10 Programmable Logic Devices (PLD)

Programmable logic Devices (PLD)

Programmable Logic Devices (PLD)

Composed with the Arrays of THREE groups of

- Inverters gates;
- AND gates;
- OR gates.

Individual groups can be programmed to connect or disconnect at the junctions in the Arrays.

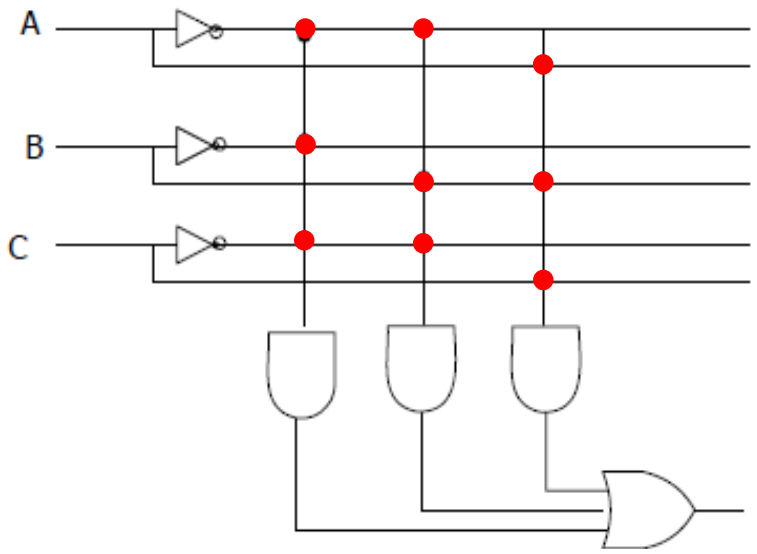
Programmable logic Devices (PLD)

Array connection may cause complicated schematic drawing,

Short-hand Notation:

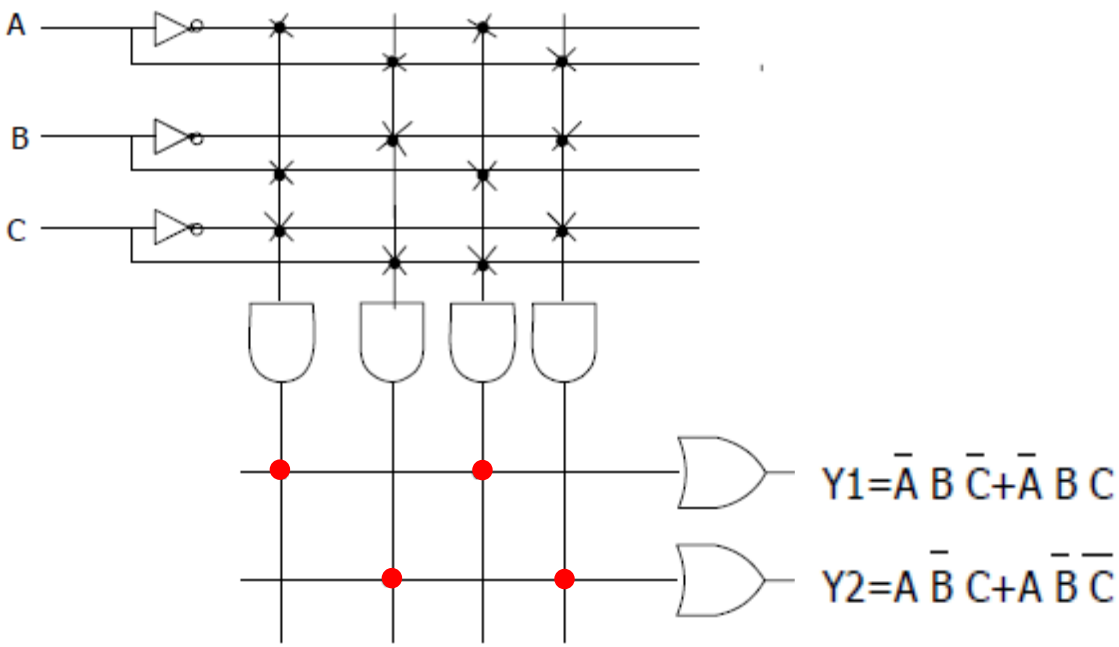
Draw the circuit connection as

- DOT for connection;
- No DOT for no connection.



$Y = A\,B\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,\bar{B}\,\bar{C}$

Programmable logic Devices (PLD)



