Notes on
Programmable Logic Devices (PLDs)

==========

OBJECTIVES:

In this lesson you will be introduced to some types of Programmable Logic Devices (PLDs):

- PROM, PAL, PLA, CPLDs, FPGAs.
- How to implement digital circuits using these PLDs.

CONTENTS:

1. INTRODUCTION:

An IC that contains large numbers of gates, flip-flops, etc. that can be configured by the user to perform different functions is called a Programmable Logic Device (PLD). It permits elaborate digital logic designs to be implemented by the user on a single device. The internal logic gates and/or connections of PLDs can be changed/configured by a programming process.

Comparison: programmable logic Vs fixed logic

The fixed logic system has circuits whose configurations are permanent. Their instructions perform only a fixed set of operations repeatedly. Once manufactured and programmed, the logic cannot be changed. This system is a fantastic asset for repeated tasks.

But one tiny mistake in the manufacturing process like uploading the wrong code in the device, and the entire system is discarded, and a new design is developed. That's quite some risk that companies aren't willing to take unless necessary. Additionally, fixed logic does not allow the users to expand or build on their existing functionalities.

Thus, we need something more flexible, easy to work, and more cost-efficient. Thus, programmable logic comes to the rescue. It is easy-to-program, affordable and equipped with better features. Inexpensive software is used to develop, code and test the required design. This design is then programmed into a device and tested in a live electronic circuit.

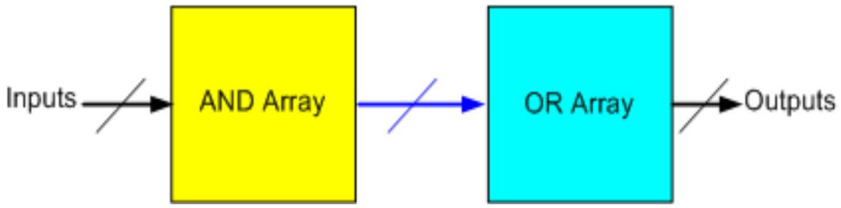
The corresponding performance then decides if the logic needs to be altered, or if the prototype is fit to be determined as the final design itself. The fixed logic system thus offers limited usability; a programmable logic seems more feasible and beneficial.

Implementing Boolean functions:

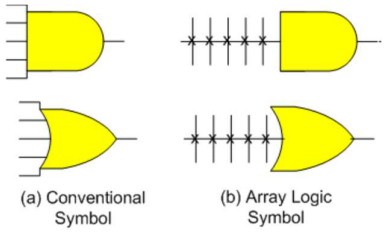
Every Boolean logic can be decomposed into product-of-sum (POS) or sum-of-product by Karnaugh map(k-map),

$$S = A \oplus B \oplus C = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$$
$$= (A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + C)(\overline{A} + \overline{B} + \overline{C})$$

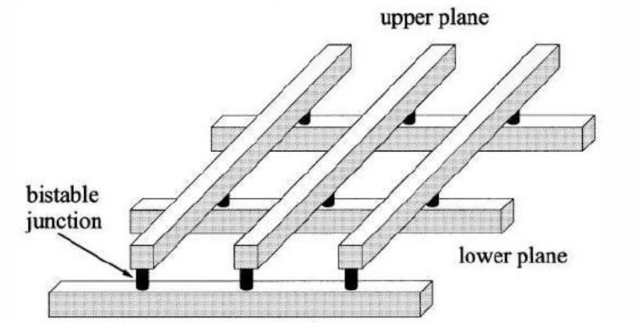
PLDs are typically built with an array of AND gates (AND-array) and an array of OR gates (OR-array) to implement the sum-of-products as shown in figure.



In order to show the internal logic diagram for such technologies in a concise form, it is necessary to have special symbols for array logic. Figure shows the conventional and array logic symbols for a multiple input AND and a multiple input OR gate.



One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact. Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function. Anti-fuse employs a thin barrier of non-conducting amorphous silicon between two metal conductors. Usually in mesh structure. When a sufficiently high voltage is applied across the amorphous silicon it is turned into a polycrystalline silicon-metal alloy with a low resistance, which is conductive



Problems of using standard ICs*:* Problems of using standard ICs in logic design are that they require hundreds or thousands of these ICs, considerable amount of circuit board space, a great deal of time and cost in inserting, soldering, and testing. Also require keeping a significant inventory of ICs.

Advantages of using PLDs*:* Advantages of using PLDs are less board space, faster, lower power requirements (i.e., smaller power supplies), less costly assembly processes, higher reliability (fewer ICs and circuit connections means easier troubleshooting), and availability of design software.

Types of PLDs:

PLDs are broadly classified into simple and complex programmable logic devices

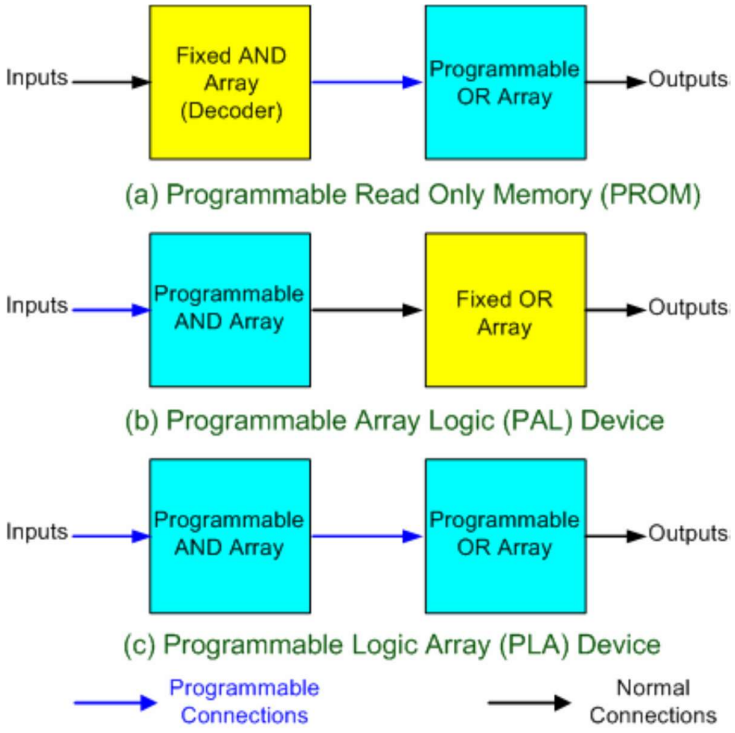
Further, this is grouped as,

- SPLDs (Simple Programmable Logic Devices)
    - ROM (Read-Only Memory)
    - PLA (Programmable Logic Array)
    - PAL (Programmable Array Logic)
    - GAL (Generic Array Logic)

- HCPLD (High Capacity Programmable Logic Device)
    - CPLD (Complex Programmable Logic Device)
    - FPGA (Field-Programmable Gate Array)

Programmable Connections in PLDs:

The programmable connections of AND-OR arrays for different types of PLDs are described here. Figure shows the locations of the programmable connections for the three types.



The PROM (Programmable Read Only Memory) has a fixed AND array (constructed as a decoder) and programmable connections for the output OR gates array. The PROM implements Boolean functions in sum-of-minterms form. The PAL (Programmable Array Logic) device has a programmable AND array and fixed connections for the OR array. The PLA (Programmable Logic Array) has programmable connections for both AND and OR arrays. So it is the most flexible type of PLD.

