# VLSI Design Automation
# Lecture-2

*By*

*Dr. Swagata Mandal*

*Assistant Professor, Electronics and Communication*

*Jalpaiguri Government Engineering College*

# Function generators

- Reconfigurable hardware device should provide the users with the possibility to dynamically implement and re-implement new functions.
- This is usually done by function generator
- Two types of function generators are available in commercial FPGA: Multiplexers and Look-up table
- A $2^n$: 1 MUX has $n$ selector lines and one output.
- A MUX can implement any function

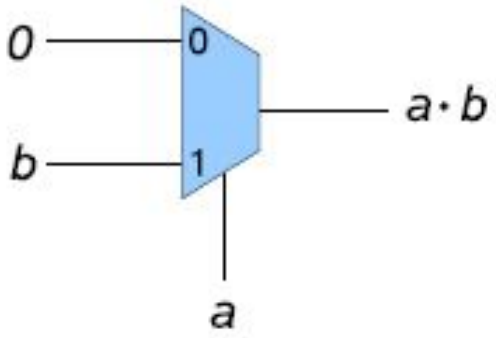



Fig: Implementation of f = axb in a 2-input MUX

- The argument b is used as input in combination with a second input 0 and the second argument a is used as selector.

# Shannon expansion theorem

- The Shannon expansion theorem can be used to decompose a function and implement it into a MUX. This theorem states that a given Boolean logic function $\sum x_1, \ldots x_n$ of *n* variables can be written as shown in the following equation.

$$F(x_1, \cdots, x_n) = F(x_1, \cdots, x_i = 1, \cdots, x_n) \times x_i + F(x_1, \cdots, x_i = 0, \cdots, x_n) \times \overline{x}_i$$

- $F(x_1, \cdots, x_i = 1, \cdots, x_n) = F_1$ is the function obtained by replacing $x_i$ with one and $F(x_1, \cdots, x_i = 0, \cdots, x_n) = F_2$ the function obtained by replacing $x_i$ by zero in *F*. $F_1$ and $F_2$ are called cofactors.

- A 2:1 MUX with input $I_0$ and $I_1$, output *O* and selector *S* is defined by the equation: $O(S, I_0, I_1) = \bar{S}(I_0) + S(I_1)$. The function *F* can be implemented on a *2:1* MUX with inputs $I_0 = F_0$, $I_1 = F_1$, and selector $S = x_i$.

- Any complex function can be implemented by broken down it into small pieces which are implemented separately using MUX and finally they will e interconnected. This process is known as technology mapping.

# Shannon Theorem (cont'd)

- In order to implement a function on a $2^n$: 1 MUX, function should have $n$ fixed variables and $2^n$ cofactors. For $n = 2$

$$O(S_0, S_1, I_0, I_1, I_2 I_3) = \overline{S_0}\,\overline{S_1}(I_0) + \overline{S_0} S_1(I_1) + S_0 \overline{S_1}(I_2) + S_0 S_1(I_3)$$

- In general Shannon expansion theorem can be written as

$$\begin{aligned}
F(x_1,\cdots,x_n) &= F(x_1,\cdots,x_i=1,\cdots,x_n)\cdot x_i + F(x_1,\cdots,x_i=0,\cdots,x_n)\cdot \overline{x}_i \\
&= [F(x_1,\cdots,x_i=1,\cdots,x_j=1,..x_n)\cdot x_i + F(x_1,\cdots,x_i=0,\cdots,x_j=1,\cdots,x_n)\cdot \overline{x_i}]x_j \\
&+ [F(x_1,\cdots,x_i=1,\cdots,x_j=0,\cdots,x_n)\cdot x_i + F(x_1,\cdots,x_i=0,\cdots,x_j=0,\cdots,x_n)\cdot \overline{x_i}]\overline{x}_j \\
&= F(x_1,\cdots,x_i=1,\cdots,x_j=1,\cdots,x_n)\cdot x_i x_j + F(x_1,\cdots,x_i=0,\cdots,x_j=1,\cdots,x_n)\cdot \overline{x}_i x_j \\
&+ F(x_1,\cdots,x_i=1,\cdots,x_j=0,\cdots,x_n)\cdot x_i \overline{x}_j + F(x_1,\cdots,x_i=0,\cdots,x_j=0,\cdots,x_n)\cdot \overline{x}_i \overline{x}_j
\end{aligned}$$

$$\begin{aligned}
I_0 &= F_0 = F(x_1,\cdots,x_i=0,\cdots,x_j=0,\cdots,x_n), \\
I_1 &= F_1 = F(x_1,\cdots,x_i=0,\cdots,x_j=1,\cdots,x_n), \\
I_2 &= F_2 = F(x_1,\cdots,x_i=1,\cdots,x_j=0,\cdots,x_n), \\
I_3 &= F_3 = F(x_1,\cdots,x_i=1,\cdots,x_j=1,\cdots,x_n)
\end{aligned}$$

# Design of Full Adder using MUX

- Let $a_i, b_i$ be the operand, $c_{i-1}$ be the carry on a previous level, $s_i$ is the sum $c_i$ be the carry for the next level. Then truth table can be written as shown in Fig 1

| $a_i$ | $b_i$ | $c_{i-1}$ | $s_i$ | $c_i$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Fig 1: Truth table of the Full adder

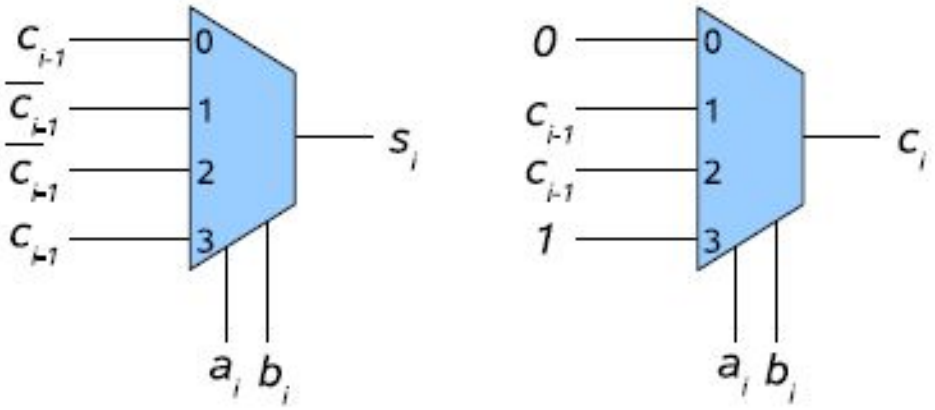



Fig 2: Implementation of a Full adder using two 4-input one output MUX

$$s_i = \bar{a}_i \bar{b}_i c_{i-1} + \bar{a}_i b_i \overline{c_{i-1}} + a_i \bar{b}_i \overline{c_{i-1}} + a_i b_i c_{i-1}$$

$$c_i = \bar{a}_i \bar{b}_i \times 0 + \bar{a}_i b_i c_{i-1} + a_i \bar{b}_i c_{i-1} + a_i b_i \times 1$$

