Reconfigurable Computing

Dr. Christophe Bobda
CSCE Department
University of Arkansas

Chapter 1
Architectures

Agenda

1. Early works
2. Programmable devices (PLD)
   - PAL /PLAs
   - CPLDs
   - FPGAs

Early Work

1. Gerald Estrin Fix-Plus Machine

- Vision of a restructurable computer system

- Pragmatic problem studies predicts gains in computation
- speeds in a variety of computational tasks when executed
- on appropriate problem-oriented configurations of the
- variable structure computer. The economic feasibility of the
- system is based on utilization of essentially the same
- hardware in a variety of special purpose structures. This
- capability is achieved by programmed or physical
- restructuring of a part of the hardware.

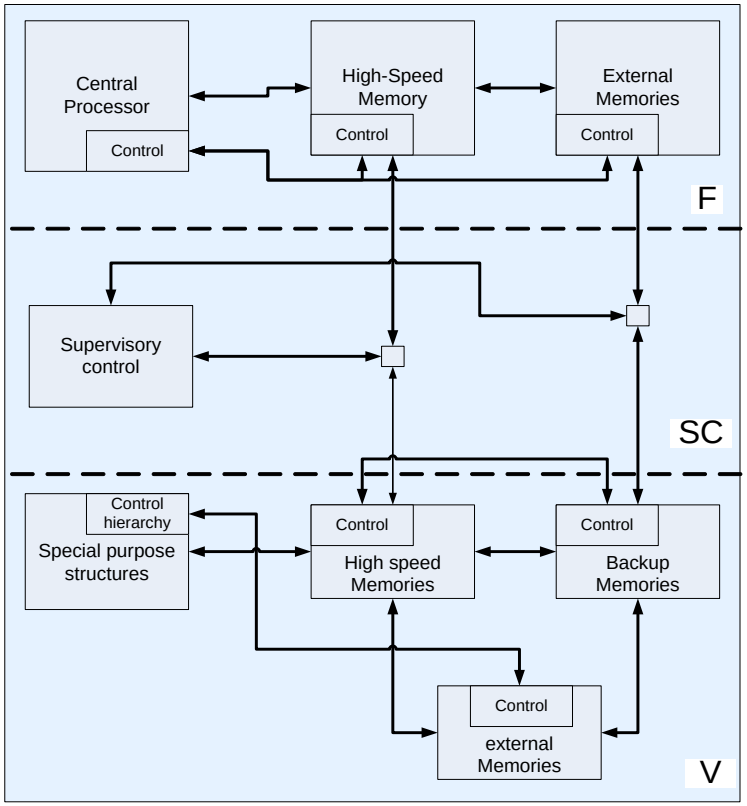
  - G. Estrin, B. Bussel, R. Turn, J Bibb (UCLA 1963)

1. Gerald Estrin Fix-Plus Machine

Fixed plus Variable structure computer

- Proposed by G. Estrin in 1959
- Consist of three parts
  1. A high speed general purpose computer (the fix part F).
  2. A variable part (V) consisting of various size high speed digital substructures which can be reorganized in a problem oriented special purpose configurations.
  3. The supervisory control (SC) coordinates operations between the fix module and the variable module.

Speed gain over IBM7090 (2.5 to 1000)



1. Gerald Estrin Fix-Plus Machine

The Fixed Part (F)

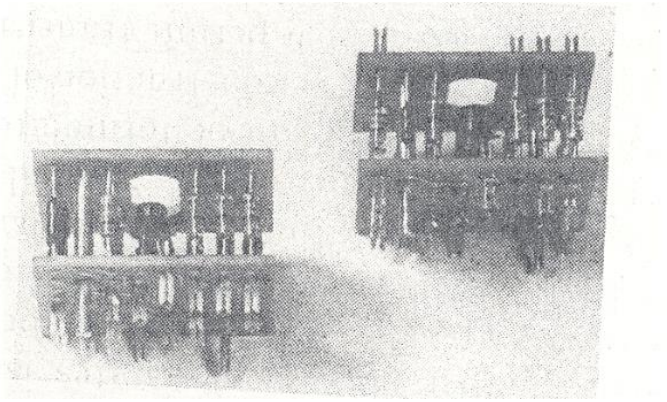
Was initially an IBM 7090, but could be any general purpose computer

The Variable Part (V)

Made upon a set of problem specific optimized functional units in the basic configuration(trigonometric functions, logarithm, exponentials, n-th power, roots, complex arithmetic, hyperbolic, matrix operation)

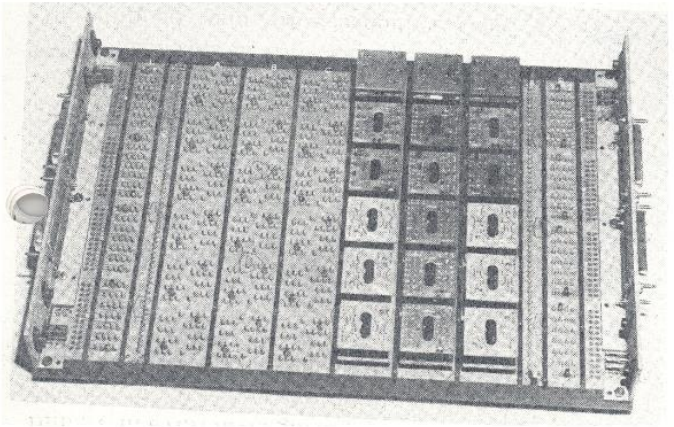
- Two types of basic building block
  - The first basic element contains four amplifiers and associated input logic for signal inversion, amplification, or high-speed storage
  - The second basic block consists of ten diodes and four output drivers and is for combinatoric application

The basic blocks

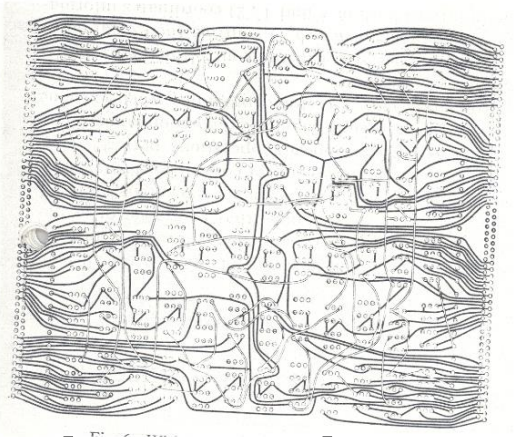


1. Gerald Estrin Fix-Plus Machine

- The basic modules can be inserted into any of 36 positions on a mother board.
- The connection between the modules is done by wiring harness
- Function Reconfiguration means changing some modules
- Routing Reconfiguration means changing some wiring harness



The mother board



The wiring harness

1. Gerald Estrin Fix-Plus Machine

Estrin at work.

Substantial efforts on Reconfiguration

1. The Rammig Machine

- Goal

- Investigation of a system, which, with no manual or
- mechanical interference, permits the building, changing,
- processing and destruction of real (not simulated) digital
- hardware

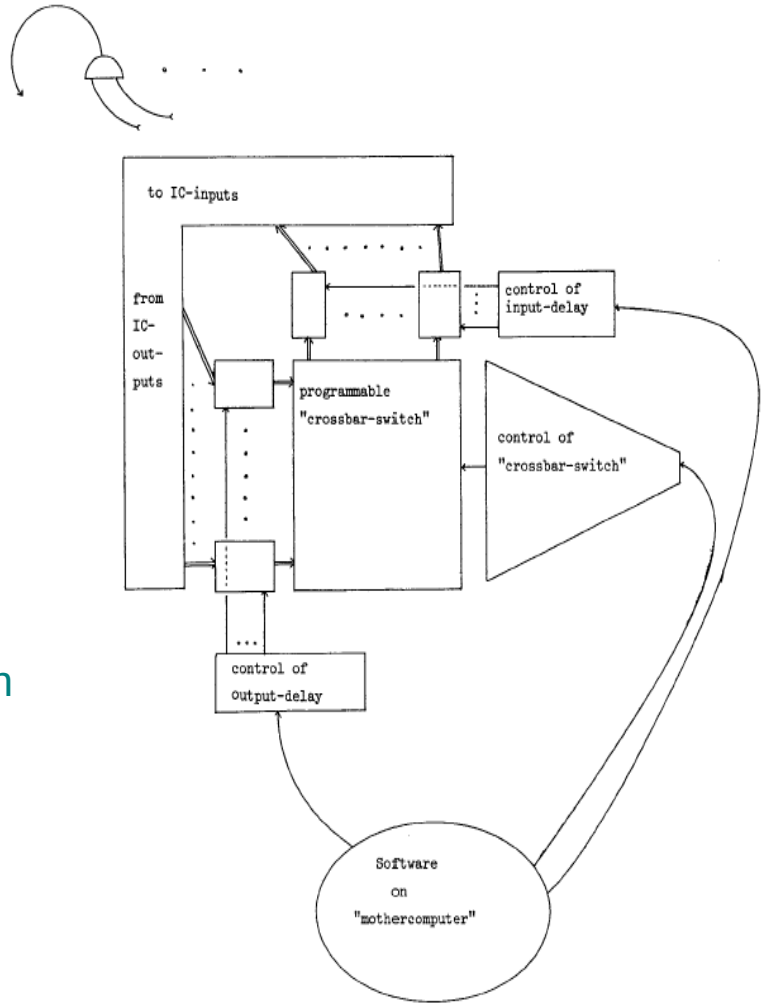
Franz J. Rammig (University of Dortmund 1977)

The concept resulted in the construction of a hardware editor. Useful to observe a circuit under test (Hardware Emulation)

