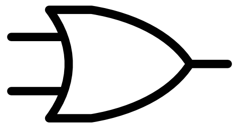
AND

A	B	C=A*B
0	0	0
0	1	0
1	0	0
1	1	1

OR

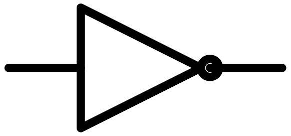
A	B	C=A+B
0	0	0
0	1	1
1	0	1
1	1	1



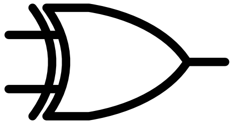


The Basics (cont)

- Inversion – The not or inverter gate



- Exclusive OR - XOR



NOT - inversion

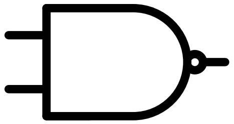
A	$\overline{A}$
0	1
1	0

XOR - exclusive OR

A	B	$X = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

And two other common gates

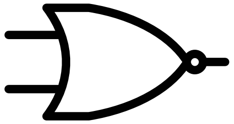
- The NAND – NOT AND



NAND

A	B	$C=\overline{A*B}$
0	0	1
0	1	0
1	0	0
1	1	0

- The NOR – NOT OR



NOR

A	B	$C=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Some Basic Theorems

● $A+0 = A$	Equality	● $A*1 = A$
● $A+1 = 1$	Identity	● $A*0 = 0$
● $A+B = B+A$	Commutative	● $A*B = B*A$
● $A+BC = (A+B)(A+C)$	Distributive	● $A(B+C) = AB+AC$
● $A+A' = 1$	Involution	● $A*A' = 0$
● $A+AB = A$	Absorption	● $A(A+B) = A$

More Theorems

- $A+\overline{A}B = A + B$ $\qquad$ $A(\overline{A} + B) = AB$
- $\overline{(A + B)} = AB$ $\qquad$ $\overline{(AB)} = \overline{A} + \overline{B}$

DeMorgan’s Law

- $A*A = A$ $\qquad$ $A + A = A$
- $(A) = A$
- $AB + \overline{A}C + BC = AB + \overline{A}C$
- Dual: $\qquad (A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C)$

Truth Tables

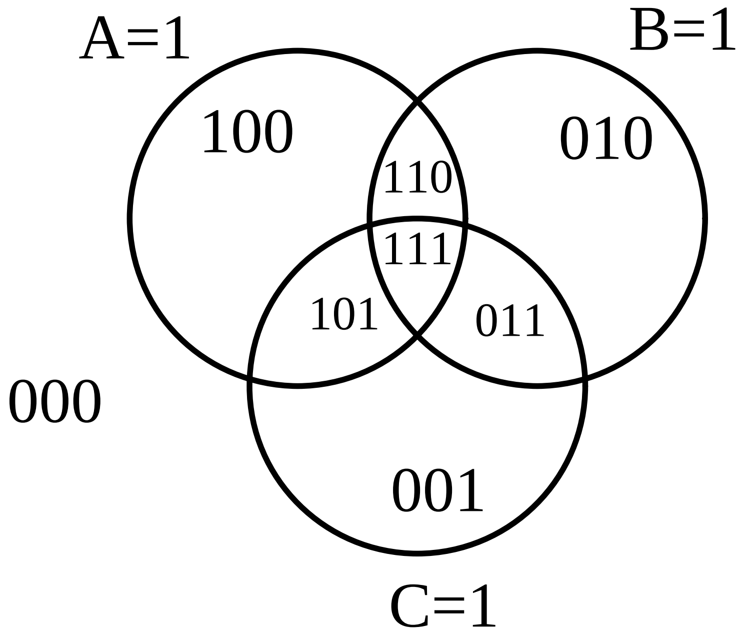
Truth tables can be used to prove equalities

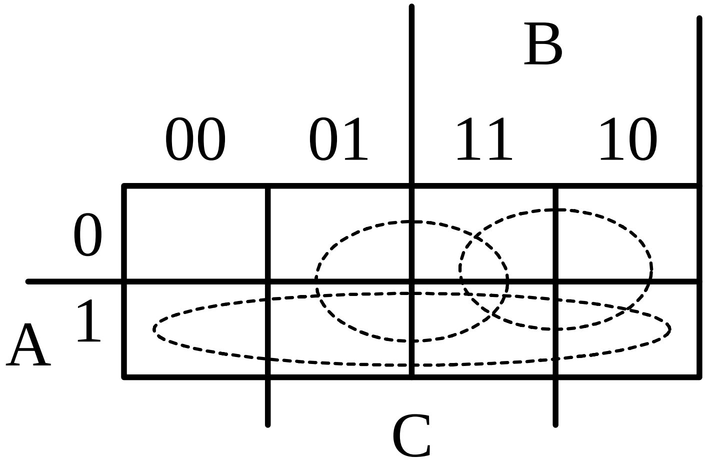
Proof of DeMorgan's Law

A	B	A+B	$\overline{(A+B)}$	$\overline{A}$	$\overline{B}$	$\overline{A}*\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

=

Venn Diagram and Karnaugh Maps





Function Simplification

- $f(a,b,c) = \bar{a}\bar{b} + bc + \bar{a}b\bar{c}$
- Simplify using Karnaugh map

Start with each term

Map $\bar{A}\,\bar{B}$

A \ BC	00	01	11	10
0	1	1		
1				

Map B C

A \ BC	00	01	11	10
0			1	
1			1	

Function Simplification

- $f(a,b,c) = \bar{a}\bar{b} + bc + \bar{a}b\bar{c}$

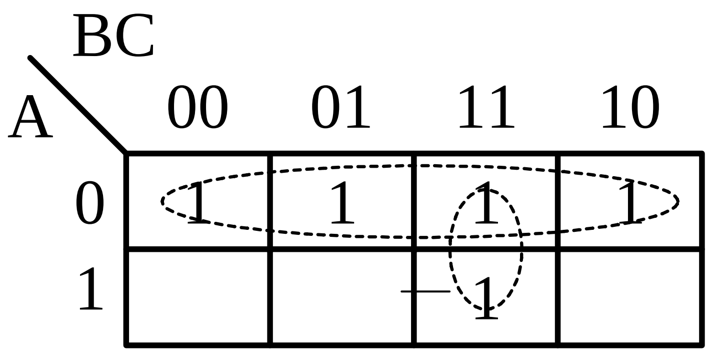
Map $\bar{A}\,B\,\bar{C}$

$$\begin{array}{c|c|c|c|c}
A \backslash BC & 00 & 01 & 11 & 10 \\
\hline
0 & & & & 1 \\
\hline
1 & & & & \\
\end{array}$$