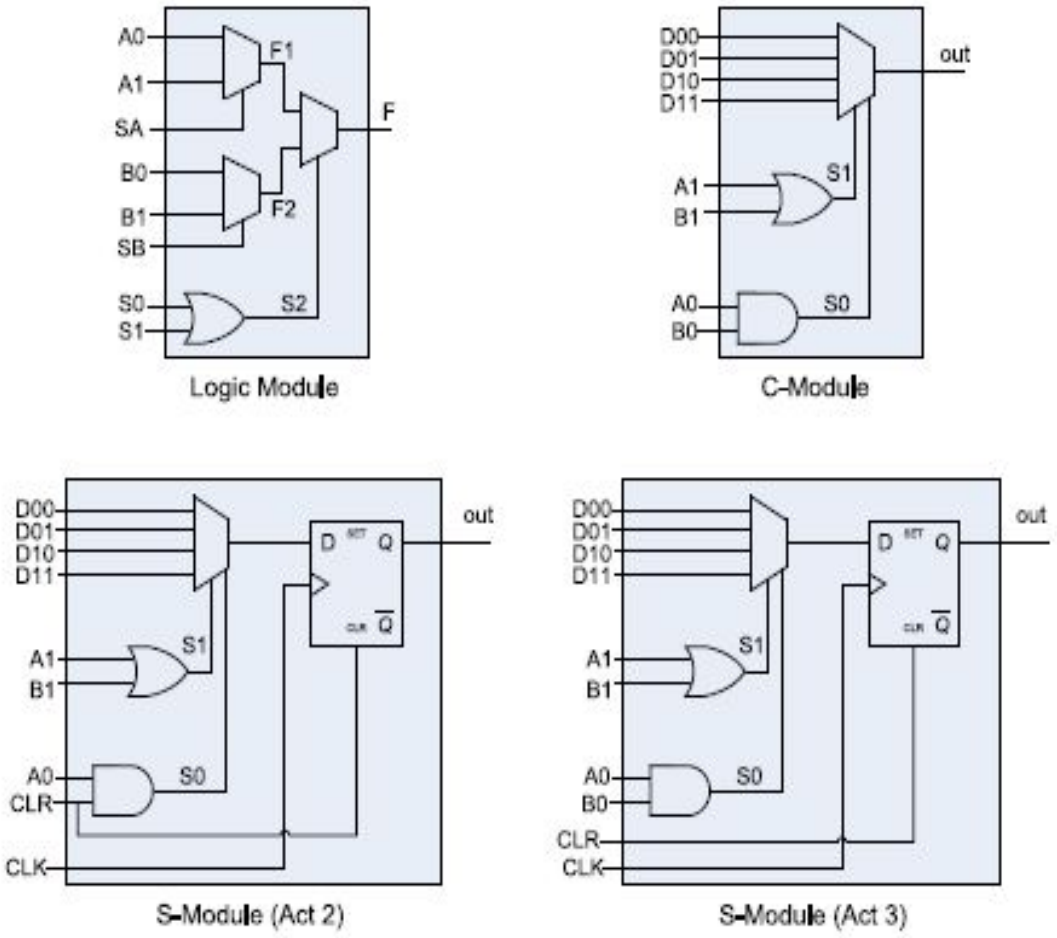
# Actel ACT X Logic module

- Basic component in Actel ACT1 device is the *logic module*. It contains 8:1 MUX which helps to implement various combinational functions and D-latches.

- The *C-modules* present in the second generation of Actel devices, the ACT2, are similar to the logic module.

- The *S-modules*, which are found in the second and third generation of actel devices, contain an additional dedicated flip-flop.

- This avoid the building of flip flop from the combinatorial logic as it is the case in the logic module.




The Actel basic computing blocks uses multiplexers as function generators

# Look-Up Table

- A *Look-Up Tables (LUT)* is a group of memory cells, which contain all the possible results of a given function for a given set of input values.
- Usually, the set of possible function values corresponds to the possible combinations of the inputs.
- The values of the function are stored in such a way that they can be retrieved by the corresponding input values.
- A n-input LUT can be use to implement up to $2^{2^n}$ different functions, each of which can take $2^n$ possible values.
- An *n*-input LUT must provide $2^n$ cells for storing the possible values of an *n*-input function.
- In FPGAs, a LUT physically consists of a set of SRAM-cells to store the values a decoder that is used to access the correct SRAM location and retrieve the result of the function, which corresponds to the input combination.
- In order to implement a complex function in a LUT-based FPGA, the function must be divided into small pieces, each of which can be implemented in a single LUT. The interconnections are used to connect small pieces together and form the complete function