1. The Rammig Machine

Implementation

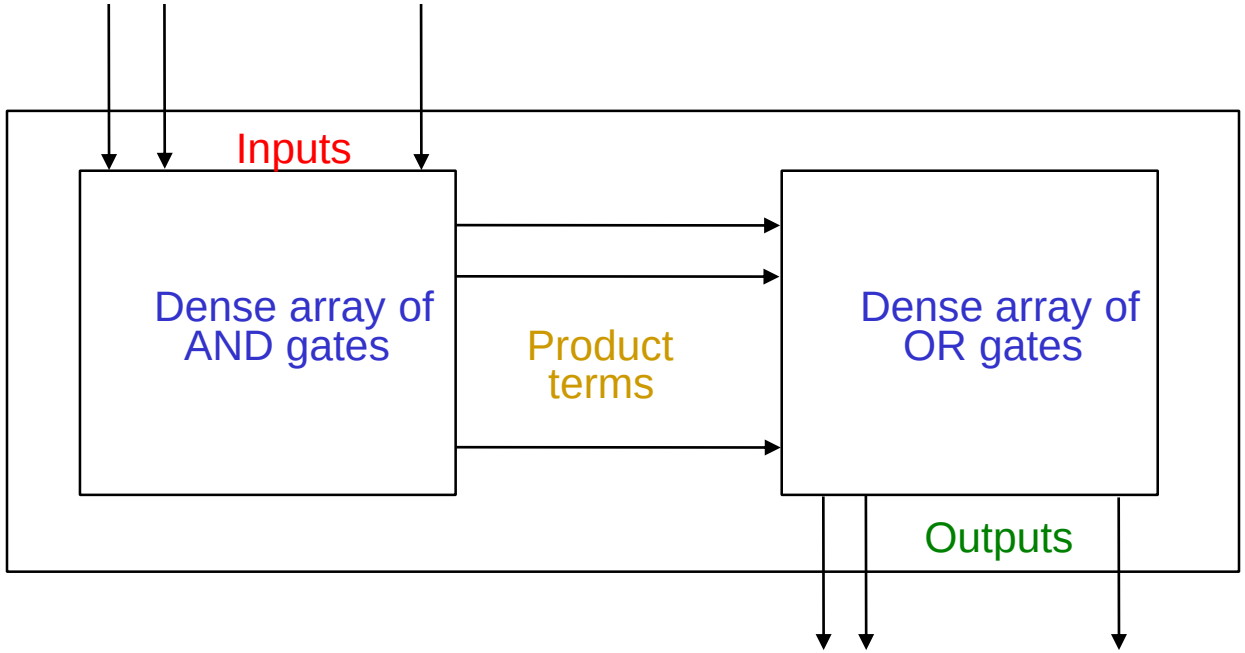
- Outputs of modules connected to Selectors and selectors output connected to module inputs.
- Software controlled modules interconnection
- Two main problems to solve
  - Because the circuit is not hard-wired, a distortion of the time behaviour is possible
  - The timing is controlled by the circuit instead of being dictated by an observation mechanism.
  - A time-controlling must therefore be provided



2. Programmable Logic

2.1 PALs and PLAs

- Pre-fabricated building block of many AND/OR gates (or NOR, NAND)
- "Personalized" by making or breaking connections among the gates



Programmable Array Block Diagram for Sum of Products Form

2.1 PALs and PLAs

Example:

Equations

$$F0 = A + \bar{B}\,\bar{C}$$
$$F1 = A\,\bar{C} + A\,B$$
$$F2 = \bar{B}\,\bar{C} + A\,B$$
$$F3 = \bar{B}\,C + A$$

Personality Matrix

Product term	Inputs A	B	C	Outputs F_0	F_1	F_2	F_3
A B	1	1	-	0	(1)	(1)	0
$\bar{B}$ C	-	0	1	0	0	0	1
A $\bar{C}$	1	-	0	0	1	0	0
$\bar{B}$ $\bar{C}$	-	0	0	(1)	0	(1)	0
A	1	-	-	(1)	0	0	(1)

Reuse of terms

Input Side:

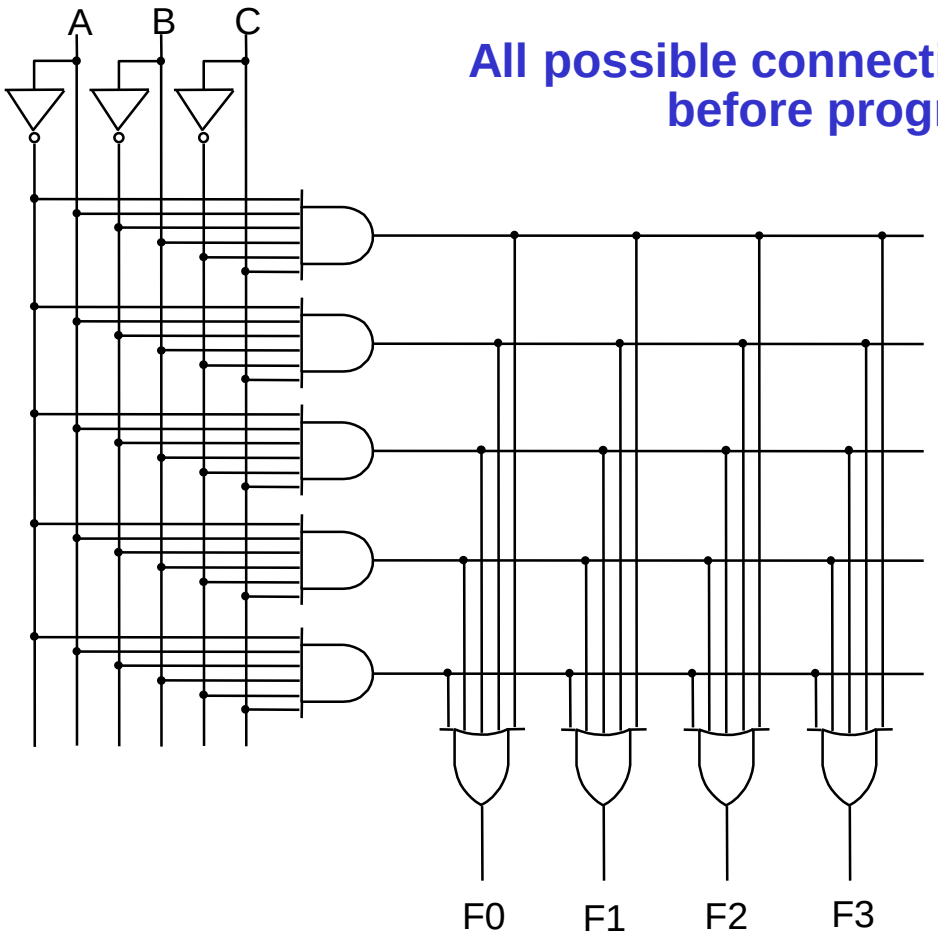
1 = asserted in term
0 = negated in term
- = does not participate

Output Side:

1 = term connected to output
0 = no connection to output

2.1 PALs and PLAs

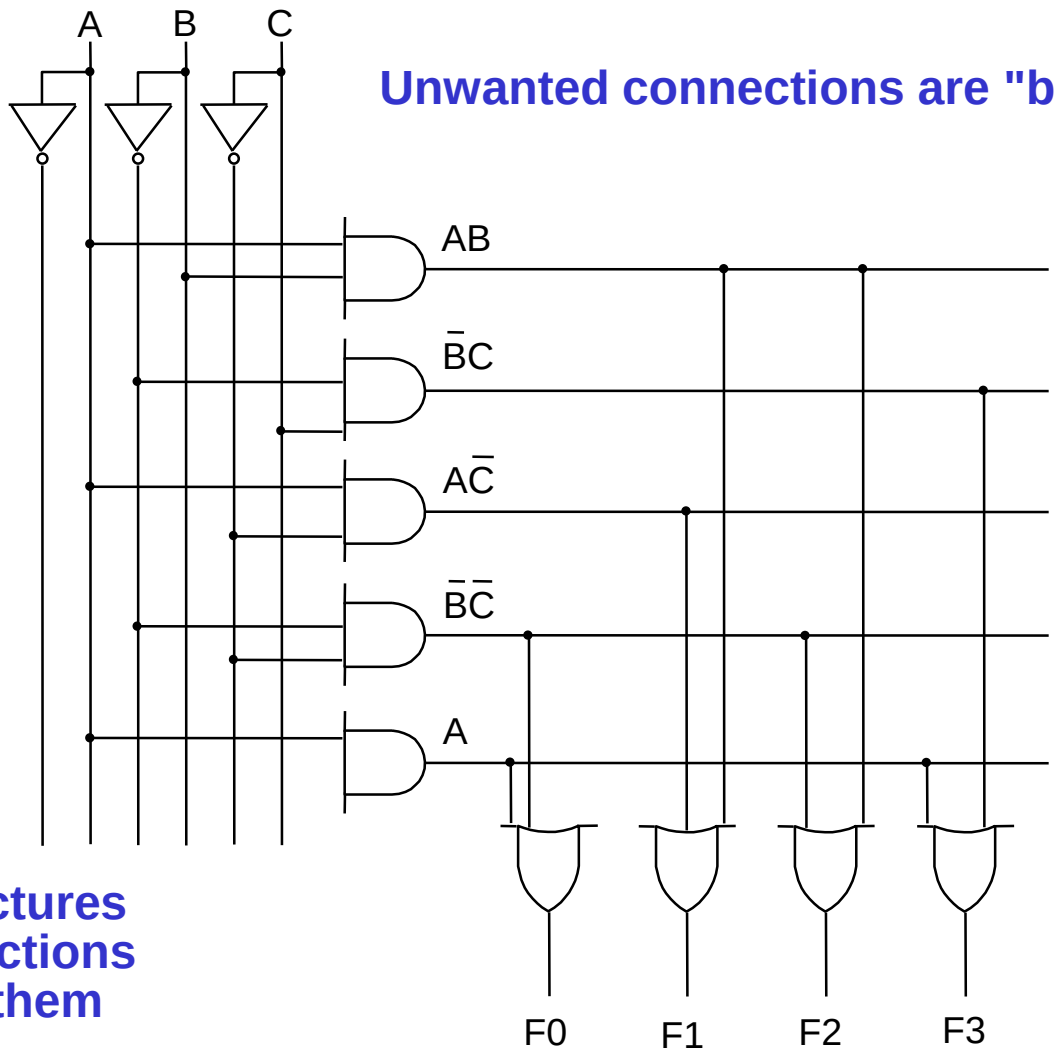
Example Continued - Unprogrammed device



All possible connections are available before programming

2.1 PALs and PLAs

Example Continued - Programmed part



Unwanted connections are "blown"