A term with a
Single element results
In 4 ones
A term with 2 elements
Gives two
A term with 3 elements
Gives a single 1

Simplify K maps

- Conbining the previous three maps
- Can represent what a K map shows by a sum of products
- Take the largest group possible
  - On one line, a square, two
  - Group needs to be a
  - power of 2



A \ BC	00	01	11	10
0	1	1	1	1
1			1	

$$f(a,b,c) = \bar{a} + bc$$

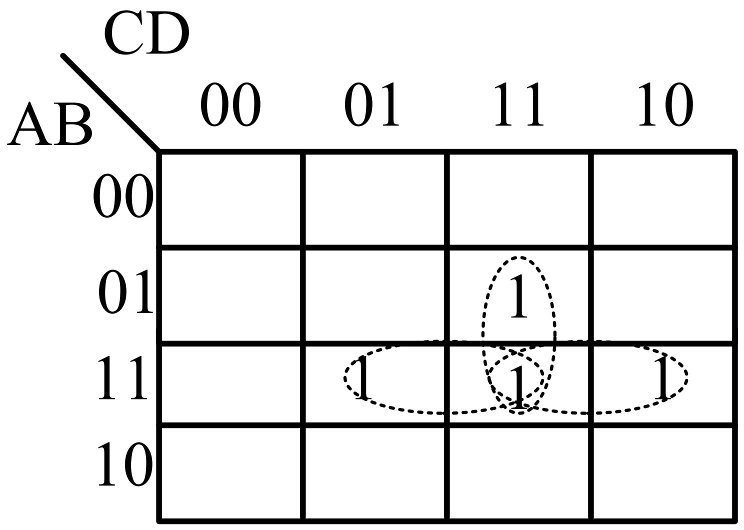
4 variable functions

- Also expressed in minterm notation
- f(a,b,c,d) = Σm(7,13,14,15)
  - So function is a 1 when abcd = 0111 (7)
    1101 (13)
    1110 (14)
    1111 (15)

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

Form into largest groups

- For 4 variables form into groups of
  - 16 – function would be a 0 or a 1
  - 8 – 1 variable
  - 4 – 2 variables
  - 2 – 3 variables
- Simplifies to
  - F = ABD + BCD + ABC



Another Example of function simplification

- From a Truth Table
- Step 1 – map to K- maps

x	y	z	F1	F2
0	0	0	1	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

F1

x \ yz	00	01	11	10
0	1		1	1
1	1		1	

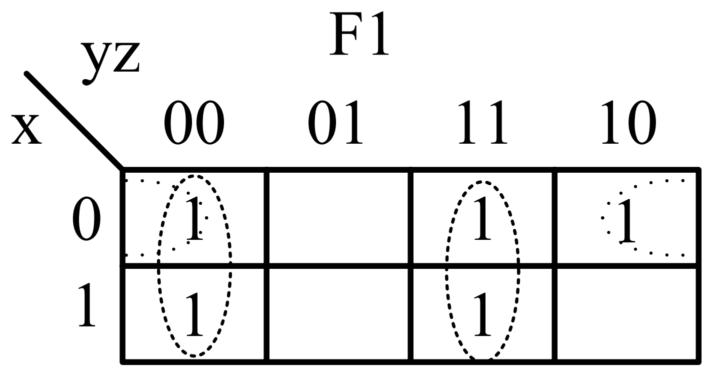
F2

x \ yz	00	01	11	10
0			1	
1		1	1	1

Step 2 – Generate simplified equation

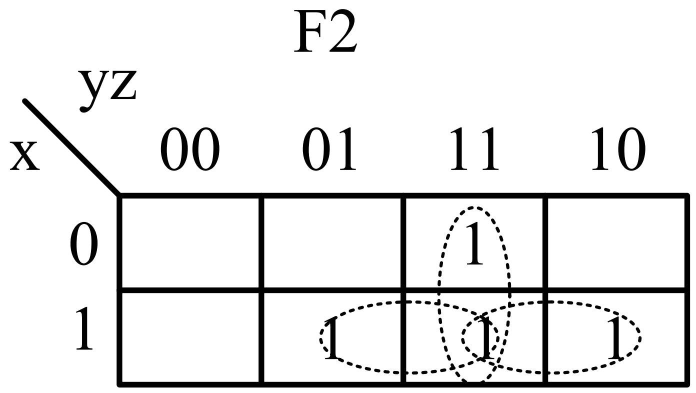
- For F1

$F1 = \overline{y}\,\overline{z} + y\,z + \overline{x}\,\overline{z}$



- For f2

$F2 = xz + xy + yz$



Example Problem

- Design a circuit having 1 output Z and 4 inputs A B C D which represents a BCD number, such that Z = 1 if the BCD number is greater than 5.
- BCD stands for binary coded decimal
  - Takes 4 binary digits
  - Only 0 through 9 are used