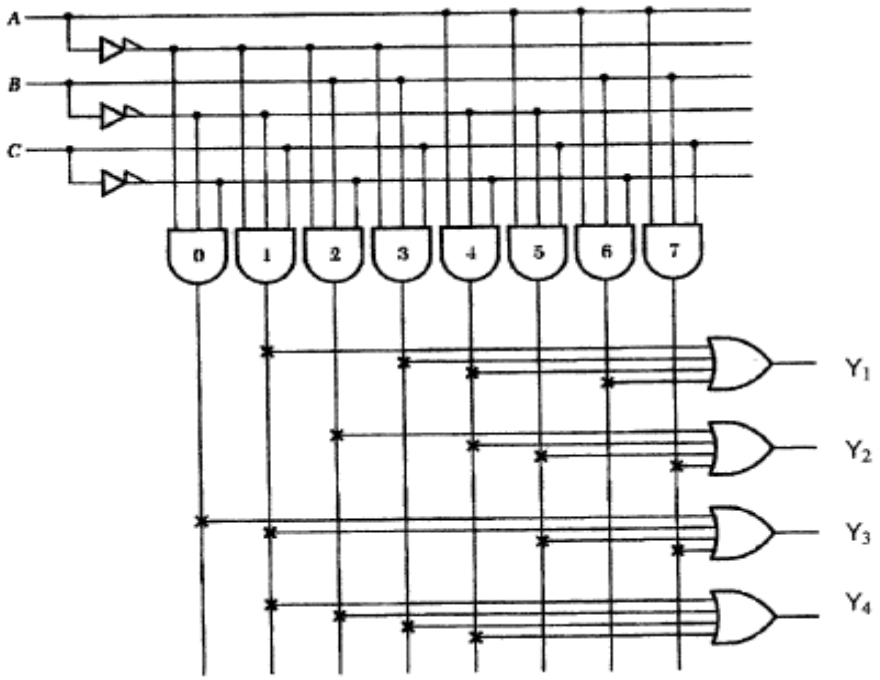
Short-Hand Notation for multiple outputs.

Programmable logic Devices (PLD)

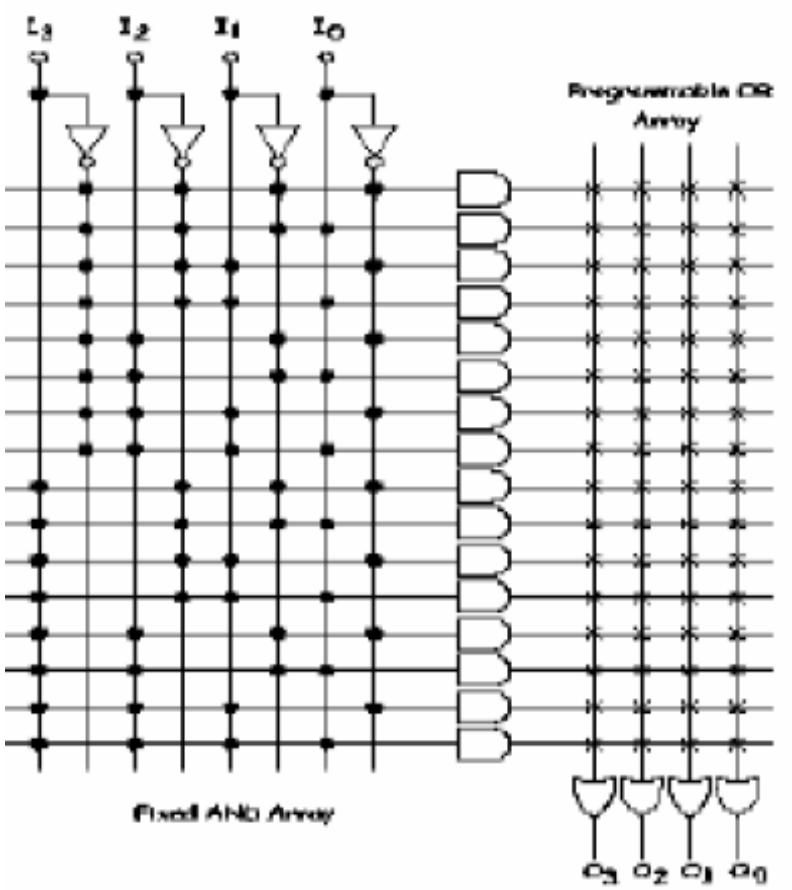
Device	AND array	OR array
PROM	Fixed	Programmable
PAL	Programmable	Fixed
PLA	Programmable	Programmable

Fixed Array is designed for all possible cases ➔ Large Structure.

PLD: Programmable ROM (PROM)