Note: some array structures work by making connections rather than breaking them

2.1 PALs and PLAs

Alternative representation for high fan-in structures

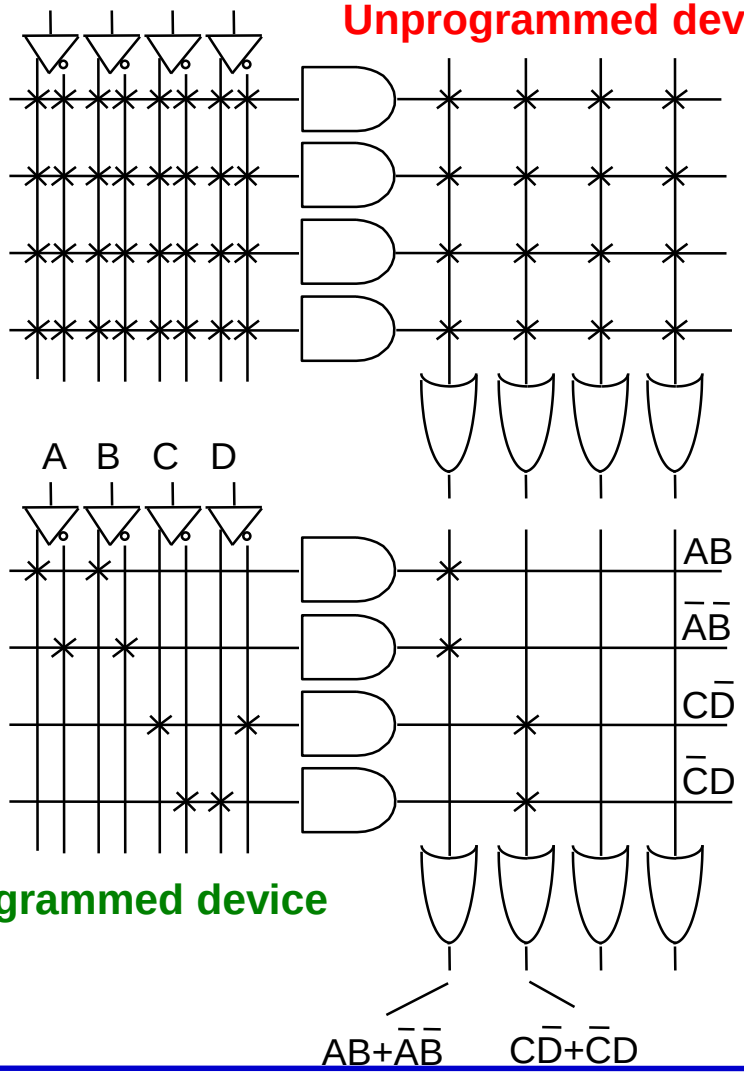
Short-hand notation
so we don't have to
draw all the wires!

X at junction indicates
a connection

Notation for implementing

$F0 = A\,B + \bar{A}\,\bar{B}$

$F1 = C\,\bar{D} + \bar{C}\,D$



2.1 PALs and PLAs

Design Example

Multiple functions of A, B, C

$F1 = A\,B\,C$

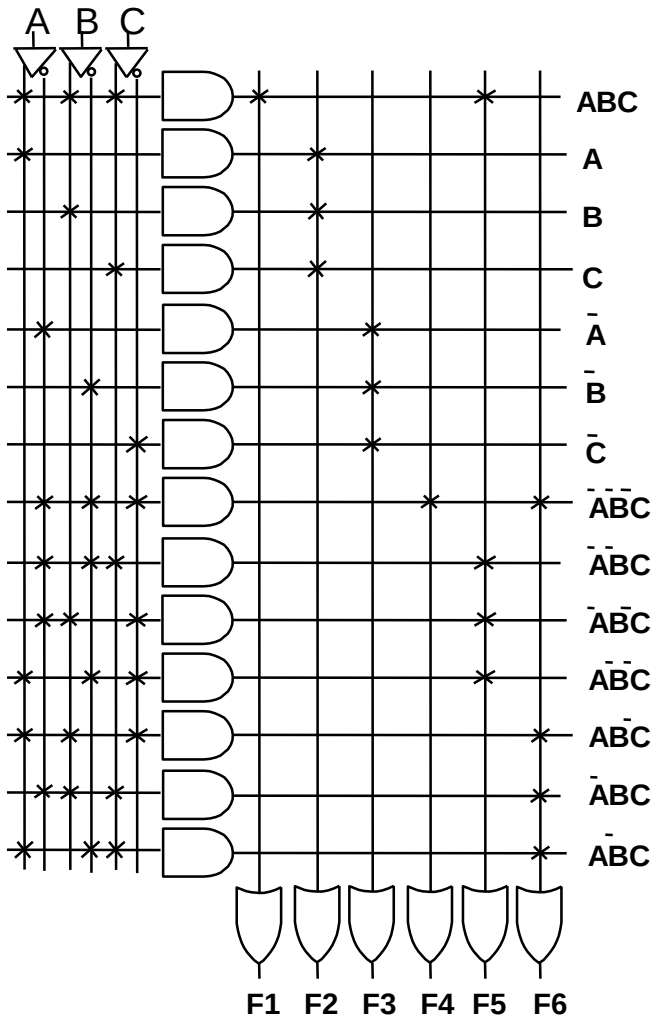
$F2 = A + B + C$

$F3 = \overline{A\,B\,C}$

$F4 = \overline{A + B + C}$

$F5 = A \oplus B \oplus C$

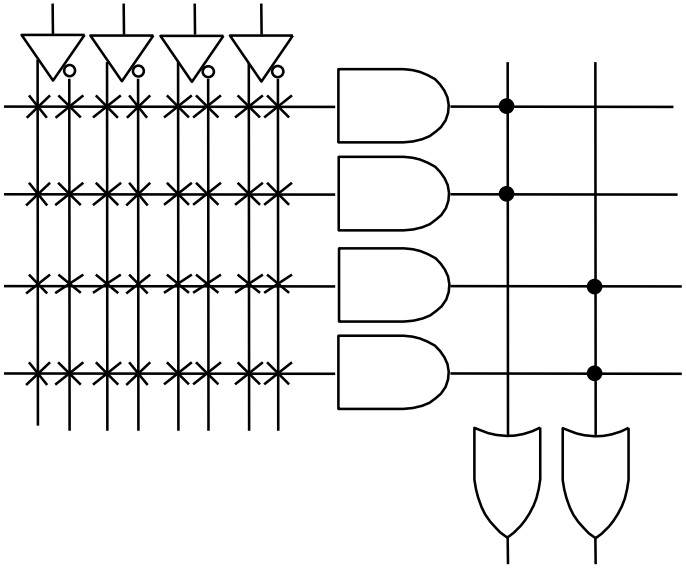
$F6 = \overline{A \oplus B \oplus C}$



2.1 PALs and PLAs

What is difference between Programmable Array Logic (PAL) and Programmable Logic Array (PLA)?

PAL : AND array is programmable, OR array is fixed at fabrication



A given column of the OR array has access to only a subset of the possible product terms

PLA: Both AND and OR arrays are programmable

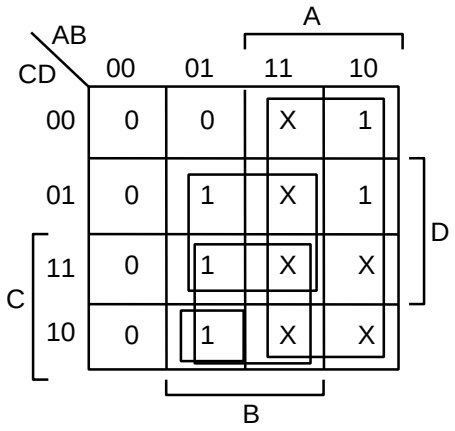
2.1 PALs and PLAs

Design Example: BCD to Gray Code Converter

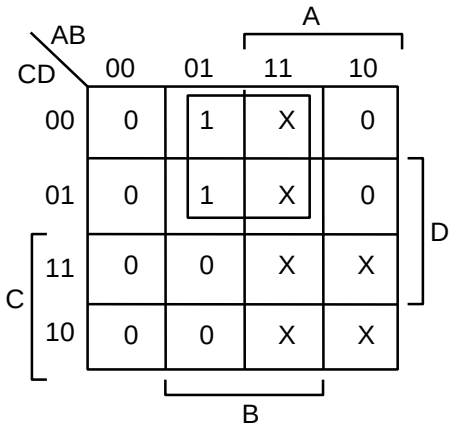
Truth Table

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

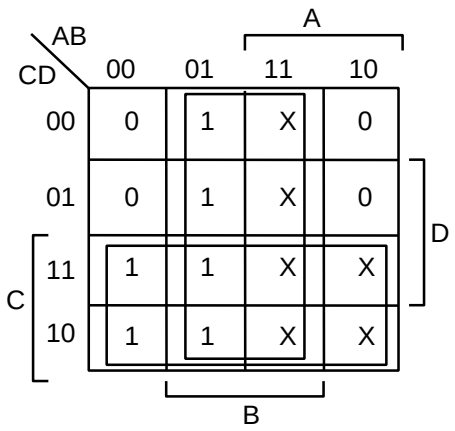
K-maps



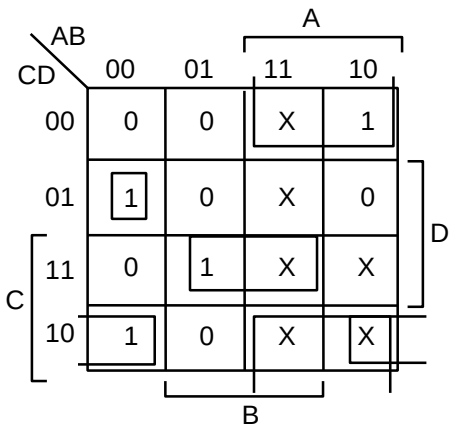
K-map for W



K-map for X



K-map for Y



K-map for Z

Minimized Functions:

$W = A + B\,D + B\,C$

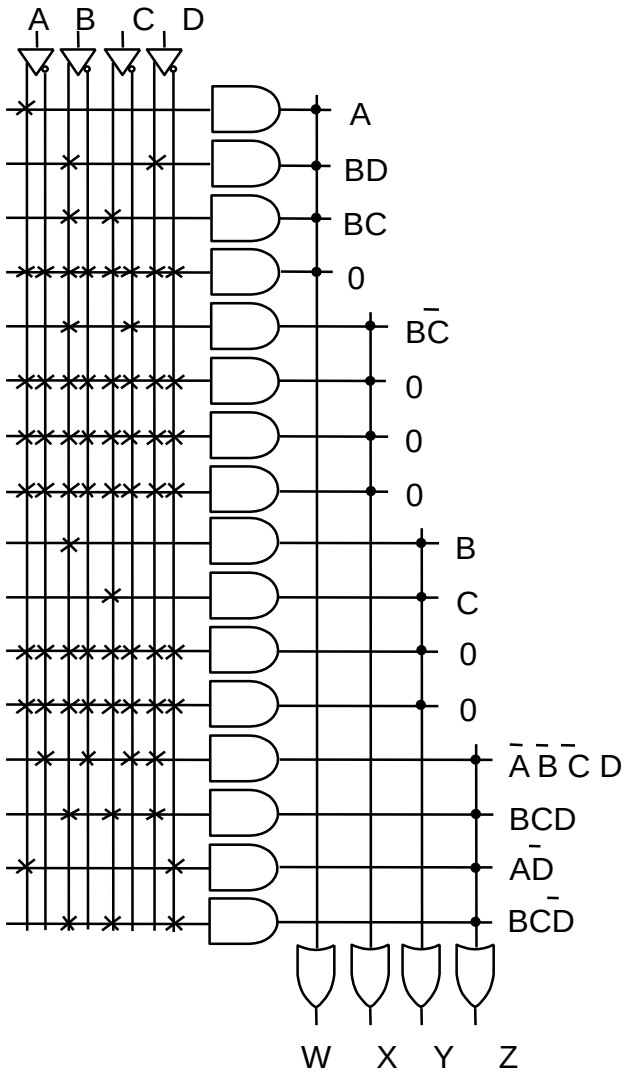
$X = B\,\overline{C}$

$Y = B + C$

$Z = \overline{A}\,\overline{B}\,\overline{C}\,D + B\,C\,D + A\,\overline{D} + B\,\overline{C}\,\overline{D}$