Applications of Programmable Logic Devices:

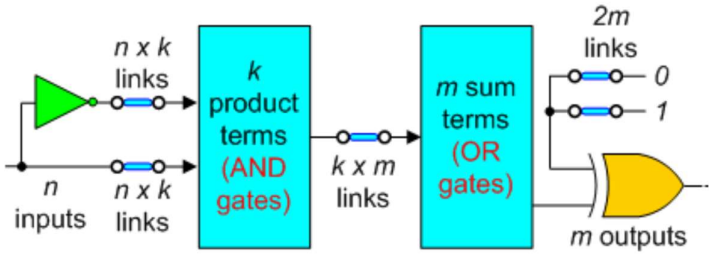
- Glue Logic
- State Machines
- Counters

- Synchronization
- Decoders
- Bus Interfaces
- Parallel-to-Serial
- Serial-to-Parallel

2. PROGRAMMABLE LOGIC ARRAY (PLA):

In PLAs, instead of using a decoder as in PROMs, a number (k) of AND gates is used where $k < 2^n$, (n is the number of inputs). Each of the AND gates can be programmed to generate a product term of the input variables and does not generate all the minterms as in the ROM. The AND and OR gates inside the PLA are initially fabricated with the links (fuses) among them. The specific Boolean functions are implemented in sum of products form by opening appropriate links and leaving the desired connections.

A block diagram of the PLA is shown in the figure. It consists of n inputs, m outputs, and k product terms. The product terms constitute a group of k AND gates each of 2n inputs. Links are inserted between all n inputs and their complement values to each of the AND gates. Links are also provided between the outputs of the AND gates and the inputs of the OR gates.



Since PLA has m-outputs, the number of OR gates is m. The output of each OR gate goes to an XOR gate, where the other input has two sets of links, one connected to logic 0 and other to logic 1. It allows the output function to be generated either in the true form or in the complement form. The output is inverted when the XOR input is connected to 1 (since $X \oplus 1 = X'$). The output does not change when the XOR input is connected to 0 (since $X \oplus 0 = X$). Thus, the total number of programmable links is 2n x k + k x m + 2m.

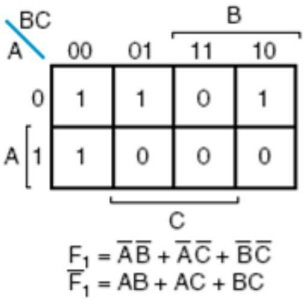
The size of the PLA is specified by the number of inputs (n), the number of product terms (k), and the number of outputs (m), (the number of sum terms is equal to the number of outputs).

Example 1:

Implement the combinational circuit having the shown truth table, using PLA.

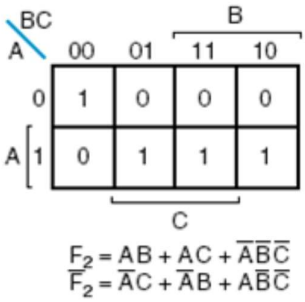
A	*B*	*C*	F_1	F_2
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Each product term in the expression requires an AND gate. To minimize the cost, it is necessary to simplify the function to a minimum number of product terms.



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

$\overline{F}_1 = AB + AC + BC$



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

$\overline{F}_2 = \overline{A}\,\overline{C} + \overline{A}\,B + A\,\overline{B}\,\overline{C}$

Designing using a PLA, a careful investigation must be taken in order to reduce the distinct product terms. Both the true and complement forms of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

The combination that gives a minimum number of product terms is,

F_1' = AB + AC + BC or F_1 = (AB + AC + BC)'

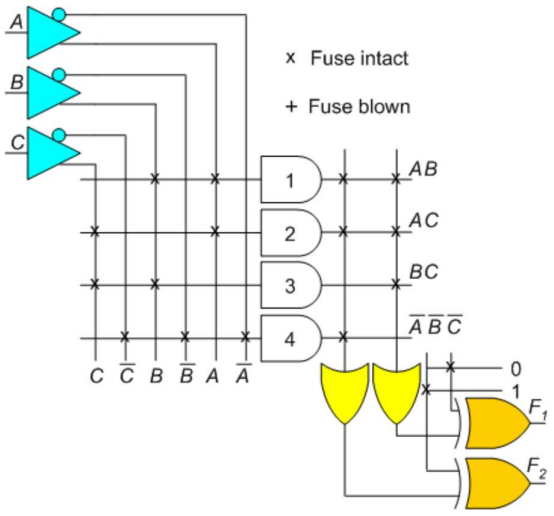
F_2 = AB + AC + A'B'C'

This gives only 4 distinct product terms: AB, AC, BC, and A'B'C'. So the PLA table will be as follows,

PLA programming table

	Product term	Inputs A B C	Outputs (C) F_1	Outputs (T) F_2
AB	1	1 1 –	1	1
AC	2	1 – 1	1	1
BC	3	– 1 1	1	–
$\overline{A}\overline{B}\overline{C}$	4	0 0 0	–	1

For each product term, the inputs are marked with 1, 0, or – (dash). If a variable in the product term appears in its normal form (unprimed), the corresponding input variable is marked with a 1. A *1* in the Inputs column specifies a path from the corresponding input to the input of the AND gate that forms the product term. A *0* in the Inputs column specifies a path from the corresponding complemented input to the input of the AND gate. A dash specifies no connection.



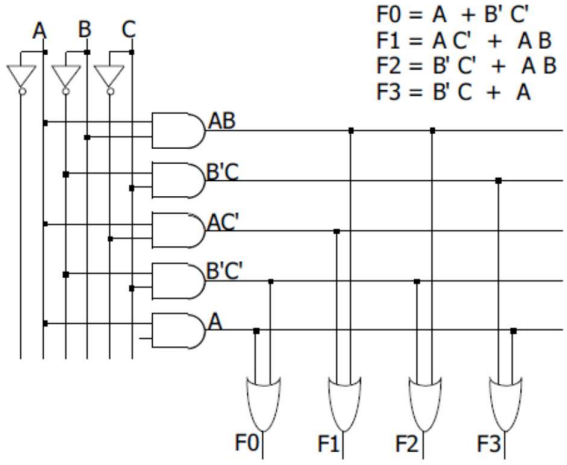
The appropriate fuses are blown and the ones left intact form the desired paths. It is assumed that the open terminals in the AND gate behave like a 1 input.

In the Outputs column, a T (true) specifies that the other input of the corresponding XOR gate can be connected to 0, and a C (complement) specifies a connection to 1.

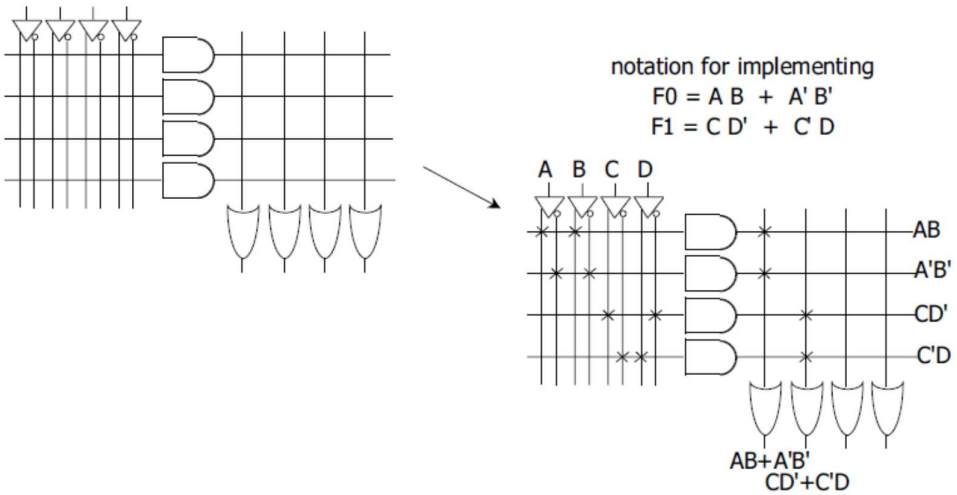
Note that output F_1 is the normal (or true) output even though a C (for complement) is marked over it. This is because F_1' is generated with AND-OR circuit prior to the output XOR. The output XOR complements the function F_1' to produce the true F_1 output as its second input is connected to logic 1.