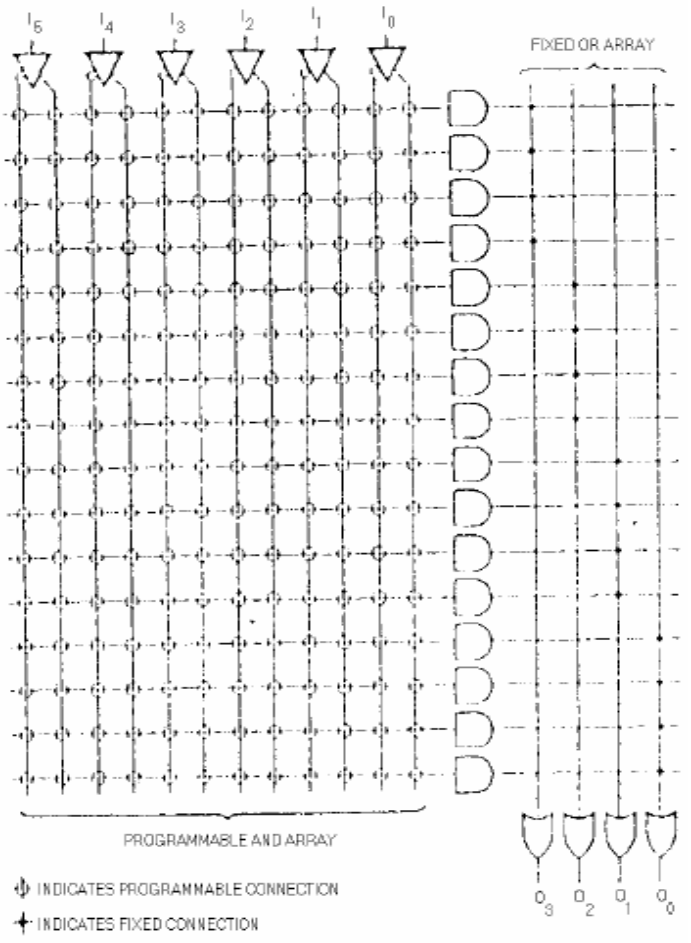


ROM

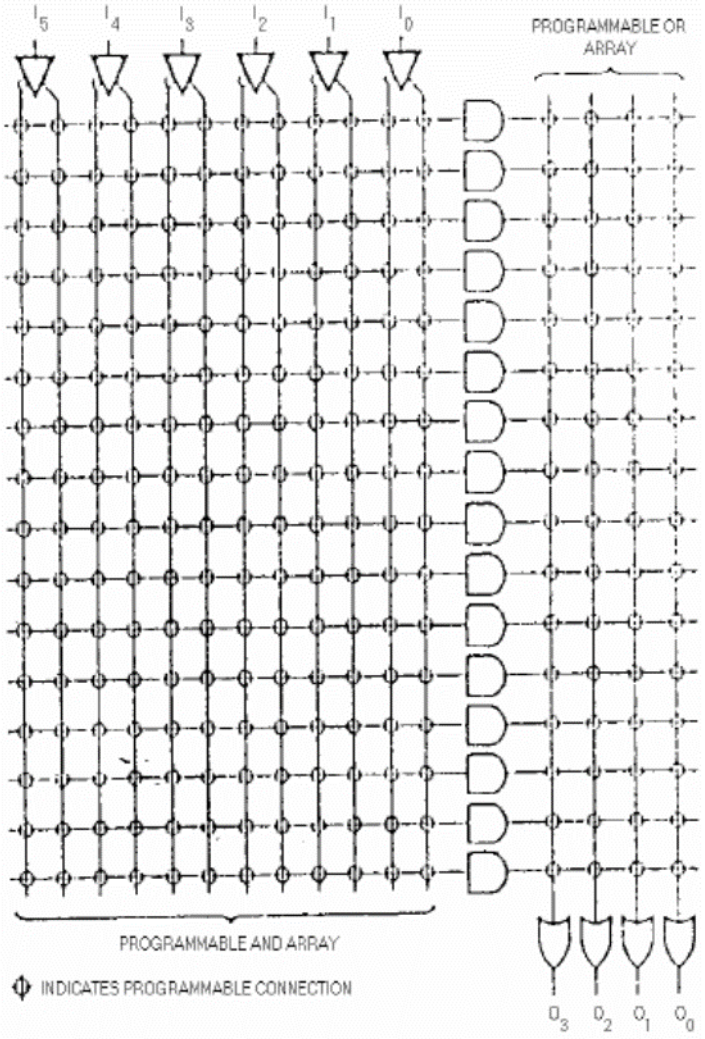


(a) Programmable ROM (Fixed AND Array)

PLD: Programmable Array Logic (PAL) and Programmable Logic Array (PLA)



(b) PAL (Fixed OR Array)



(c) PLA (AND-OR Programmable)

PLD Example: How to Use PAL

PAL Code: PAL*nXm*

- *n* = The Number of the inputs;
- *m* = The Number of the outputs;
- *X* = The Type of the outputs:

- L: Active low
- C: Complementary
- R: Register
- X: EX-OR
- V: variable output
- H: Active high
- P: Programmable Polarity
- RP: Register Programmable Polarity
- A: Register with calculations

Example: PAL16L8

The meaning of PAL codes:

- There are 16 inputs
- 8 outputs and
- L means that the output is active low.

PLD Example: Examples of PAL IC

Type No.*	No. of Inputs	No. of Outputs	Gate Configuration	No. of Inputs per OR GATE
10H8	10	8	AND-OR	2
12H6	12	6	AND-OR	4, 2, 2, 2, 2, 4
14H4	14	4	AND-OR	4
16H2	16	2	AND-OR	8
16C1	16	1	AND-OR/NOR	16
20C1	20	1	AND-OR/NOR	16
10L8	10	8	AND-NOR	2
12L6	12	6	AND-NOR	4, 2, 2, 2, 2, 4
14L4	14	4	AND-NOR	4
16L2	16	2	AND-NOR	8
12L10	12	10	AND-NOR	2
14L8	14	8	AND-NOR	4, 2, 2, 2, 2, 2, 2, 4
16L6	16	6	AND-NOR	4, 4, 2, 2, 4, 4
18L4	18	4	AND-NOR	4
20L2	20	2	AND-NOR	8
16L8	16	8	AND-NOR	8
20L8	20	8	AND-NOR	8
20L10	20	10	AND-NOR	4

PLD: PAL Example 1

Example: Design a logic circuit using PAL16L8 with Multiple outputs:

$F_1(A, B, C) = \Sigma m(1, 3, 4, 5, 6, 7)$; $F_2(A, B, C) = \Sigma m(0, 1, 4, 5, 6)$;

$F_3(A, B, C) = \Sigma m(1, 2, 5)$; $F_4(A, B, C) = \Sigma m(0, 1, 3, 7)$.