# LUT based Design

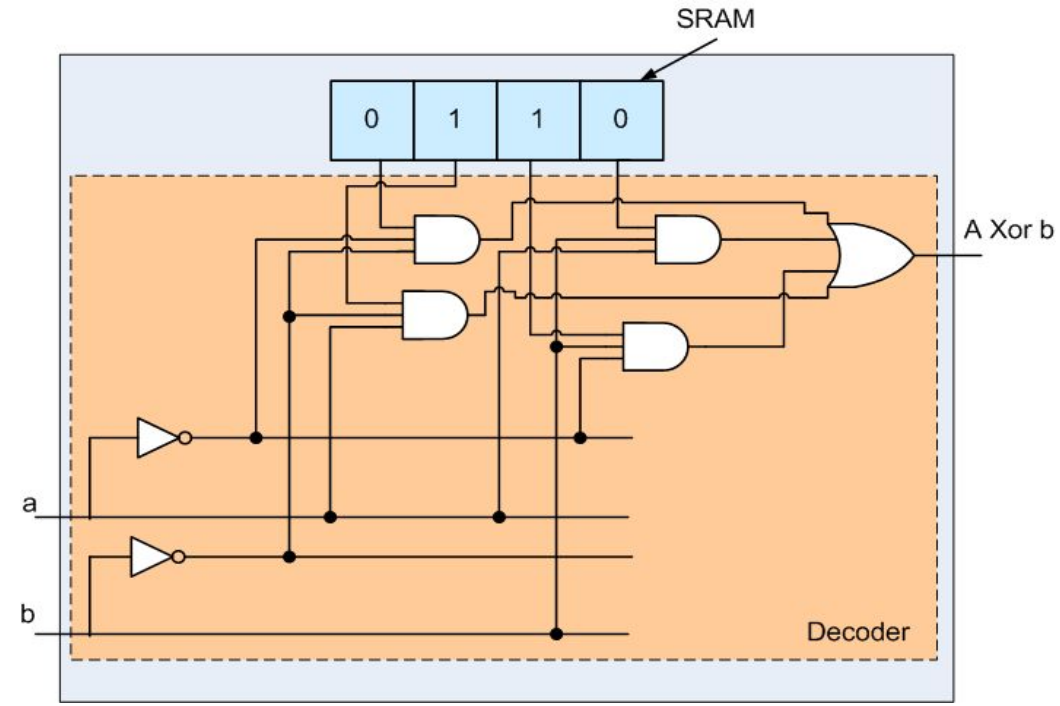



Fig1: 2-input LUT

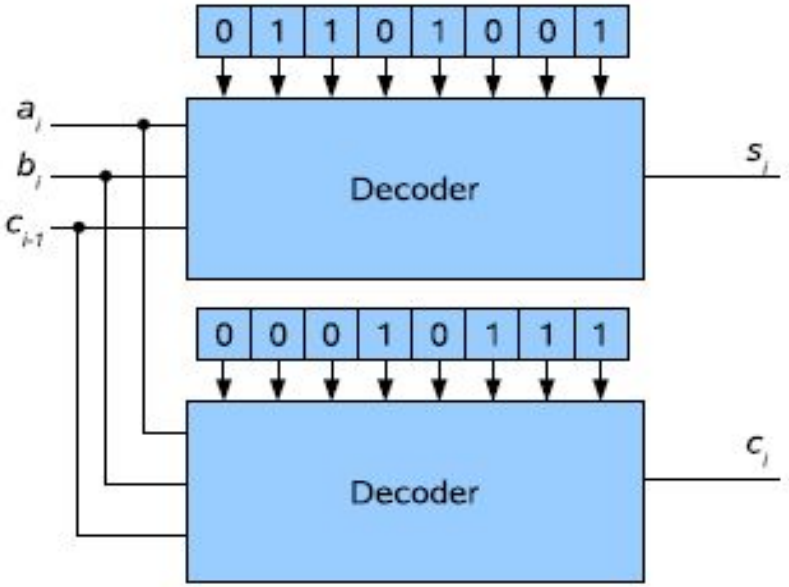
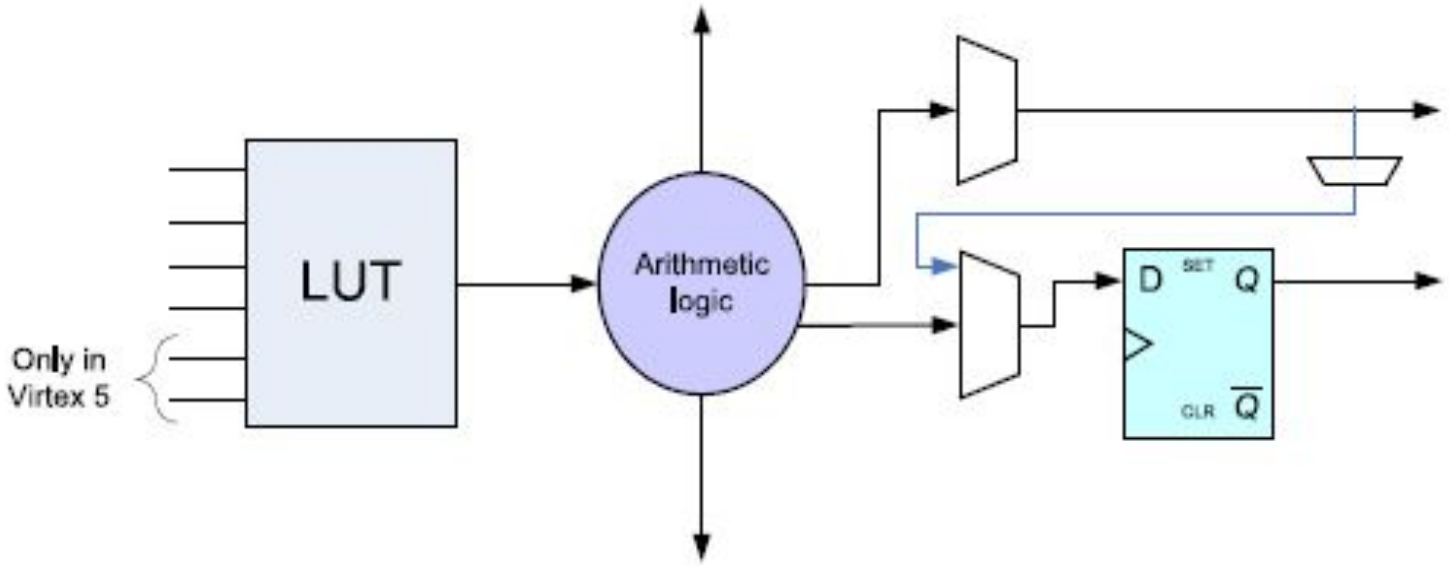



Fig2: Implementation of a full adder in two 3-input LUTs

- SRAM-based LUT are used in the most commercial FPGAs as function generators.
- Several LUTs are usually grouped in a large module in which other functional elements like flip flops and multiplexers are available.
- The connection between the LUTs inside such modules is faster than connections via the routing network, because dedicated wires are used.

# Xilinx CLB




Basic block of the Xilinx FPGAs

- The basic computing block in the Xilinx FPGAs consists of a LUT with variable number of inputs, a set of multiplexers, arithmetic logic and a storage element.
- The LUT is used to store the configuration while the multiplexers selects the right inputs for the LUT and the storage element as well as the right output of the block.
- The arithmetic logic provides some facilities like XOR-gate and faster carry chain to build faster adder without wasting too much LUT-resources.

# Xilinx CLB(cont'd)

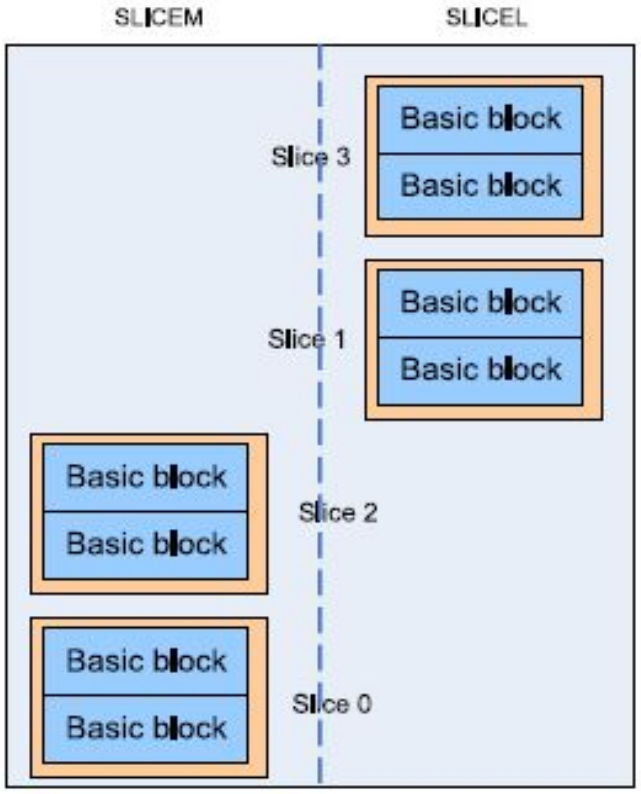



Fig 1: CLB in the Spartan, Virtex-II and Virtex-4

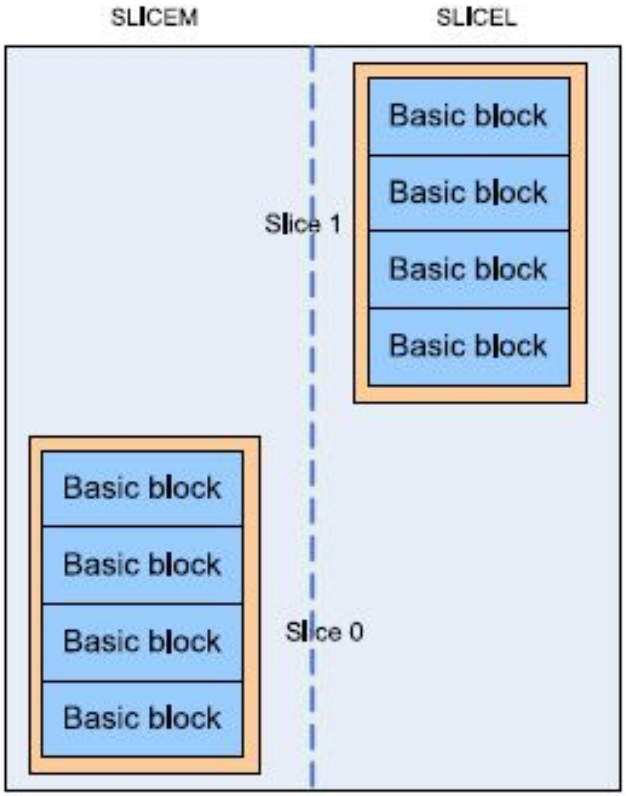



Fig 2: CLB in Virtex-5

Several basic computing blocks are grouped in a coarse-grained element called the *Configurable Logic Block (CLB)*

# Xilinx CLB(cont'd)

- In the older devices like the 4000 series, the Virtex and Virtex E and the Spartan devices, two basic blocks were available in a CLB.

- In the newer devices like the Spartan 3, the Virtex-II, the Virtex II-Pro and the Virtex 4, the CLBs are divided in four slices each of which contains two basic blocks.

- In the newer devices, the left part slices of a CLB, also called SLICEM can be configure either as combinatorial logic, or can be use as 16-bit SRAM or as shift register while right-hand slices, the SLICEL, can only be configured as combinatorial logic.

- Except for the Virtex5, all LUTs in Xilinx devices have four inputs and one output.