Example 2:

All possible connections are available ***before programming*** as follows,



Unwanted connections are blown in the fuse (normally connected, break the unwanted ones) and in the anti-fuse (normally disconnected, make the wanted ones) ***after programming*** **for the given example** as follows,



Limitations of PLAs

PLAs come in various sizes. Typical size is 16 inputs, 32 product terms, 8 outputs

- Each AND gate has large fan-in. This limits the number of inputs that can be provided in a PLA
- 16 inputs forms 2^{16}, possible input combinations; only 32 permitted (since 32 AND gates) in a typical PLA
- 32 AND terms permitted large fan-in for OR gates as well
    - This makes PLAs slower and slightly more expensive than some alternatives to be discussed shortly
- 8 outputs could have shared min-terms, but not required

Applications of PLA:

- PLA is used to provide control over datapath.
- PLA is used as a counter.
- PLA is used as a decoders.
- PLA is used as a BUS interface in programmed I/O.

3. PROGRAMMABLE READ ONLY MEMORY (PROM):

Read Only Memory (ROM) is a memory device, which stores the binary information permanently. If the ROM has programmable feature, then it is called as Programmable ROM PROM. The user has the flexibility to program the binary information electrically once by using PROM programmer. The input lines to the AND array are hard-wired and the output lines to the OR array are programmable. Thus, we generate 2^n product terms using 2^n AND gates having n inputs each, using n x 2^n decoder. This decoder generates 'n' min-terms. Each AND gate generates one of the possible AND products (i.e., min-terms).

Given a 2^k x n ROM, we can implement ANY combinational circuit with at most *k* inputs and at most n outputs. Because,

- k-to-2^k decoder will generate all 2^k possible min-terms
- Each of the OR gates must implement a ∑m()
- Each ∑m() can be programmed

The procedure for implementing a ROM-based circuit is as follows for the given example,

f(a,b,c) = a'b' + abc

g(a,b,c) = a'b'c' + ab + bc

h(a,b,c) = a'b' + c

and its solution can be obtained as,

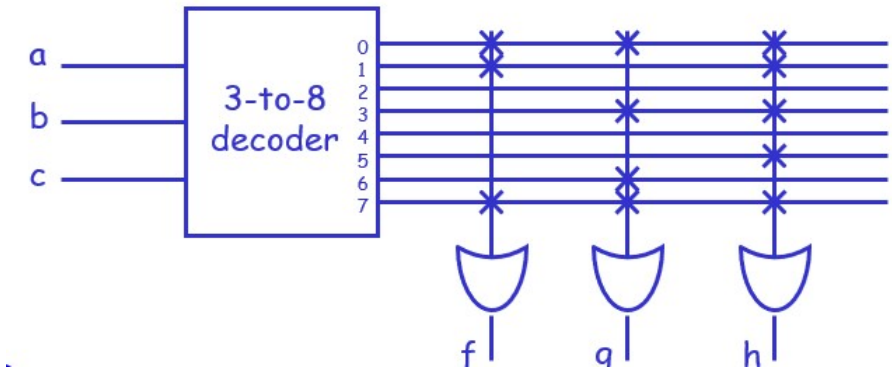
Express f(), g(), and h() in ∑m() format (use truth tables)

Program the ROM based on the 3 ∑m()'s

Example:

There are 3 inputs and 3 outputs, thus we need a 8x3 ROM block.

- f = ∑m(0, 1, 7)
- g = ∑m(0, 3, 6, 7)
- h = ∑m(0, 1, 3, 5, 7)



Another practical application of PROM device is BCD to 7 Segment Display Controller and the corresponding input and output relationship are shown in the following table.

A B C D	C0	C1	C2	C3	C4	C5	C6
0 0 0 0	1	1	1	1	1	1	0
0 0 0 1	0	1	1	0	0	0	0
0 0 1 0	1	1	0	1	1	0	1
0 0 1 1	1	1	1	1	0	0	1
0 1 0 0	0	1	1	0	0	1	1
0 1 0 1	1	0	1	1	0	1	1
0 1 1 0	1	0	1	1	1	1	1
0 1 1 1	1	1	1	0	0	0	0
1 0 0 0	1	1	1	1	1	1	1
1 0 0 1	1	1	1	0	0	1	1
1 0 1 0	X	X	X	X	X	X	X
1 0 1 1	X	X	X	X	X	X	X
1 1 0 0	X	X	X	X	X	X	X
1 1 0 1	X	X	X	X	X	X	X
1 1 1 0	X	X	X	X	X	X	X
0 1 1 1	X	X	X	X	X	X	X

Comparison: ROM Vs PLA

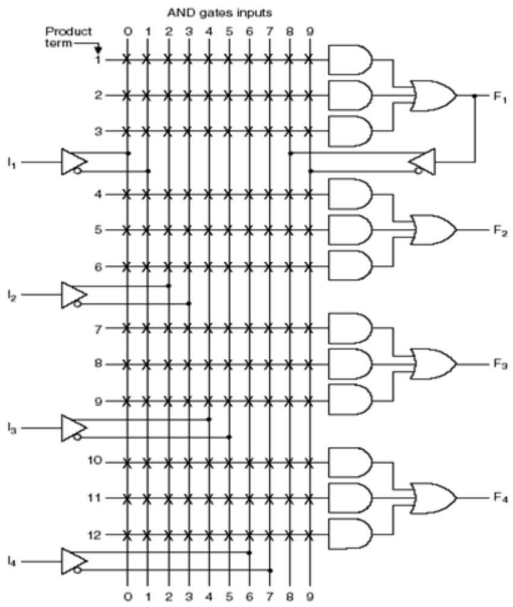
- ROM approach advantageous when
  - design time is short (no need to minimize output functions)
  - most input combinations are needed (e.g., code converters)
  - little sharing of product terms among output functions
- ROM problems
  - size doubles for each additional input (32x4 for Calendar example)
  - can't exploit don't cares
- PLA approach advantageous when
  - design tools are available for multi-output minimization
  - there are relatively few unique minterm combinations
  - many minterms are shared among the output functions
  - Supports multilevel implementation using feedback
- PAL problems
  - constrained fan-ins on OR plane
  - Difficulty of common term re-use??

4. PROGRAMMABLE ARRAY LOGIC (PAL):

PAL is a programmable logic device that has Programmable AND array & fixed OR array. The advantage of PAL is that we can generate only the required product terms of Boolean function instead of generating all the min terms by using programmable AND gates. As only AND gates are programmable, the PAL device is easier to program but it is not as flexible as the PLA. Here, the inputs of AND gates are programmable. That means each AND gate has both normal and complemented inputs of variables. So, based on the requirement, we can program any of those inputs. So, we can generate only the required product terms by using these AND gates. Here, the inputs of OR gates are not of programmable type. So, the number of inputs to each OR gate will be of fixed type. Hence, apply those required product terms to each OR gate as inputs. Therefore, the outputs of PAL will be in the form of sum of products form.

The device shown in the below figure has 4 inputs and 4 outputs. Each input has a buffer-inverter gate, and each output is generated by a fixed OR gate. The device has 4 sections, each composed of a 3-wide AND-OR array, meaning that there are 3 programmable AND gates in each section.