Solution:

Since the device's output is active LOW,

➔ So, we have to consider the invert of the outputs: $\overline{F_n}$,

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

F_1

$F_1 = A + C$

$\overline{F_1} = \overline{A}\,\overline{C}$

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

F_2

$F_2 = B + A\overline{C}$

$\overline{F_2} = \overline{A}\,B + B\,C$

PLD: PAL Example 1

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

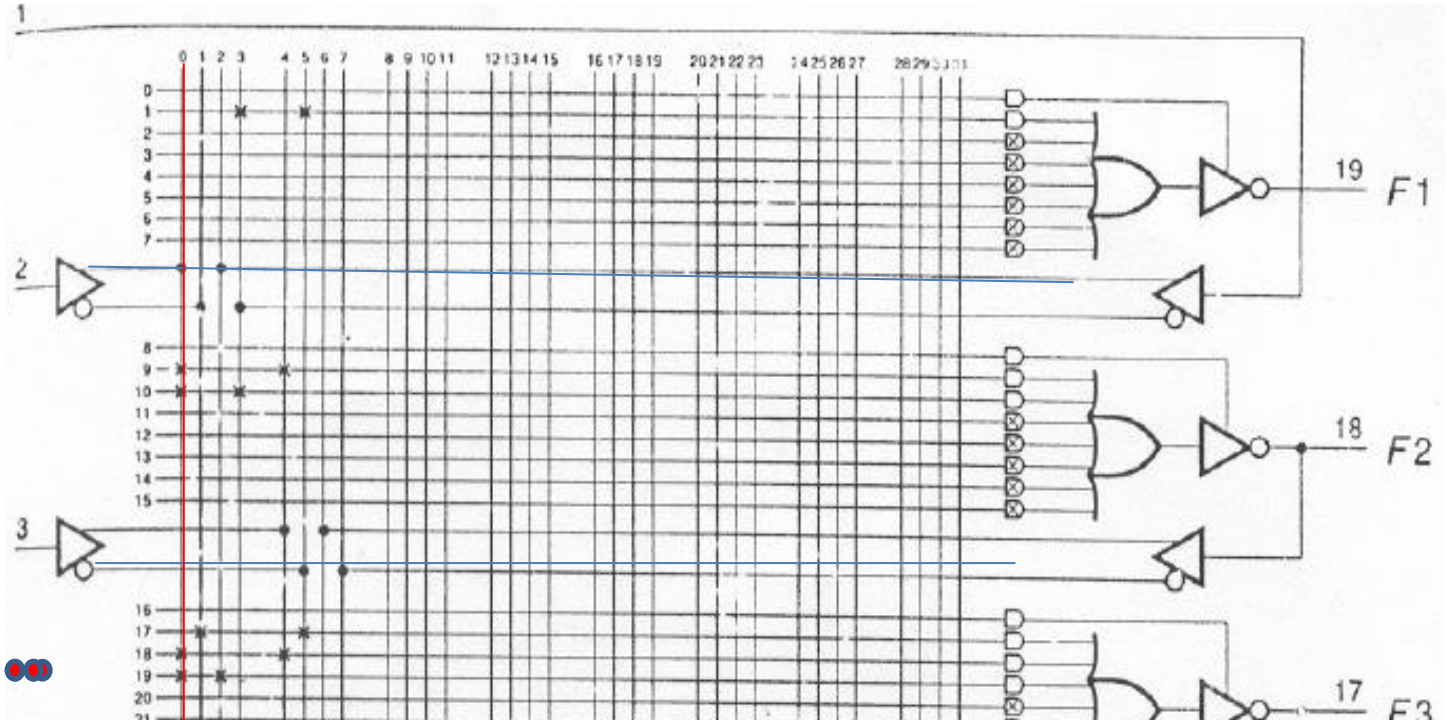
A \ BC	00	01	11	10
0	1	1	1	0
1	0	0	1	0

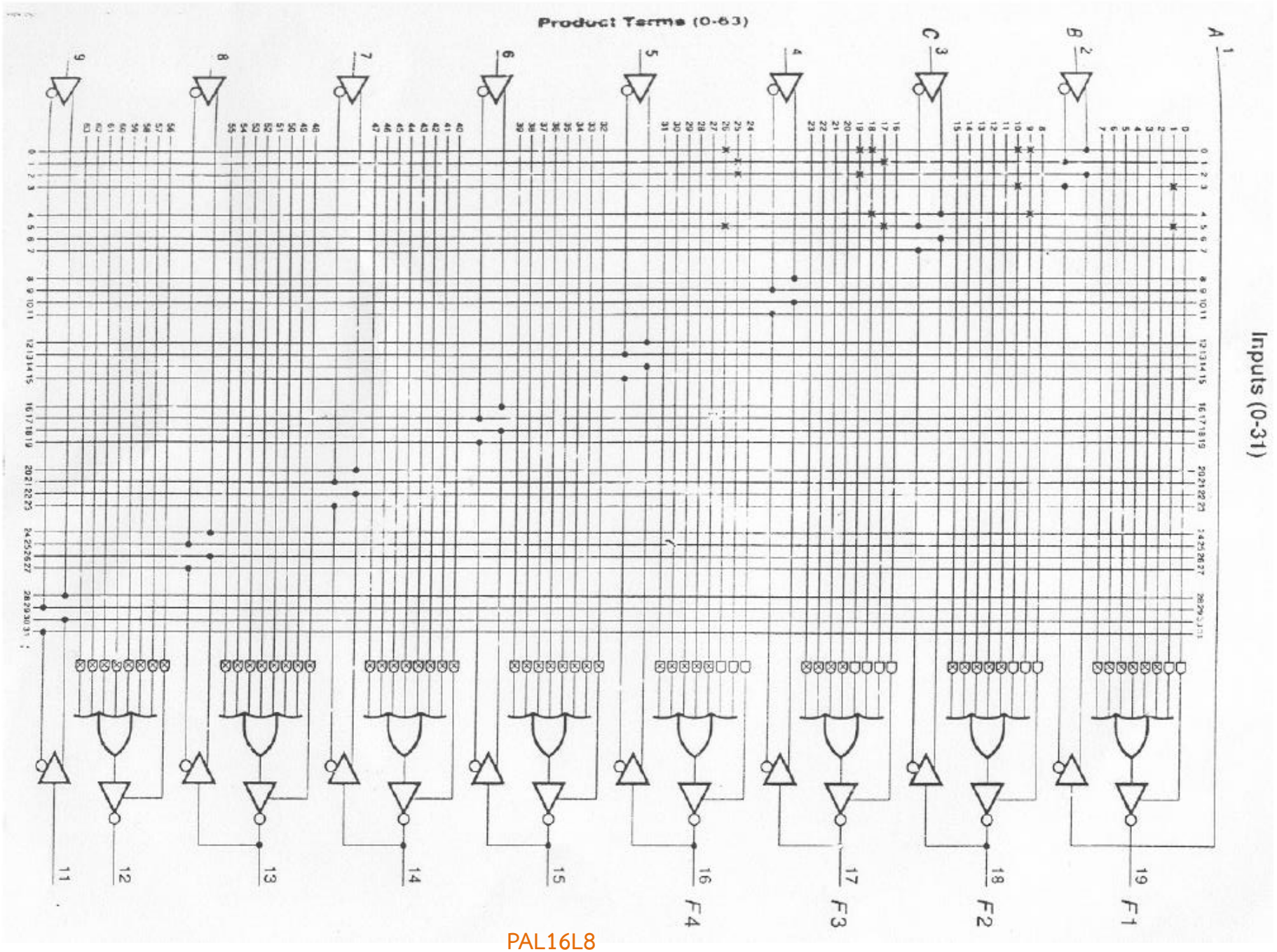
$F_3 = \overline{A}BC + \overline{B}\overline{C}$

$\overline{F}_3 = AB + BC + \overline{B}\overline{C}$

$F_4 = \overline{A}\overline{B} + BC$

$\overline{F}_4 = A\overline{B} + B\overline{C}$





PAL16L8

PLD: PAL Example 2

Example: Design a logic circuit using PAL16P8 with Multiple outputs:

$F_1(A, B, C, D) = \Sigma m(1,3,12,13,14,15)$;

$F_2(A, B, C, D) = \Sigma m(5, 10, 11, 12, 13, 15)$;

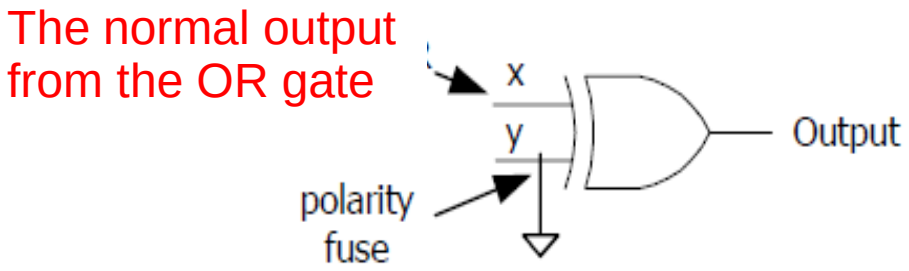
$F_3(A, B, C, D) = \Sigma m(0, 3, 4, 7, 8, 10, 11)$.

