ASIC & FPGA Chip Design:

FPGA Architectures

Mahdi Shabany

Department of Electrical Engineering
Sharif University of technology

Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

Outline

- **Introduction**
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

Introduction: Digital System Design

- To design digital systems there are three options:
  - **Microprocessors and DSP** [software-based]
    - Fetch & execute software instructions (e.g., running a word processing program)
    - Very efficient for complex sequential math-intensive tasks
    - Slow & Power hungry

Course Focus

  - **Programmable Logic devices (PLDs)** [Hardware-based]
    - Directly implements logic functions on hardware
    - Faster
    - Less power consumption
  - **Application Specific Integrated Circuit (ASIC)** [Hardware-Based]
    - Fastest
    - Lowest power consumption

Introduction: Digital System Design

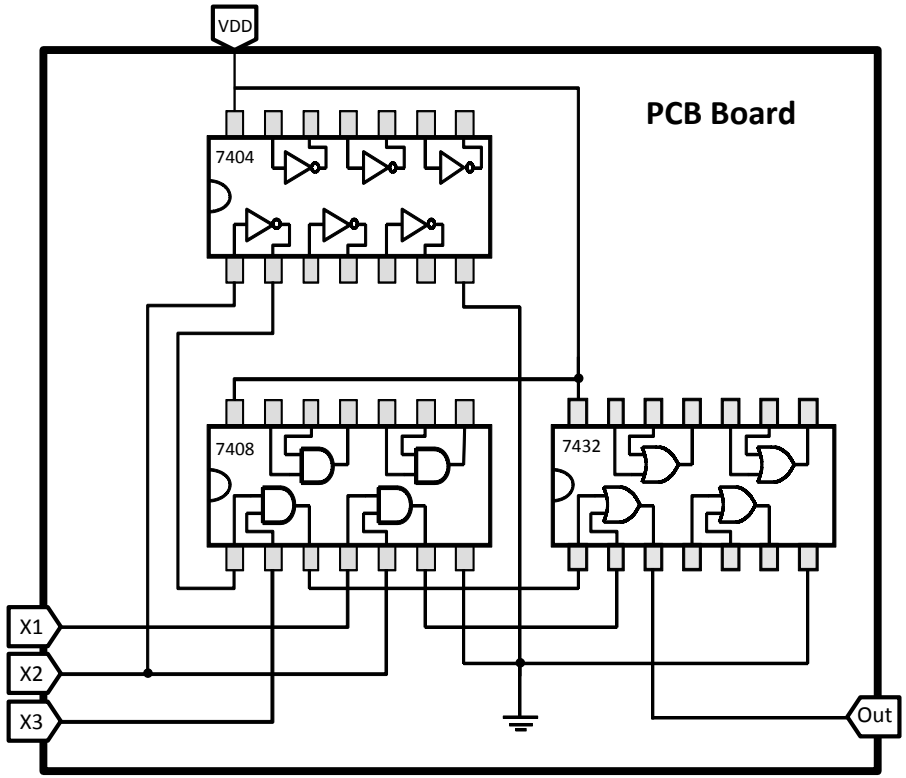
- **DSP** [software-based]
  - Easy to program (usually standard C)
  - Very efficient for complex sequential math-intensive tasks
  - Fixed data path-width. Ex: 24-bit adder, is not efficient for 5-bit addition
  - Limited resources

- **FPGA & ASIC** [Hardware-based]
  - Requires HDL language programming
  - Efficient for highly parallel applications
  - Efficient for bit-level operations
  - Large number of gates and resources
  - Does not support floating point, must construct your own

Introduction: Digital System Design

I. Conventional Approach:

- Board-based designs
  - Large # of chips (containing basic logic gates) on a single Printed Circuit Board (PCB)