Each AND gate has 10 programmable input connections indicating by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple input configuration of an AND gate. One of the outputs $\mathbf{F_1}$ is connected to a buffer-inverter gate and is fed back into the inputs of the AND gates through programmed connections.



Designing using a PAL device, the Boolean functions must be simplified to fit into each section. The number of product terms in each section is fixed and if the number of terms in the function is too large, it may be necessary to use two or more sections to implement one Boolean function.

Example:

Implement the following Boolean functions using the PAL device as shown above,

W(A, B, C, D) = Σm(2, 12, 13)

X(A, B, C, D) = Σm(7, 8, 9, 10, 11, 12, 13, 14, 15)

Y(A, B, C, D) = Σm(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)

Z(A, B, C, D) = Σm(1, 2, 8, 12, 13)

Simplifying the 4 functions to a minimum number of terms results in the following Boolean functions:

W = ABC’ + A’B’CD’

X = A + BCD

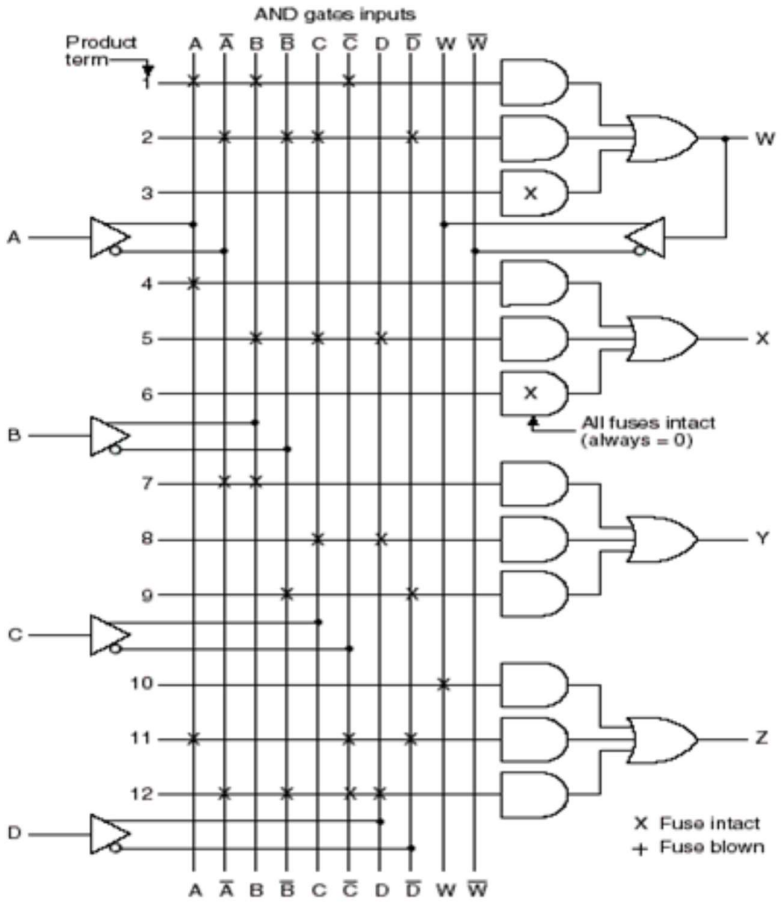
Y = A’B + CD + B’D’

Z = ABC’ + A’B’CD + AC’D’ + A’B’C’D =W +AC’D’ + A’B’C’D

Note that the function for **Z** has four product terms. The logical sum of two of these terms is equal to **W**. Thus, by using **W**, it is possible to reduce the number of terms for **Z** from four to three, so that the function can fit into the given PAL device.

Product term	AND Inputs A	B	C	D	W	Outputs
1	1	1	0	—	—	$W = AB\overline{C}$
2	0	0	1	0	—	$+\overline{A}\,\overline{B}CD$
3	—	—	—	—	—	
4	1	—	—	—	—	$X = A$
5	—	1	1	1	—	$+BCD$
6	—	—	—	—	—	
7	0	1	—	—	—	$Y = \overline{A}B$
8	—	—	1	1	—	$+CD$
9	—	0	—	0	—	$+\overline{B}\,\overline{D}$
10	—	—	—	—	1	$Z = W$
11	1	—	0	0	—	$+A\overline{C}\,\overline{D}$
12	0	0	0	1	—	$+\overline{A}\,\overline{B}\,\overline{C}D$

The PAL programming table is similar to the table used for the PLA, except that only the inputs of the AND gates need to be programmed. The following figure shows the connection map for the PAL device, as specified in the programming table.



Since both W and X have two product terms, third AND gate is not used. If all the inputs to this AND gate left intact, then its output will always be 0, because it receives both the true and complement of each input variable i.e., AA' =0

Inferences:

If an I/O pin's output-control gate produces a constant 1, the output is always enabled, but the pin may still be used as an input too.

Outputs can be used to generate first-pass "*helper terms*" for logic functions that cannot be performed in a single pass with the limited number of AND terms available for a single output.

Comparison: PAL Vs PLA

- PALs have the same limitations as PLAs (small number of allowed AND terms) plus they have a fixed OR plane i.e., less flexibility than PLAs
- PALs are simpler to manufacture, cheaper, and faster (better performance)
- PALs also often have extra circuitry connected to the output of each OR gate
    - The OR gate plus this circuitry is called a macro-cell

Comparison of ROM, PAL and PLA

- ROM – full AND plane, general OR plane
    - cheap (high-volume component)
    - can implement any function of n inputs
    - medium speed
- PAL – programmable AND plane, fixed OR plane
    - intermediate cost
    - can implement functions limited by number of terms
    - high speed (only one programmable plane that is much smaller than ROM's decoder)
- PLA – programmable AND and OR planes
    - most expensive (most complex in design, need more sophisticated tools)
    - can implement any function up to a product term limit
    - slow (two programmable planes)

PROGRAMMING SPLDs:

SPLDs must be programmed so that the switches are in the correct places CAD tools are usually used to do this. A fuse map is created by the CAD tool and then that map is downloaded to the device via a special programming unit.

There are two basic types of programming techniques,

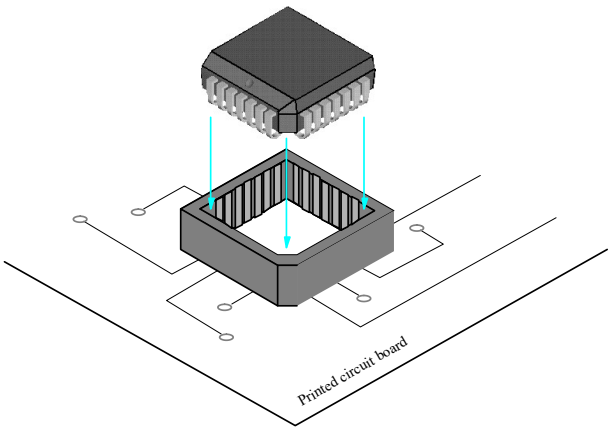
- Removable sockets on a PCB

- In system programming (ISP) on a PCB

This approach is not very common for PLAs and PALs but it is quite common for more complex PLDs.

Removable SPLD Socket Package:

The SPLD is removed from the PCB, placed into the unit and programmed there.



In-System Programming (ISP):

Used when the SPLD cannot be removed from the PCB. A special cable and PCB connection are required to program the SPLD from an attached computer. Very common approach to programming more complex PLDs like CPLDs, FPGAs, etc.