2.1 PALs and PLAs

Programmed *PAL:*



Minimized Functions:

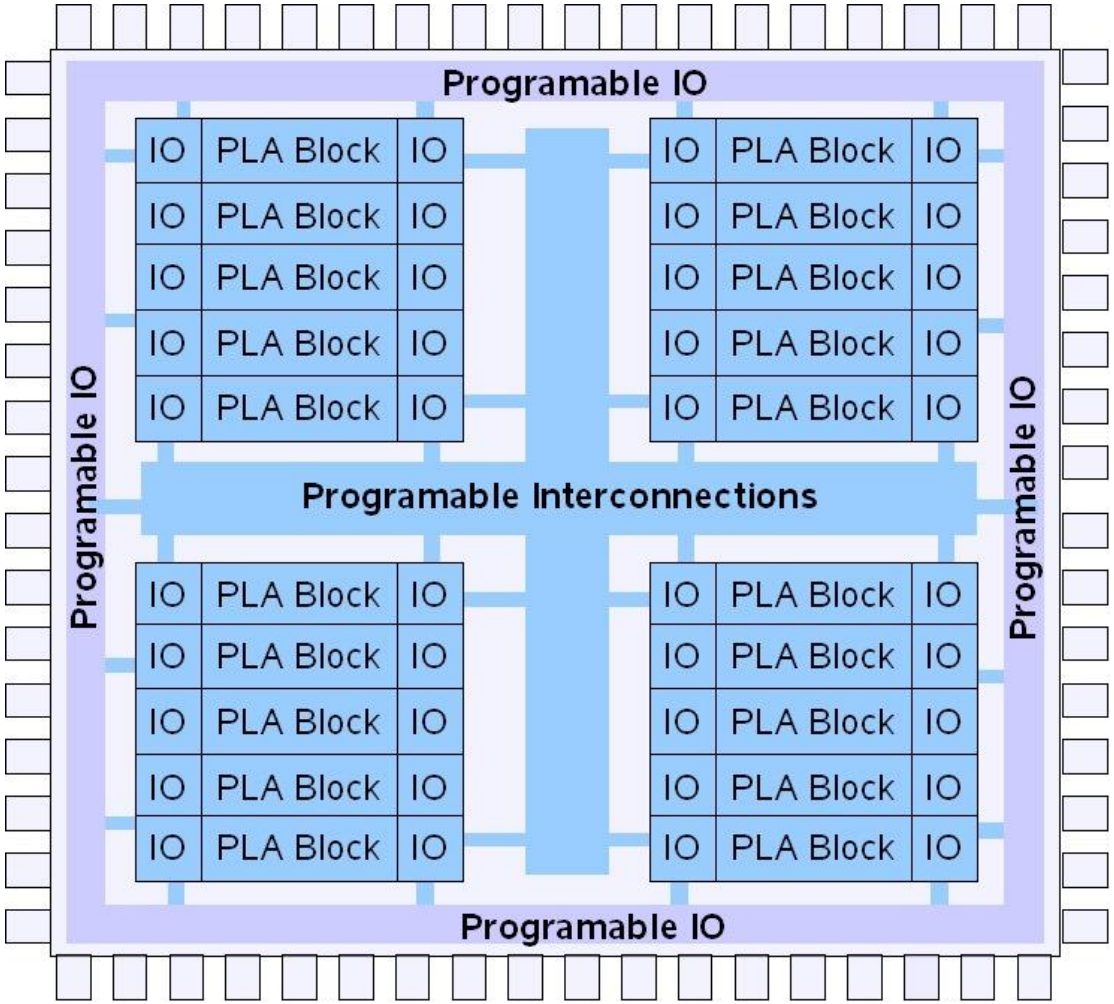
$W = A + B\,D + B\,C$
$X = B\,\overline{C}$
$Y = B + C$
$Z = \overline{A}\,\overline{B}\,\overline{C}\,D + B\,C\,D + A\,\overline{D} + B\,\overline{C}\,\overline{D}$

4 product terms per each OR gate

2.2 Complex Programmable Logic Devices

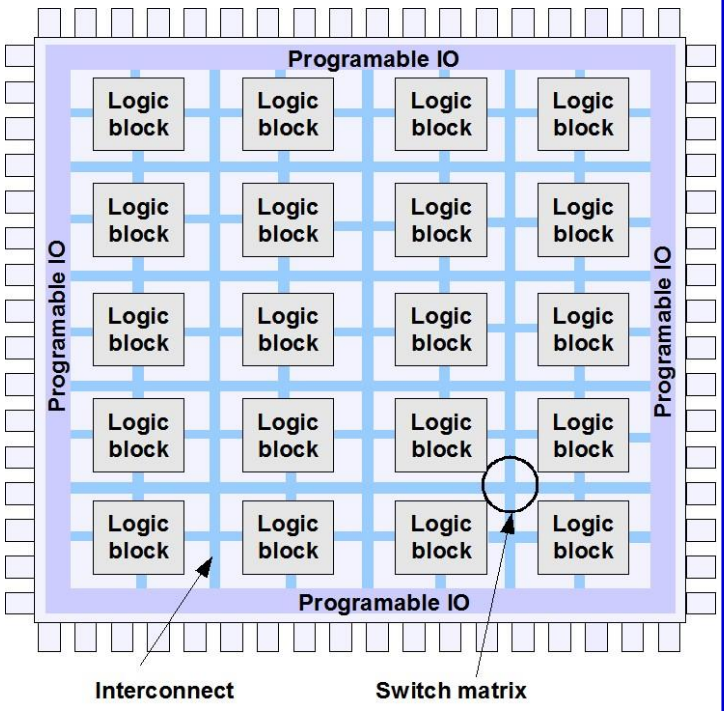
- Complex PLDs (CPLD) typically combine PAL combinational logic with Flip Flops
  - Organized into logic blocks connected in an interconnect matrix
  - Combinational or registered output
- Usually enough logic for simple counters, state machines, decoders, etc.
- CPLDs logic is not enough for complex operation
- FPGAs have much more logic than CPLDs
- e.g. Xilinx Coolrunner II, etc.

2.2 Complex Programmable Logic Devices



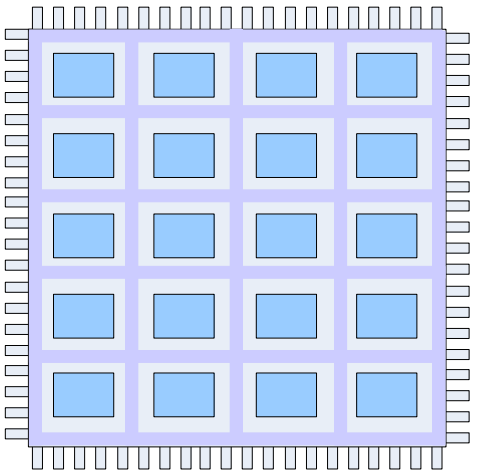
2.3 Field Programmable Gate Arrays (FPGAs)

- Introduced in 1985 by Xilinx
- Roughly seen, an FPGA consist of:
  - A set of programmable macro cells
  - A programmable interconnection network
  - Programmable input/outputs
  - Subparts of a (complex) function are implemented in macro cells which are then connected to build the complete function
  - The IO can be programmed to drive the macro cell's inputs or to be driven by the macro cell's outputs
  - Unlike traditional application-specific integrated circuit (ASIC), function is specified by the user *after* the device is manufactured
  - Physical structure and programming method is vendor dependant

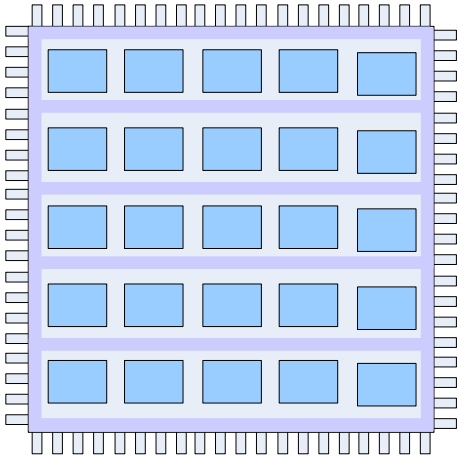


2.3 FPGA Structure

- Typical organization
  1. Symmetrical Array
     - 2 D array of processing elements (PE) embedded in an interconnection network
     - Interconnection points at the horizontal-vertical intersection
  2. Row based
     - Rows of Processing elements
     - Horizontal routing via horizontal channels
     - Channels divided in segments
     - Vertical connections via dedicated vertical tracks (not on the graphic)



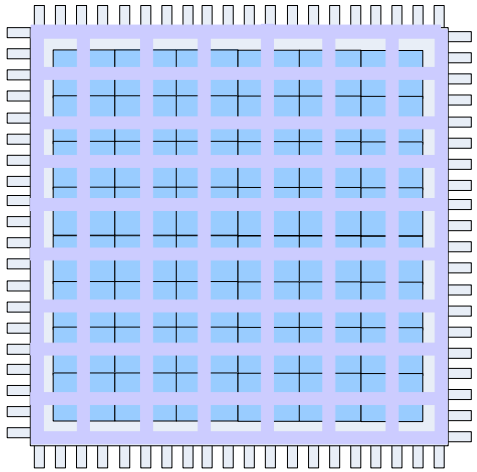
Symetrical array



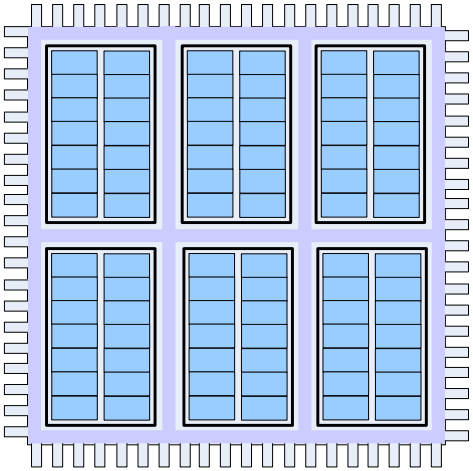
Row-based

2.3 FPGA Structure

- Typical organization (cont)
  3. Sea of gates
     - 2 D array of processing elements
     - No space left aside the PEs for routing
     - Connection is done on a separate layer on top of the cells
  4. Hierarchical
     - Hierarchically placed Macro cells
     - Low-level macro cells are grouped to build the higher-level's PEs