Solution:

- Since the device's output is Programmable Polarity,

➔ so, we have the choice to use either normal or invert of the outputs.

- The structure of the output with Programmable Polarity:



Using the EX-OR gate: Output = $x \oplus y$

- If $y = 1$ (Not connected to Ground): $x \oplus 1 = \overline{x}$ ➔ The invert of the output
- If $y = 0$ (connected to Ground): $x \oplus 0 = x$ ➔ The normal output

PLD: PAL Example 2

$F_1(A, B, C, D) = \Sigma m(1,3,12,13,14,15)$;

$F_1 = AB + \overline{A}\,\overline{B}\,D$ and $\overline{F}_1 = \overline{A}B + A\overline{B} + \overline{B}\,\overline{D}$

or $\overline{F}_1 = \overline{A}B + A\overline{B} + \overline{A}\overline{D}$

$F_2(A, B, C, D) = \Sigma m(5, 10, 11, 12, 13, 15)$;

$F_2 = AB\overline{C} + B\overline{C}D + A\overline{B}C + ACD$ and $\overline{F}_2 = \overline{A}\overline{D} + A\overline{C} + \overline{B}\overline{C} + BC\overline{D}$

or $F_2 = AB\overline{C} + B\overline{C}D + A\overline{B}C + ABD$

$F_3(A, B, C, D) = \Sigma m(0, 3, 4, 7, 8, 10, 11)$.