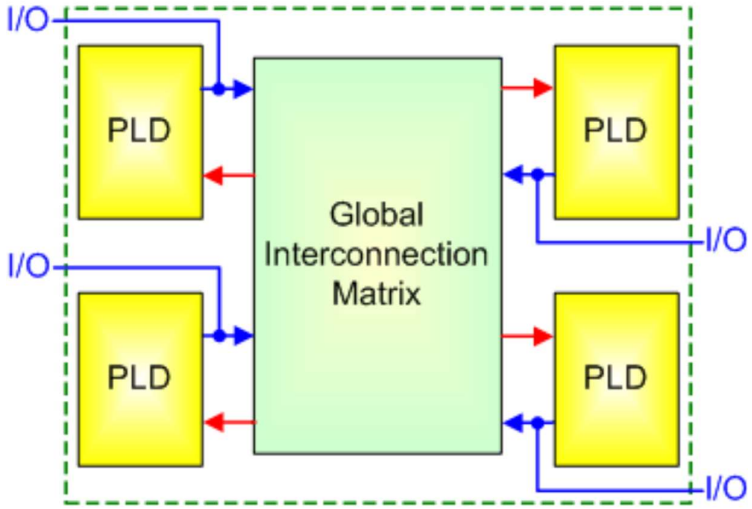
5. COMPLEX PROGRAMMABLE LOGIC DEVICES (CPLDs):

A CPLD contains a bunch of PLD blocks whose inputs and outputs are connected together by a global interconnection matrix. A CPLD is an arrangement of many SPLD-like blocks on a single chip. These circuit blocks might be either PAL-like or PLA-like blocks. Thus a CPLD has two levels of programmability: each PLD block can be programmed, and then the interconnections between the PLDs can be programmed.

Characteristics:

- They have a higher input to logic gate ratio.
- These devices are denser than SPLDs but have better functional abilities.
- CPLDs are based on EPROM or EEPROM technology.
- If you require a larger number of macrocells for a given application, ranging anywhere between 32 to 1000 macrocells, then a Complex Programmable Logic Device is the solution.

- Thus, we use CPLD in applications involving larger I/Os, but data processing is relatively low.



CASE STUDY: Xilinx XC9500 CPLD Family:

The Xilinx XC9500 series is a family of CPLDs with a similar architecture but varying numbers of external input/output (I/O) pins and internal PLDs which is called as function blocks (FBs). Each internal PLD has 36 inputs and 18 macrocells according to the number of chip family. Macro-cells whose outputs are usable only internally are sometimes called buried macrocells.

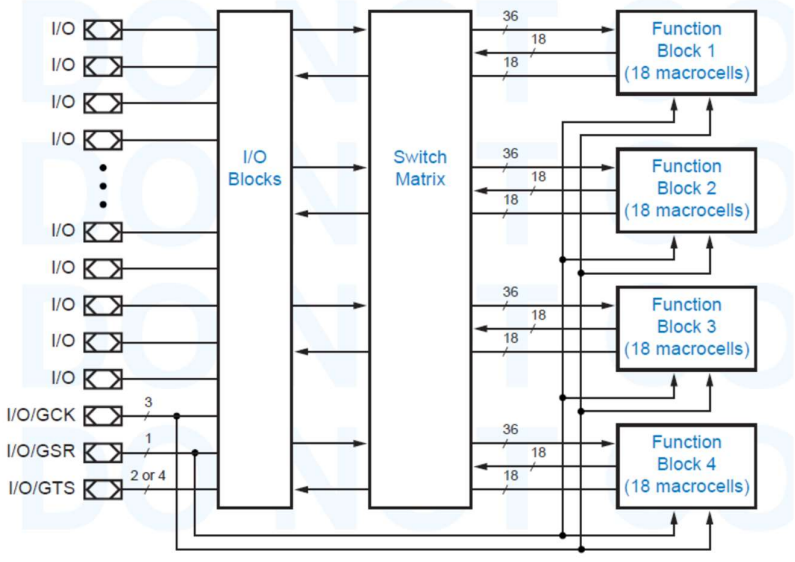
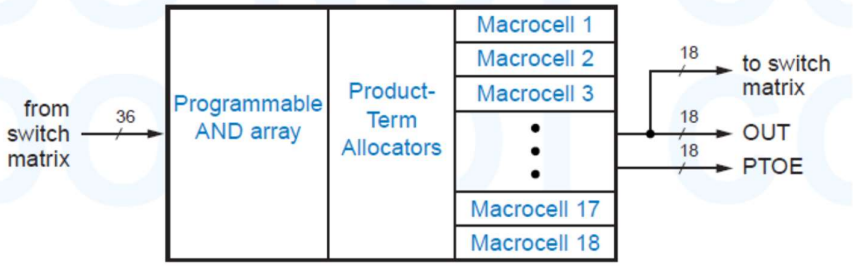


Figure shows the block diagram of the internal architecture of a typical XC9500-family CPLD. The external I/O pins can be used as input, output or bi-directional pins according to device programming. The pins marked I/O/GCK, I/O/GSR and I/O/GTS are special purpose

pins. Any of these pins can b e used as global clocks (GCK). The same pin can be used as an asynchronous preset or clear. Two or four pins can be used as global three state controls (GTS).

Function Block Architecture

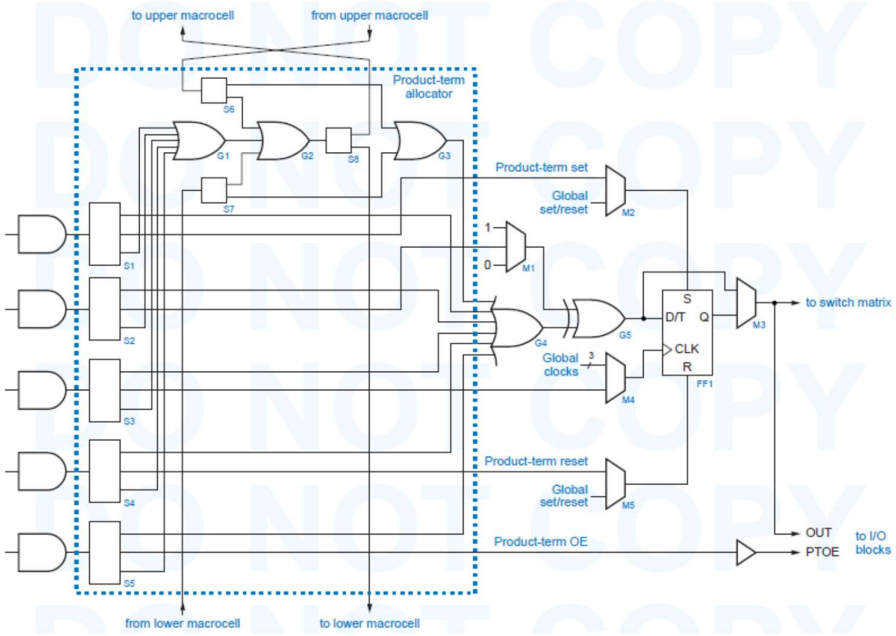
The basic structure of an XC9500 FB is shown in following figure. The programmable AND array has just 90 product terms. It has fewer AND terms per microcell. Each FB will receive 36 signals from the switch matrix, the macrocell outputs from each of the FB and the external inputs from the I/O pins are applied to the switching matrix. Each FB has 18 outputs which run "under" the switch matrix and connect to the I/O blocks. These signals are only the output enable signals for the I/O block output drivers



Macrocell:

The XC9500 and other CPLDs have product-term allocators that allow a macrocell's unused product terms to be used by other nearby macrocells in the same FB. Figure shows the logic diagram of the XC9500 product-term allocator and macrocell. In this figure, the rectangular boxes labelled S1–S8 are programmable signal-steering elements that connect their input to one of their two or three outputs. The trapezoidal boxes labeled M1–M5 are programmable multiplexers that connect one of their two to four inputs to their output. The five AND gates associated with the macrocell appear on the left-hand side of the figure. Each one is connected to a signal-steering box whose top output connects the product term to the macrocell's main OR gate G4. Considering just this, only five product terms are available per macrocell. However, the top, sixth input of G4 connects to another OR gate G3 that receives product terms from the macrocells above and below the current one. Any of the macrocell's product terms that are not otherwise used can be steered through S1–S5 to be combined in an OR gate G1 whose output can eventually be steered to the macrocell above or below by S8. Before steering, product-term allocator these product terms may be combined with product terms from below or above through S6, S7, and G2. Thus, product terms can be "daisy-chained" through successive macrocells to create larger sums of products. In principle, all 90 product

terms in the FB could be combined and steered to one macrocell, although that would leave 17 out of the FB's 18 macrocells with no product terms at all.