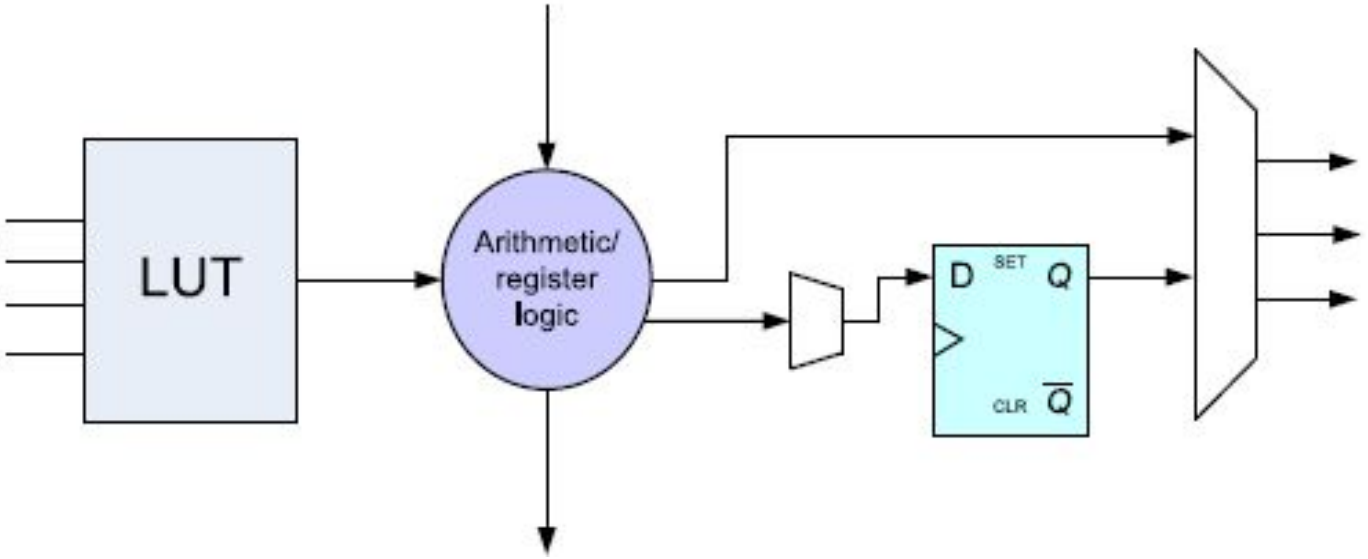
- In the Virtex 5 each LUT has six inputs and two outputs.

- The LUT can be configured either as a 6-input LUT, in which case only one output can be used, or as two 5-input LUTs, in which case each of the two outputs is used as output of a 5-input LUT.

# Altera Logic Array Block

- Altera's FPGAs (Cyclone, FLEX and Stratix) are also LUT based

- In the Cyclone II as well as in the FLEX architecture, the basic unit of logic is the *logic element* (LE) that typically contains a LUT, a filp-flop, a multiplexer and additional logic for carry chain and register chain.




Logic Element in the Cyclone II

- Architecture of Cyclone is very similar to that of ALTERA FLEX devices

# Adaptive Logic Module

- In the Stratix II devices, the basic computing unit is called *adaptive logic module* (ALM)
- The ALM is made upon a mixture of 4-input and 3-input LUTs that can be used to implement logic functions with variable number of inputs.

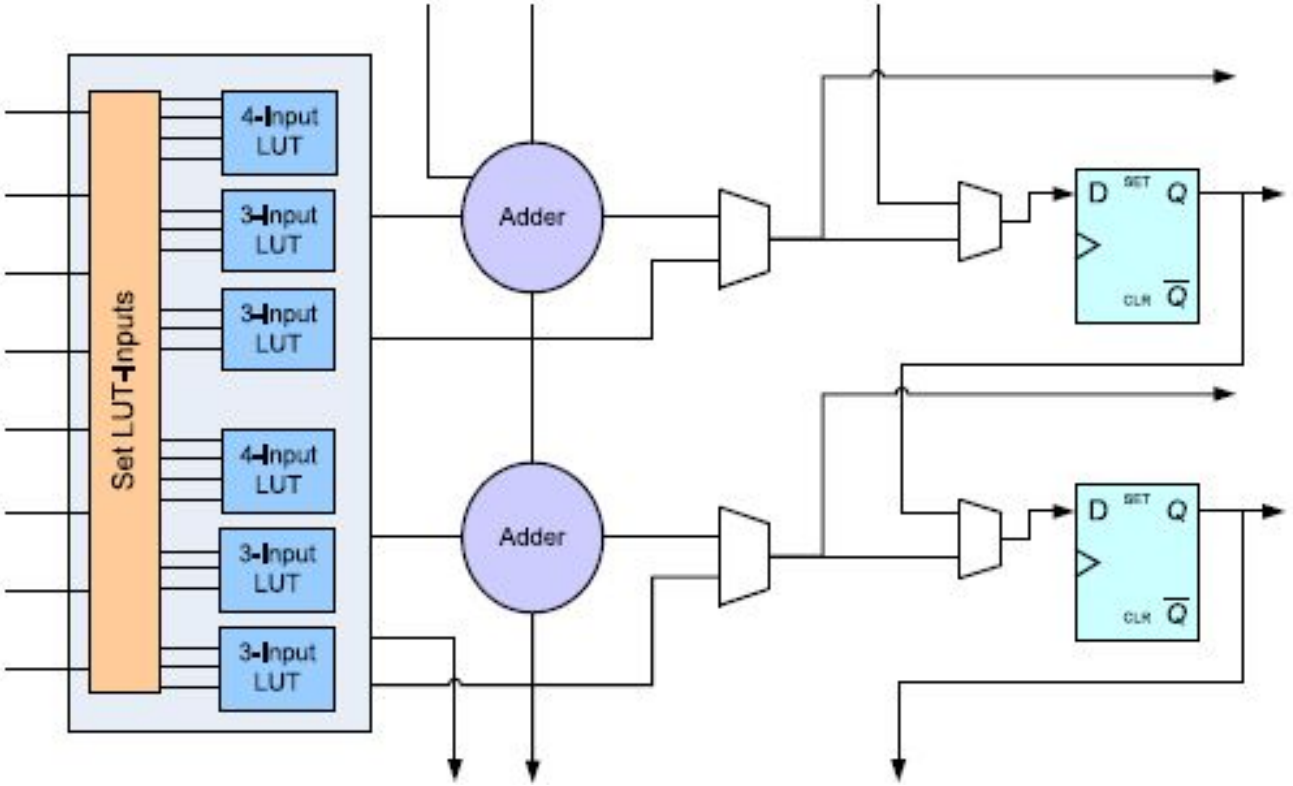



Fig1 : Stratix-II Adaptive Logic Module

# Logic Array Block

- ALM architecture insures a backward compatibility to 4-input based designs, while providing the possibility to implement coarse-grained module with variable number (up to 8) inputs.

- Additional modules including flip flops, adders and carry logic are also provided.

- Altera logic cells are grouped to form coarse-grained computing elements called *Logic Array Blocks* (LAB).

- The number of logic cells per LAB varies from the device to device.

- The Flex 6000 LABs contains ten logic elements while the FLEX 8000 LAB contains only eight.

- Sixteen LEs are available for each LAB in the cyclone II while the Stratix II LAB contains eight ALMs.