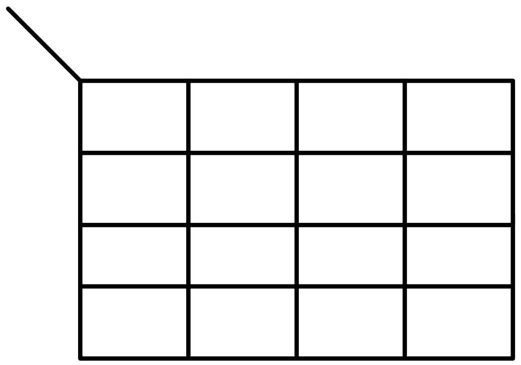
Map onto K map

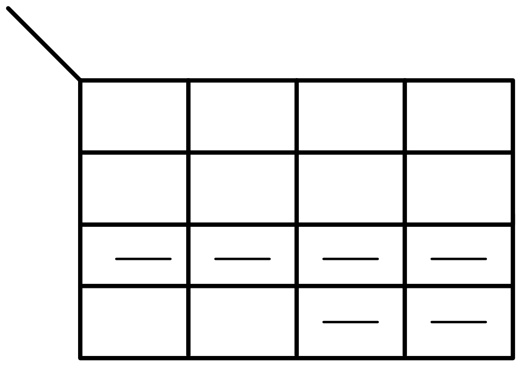
- Fill in a 1 whenever output should be a 1
- Here that would be 6, 7, 8 or 9
- And a 0 for blocks 0 through 5

AB \ CD	00	01	11	10
00				
01				
11				
10				

The K map cont

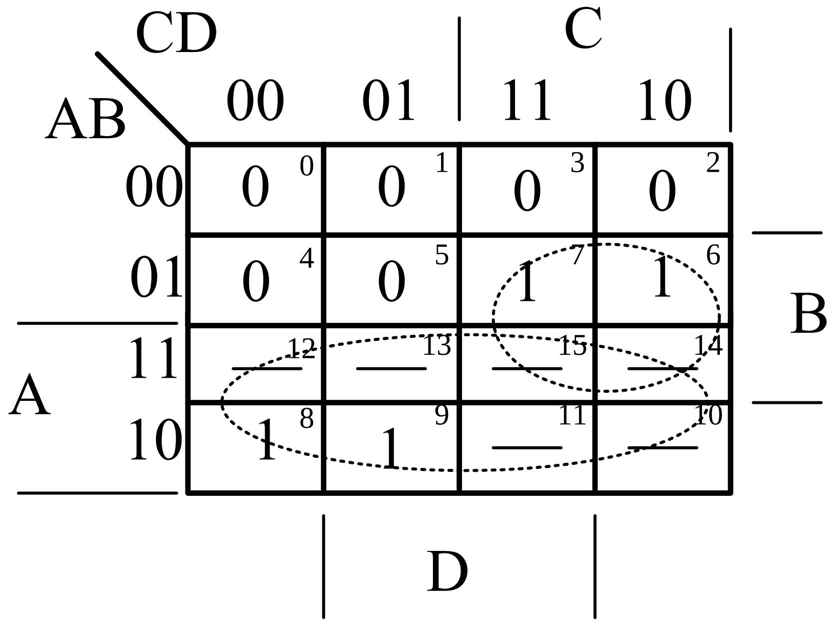
- Filled in
- What should other blocks be?
- They would be don’t cares as in BCD notation they will never occur
- And final K map for simplification is



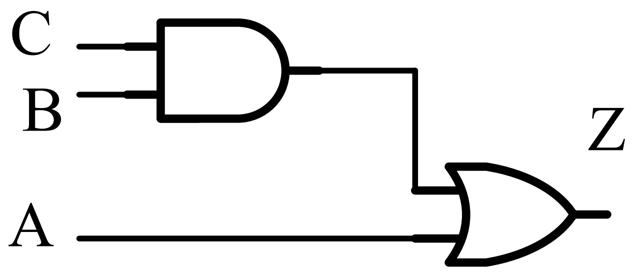


Simplified using don’t cares

- Form the largest power of 2 grouping from 1’s and don’t cares
- Cover all the 1’s but don’t need to cover all the don’t cares
- Get Z = A + BC

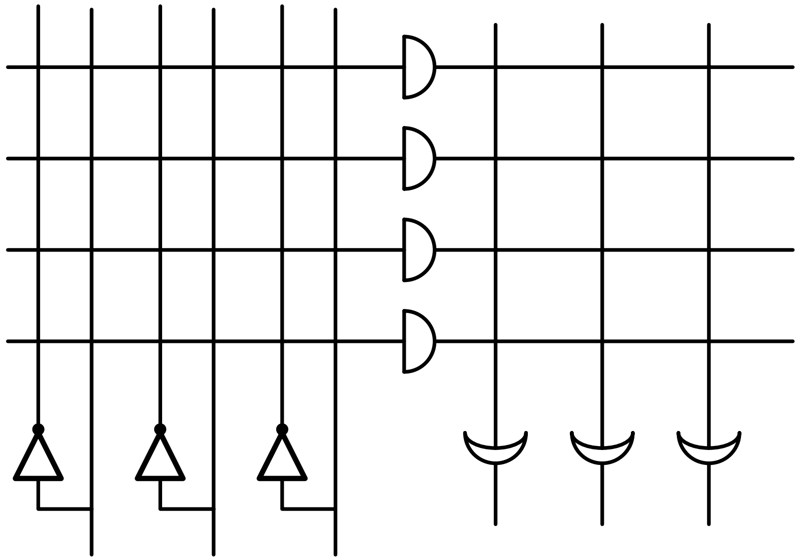


- Gate implementation



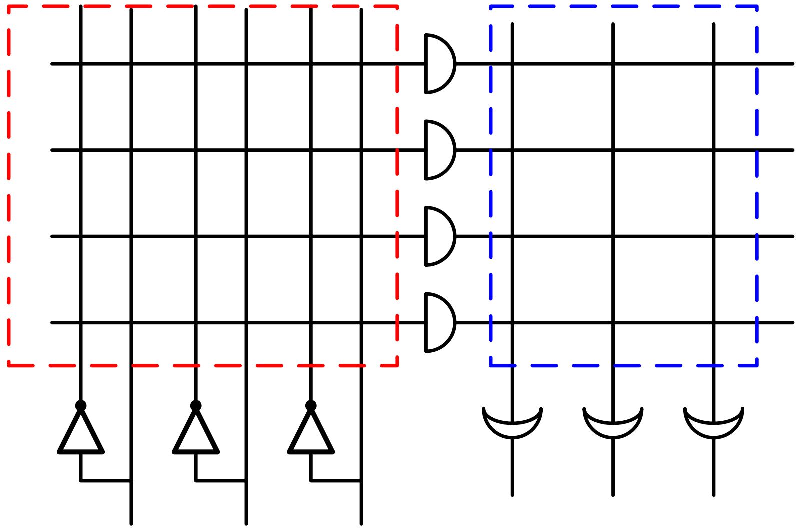
PLAs

- PLA – Programmable Logic Array
- PAL – Programmable Array Logic – much like a PLA but restricted connections
- Formed of an AND plane and an OR plane