Switch Matrix:

The Fast CONNECT switch matrix connects signals to the FB inputs. All IOB outputs (corresponding to user pin inputs) and all FB outputs drive the Fast CONNECT matrix. Any of these (up to a FB fan-in limit of 36) may be selected, through user programming, to drive each FB with a uniform delay. The switch matrix is capable of combining multiple internal connections into a single wired-AND output before driving the destination FB. This provides additional logic capability and increases the effective logic fan-in of the destination FB without any additional timing delay.

I/O Block:

The I/O Block (IOB) interfaces between the internal logic and the device user I/O pins. Each IOB includes an input buffer, output driver, output enable selection multiplexer, and user programmable ground control. The input buffer is compatible with standard 5V CMOS, 5V TTL, and 3.3V signal levels. The input buffer uses the internal 5V voltage supply (VCCINT) to ensure that the input thresholds are constant and do not vary with the VCCIO voltage.

The output enable may be generated from one of four options: a product term signal from the macrocell, any of the global OE signals, always [1], or always [0]. There are two global output enables for devices with up to 144 macrocells, and four global output enables for

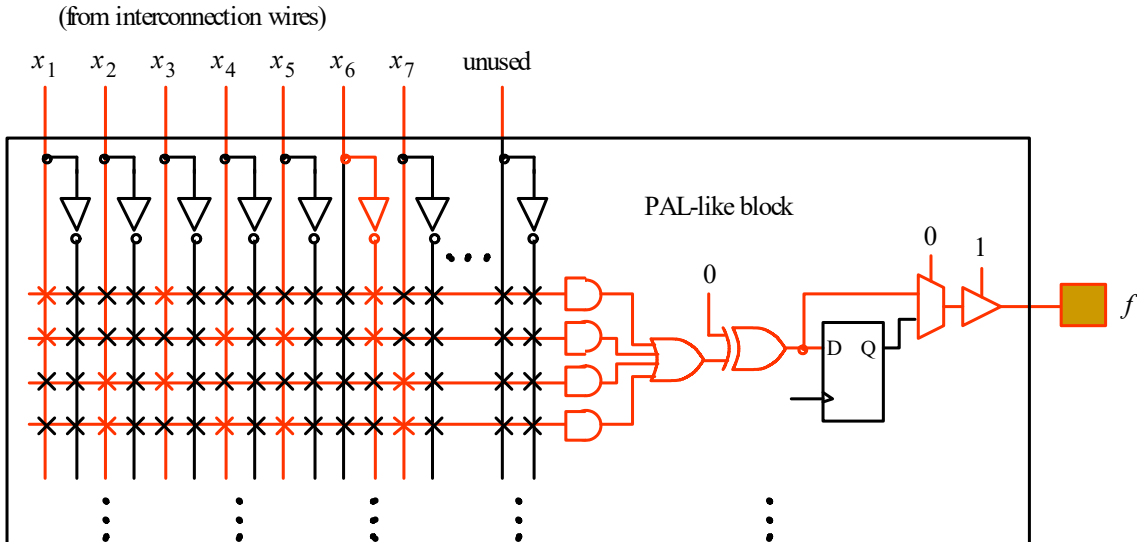
the rest of the devices. Both polarities of any of the global 3-state control (GTS) pins may be used within the device.

Features:

- High-performance
- Large density range: 36 to 288 macrocells with 800 to6,400 usable gates
- 5V in-system programmable
- Endurance of 10,000 program/erase cycles
- Program/erase over full commercial voltage and temperature range
- Enhanced pin-locking architecture
- Flexible 36V18 Function Block
- 90 product terms drive any or all of 18 macrocells within Function Block
- Global and product term clocks, output enables, set and reset signals
- Extensive IEEE Std 1149.1 boundary-scan (JTAG) support
- Programmable power reduction mode in each macrocell
- Slew rate control on individual outputs - User programmable ground pin capability
- Extended pattern security features for design protection
- High-drive 24 mA outputs
- 3.3V or 5V I/O capability

Example:

Use a CPLD to implement the function, $f = x_1x_3x_6' + x_1x_4x_5x_6' + x_2x_3x_7 + x_2x_4x_5x_7$



Applications of CPLD:

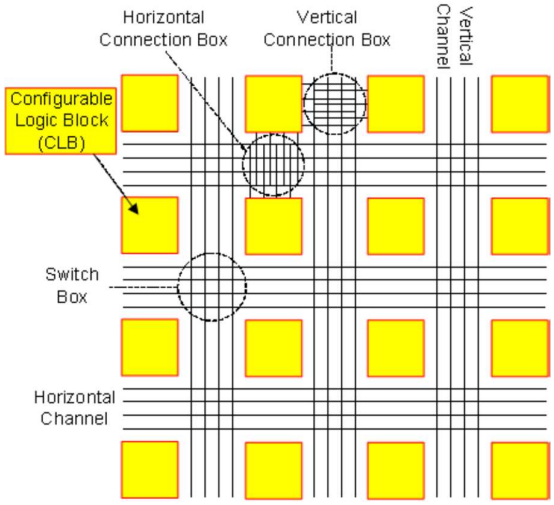
- Complex programmable logic devices are ideal for high performance, critical control applications.
- CPLD can be used in digital designs to perform the functions of boot loader
- CPLD is used for loading the configuration data of a field programmable gate array from non-volatile memory.
- Generally, these are used in small design applications like address decoding
- CPLDs are frequently used many applications like in cost sensitive, battery operated portable devices due to its low size and usage of low power.

6. FIELD PROGRAMMABLE GATE ARRAYS (FPGAs):

A Field Programmable Gate Array has an entire logic system integrated on a single chip. It offers excellent flexibility for reprogramming to the system designers. Logic circuitry involving more than a thousand gates use FPGAs. Compared to a normal custom system chip, the FPGA has ten times better integration density.

The FPGA consists of 3 main structures:

1. Programmable logic structure,
2. Programmable routing structure, and
3. Programmable Input/Output (I/O).



Programmable Logic Structure:

The programmable logic structure FPGA consists of a 2-dimensional array of configurable logic blocks (CLBs). These logic blocks have a lookup table in which the sequential circuitry is implemented. Each CLB can be configured (programmed) to implement

any Boolean function of its input variables. Typically CLBs have between 4-6 input variables. Functions of larger number of variables are implemented using more than one CLB. In addition, each CLB typically contains 1 or 2 FFs to allow implementation of sequential logic.