# FPGA structures

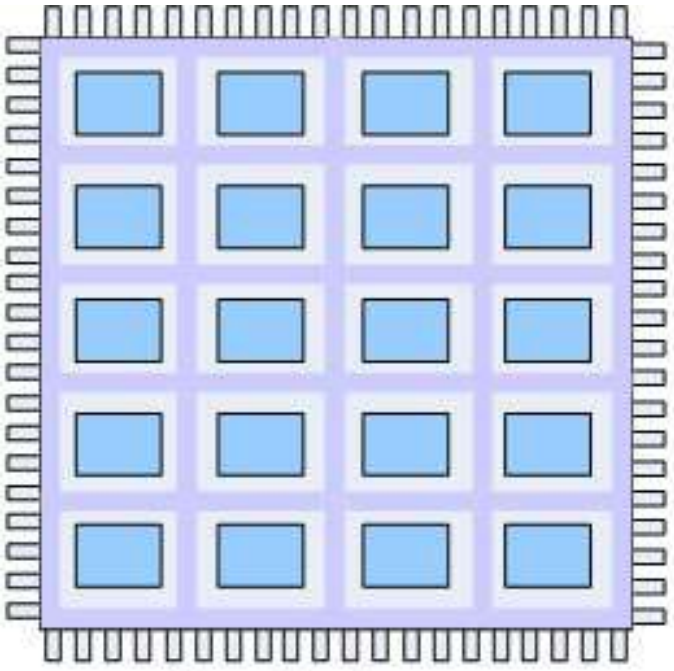
- FPGAs consist of a set of programmable logic cells placed on the device such as to build an array of computing resources.
- The resulting structure is vendor dependant.
- According to the arrangement of logic blocks and the interconnection paradigm of the logic blocks on the device, FPGAs can be classified in four categories:

  ***Symmetrical array*, *Row-based*, *Hierarchy-based* and *Sea of gates***

**Symmetrical Array:**

- A symmetrical array-based FPGA consists of a two dimensional array of logic blocks immersed in a set of vertical and horizontal lines.

- Switch elements exist at the intersections of the vertical and horizontal lines to allow for the connections of vertical and horizontal lines.



Symmetrical Array based architecture

# Example of Symmetrical Array FPGA

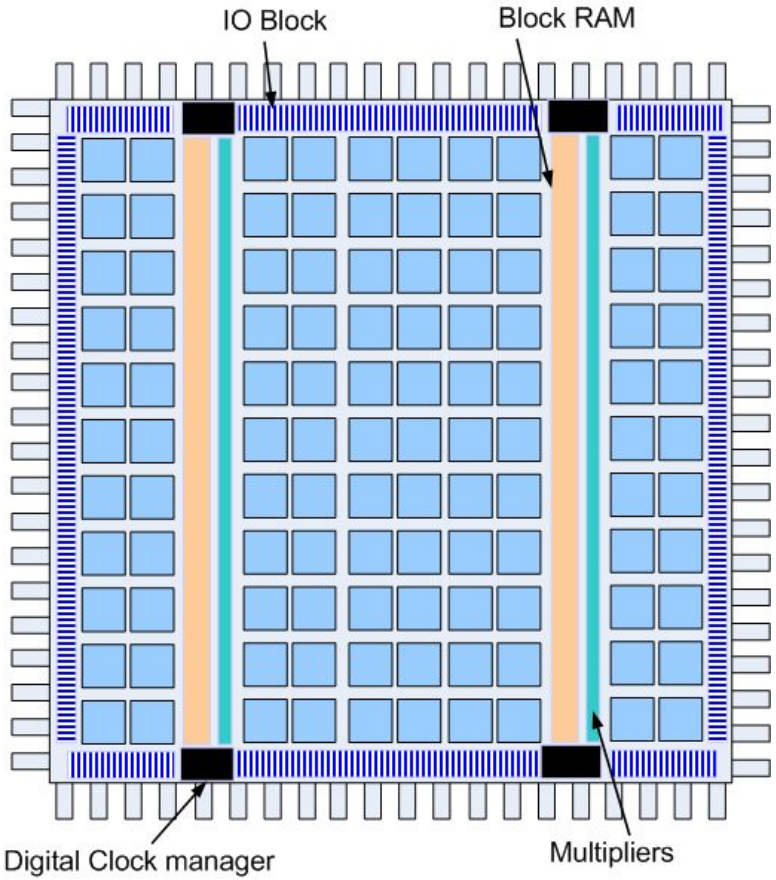



Fig 1: Xilinx Virtex II

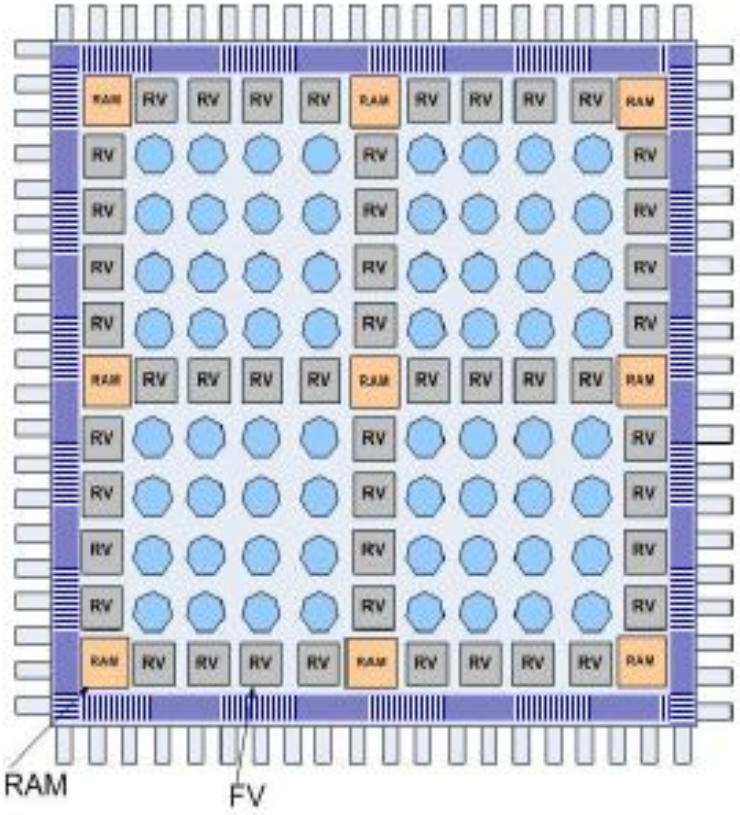



Fig 2: Atmel AT40K FPGAs

- On the Xilinx devices, CLBs are embedded in the routing structure that consists of vertical and horizontal wires.

# Symmetrical Array FPGA(cont'd)

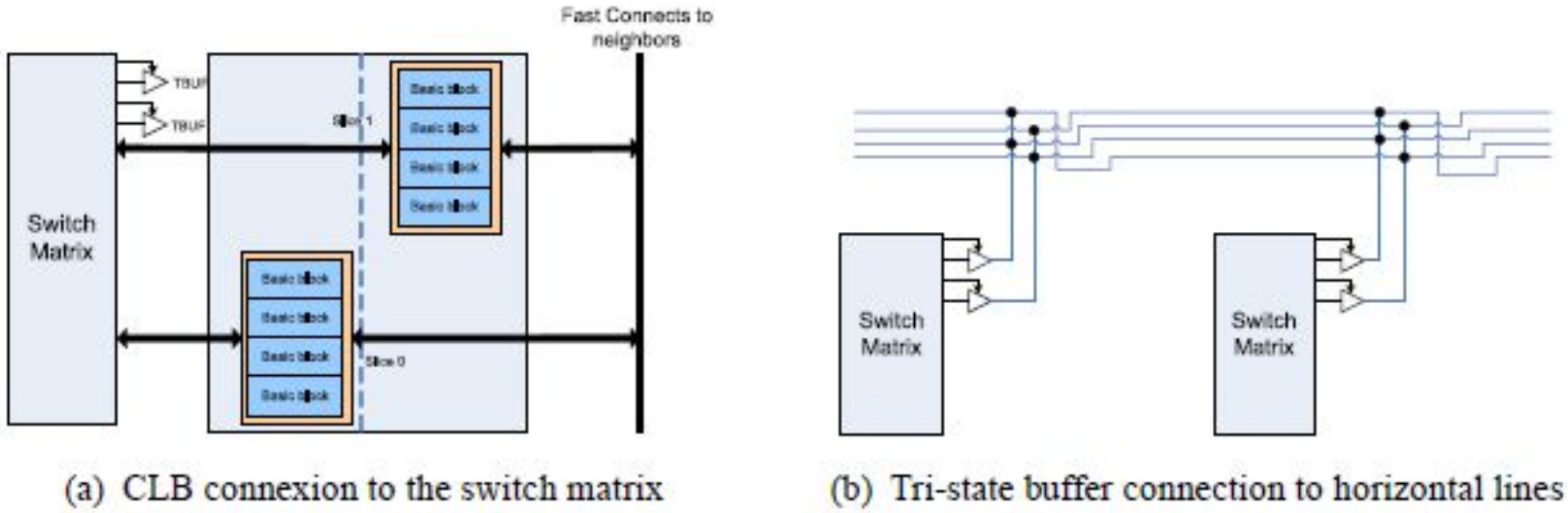



Fig 1: Virtex routing resource

- Each CLB element is tied to a switch matrix to access the general routing structure, as shown in Fig 1 (a).