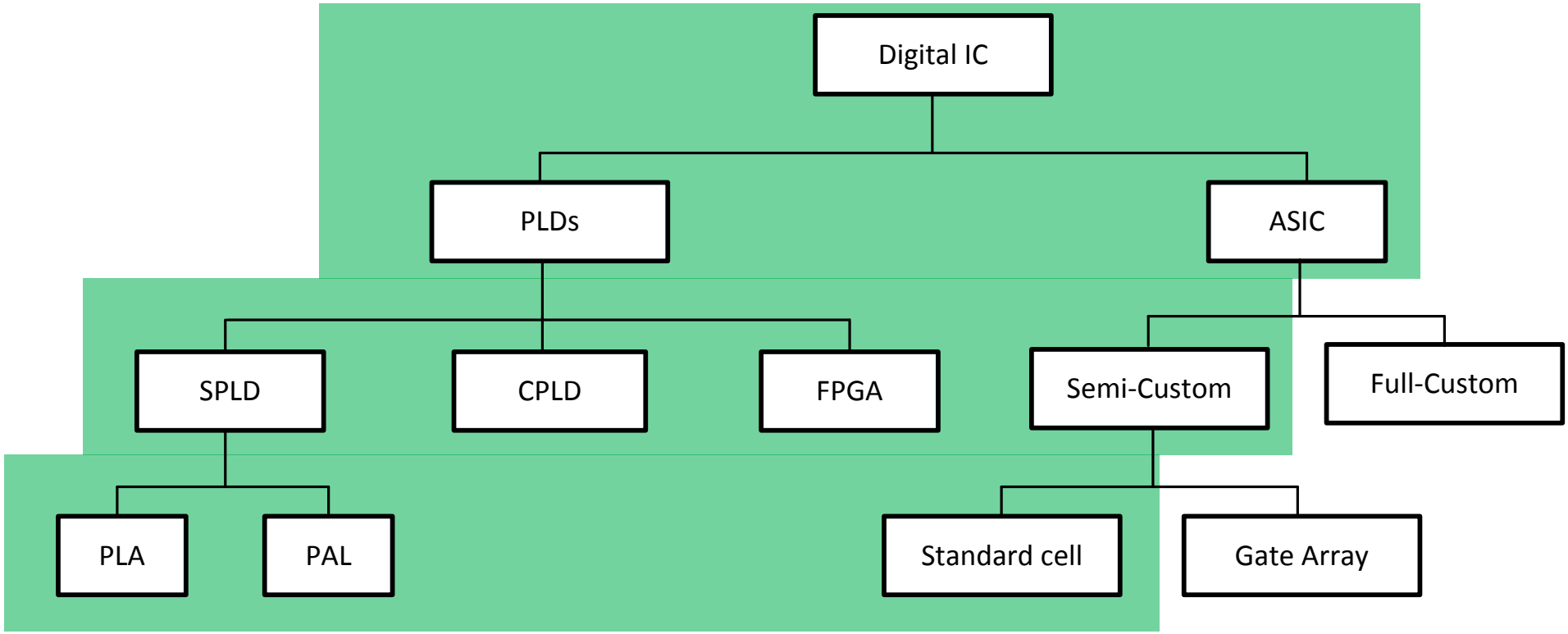


Introduction: Digital System Design

II. High-density Single Chip

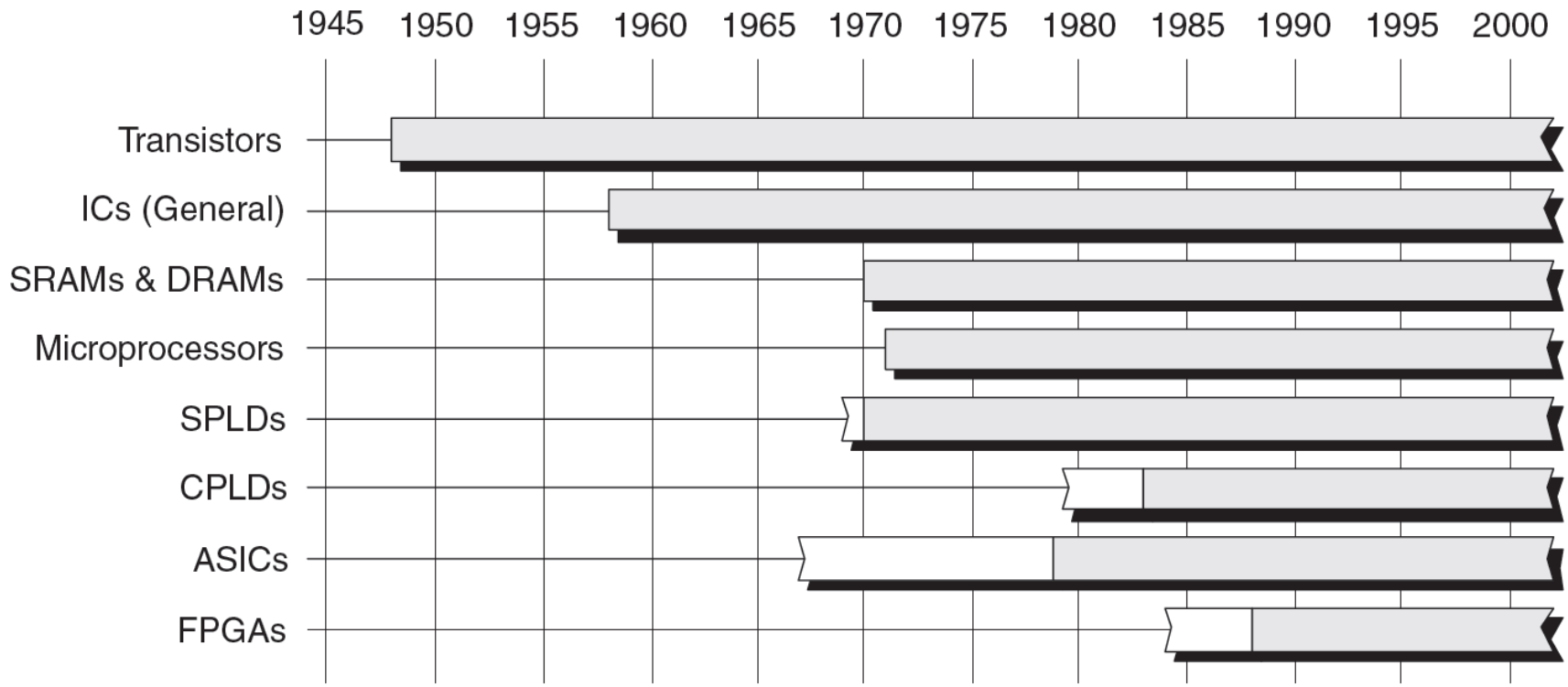
- A single chip replaces the whole multi-chip design on PCB
- Programmable Logic Designs (PLDs) or
- Application Specific Integrated Circuits (ASICs)
  - Lower overall cost
  - On-chip interconnects are many times faster than off-chip wires
  - Lower area with the same functionality
  - Lower power consumption
  - Lower noise

Programmable Logic Designs (PLDs)