$F_3 = \overline{A}\,\overline{C}\,\overline{D} + A\overline{B}C + \overline{A}CD + \overline{B}\,\overline{C}\,\overline{D}$ (or $A\overline{B}\overline{D}$ or $\overline{B}CD$)

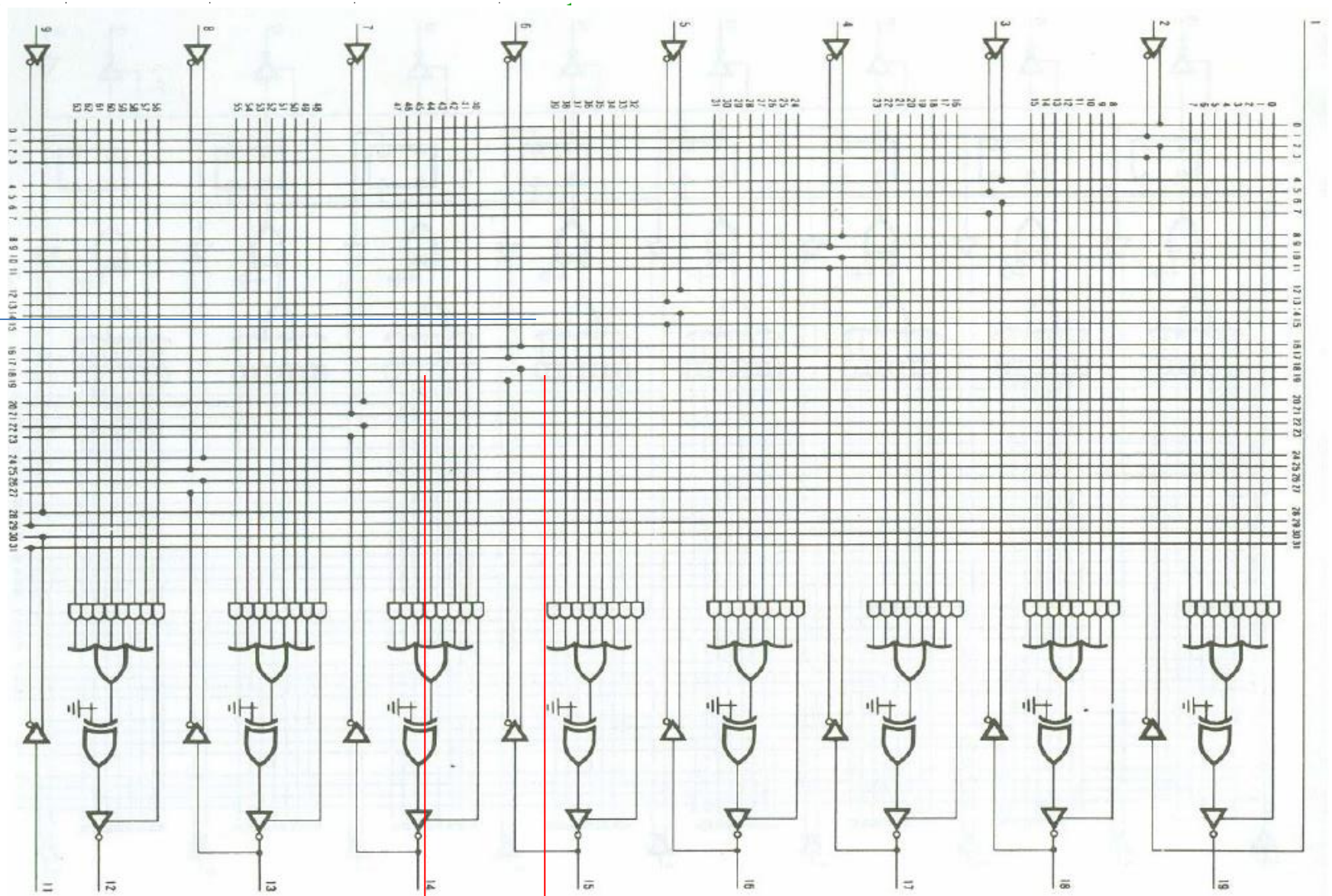
and $\overline{F}_3 = AB + \overline{C}D + \overline{A}C\overline{D}$

For the easy connections, we could choose F_1 , $\overline{F}_2$ and $\overline{F}_3$
By setting y = 0, 1, 1 for F_1 , F_2 and F_3, respectively.

$F_2 = \overline{A}\overline{D} + A\overline{C} + \overline{B}\overline{C} + BC\overline{D}$