- The switch matrix provides programmable multiplexers, which are used to select the signals in the given routing channel that should be connected to the CLB terminals.

- The switch matrix can also connect vertical and horizontal lines, thus making routing possible on the FPGA.

# Symmetrical Array FPGA(cont'd)

- Each CLB has access to two tri-state driver (TBUF) over the switch matrix which can be used to drive on-chip buses.
- Each tri-state buffer has its own tri-state control pin and its own input pin that are controlled by the logic built in the CLB.
- Four horizontal routing resources per CLB are provided for on chip tri-state busses.
- Each tri-state buffer has access alternately to two horizontal lines, which can be partitioned as shown in Fig 1 (b).
- Besides the switch matrix, CLBs connect to their neighbours using dedicated fast connexion tracks.

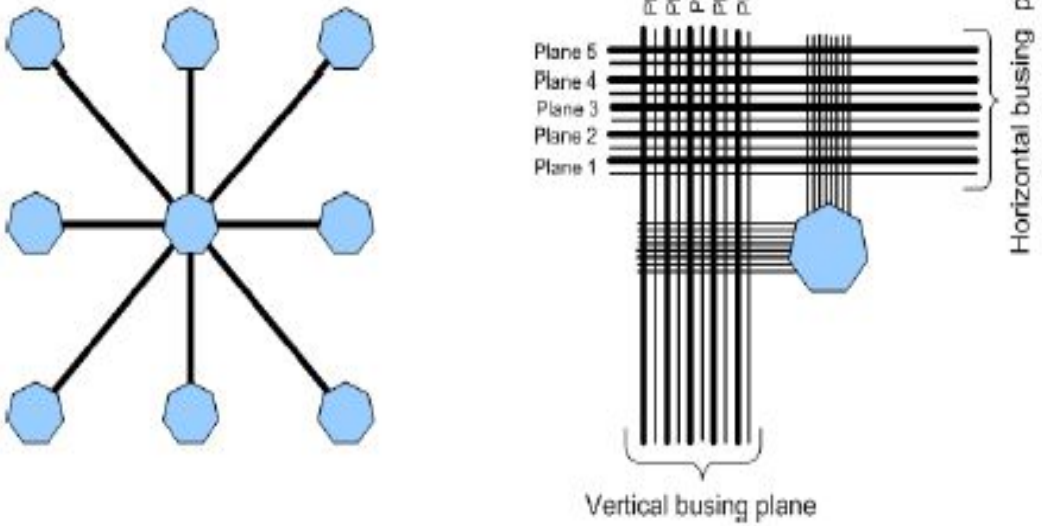



Fig: Local connection of Atmel Cell

# Row-Based FPGAs

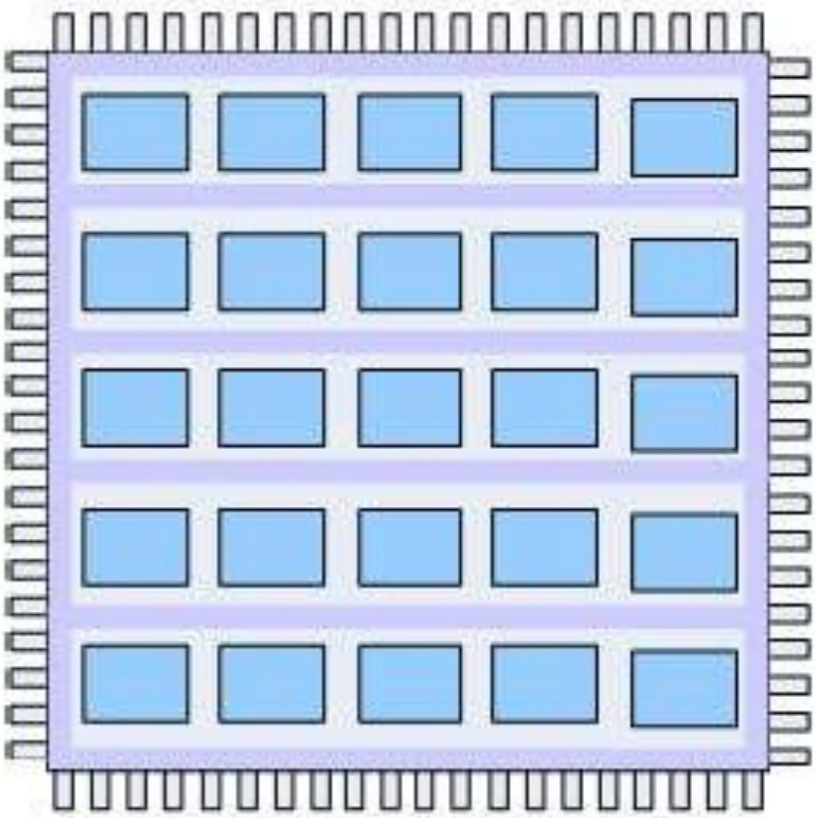


Fig1 : General Architecture of Row based FPGA

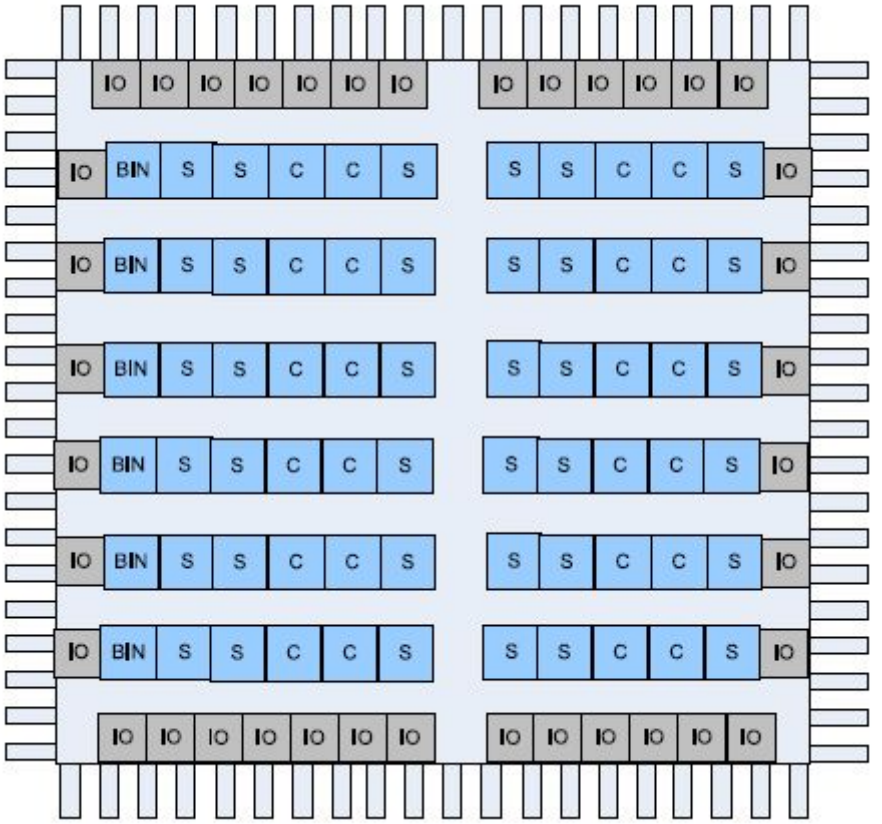



Fig 2: Row based arrangement on the Actel ACT3 FPGA Family

- A row-based FPGA consists of alternating rows of logic block or macro cells and channels as shown in Fig 1 and 2.