Large designs are partitioned and mapped to a number of CLBs with each CLB configured (programmed) to perform a particular function. These CLBs are then connected together to fully implement the target design. Connecting the CLBs is done using the FPGA programmable routing structure.

Configurable Logic Blocks (CLBs):

Look-up Table (LUT)-Based CLB :

The basic unit of look-up table based FPGAs is the configurable logic block. The configurable logic block implements the logic functions. The look-up table based FPGA is the Xilinx 4000-series FPGA. Further, configurable logic block implements functions.

PLA-Based CLB :

PLA-based FPGA devices are based on conventional PLDs. The important advantage of this structure is the logic circuits are implemented using only a few level logic. To improve integration density logic expander is used.

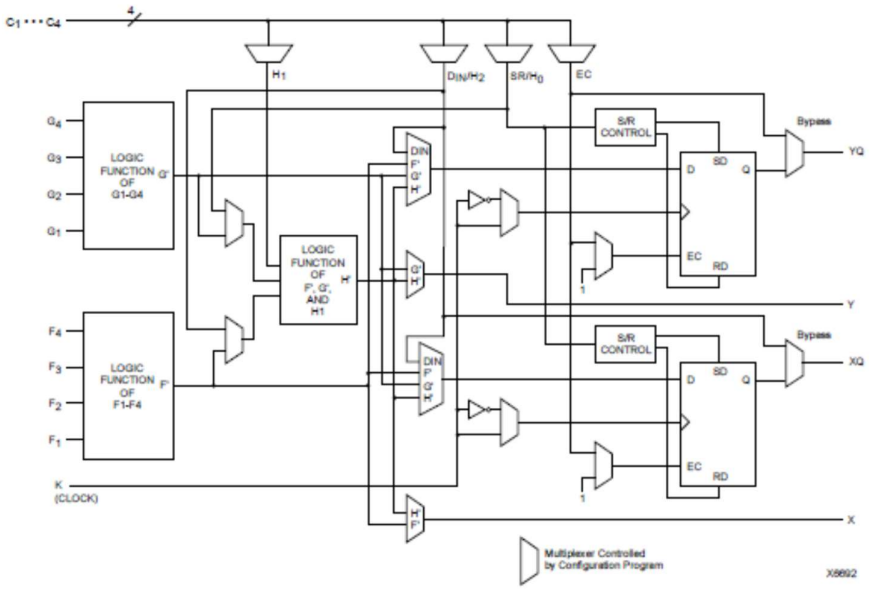
Multiplexer-Based CLB :

In Multiplexer-based FPGAs to implement the logic circuits the multiplexers are used. The main advantage of multiplexer-based FPGA is to provide more functionality by using minimum transistors. Due to large number of inputs, multiplexer-based FPGAs place high demands on routing.

CASE STUDY: Xilinx 4000 FPGA Family:

The principle CLB elements are shown in following figure. Each CLB contains a pair of flip-flops and two independent 4-input function generators. These function generators have a good deal of flexibility as most combinatorial logic functions need less than four inputs. Thirteen CLB inputs and four CLB outputs provide access to the functional flip-flops. Configurable Logic Blocks implement most of the logic in an FPGA. Two 4-input function generators (F and G) offer unrestricted versatility. Most combinatorial logic functions need four or fewer inputs. However, a third function generator (H) is provided. The H function generator has three inputs. One or both of these inputs can be the outputs of F and G; the other

input(s) are from outside the CLB. The CLB can therefore implement certain functions of up to nine variables, like parity check or expandable-identity comparison of two sets of four inputs.



Each CLB contains two flip-flops that can be used to store the function generator outputs. However, the flip-flops and function generators can also be used independently. DIN can be used as a direct input to either of the two flip-flops. H1 can drive the other flip-flop through the H function generator. Function generator outputs can also be accessed from outside the CLB, using two outputs independent of the flip-flop outputs. This versatility increases logic density and simplifies routing. Thirteen CLB inputs and four CLB outputs provide access to the function generators and flip-flops. These inputs and outputs connect to the programmable interconnect resources outside the block.

Four independent inputs are provided to each of two function generators (F1 - F4 and G1 - G4). These function generators, whose outputs are labelled F' and G', are each capable of implementing any arbitrarily defined Boolean function of four inputs. The function generators are implemented as memory look-up tables. The propagation delay is therefore independent of the function implemented. A third function generator, labelled H', can implement any Boolean function of its three inputs. Two of these inputs can optionally be the F' and G' functional generator out-puts. Alternatively, one or both of these inputs can come from outside the CLB (H2, H0). The third input must come from outside the block (H1).

Signals from the function generators can exit the CLB on two outputs. F' or H' can be connected to the X output. G' or H' can be connected to the Y output. A CLB can be used to implement any of the following functions:

- any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables
- any single function of five variables
- any function of four variables together with some functions of six variables
- some functions of up to nine variables

Implementing wide functions in a single block reduces both the number of blocks required and the delay in the signal path, achieving both increased density and speed. The versatility of the CLB function generators significantly improves system speed. In addition, the design-software tools can deal with each function generator independently. This flexibility improves cell usage.

The flexibilty and symmetry of the CLB architecture facilitates the placement and routing of a given application. Since the function generators and flip-flops have independent inputs and outputs, each can be treated as a separate entity during placement to achieve high packing density. Inputs, outputs and the functions themselves can freely swap positions within thew CLB to avoid routing congestion during the placement and routing operation.

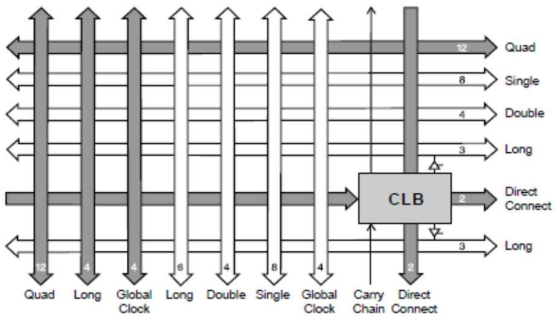
Programmable Routing Structure:

To allow for flexible interconnection of CLBs, FPGAs have 3 programmable routing resources:

1. Vertical and horizontal routing channels which consist of different length wires that can be connected together if needed. These channel run vertically and horizontally between columns and rows of CLBs as shown in the Figure.

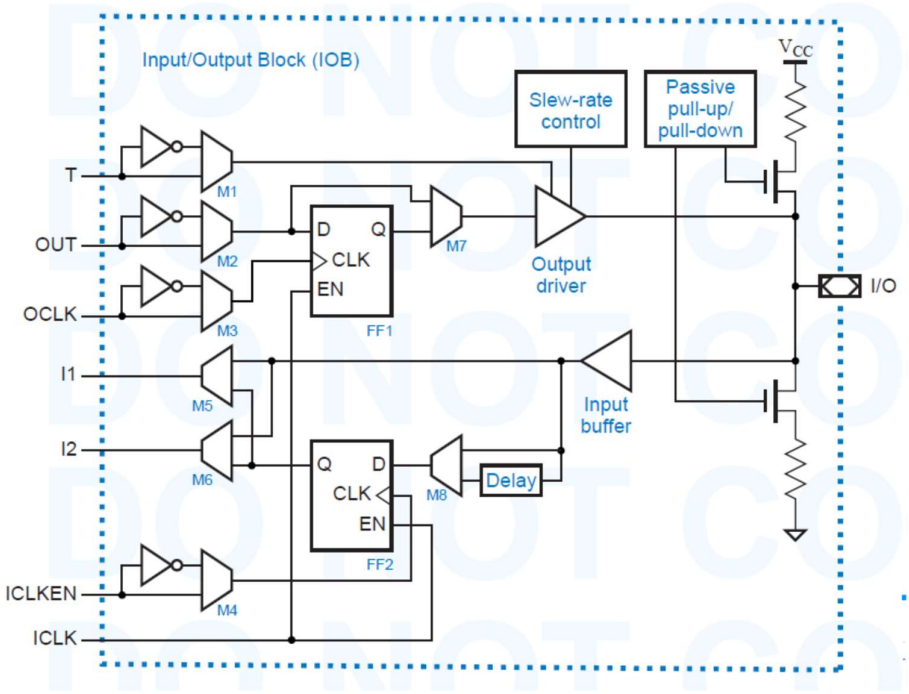
2. Connection boxes, which are a set of programmable links that can connect input and output pins of the CLBs to wires of the vertical or the horizontal routing channels.