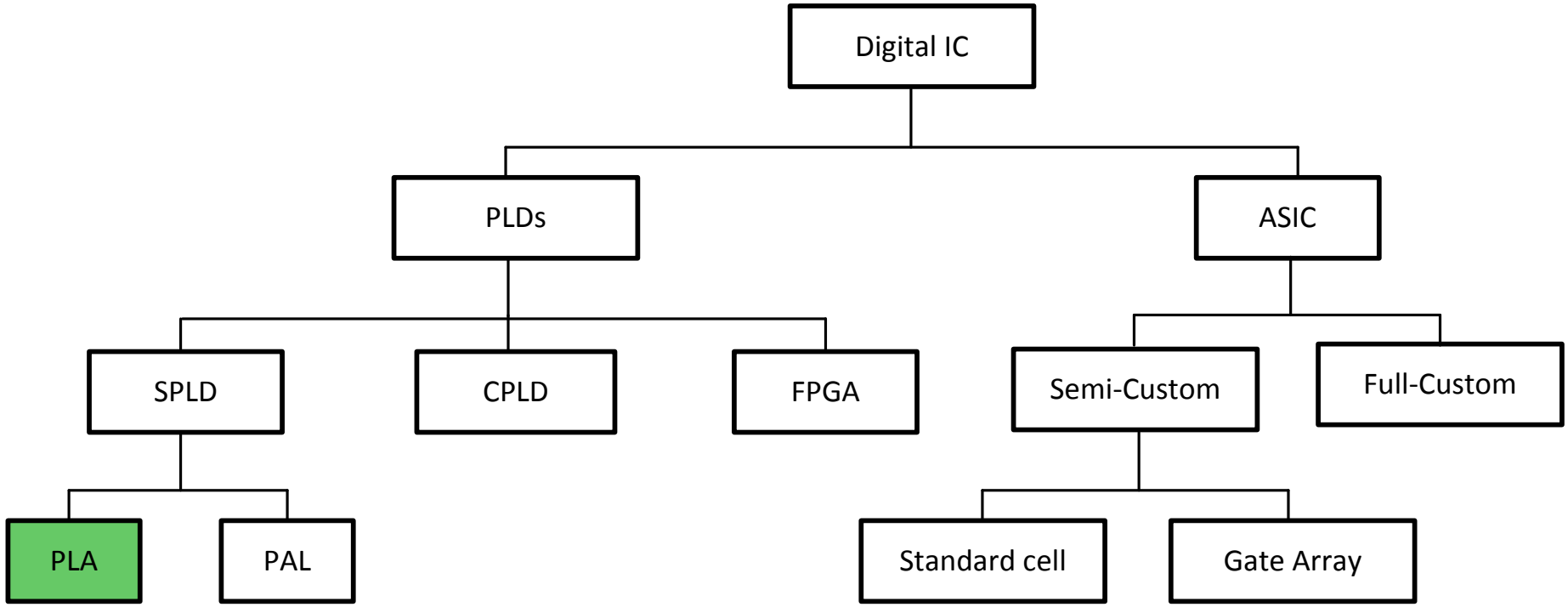
This course

Technology Timeline



The white portions of the timeline bars indicate that although early incarnations of these technologies may have been available, they weren't enthusiastically received by the engineers working in the trenches during this period. For example, although Xilinx introduced the world's first FPGA as early as 1984, design engineers didn't really start using it until the early 1990s.

Programmable Logic Designs (PLDs)



Outline

- Introduction
- **Simple Programmable Logic Designs (SPLDs)**
  - **PLA**
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

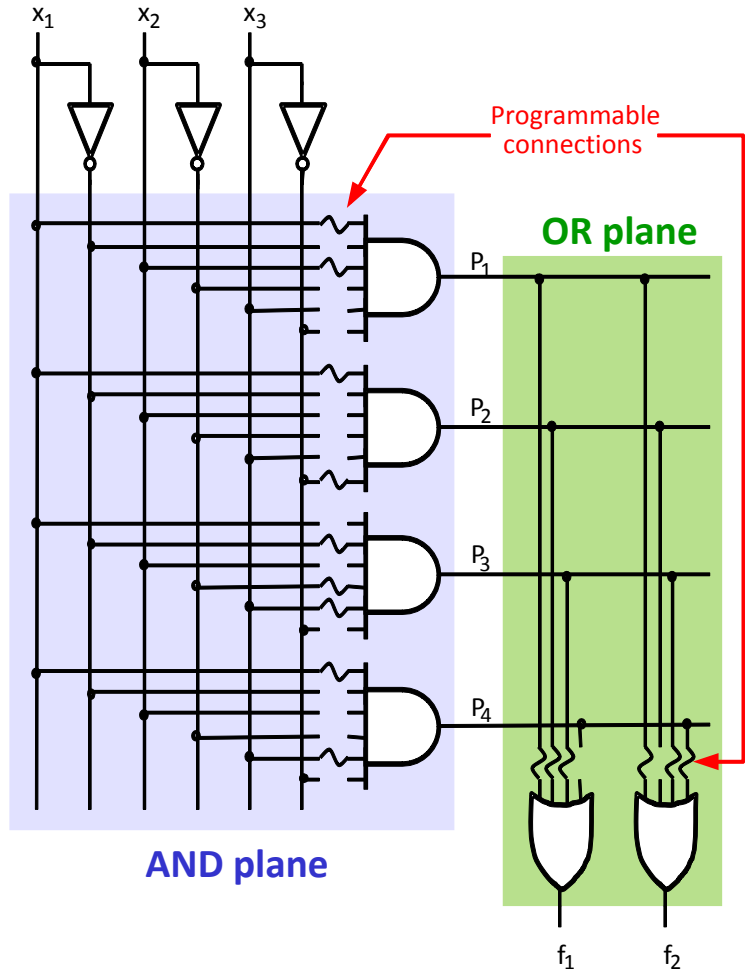
Simple Programmable Logic Designs (SPLDs)

❑ Field Programmable Logic Arrays (FPLA or PLA)

- ➢ Introduced in early 1970s by Philips
- ➢ Consists of two levels of logic gates
  - ▪ Programmable "wired" **AND**-plane
  - ▪ Programmable "wired" **OR**-plane
- ➢ Two levels of programmability
- ➢ Well-suited for implementing functions in sum-of-product (SOP) form.