- The space between the logic blocks is called channel and used for signal routing.

# Row-Based FPGAs(cont'd)

- The routing is done via the in the horizontal direction using the channels.
- In the vertical direction, dedicated vertical tracks are used.

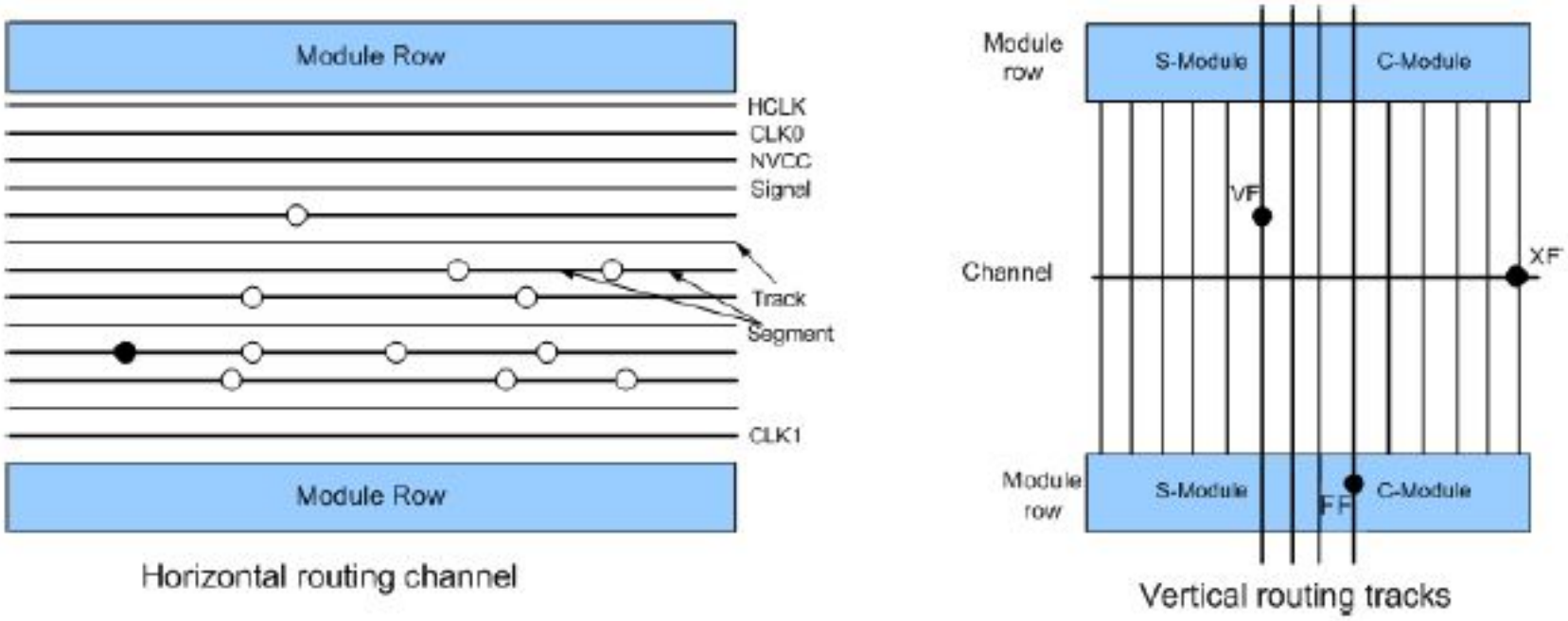
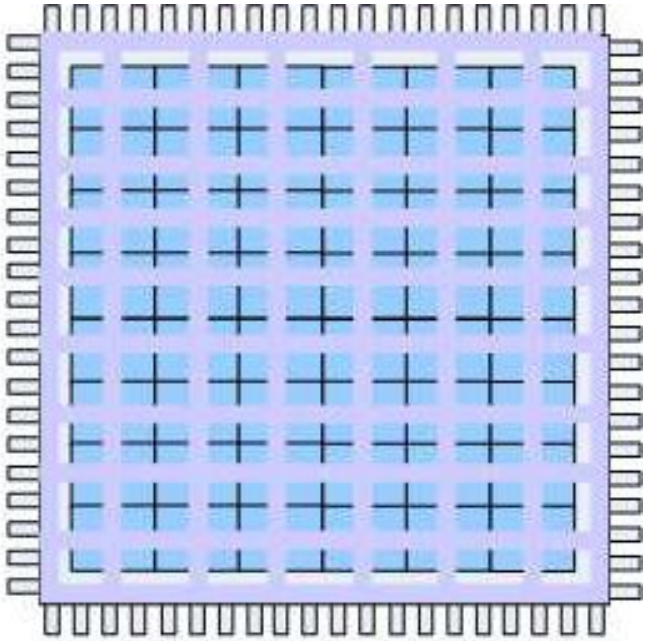


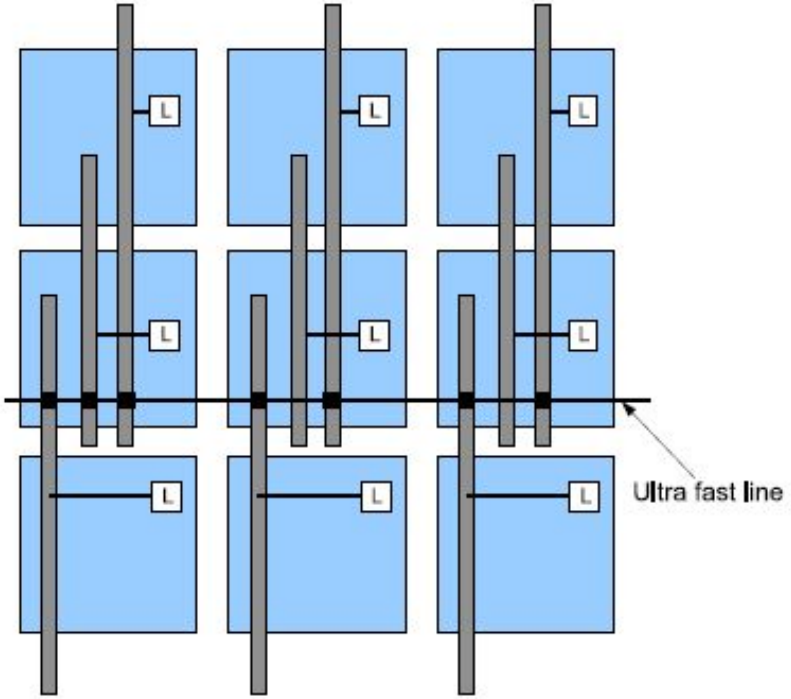

Fig: Actel's ACT3 FPGA horizontal and vertical routing resources

- As shown in Fig, a channel consists of several routing tracks divided into segments.
- The minimum length of a segment is the width of a module pair and its maximum length is the length of a complete channel, i.e. the width of the device.
- Four types of antifuse are used: XF, HF, FF, fast vertical

# Sea-of-gates



Sea of Gate




Actel ProASIC local routing resources

- Macro cells are arranged in 2-D fashion like symmetrical array but unlike symmetrical array there is no space left aside between the macro cells for routing.
- The interconnection wires are fabricated on top of the cells.
- The Actel ProASIC FPGA family is an implementation of the sea-of-gate approach.
- It's core consist of a sea-of-gates called *sea-of-tiles*