$F_1 = AB + \overline{A}\,\overline{B}\,D$

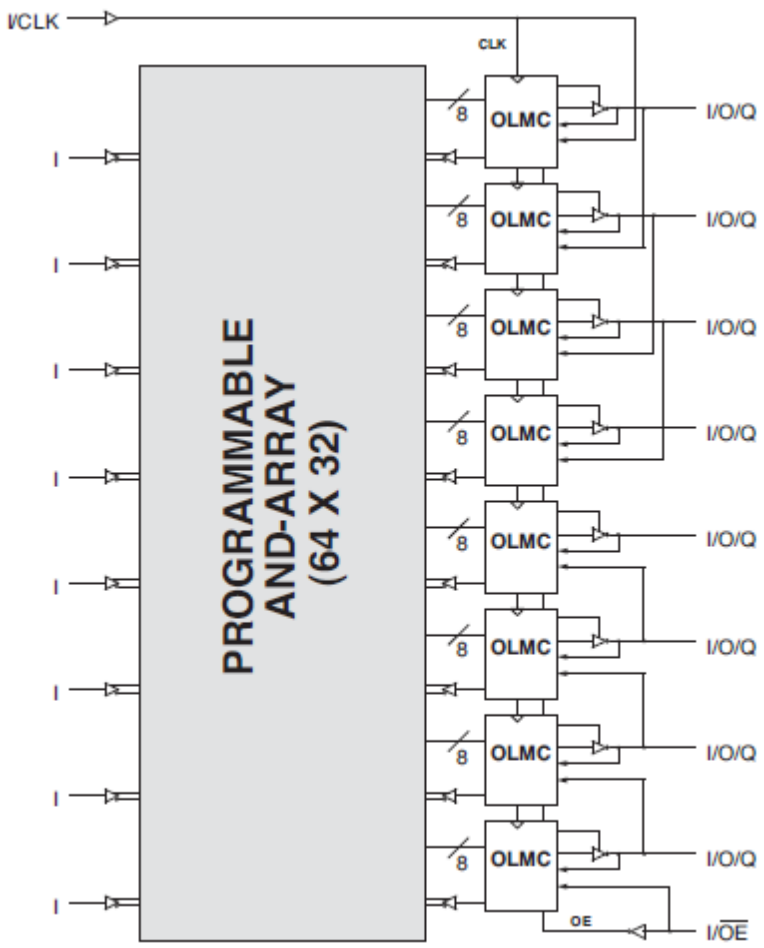
$F_3 = AB + CD + \overline{A}CD$



Generic Array Logic (GAL)

Generic Array Logic (GAL) is the improvement of PAL with

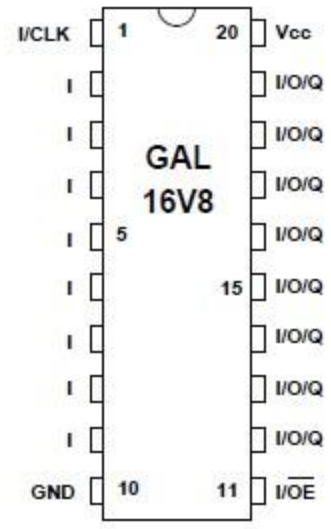
- Low current – power dissipation
- Faster Response and
- Modifying the output section named
- Output Logic Macro Cell (OLMC):
  - Buffer
  - Flipflop
  - Register



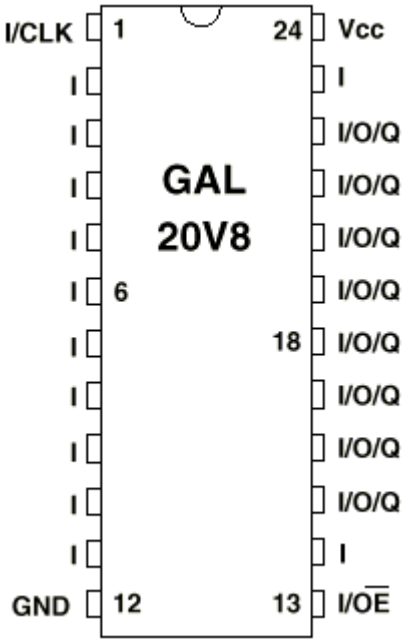
Therefore, PAL is replaced with GAL;

Similar IC structure and types:

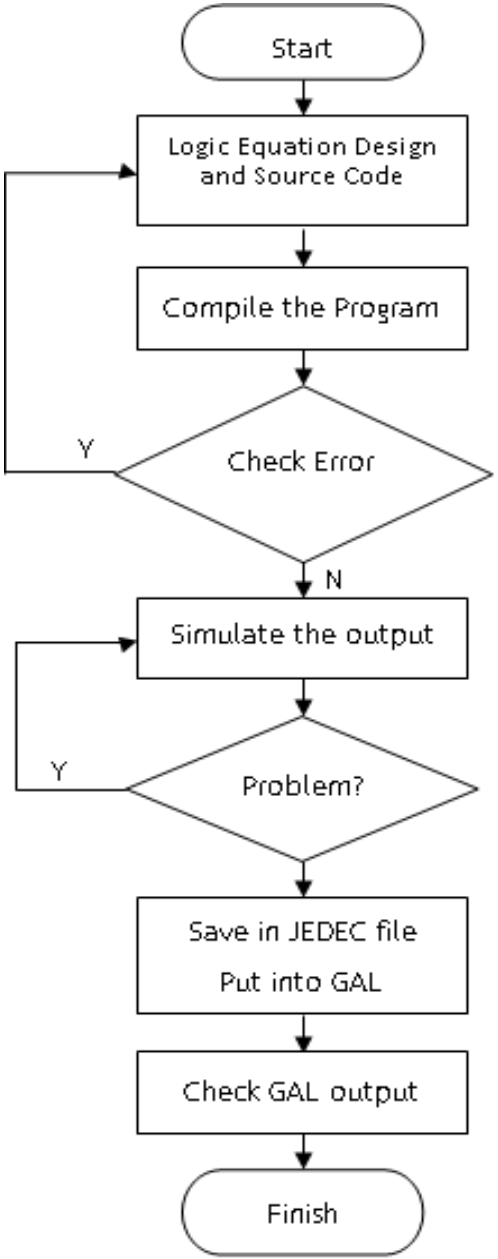
Generic Array Logic (GAL): PAL IC Replacement



GAL16V8		GAL20V8	
PAL16L8	PAL14L4	PAL20L8	PAL18L4
18H8	16L2	20H8	20L2
16R8	10H8	20R8	14H8
16R6	12H6	20R6	16H6
16R4	14H4	20R4	18H4
16P8	16H2	20P8	20H2
16RP8	10P8	20RP8	14P8
16RP6	12P6	20RP6	16P6
16RP4	14P4	20RP4	18P4
10L8	16P2	14L8	20P2
12L6		16L6	



PLD Programming with Design Tools



Array Connections

Signal Check

IC Check

PLD Programming with Design Tools

Software Tools

Currently, there are many types of design software available.