PLAs

- PLA – Programmable Logic Array
- PAL – Programmable Array Logic – much like a PLA but restricted connections
- Formed of an AND plane and an OR plane



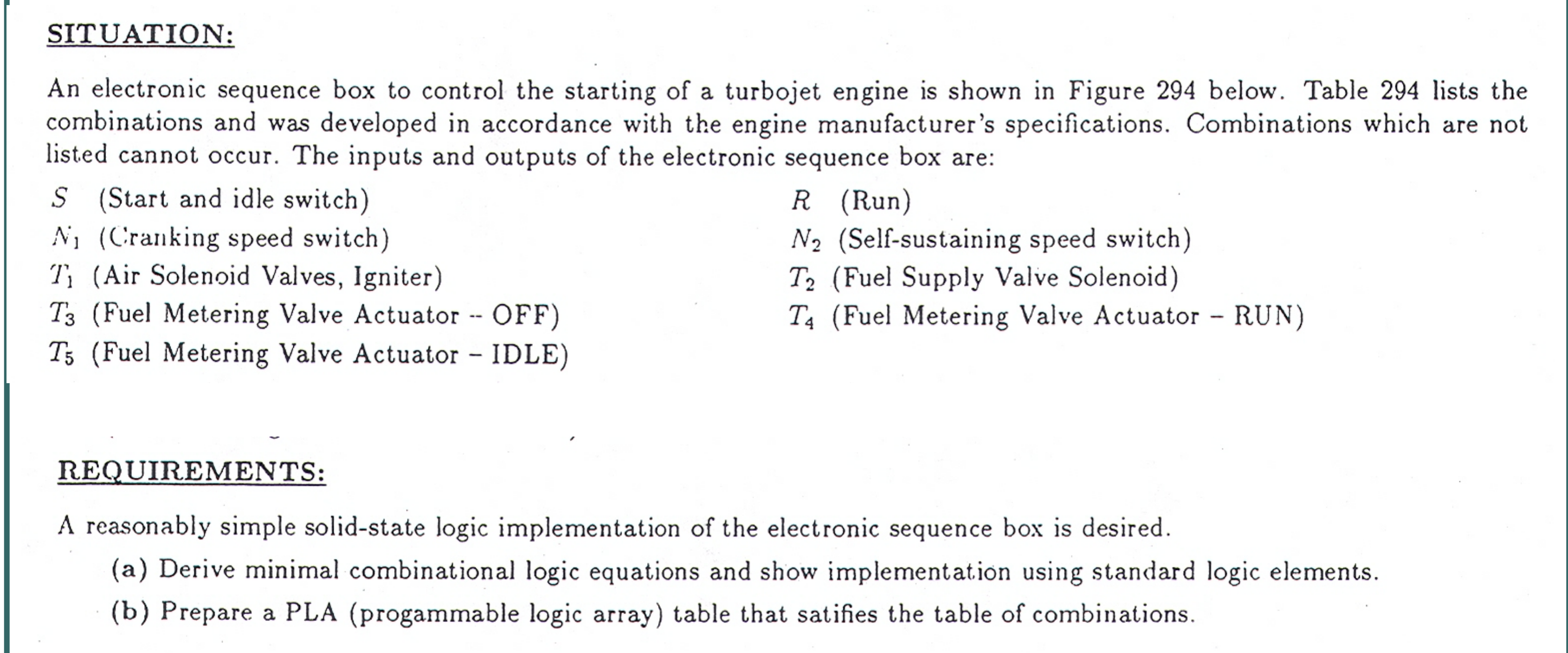
PLA example

- Problem: A sequence box to control automatic starting of a jet engine is to be designed. It has the following signals

- Problem stated on next slide.

Problem statement

- XX



SITUATION:

An electronic sequence box to control the starting of a turbojet engine is shown in Figure 294 below. Table 294 lists the combinations and was developed in accordance with the engine manufacturer's specifications. Combinations which are not listed cannot occur. The inputs and outputs of the electronic sequence box are:

S	(Start and idle switch)	R	(Run)
N_1	(Cranking speed switch)	N_2	(Self-sustaining speed switch)
T_1	(Air Solenoid Valves, Igniter)	T_2	(Fuel Supply Valve Solenoid)
T_3	(Fuel Metering Valve Actuator -- OFF)	T_4	(Fuel Metering Valve Actuator – RUN)
T_5	(Fuel Metering Valve Actuator – IDLE)		

REQUIREMENTS:

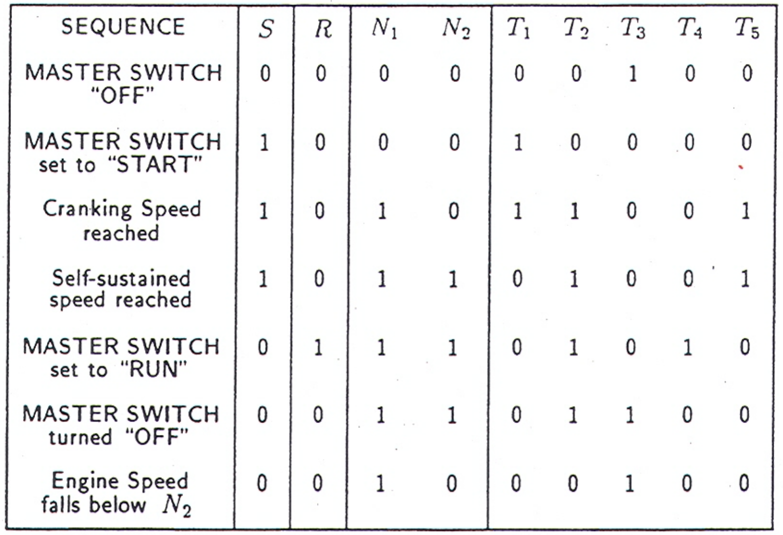
A reasonably simple solid-state logic implementation of the electronic sequence box is desired.