Sea of gates

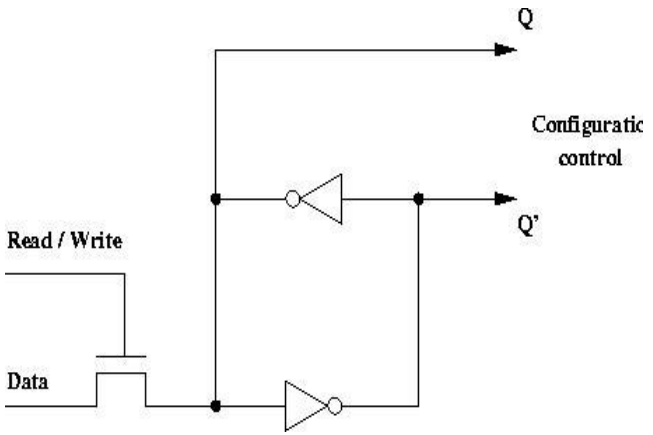


Hierarchical

2.3 FPGA Programming Technologies

SRAM (LUT-based)

- An SRAM is used to store all possible values of a function
- Value of a function for a given input is retrieved using the inputs as SRAM-Address
- SRAM implementing a function is called a look-up table (LUT)
- A new function is implemented by writing new values into the LUT
  - ➔ SRAM-based FPGA can therefore be reprogrammed (configured) on the fly
  - ➔ Since a LUT is volatile, a LUT configuration is lost when switching off the system



2.3 FPGA Programming Technologies

Anti-fuse

- An anti-fused normally presents a high-impedance state
- can be “fused” into a low-impedance state when programmed by a high voltage.
- The anti-fuse used in each of FPGAs from different company differs in construction.

Advantages:

➔ small area,

➔ low resistance and parasitic capacitance than transistors

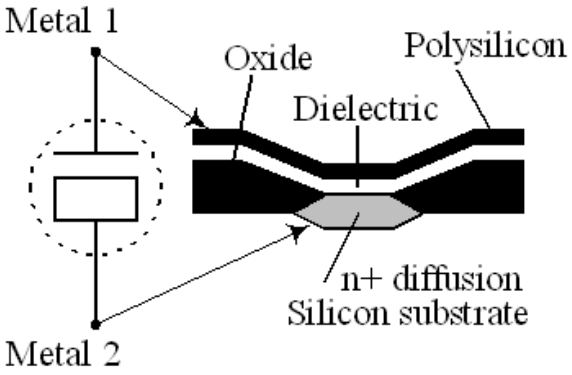
➔ reduce delays in the routing.

Drawback: No reprogrammation possible

2.3 FPGA Programming Technologies

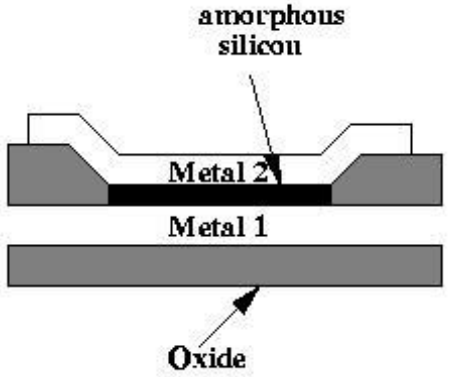
Poly-diffusion Anti-fuse: ACTEL PLICE

- programmable low-impedance circuit element
- Poly-silicon terminal
- Oxide-Nitride-Oxide dielectric
- Melting the dielectric establish connection



Metal Anti-fuse: Q-Logic Vialink

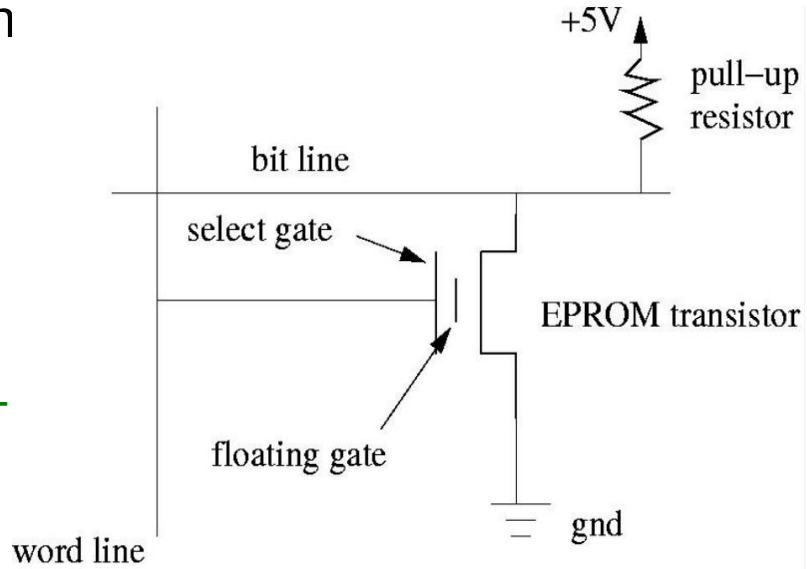
- 2 Metal terminal layers (Titanium-Tungsten)
- Programming points isolated by amorphous Silicon film



2.3 FPGA Programming Technologies

EEPROM (Flash)

- The same technology as that used in
- EPROM and EEPROM memories.
- Advantages:
  - EPROMs require re-programmable but do not require external storage.
  - EEPROM can be re-programmed in-circuit.
- Drawbacks:
  - EPROM's resistors consume static power.
  - EEPROM requires more chip area and multiple voltage sources.

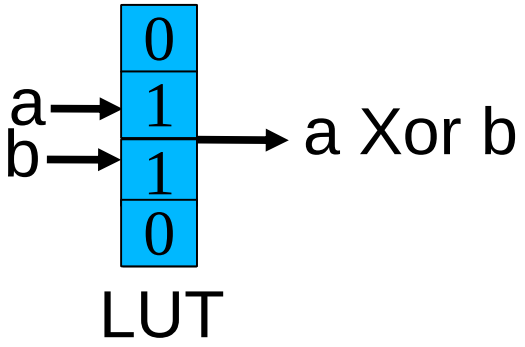


2.3 FPGA Function generators

LUT

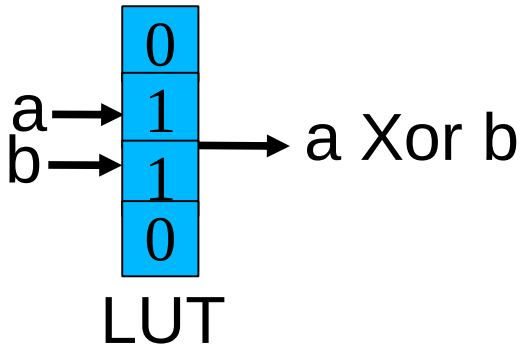
- LUT are used as function generators in SRAM-based FPGA
- A k-inputs LUT can implement up to 2^k different functions
- A k-input LUT has 2^k SRAM locations
- A function is implemented by writing all possible values that the function can take in the LUT
- The inputs values are used to address the LUT and retrieve the value of the function corresponding the the input values

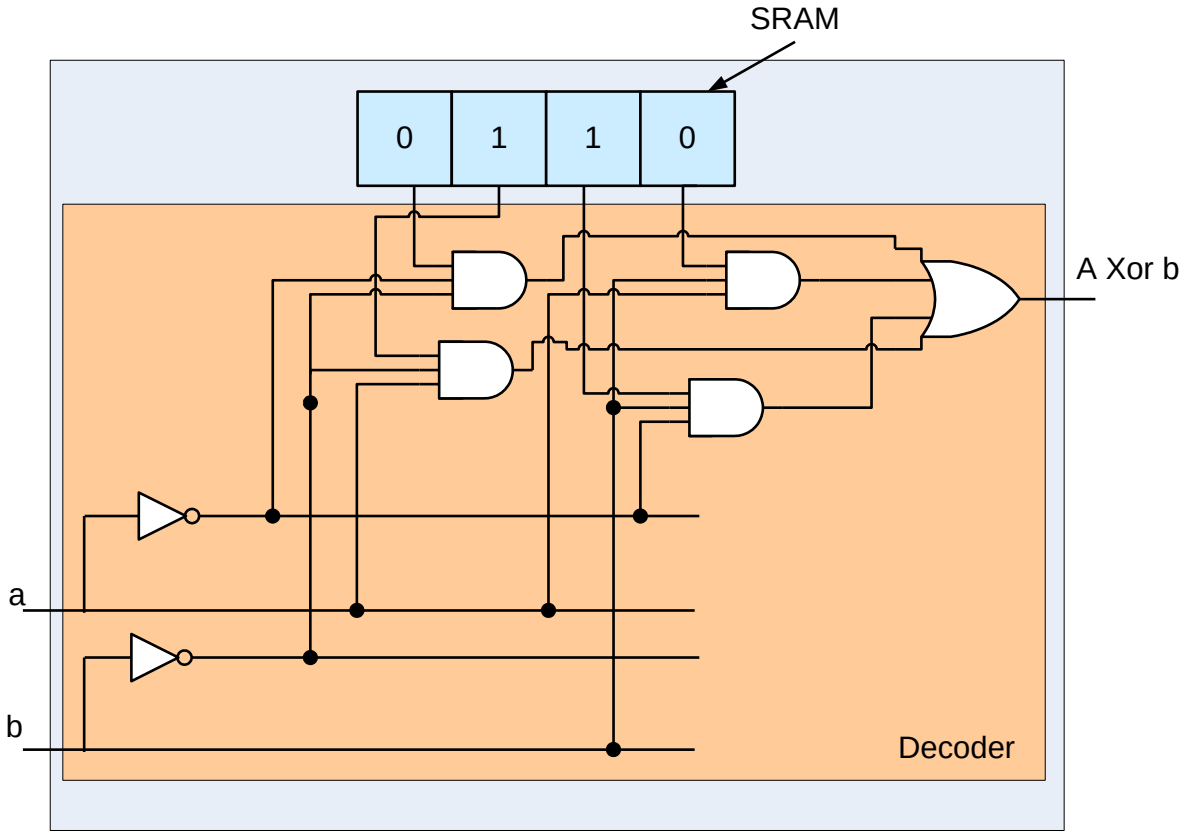
a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0