- ❑ Machine level called Assemblers
  - ○ PALASM: not very effective,
- ❑ High level software called a compiler
  - ○ CUPL (Universal Compiler for Programmable Logic Devices),
  - ○ ABEL (Advance Boolean Expression Language) or
  - ○ ELWG-GDS (GAL Design Software)
  - ○ etc.

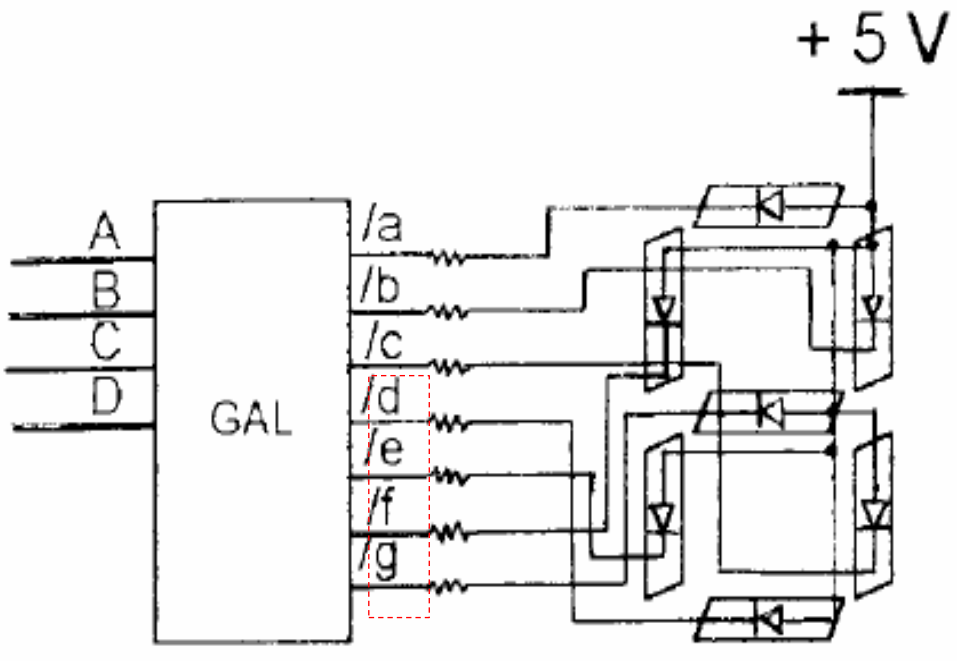
Boolean function for different Softwares

Logic Operator	Boolean	ABEL	CUPL	ELW-GDS	PALASM
AND	A•B	A&B	A•B	A*B	A*B
OR	A+B	A#B	A#B	A+B	A+B
NOT	$\overline{A}$	!A	!A	/A	/A
EX-OR	$A \oplus B$	A $ B	A $ B	/A*B+A*/B	/A*B+A*/B

GAL Programming: CUPL – gates invention

```
Name Gates;
Partno CA0001;
Assembly None;
Device G16V8;
Format J;
/*****************************************************************/
/* This is a example to demonstrate how CUPL */
/* compiles simple gates. */
/*****************************************************************/
/* Inputs: define inputs to build simple gates */
Pin 1 = a;
Pin 2 = b;
/* Outputs: define outputs as active HI levels */
Pin 12 = inva;
Pin 13 = invb;
Pin 14 = and;
Pin 15 = nand;
Pin 16 = or;
Pin 17 = nor;
Pin 18 = xor;
Pin 19 = xnor;
/* Logic: examples of simple gates expressed in CUPL */
inva = !a; /* inverters */
invb = !b;
and = a & b; /* and gate */
nand = !(a & b); /* nand gate */
or = a # b; /* or gate */
nor = !(a # b); /* nor gate */
xor = a $ b; /* exclusive or gate */
xnor = !(a $ b); /* exclusive nor gate */
```

GAL Programming: Hex To Seven-Segment Decoder



GAL Programming: Hex To Seven-Segment Decoder

N	Inputs				Outputs						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	1	0	1	0	1	1	1	0	1	1	1
B	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
D	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

GAL Programming: Hex To Seven-Segment Decoder

```
Name HexLed;
PartNo xxxxxx;
Assembly None;
Device G16V8;
Format J;
/*************************************************************/
/* */
/* Figure 1 : Hex to 7 Segment Decoder */
/* */
/** Inputs **/
Pin 2 = A;
Pin 3 = B;
Pin 4 = C;
Pin 5 = D;
/** Outputs **/
Pin 12 = a;
Pin 13 = b;
Pin 14 = c;
Pin 15 = d;
Pin 16 = e;
Pin 17 = f;
Pin 18 = g;
/** Declarations and Intermediate Variable Definitions **/
FIELD input = [D, C, B, A];
FIELD output = [a, b, c, d, e, f, g];
/** Logic Equations **/
TABLE input => output {
0 => 7E; 1 => 30; 2 => 6D; 3 => 79;
4 => 33; 5 => 5B; 6 => 5F; 7 => 70;
8 => 7F; 9 => 73; A => 77; B => 1F;
C => 4E; D => 3D; E => 4F; F => 47;
}
```