(a) Derive minimal combinational logic equations and show implementation using standard logic elements.

(b) Prepare a PLA (progammable logic array) table that satifies the table of combinations.

Problem statement translated to a table

- The table



SEQUENCE	S	R	N_1	N_2	T_1	T_2	T_3	T_4	T_5
MASTER SWITCH "OFF"	0	0	0	0	0	0	1	0	0
MASTER SWITCH set to "START"	1	0	0	0	1	0	0	0	0
Cranking Speed reached	1	0	1	0	1	1	0	0	1
Self-sustained speed reached	1	0	1	1	0	1	0	0	1
MASTER SWITCH set to "RUN"	0	1	1	1	0	1	0	1	0
MASTER SWITCH turned "OFF"	0	0	1	1	0	1	1	0	0
Engine Speed falls below N_2	0	0	1	0	0	0	1	0	0

TABLE 294

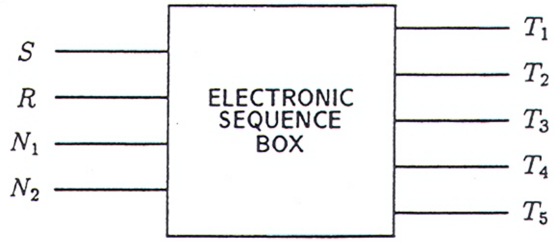
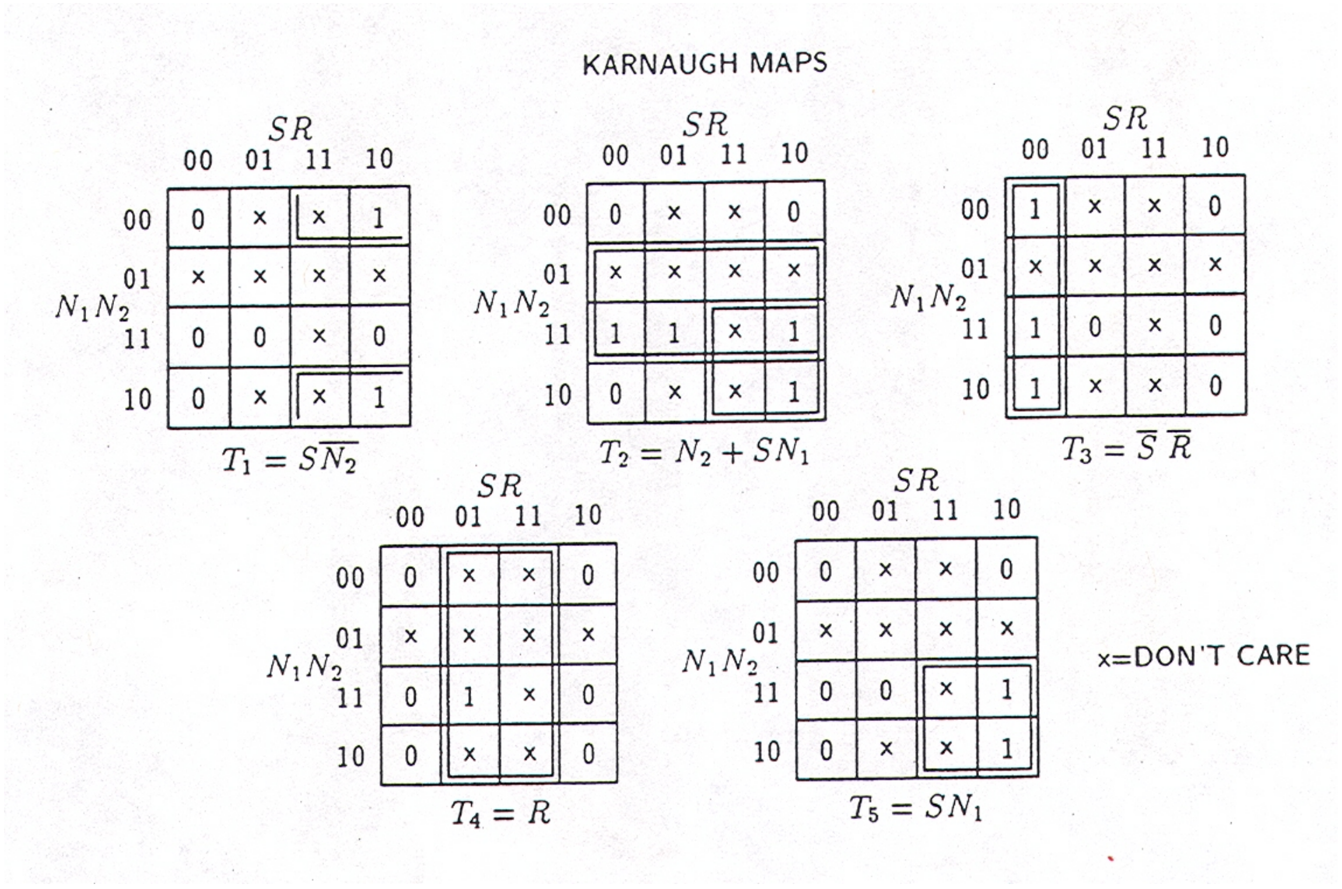