$$f_1 = x_1x_2 + x_1\bar{x}_3 + \bar{x}_1\bar{x}_2x_3$$

$$f_2 = x_1x_2 + x_1x_3 + \bar{x}_1\bar{x}_2x_3$$

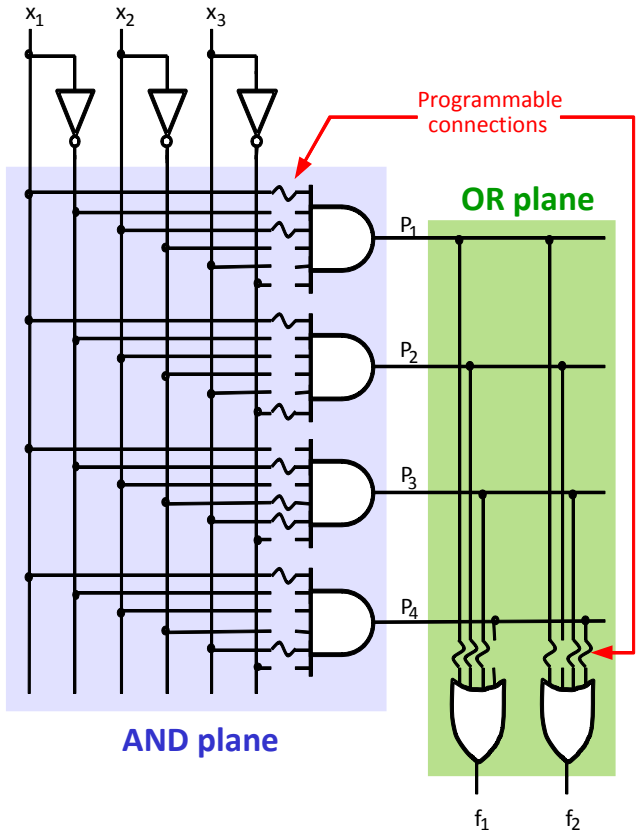


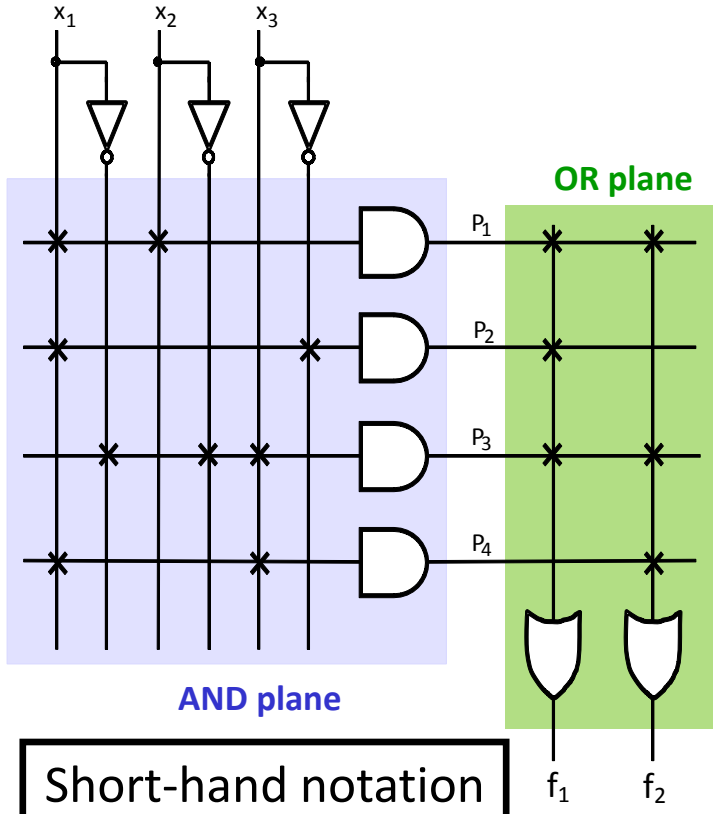
SPLD: Programmable Logic Arrays (PLA)

- Each "AND" gate or "OR" gate can have many inputs
  - **Wide** AND/OR gates

$$f_1 = x_1x_2 + x_1\bar{x}_3 + \bar{x}_1\bar{x}_2x_3$$
$$f_2 = x_1x_2 + x_1x_3 + \bar{x}_1\bar{x}_2x_3$$

Unwanted connections are "blown"





Short-hand notation

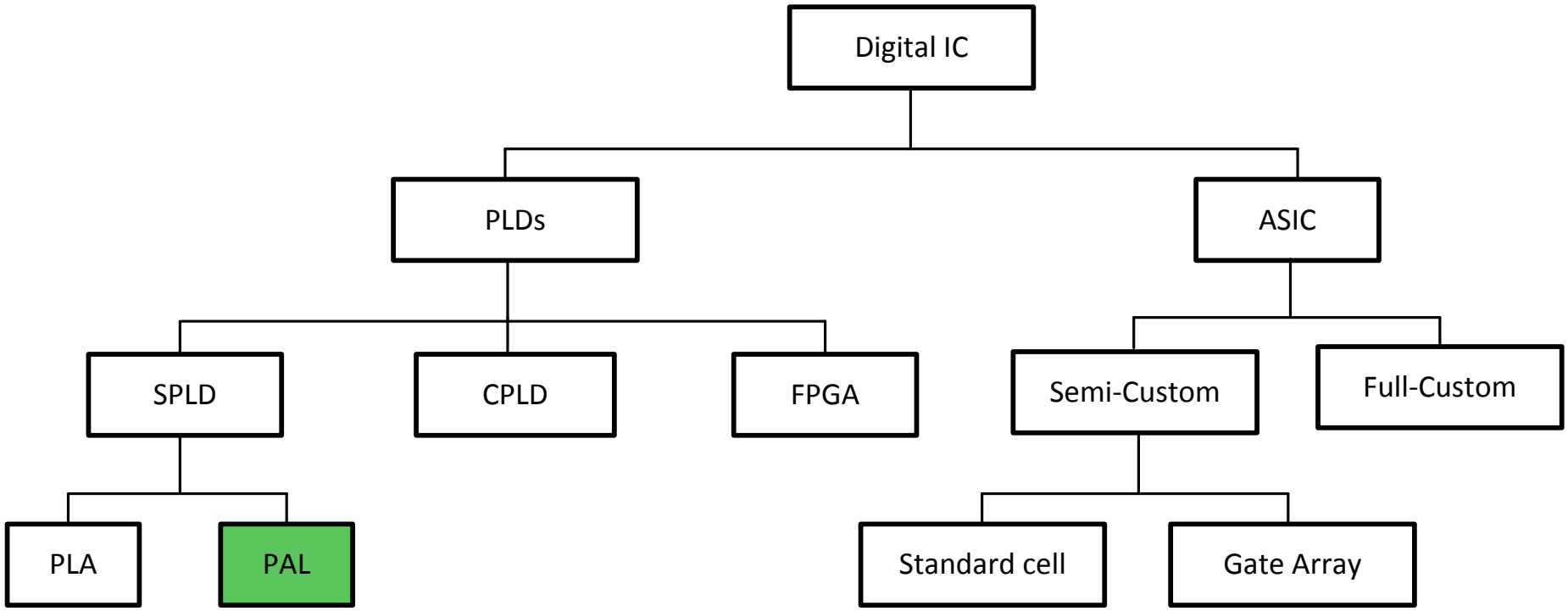
SPLD: PLAs

- **Advantages:**
  - PLA is efficient in terms of its required area for its implementation on IC
  - Often used as part of larger chips, e.g., microprocessors