2.3 FPGA Function generators

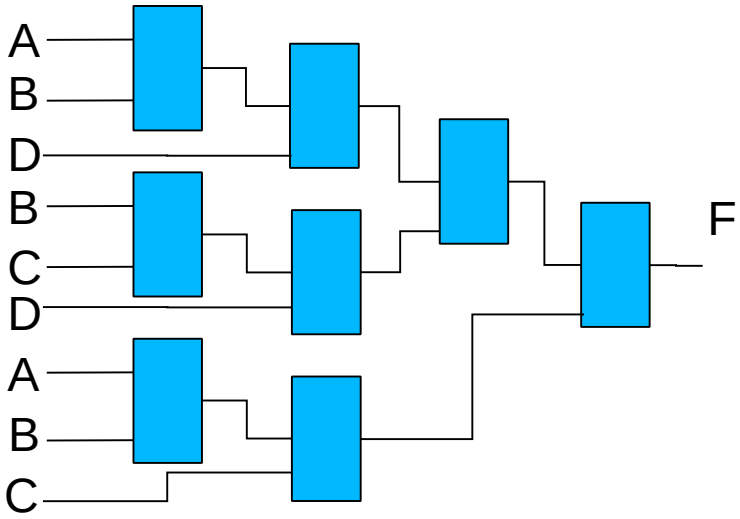
a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

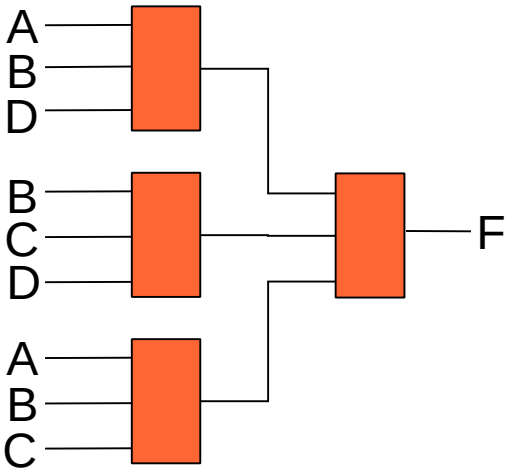


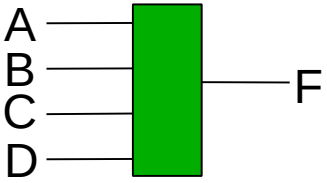


2.3 FPGA Function generators

- LUT Example: Implement the function $F = ABD + BC\overline{D} + \overline{A}\,\overline{B}\,\overline{C}$
- using:
  - 2-input LUTs
  - 3-input LUTs
  - 4-input LUTs



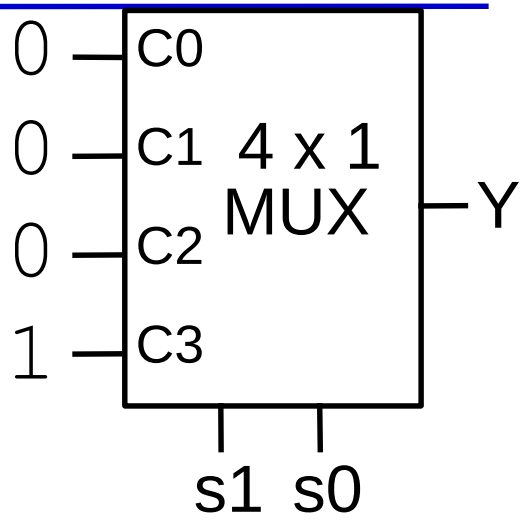




2.3 FPGA Function generators

Multiplexers (MUX)

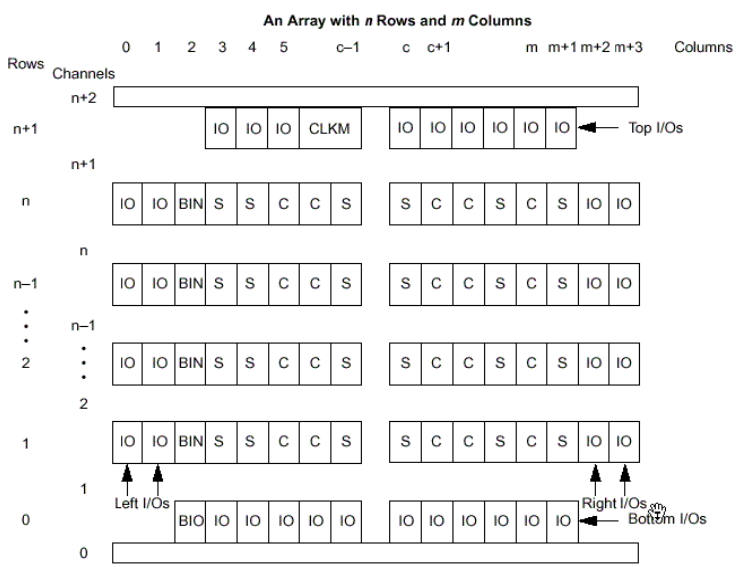
- A 2^kx1 MUX can implement up to 2^k different functions
- A function is implemented by writing all possible values that the function can take as constant at the MUX-Inputs
- The selector-values are used to pass the corresponding input to the MUX output
- Complex function can be decomposed and implement using many MUXes using the Shannon expansion theorem

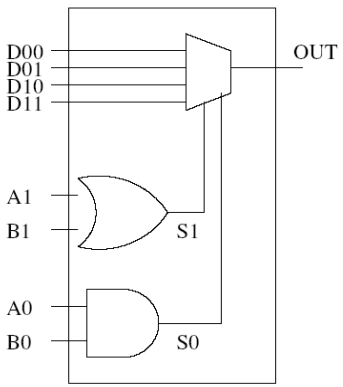


s1	s0	Y =AND	
0	0	C0	0
0	1	C1	0
1	0	C2	0
1	1	C3	1

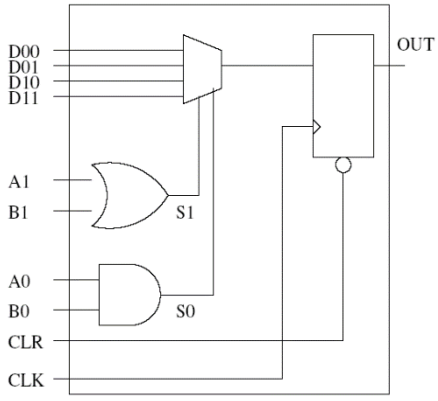
2.3 The Actel ACT3 Family (row-based)

- Row-based FPGA
  - Modules rows separated by routing channels
- MUX-based macro-cells
  - C-Module
    - 4:1 MUX + 1 OR + 1 AND
  - S-Module
    - 4:1 MUX + 1 OR + 1 AND
    - 1 Flip Flop
- IO placed aside the device





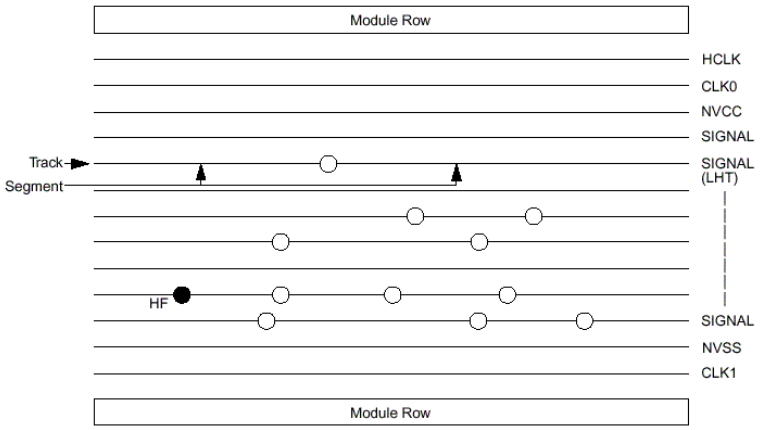
C-Module

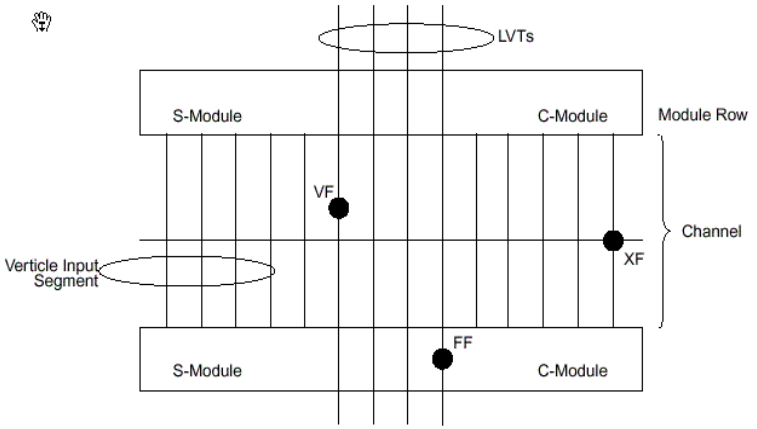


S-Module (ACT 3)

2.3 The Actel ACT3 Family (row-based)

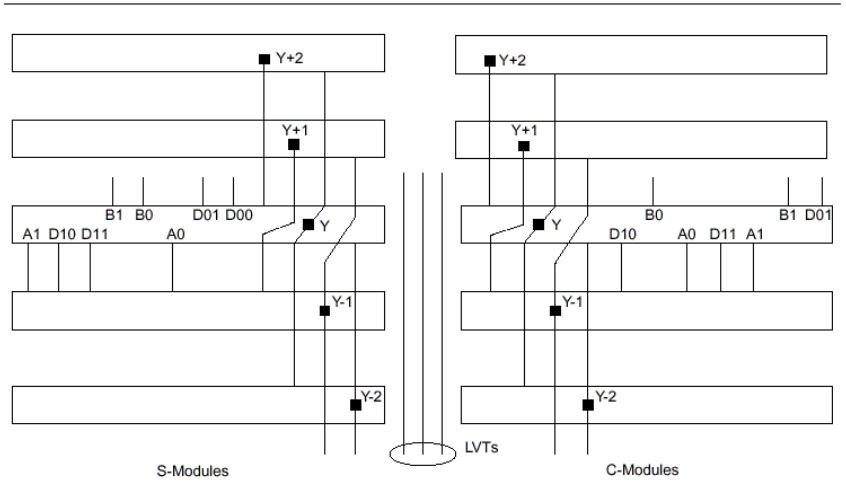
- Channels are composed of several segmented routing tracks
  - Minimum length = module pair width
  - Maximum length = row width
  - Long segment if segment width > 3
  - Connections are anti-fuse based
    - Horizontal-to-vertical (XF)
    - Horizontal-to-horizontal (HF)
    - Vetical-to-vertical (VF)
    - Fast vertical connection (FF)
- Tracks for module inputs are segmented by pass transistor (inactive during normal operation)
- Vertical inputs span the channels above and below