FIGURE 294

K maps for the sequencer



KARNAUGH MAPS

N_1N_2 \ SR	00	01	11	10
00	0	×	×	1
01	×	×	×	×
11	0	0	×	0
10	0	×	×	1

$T_1 = S\overline{N_2}$

N_1N_2 \ SR	00	01	11	10
00	0	×	×	0
01	×	×	×	×
11	1	1	×	1
10	0	×	×	1

$T_2 = N_2 + SN_1$

N_1N_2 \ SR	00	01	11	10
00	1	×	×	0
01	×	×	×	×
11	1	0	×	0
10	1	×	×	0

$T_3 = \overline{S}\,\overline{R}$

N_1N_2 \ SR	00	01	11	10
00	0	×	×	0
01	×	×	×	×
11	0	1	×	0
10	0	×	×	0

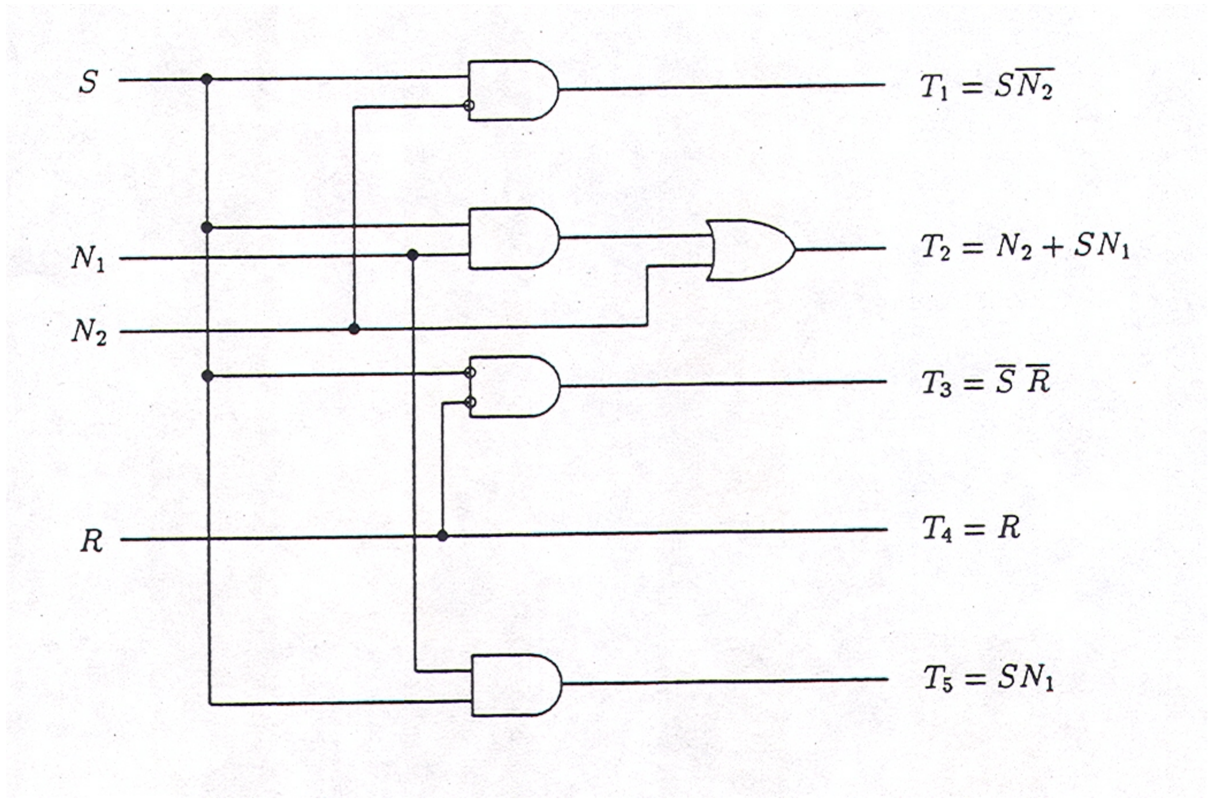
$T_4 = R$

N_1N_2 \ SR	00	01	11	10
00	0	×	×	0
01	×	×	×	×
11	0	0	×	1
10	0	×	×	1

$T_5 = SN_1$

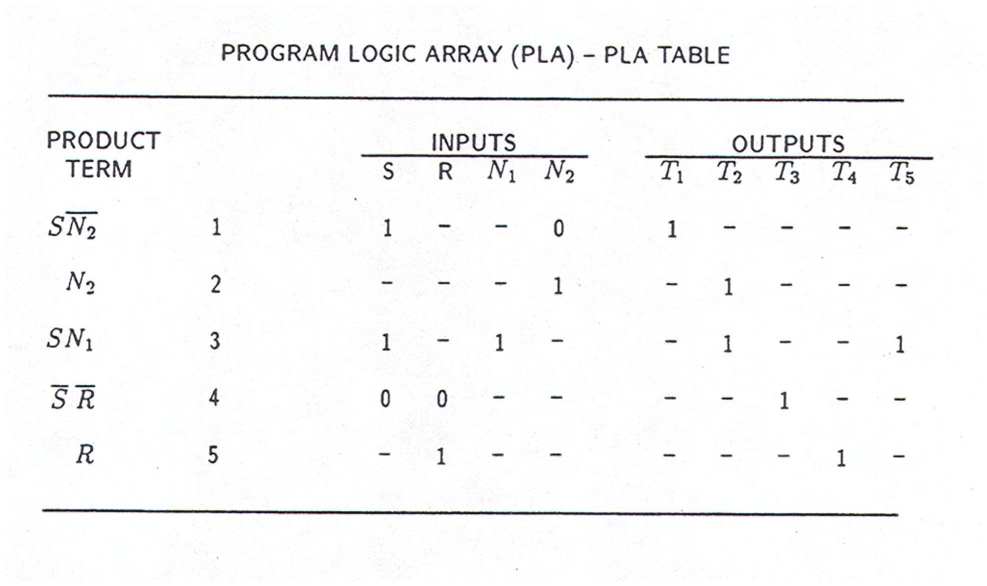
x=DON'T CARE

The logic implementation in gates



And a PLA implementation

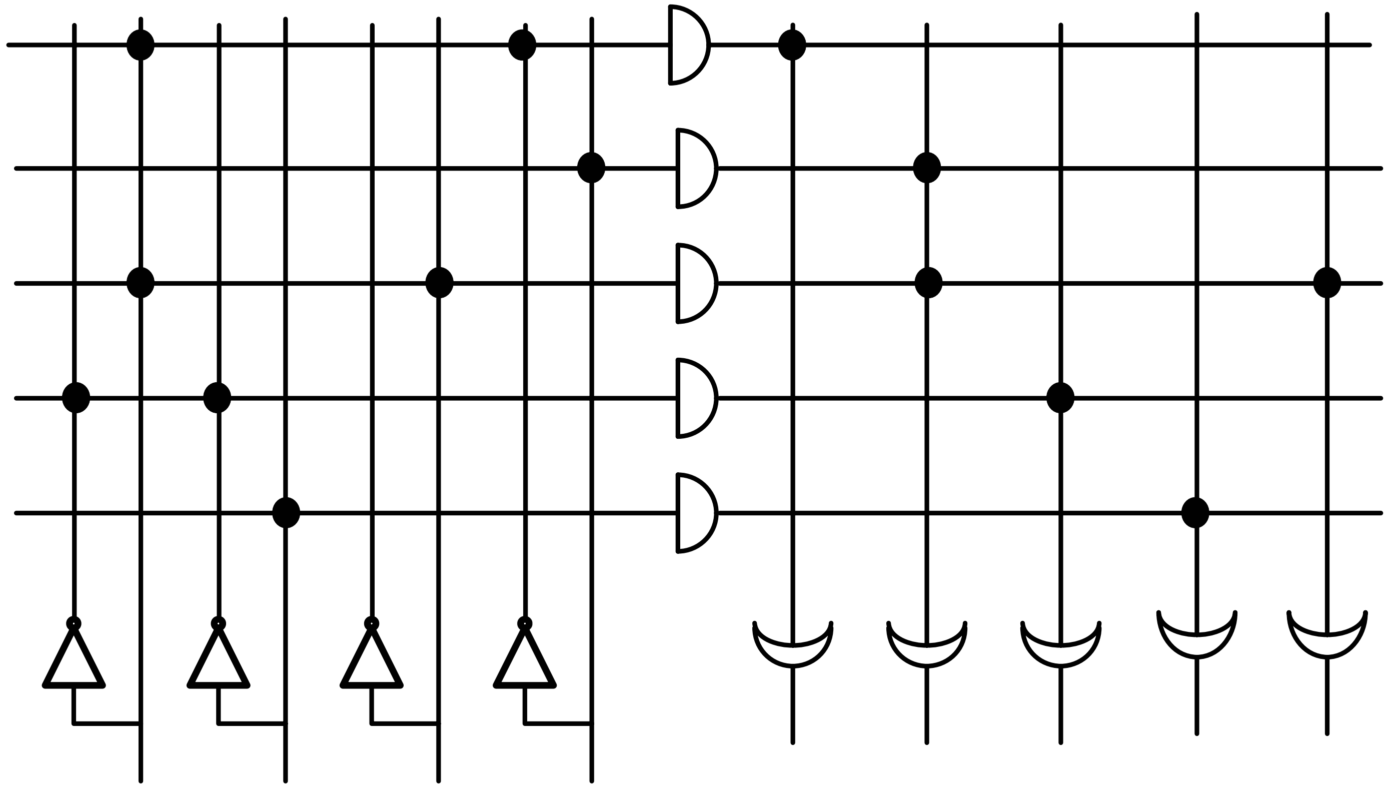