- **Drawbacks:**
  - Two-level programmable logic planes are difficult to fabricate
  - Two-level programmable structure introduces significant propagation delay
  - Normally many pins, large package thus, high fabrication cost

To overcome these drawbacks, PAL was introduced

Programmable Logic Designs (PLDs)



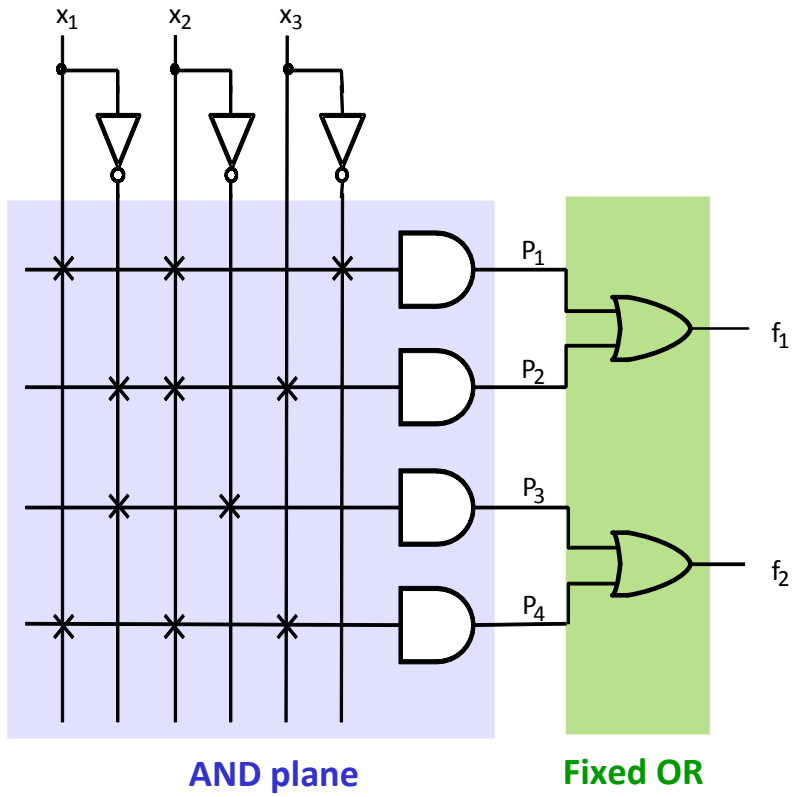
Outline

- Introduction
- **Simple Programmable Logic Designs (SPLDs)**
  - PLA
  - **PAL**
- Complex Programmable Logic Designs (CPLDs)
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

SPLD: Programmable Array Logic (PAL)

- **PAL:**
  - Consists of two levels of logic gates
    - Programmable "wired" **AND**-plane
    - Fixed **OR**-gates
  - Single level of programmability

- **Advantages**:
  - Simpler to fabricate
  - Better performance
- **Drawbacks**:
  - Less flexibility