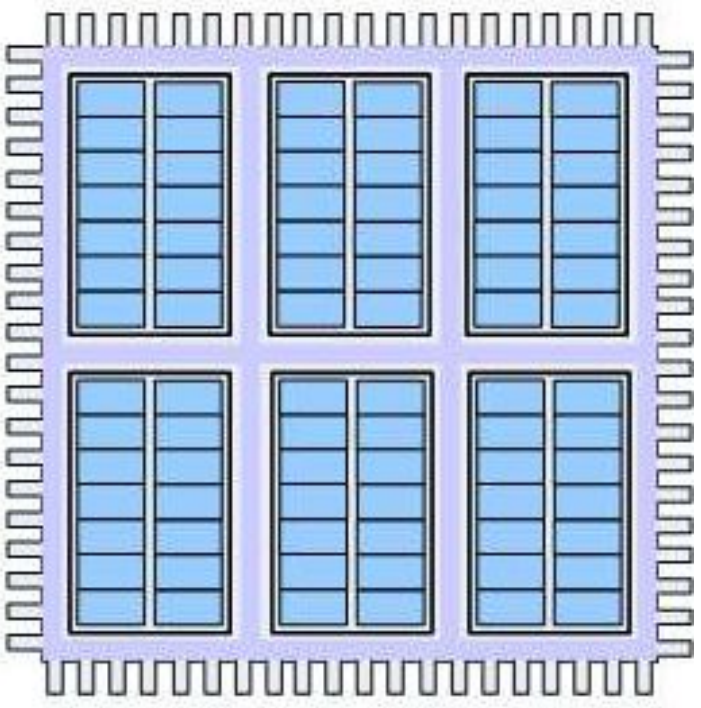
# Sea-of-gates(cont'd)

- The macro cells are the EEPROM-based tiles as seen previously.
- The device uses a four level of hierarchy routing resource to connect the logic tiles: the *local resources*, the *long-line resources*, the *very long-line* resources and the *global networks*.
- The local resources allow the output of the tile to be connected to the inputs of one of the eight surrounding tiles.
- The long-line resources provide routing for longer distances and higher fanout connections.
- These resources, which vary in length (spanning one, two, or four tiles), run both vertically and horizontally and cover the entire device.
- The very long lines span the entire device. They are used to route very long or very high fanout nets.

# Hierarchical-based

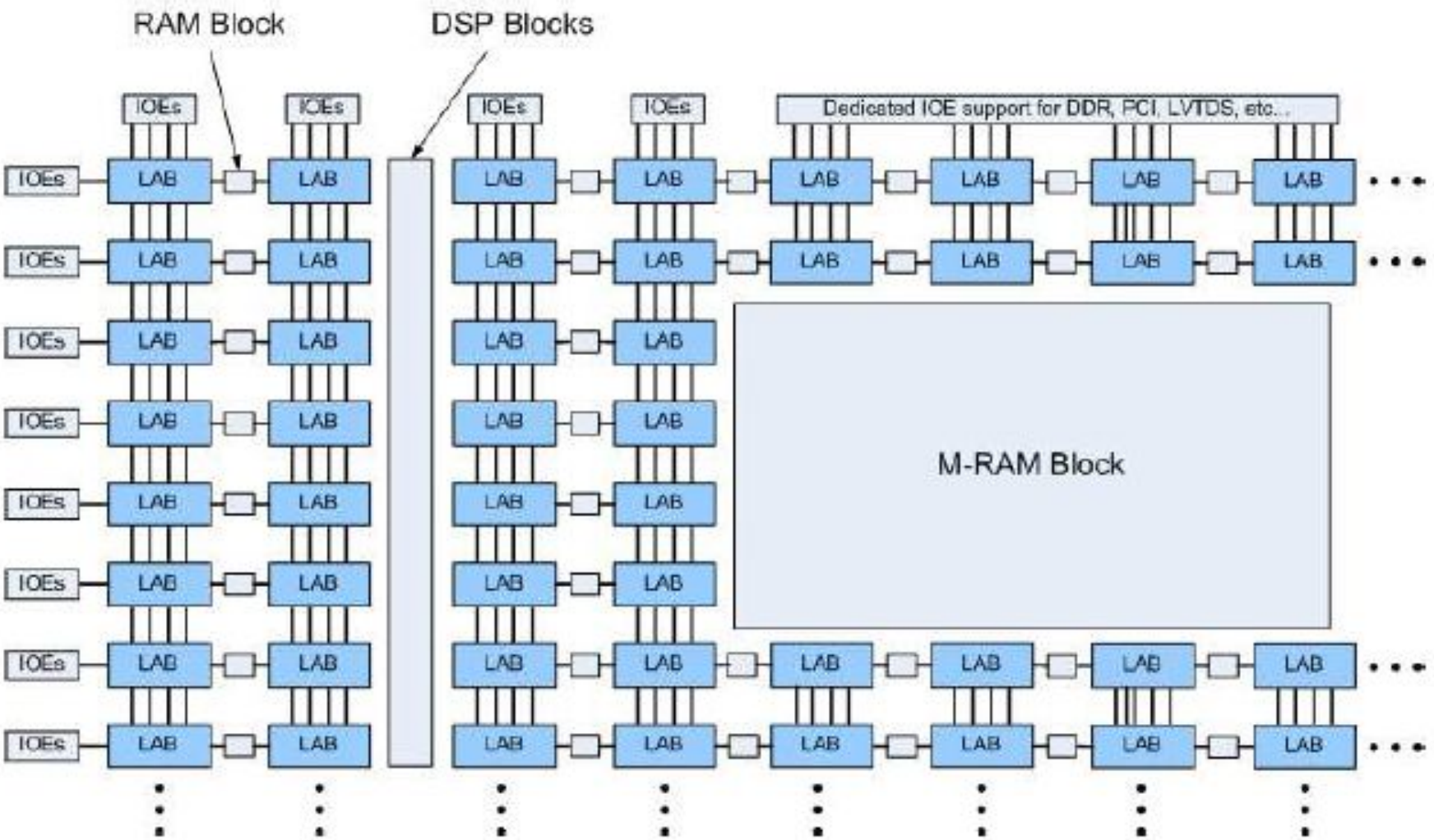
- In hierarchical-based FPGAs, macro cells are hierarchically placed on the device.
- Elements with the lowest granularity are at the lowest level hierarchy. They are grouped to form the elements of the next level.
- Each element of a level $i$ consists of a given number of elements from level $i - 1$

- Altera FPGAs (FLEX, Cyclone II and Stratix II) have two hierarchical levels.
- The logic cells (in the Cyclone II and FLEX) and the ALM in the Stratix II are on the lowest level of the hierarchy.
- The Logic Array Blocks (LAB) build the higher level
- Each LAB contains a given number of logic elements



Hierarchical-based structure

# Hierarchical-based(cont'd)




Hierarchical arrangement on the Altera Stratix II FPGA

# Hierarchical-based(cont'd)

- Signal connections to and from device pin are provided via a routing structure called *FastTrack* in the FLEX and *MultiTrack* in Cyclone II and Stratix-II.

- The *FastTract* as well as the *MultiTrack* interconnects consist of a series of fast, continuous row and column channels that run the entire length and width of the device.

- Signals between LEs or ALMs in the same LAB and those in the adjacent LABs are routed via local interconnect signals.

- Each row of a LAB is served by a dedicated row interconnect, which routes signals between LABs in the same row.

- The column interconnect routes signals between rows and routes signals from I/O pin rows.
- LEs can drive global control signals. This is helpful for distributing the internally generated clock, asynchronous clear and asynchronous preset signals and high-fan-out data signals.

# THANK YOU