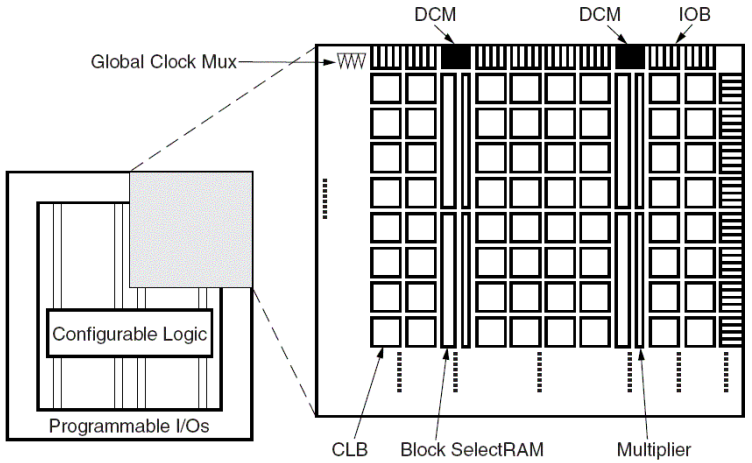
2.3 The Actel ACT3 Family (row-based)

1. Module outputs have dedicated channels which extend vertically two channels above and two channels below, except at the bottom and the top



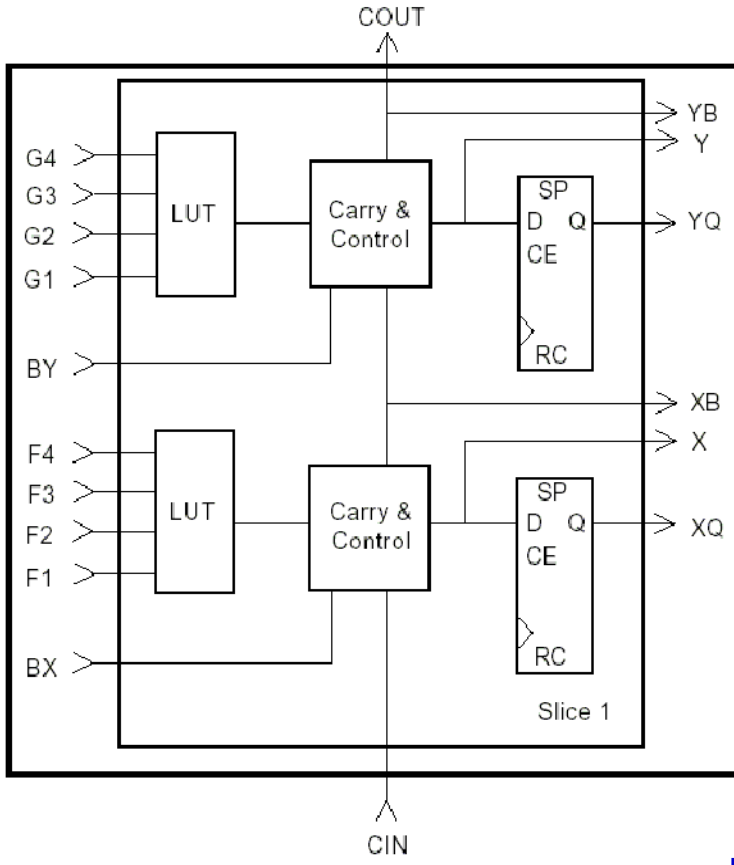
2.3 The Xilinx Virtex Family (symmetrical array)

- Symmetrical-array Based FPGA
  - Macro cells are configurable logic block (CLBs), placed on line column intersection.
  - Additional modules exist:
    - Block RAM for internal use
    - Digital clock manager (DCM) for user specific clock frequency generation)
    - Embedded multiplier (Virtex II or newer Virtex series)
    - Global clock Multiplexers
    - Input output block (IOB) for off-chip communication



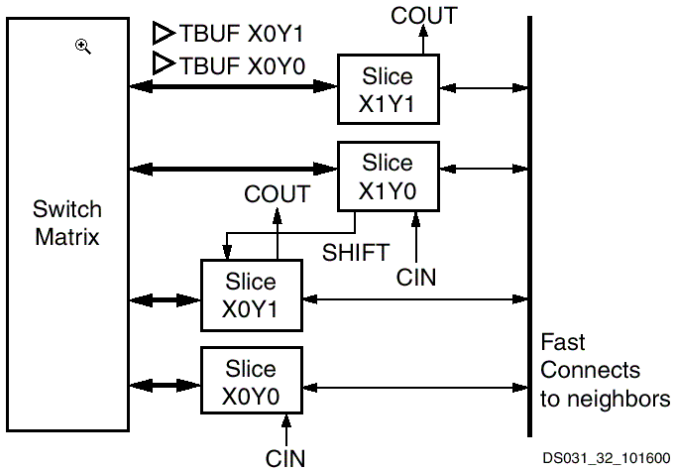
2.3 The Xilinx Virtex Family (symmetrical array)

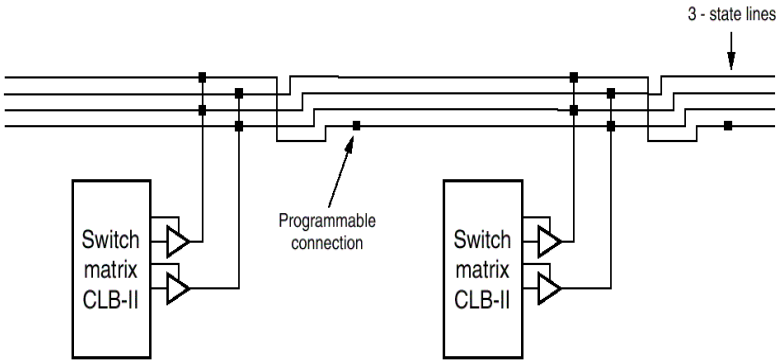
- Macro cells are CLBs. A CLB contains 4 identical slices on virtexII and newer and 2 slices on Virtex and Virtex E
- 4 slices split in two columns of 2 slices each
- 1 slice contains:
  - 2 4-inputs LUT
  - 2 FF for storing LUT results
  - MUX to feed LUT either to a FF or the the output
  - Carry in and carry out help to construct fast adder circuits using neighbour CLBs



2.3 The Xilinx Virtex Family (symmetrical array)

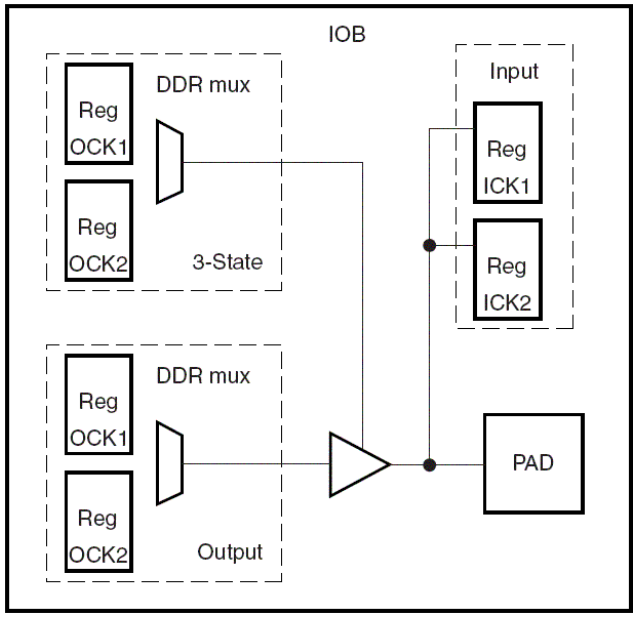
- A CLBs access the general routing matrix via a switch matrix
- Fast connection lines are used for local connections
- A switch matrix connects CLB terminal on the routing reource using multiplexers
- 4 horizontal resource per CLB for on-chip tri-state busses
- Each CLB have two tri-state driver (TBUF) that can drive on chip busses
- Each TBUF has its own control pin and its own input pin
- TBUF are AND-OR based, i.e timing is more predictable.





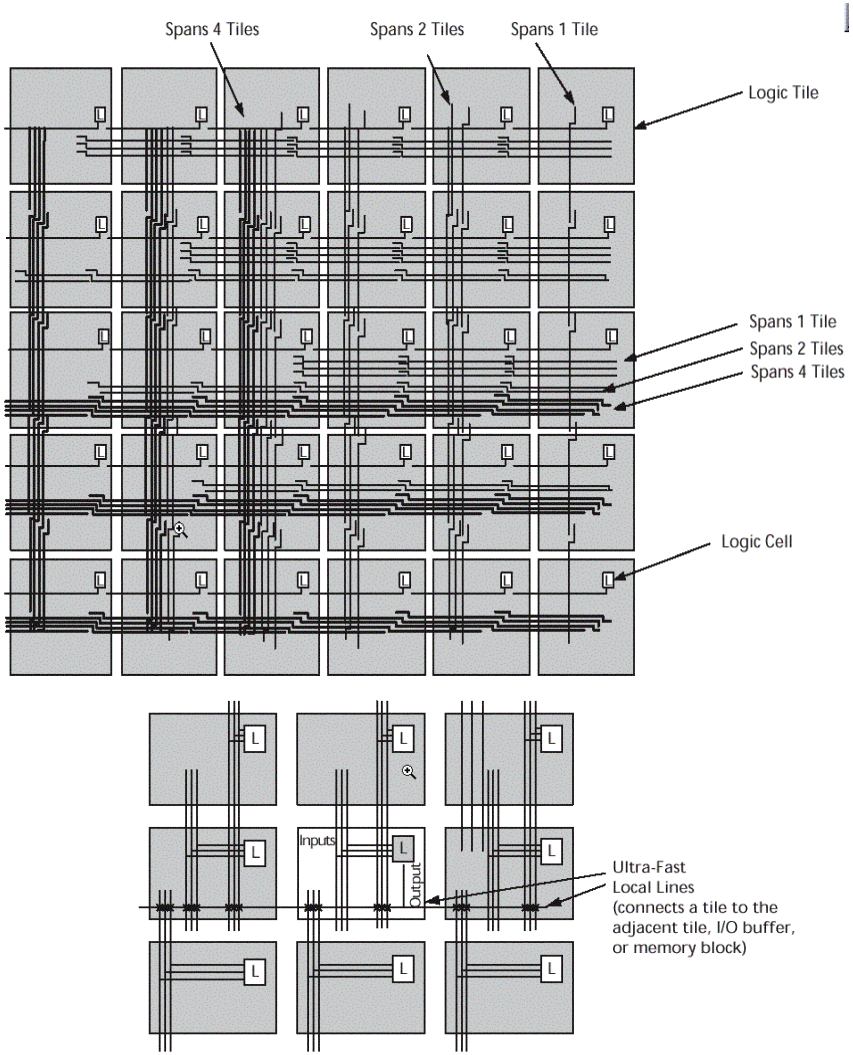
2.3 The Xilinx Virtex Family (symmetrical array)

- IOB for off-chip communication
  - Programmability allows the use of an IOB by any CLB.
  - Connection can be input, output or bidirectional.
  - 6 IOB latched for double data rate (DDR) transmission.
  - One of the DDR register can be used on input, output or tri-state.
  - DDR accomplished by the two register on each path clocked by rising or falling edge from different clock nets.
  - The two Clock signals are generated by the DCM.