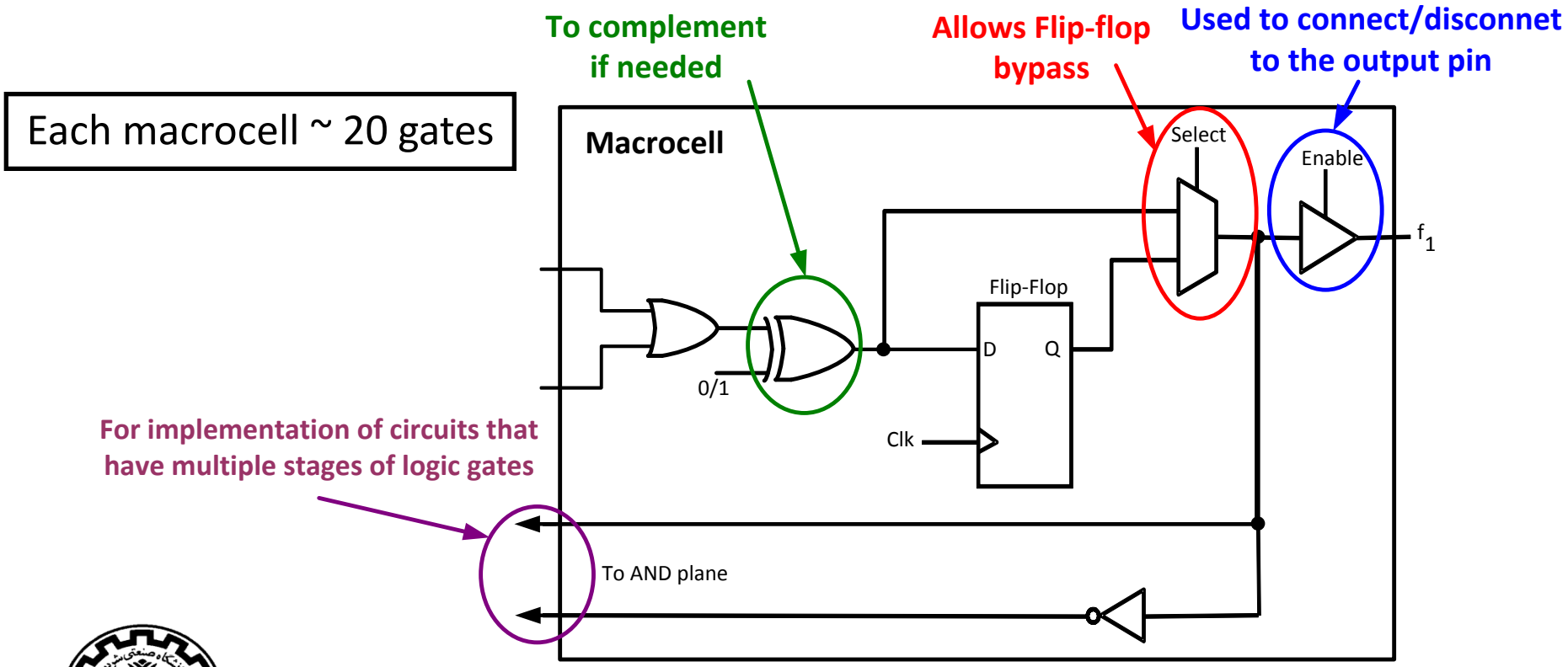


$$f_1 = \bar{x}_1 x_2 x_3 + x_1 x_2 \bar{x}_3$$

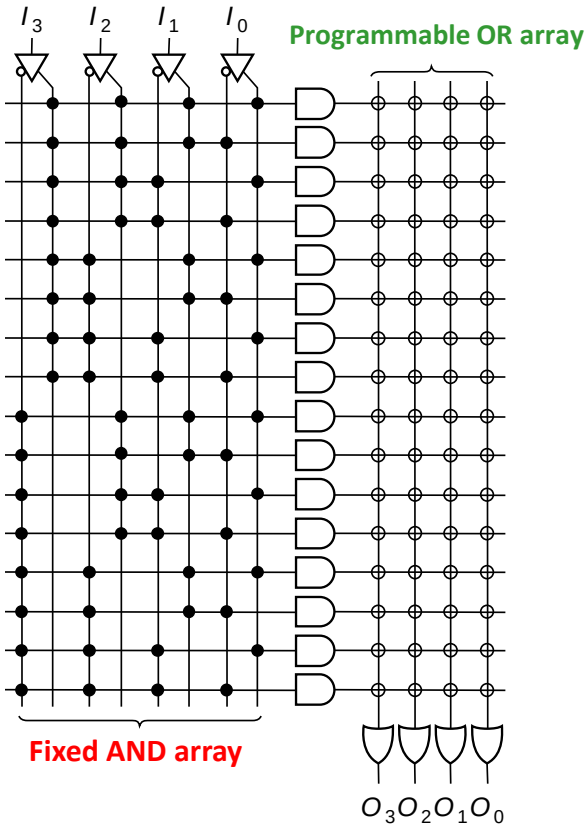
$$f_2 = x_1 x_2 x_3 + \bar{x}_1 \bar{x}_2$$

SPLD: Programmable Array Logic (PAL)

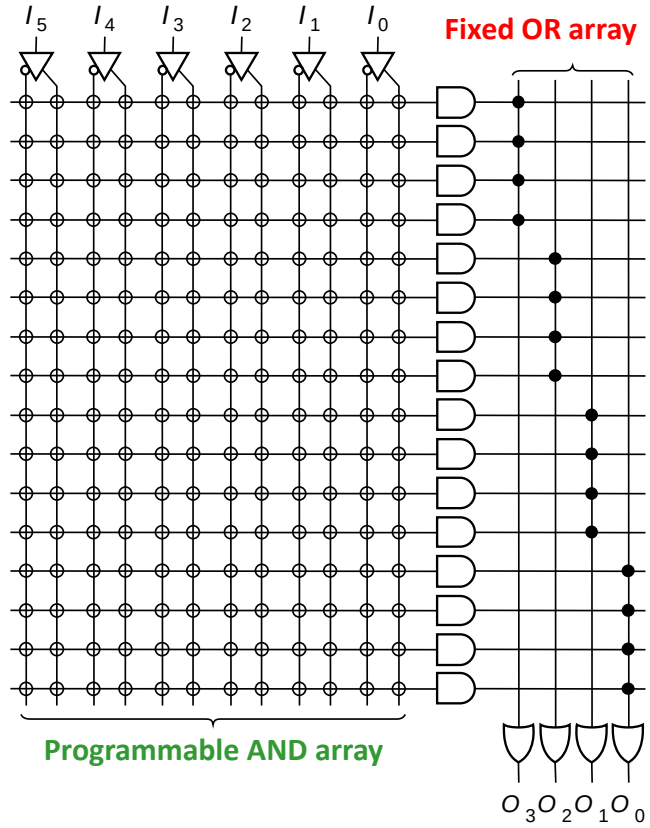
- **To increase flexibility:**
  - PALs with various sizes of OR-gates.
  - Add extra circuitry to the OR-gate output (Called “Macrocell”)



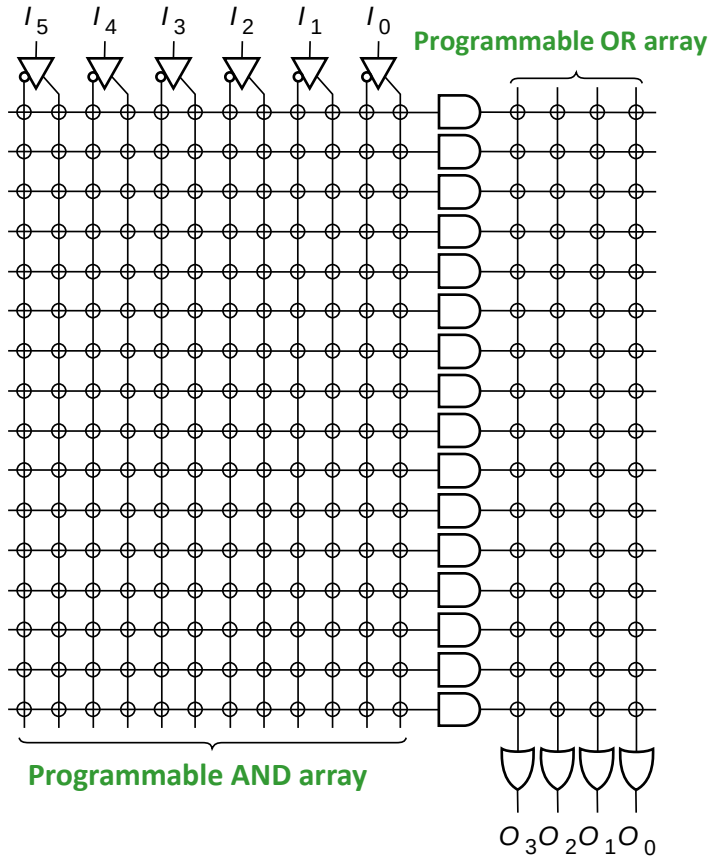
PAL vs. PLA vs. ROM



PROM



PAL



PLA

Indicates programmable connection

Indicates fixed connection

Commercial SPLD Products

- Commercial SPLD Products:

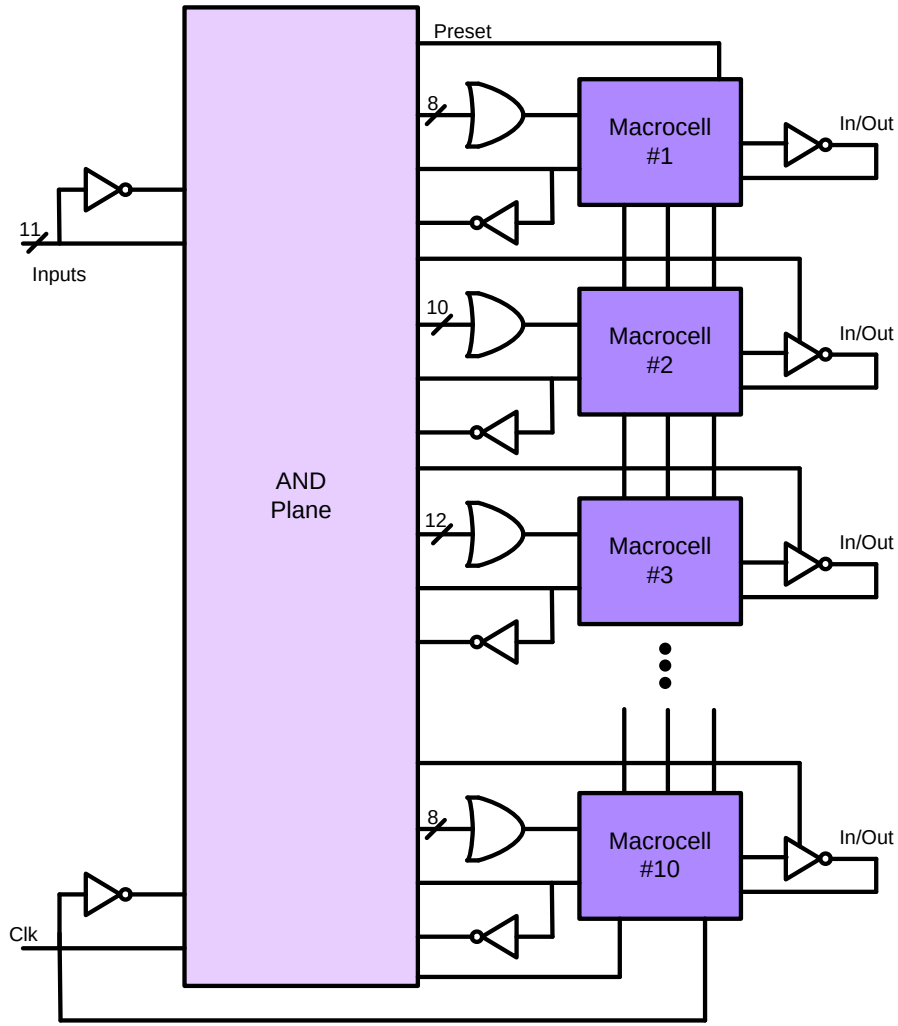
Manufacturer	Product
Altera	Classic
Atmel	PAL
Lattice	ispGAL

- Part number: NN X MM – S
  - NN: Max # of inputs
  - MM: Max # of outputs (some can be used as inputs)
  - X=R (outputs are registered by a D-FF)
  - X=V (Volatile)
  - S: Speed grade

Example:
- 22 V 10-1
- 16 R 8-2

PAL: 22V10 (Lattice Semiconductors)

- Maximum of 22 inputs
  - 11 inputs, one clock, 10 in/outs
- 10 inputs/outputs
- Variable OR gates (8 to 16 inputs)