- You will most like be asked for this in a table representation form like this



PROGRAM LOGIC ARRAY (PLA) – PLA TABLE

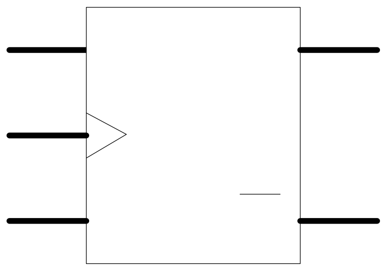
PRODUCT TERM		INPUTS				OUTPUTS				
		S	R	N_1	N_2	T_1	T_2	T_3	T_4	T_5
$S\overline{N_2}$	1	1	–	–	0	1	–	–	–	–
N_2	2	–	–	–	1	–	1	–	–	–
SN_1	3	1	–	1	–	–	1	–	–	1
$\overline{S}\,\overline{R}$	4	0	0	–	–	–	–	1	–	–
R	5	–	1	–	–	–	–	–	1	–

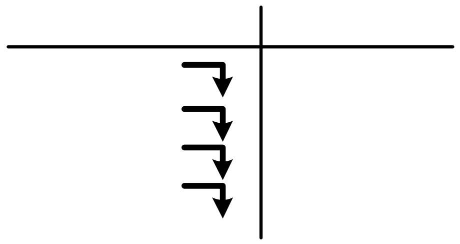
The PLA



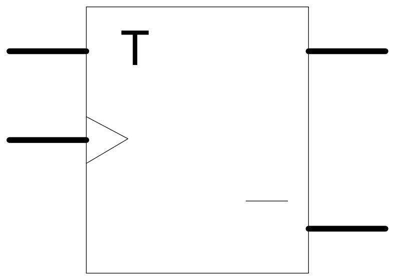
Basic Sequential Logic

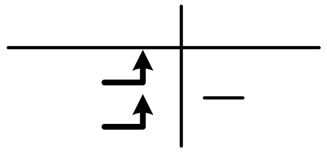
- Flip Flops
- Set Reset FF
  - Q*=S+R’Q





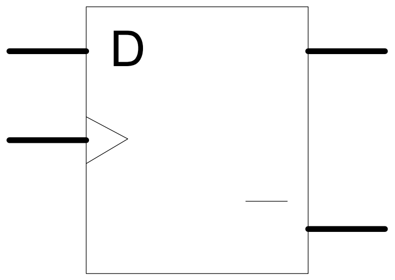
- Toggle FF
  - Q*=Q’
  - Q* is next value
  - Or next state