3. Switch boxes, located at the intersection of the vertical and horizontal channels. These are a set of programmable links that can connect wire segments in the horizontal and vertical channels.



Programmable I/O:

These are mainly buffers that can be configured either as input buffers, output buffers or input/output buffers. They allow the pins of the FPGA chip to function either as input pins, output pins or input/output pins. The IOBs provide a simple interface between the internal user logic and the package pins.



Input Signals:

Two paths, labelled I1 and I2, bring input signals into the array. Inputs also connect to an input register that can be programmed as either an edge-triggered flip-flop or a level-sensitive transparent-Low latch. The choice is made by placing the appropriate primitive from the symbol library. The inputs can be globally configured for either TTL (1.2V) or CMOS (2.5V) thresholds.

The two global adjustments of input threshold and output level are independent of each other. There is a slight hysteresis of about 300mV.Seperate clock signals are provided for the input and output registers; these clocks can be inverted, generating either falling-edge or rising-edge triggered flip-flops. As is the case with the CLB registers, a global set/reset signal can be used to set or clear the input and output registers whenever the RESET net is alive.

Registered Inputs:

The I1 and I2 signals that exit the block can each carry either the direct or registered input signal. The input and output storage elements in each IOB have a common clock enable input, which through configuration can be activated individually for the input or output flip-

flop or both. This clock enable operates exactly like the EC pin on the XC4000E CLB. It cannot be inverted within the IOB.

Example:

Use an FPGA with 2 input LUTS to implement the function,

$$f = x_1x_3x_6' + x_1x_4x_5x_6' + x_2x_3x_7 + x_2x_4x_5x_7$$

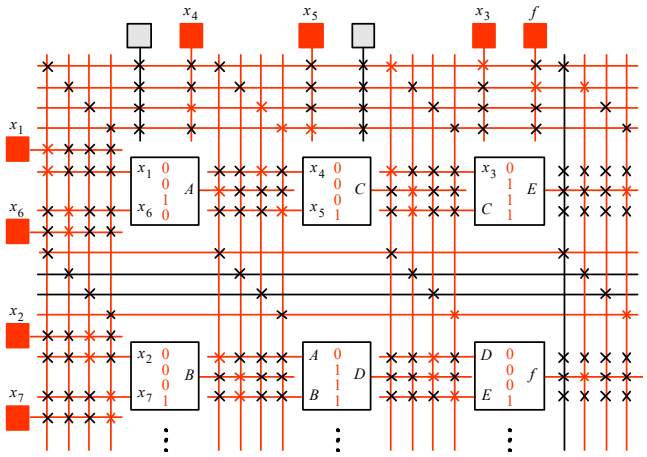
Fan-in of expression is too large for FPGA. This was simple to do in a CPLD

Factor *f* to get sub-expressions with max fan-in = 2

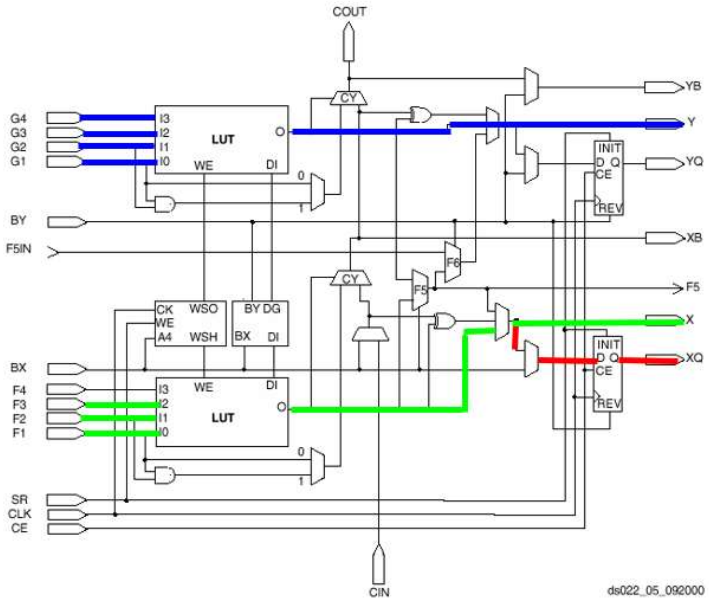
$$f = x_1x_6'(x_3 + x_4x_5) + x_2x_7(x_3 + x_4x_5)$$

$$= (x_1x_6' + x_2x_7)(x_3 + x_4x_5)$$ (Implementation shown in figure)

Could use Shannon's expansion instead. Goal is to build expressions out of 2-input LUTs



Example for four and three input functions:



4-input function

3-input function; registered

Applications of FPGAs:

- Implementation of random logic
  - easier changes at system-level (one device is modified)
  - can eliminate need for full-custom chips
- Prototyping
  - ensemble of gate arrays used to emulate a circuit to be manufactured
  - get more/better/faster debugging done than possible with simulation
- Reconfigurable hardware
  - one hardware block used to implement more than one function
  - functions must be mutually-exclusive in time
  - can greatly reduce cost while enhancing flexibility
  - RAM-based only option
- Special-purpose computation engines
  - hardware dedicated to solving one problem (or class of problems)
  - accelerators attached to general-purpose computers

CONCLUSION:

Over the last few years, programmable logic suppliers have made such phenomenal technical advances that PLDs are now seen as the logical solution of choice from many designers. One reason for this is that PLD suppliers such as Xilinx are "fabless" companies; instead of owning chip manufacturing foundries, Xilinx out sources that job to partners like IBM Microelectronics and UMC, whose chief occupation is making chips. This strategy allows Xilinx to focus on designing new product architectures, software tools, and intellectual property cores while having access to the most advanced semiconductor process technologies. Advanced process technologies help PLDs in a number of key areas: faster performance, integration of more features, reduced power consumption, and lower cost. Today Xilinx is producing programmable logic devices on a state-of-the-art 0.13-micron low-k copper process, one of the best in the industry.

Just a few years ago, for example, the largest FPGA was measured in tens of thousands of system gates and operated at 40 MHz. Today, however, FPGAs with advanced features offer millions of gates of logic capacity, operate at 300 MHz, and offer a new level of integrated functions such as processors and memory.

Just as significant, PLDs now have a growing library of intellectual property (IP) or cores – these are predefined and tested software modules that customer can use to create system functions instantly inside the PLD. Cores include everything from complex digital signal processing algorithms and memory controllers to bus interfaces and full-blown software-based microprocessors. Such cores save customers a lot of time and expense. It would take customers months to create these functions, further delaying a product introduction.

The value of programmable logic has always been its ability to shorten development cycles for electronic equipment manufacturers and help them get their product to market faster. As PLD suppliers continue to integrate more functions inside their devices, reduce costs, and increase the availability of time-saving IP cores, programmable logic is certain to expand its popularity with digital designers. Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.