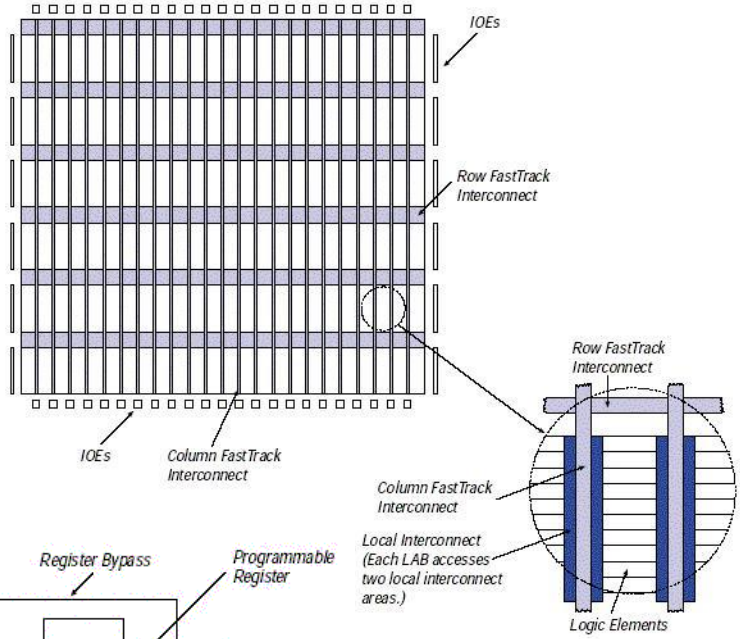
2.3 The Actel ProAsic Family (sea-of-gates)

- Sea-of-gates style (sea-of-tiles)
  - Macro cells are EEPROM based tiles
  - Four level of hierarchy routing resource.
    - Local resource connects a tile to one of its 8 neighbours
    - Long-lines resource provides routing for long distance and high fan-out (spanns 1, 2 or 4 tiles). Runs both horizontal and vertical
    - Very long-line resource spans the entire device
    - Global network (clocks, reset)
  - Connection via anti-fused



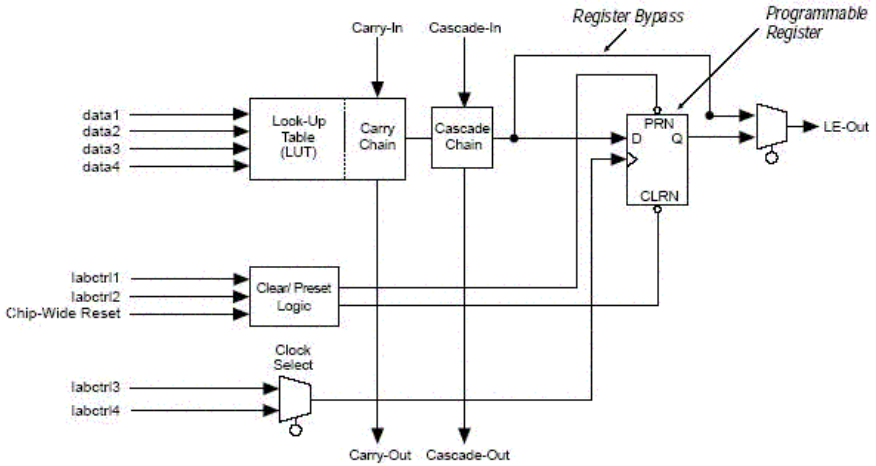
2.3 The Altera Flex family (hierarchical)

- Hierarchical-based FPGA
- Logic elements (LE) are grouped in Logic array block (LAB), on the higher level
  - 8 LE / LAB for the FLEX8000
  - 10 LE / LAB for the FLEX8000
- LAB arranged as array on the device



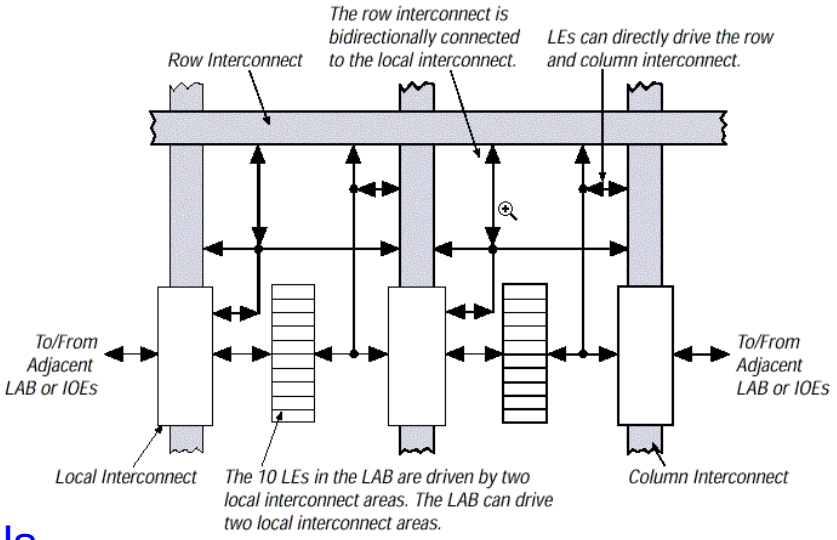
An LE contains:

- 1 4-input LUT
- 1 FF
- Carry-in, carry-out
- MUX
- Additional logic



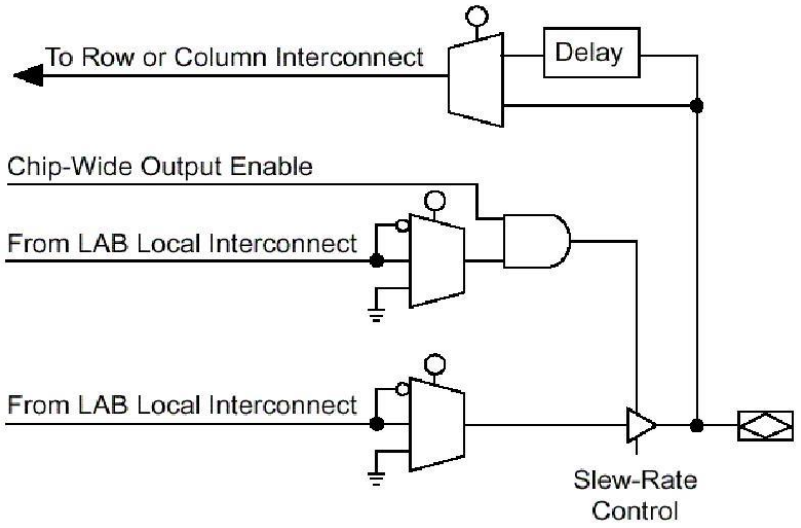
2.3 The Altera Flex family (hierarchical)

- FastTrack interconnect provides on-chip routing resource
- Connections among LEs and adjacent LABs via local interconnect signals
- Connection inside each row of LAB is done by a dedicated row interconnect
- Each column of LAB is served by a dedicated column interconnect.
- LEs can drive the row or column channels
- Column interconnect can drive row interconnect.
- A signal from the column interconnect must be routed to the row interconnect before entering an LAB
- LEs can drive global signals (Clocks, reset, asynchronous clear, high fanout, etc...)



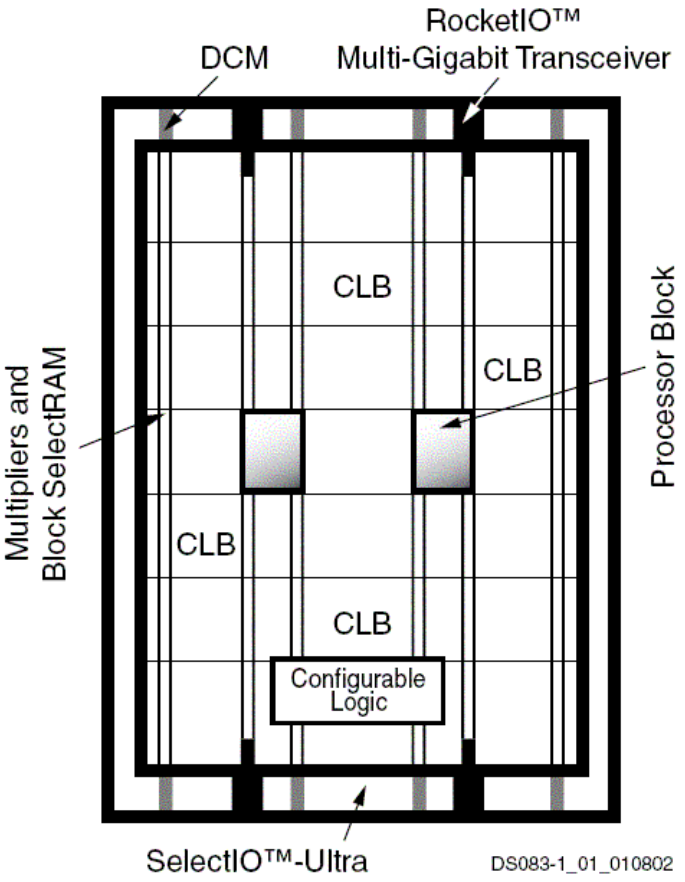
2.3 The Altera Flex family (hierarchical)

- Programmable IO Element (IOE) allows on-chip and off-chip programmable communication
- An IOE can be programmed as input, output or bidirectional.
- IOE receives data from adjacent interconnect (can be driven by row or column interconnect)
- IOE receives its chip enable (ce) from an adjacent LE.
- One pin per output element (OE) -> possible open drain emulation



2.3 Hybrid FPGAs

- The Xilinx VirtexII-Pro
- Basic structure: VirtexII
- Additional features
  - Up to 4 hard-core embedded IBM power pc 405 RISC processors with 300+ Mhz
  - Advanced 18bit x 18bit embedded multipliers
  - Dual-ported RAM
  - Embedded high speed serial RocketIO multi-gigabit transceivers



2.3 Hybrid FPGAs

- The Altera Excalibur
- Specific features:
  - One ARM922T 32-bits RISC processor with 200 Mhz
  - Embedded multipliers
  - Internal single and dual-ported RAM and SDRAM controller
  - Expansion bus interface for flash-RAM connection
  - Embedded SignalTap logic analyzer