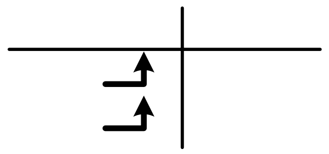




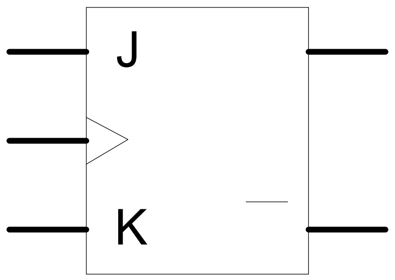
Basic Sequential Logic

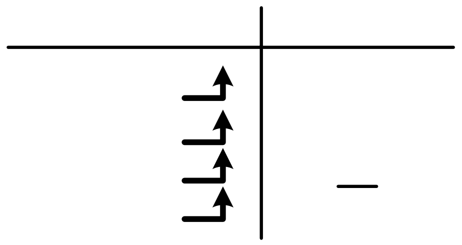
- D F/F
  - Q* = D





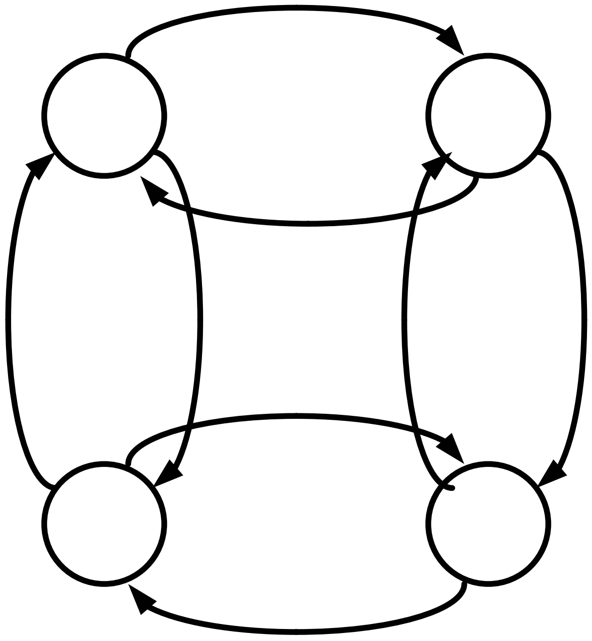
- JK F/F
  - Q*=JQ'+K'Q





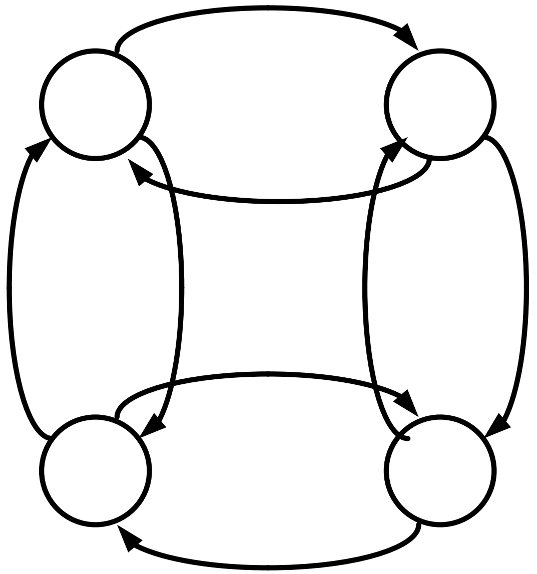
A Simple up/down counter

- Start with a state diagram



A Simple up/down counter

- Start with a state diagram
- And a state table for T F/Fs



K Maps for the toggle F/Fs

T1

x \ y1 y2	00	01	11	10
0	0	1	1	0
1	1	0	0	1

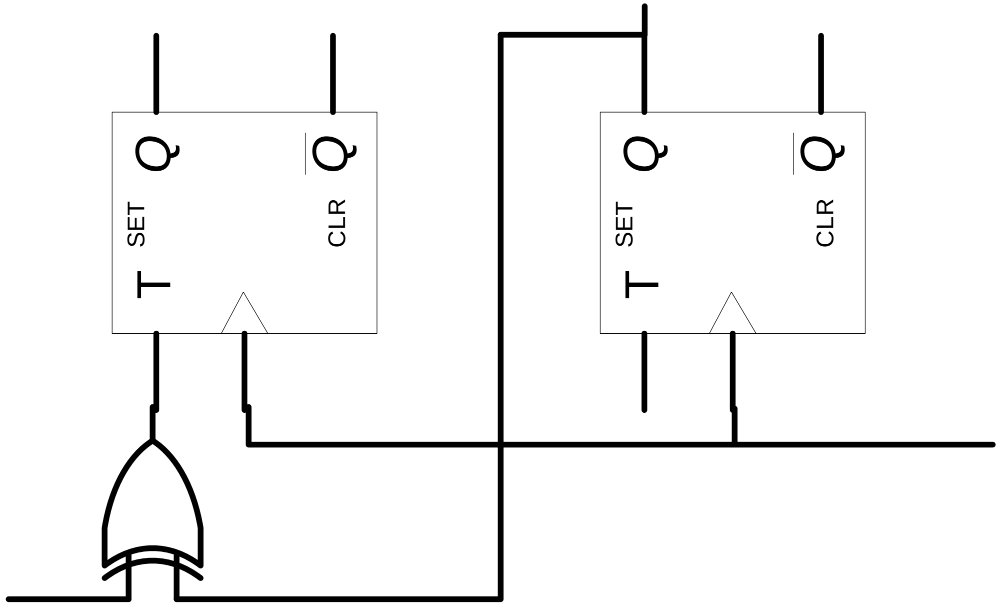
T2

x \ y1 y2	00	01	11	10
0	1	1	1	1
1	1	1	1	1

T1 = x'y2 + x y2'
= x xor y2

T2 = 1

Implementation


Q
Q
SET
CLR
T
Q
Q
SET
CLR
T