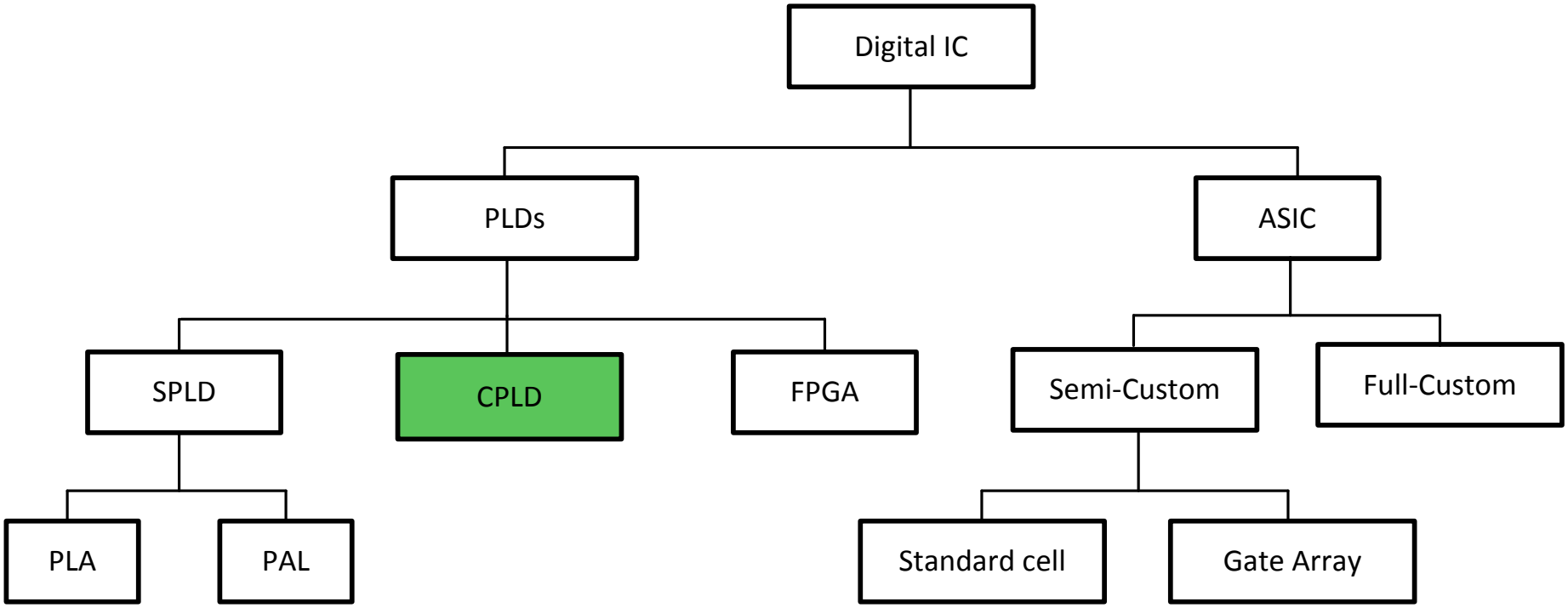
SPLD Scalability

- It is very hard to scale SPLDs for more complex designs
  - b/c the structure of the logic planes grow too quickly in size as the # of inputs increases

- **Solution:**
  - Integrate multiple SPLDs onto a single chip
  - Plus internal programmable interconnect to connect them together

→ **Complex PLDs (CPLD)**

Programmable Logic Designs (PLDs)

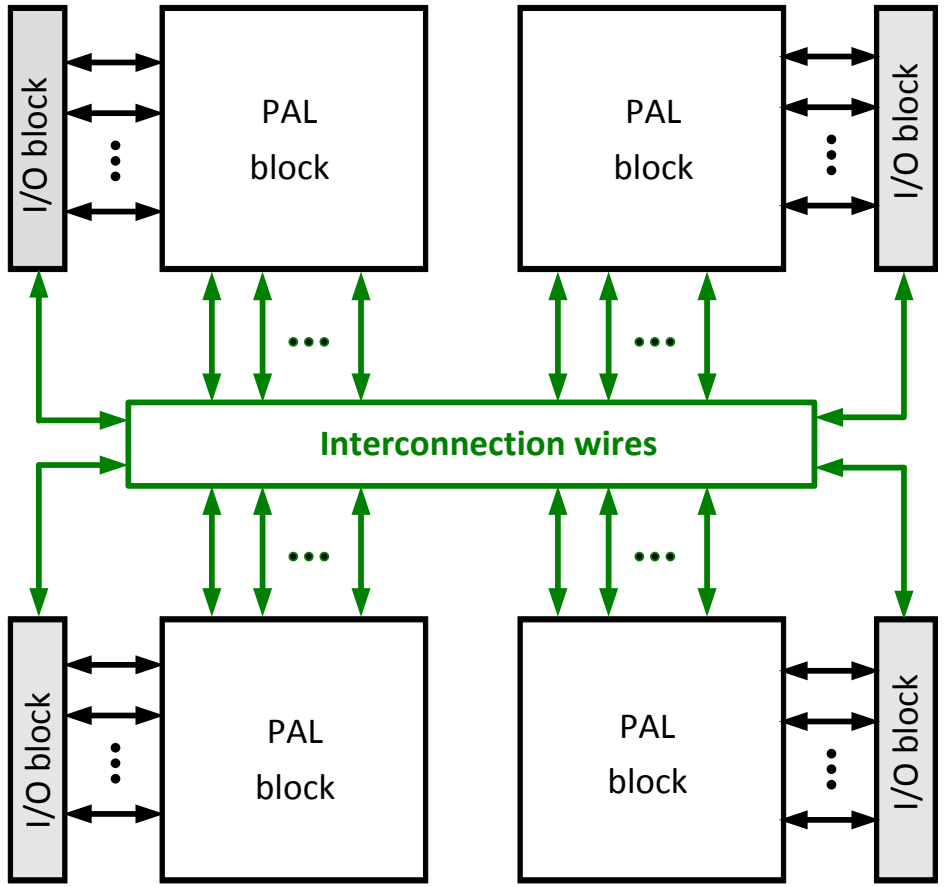


Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- **Complex Programmable Logic Designs (CPLDs)**
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

CPLD

- Consists of 2 to 100 PAL blocks
- Interconnection contains programmable switches
- The number of switches is critical

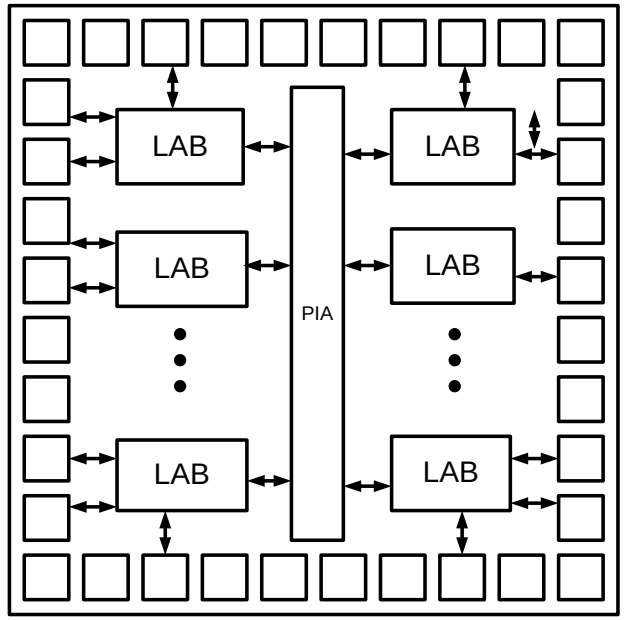


Commercial CPLDs:

Manufacturer	Product
Altera	MAX 7000, MAX 10K
Atmel	ATF
Xilinx	XC9500
AMD	Mach series
ICT	PEELArray
Lattice	ispLSI series

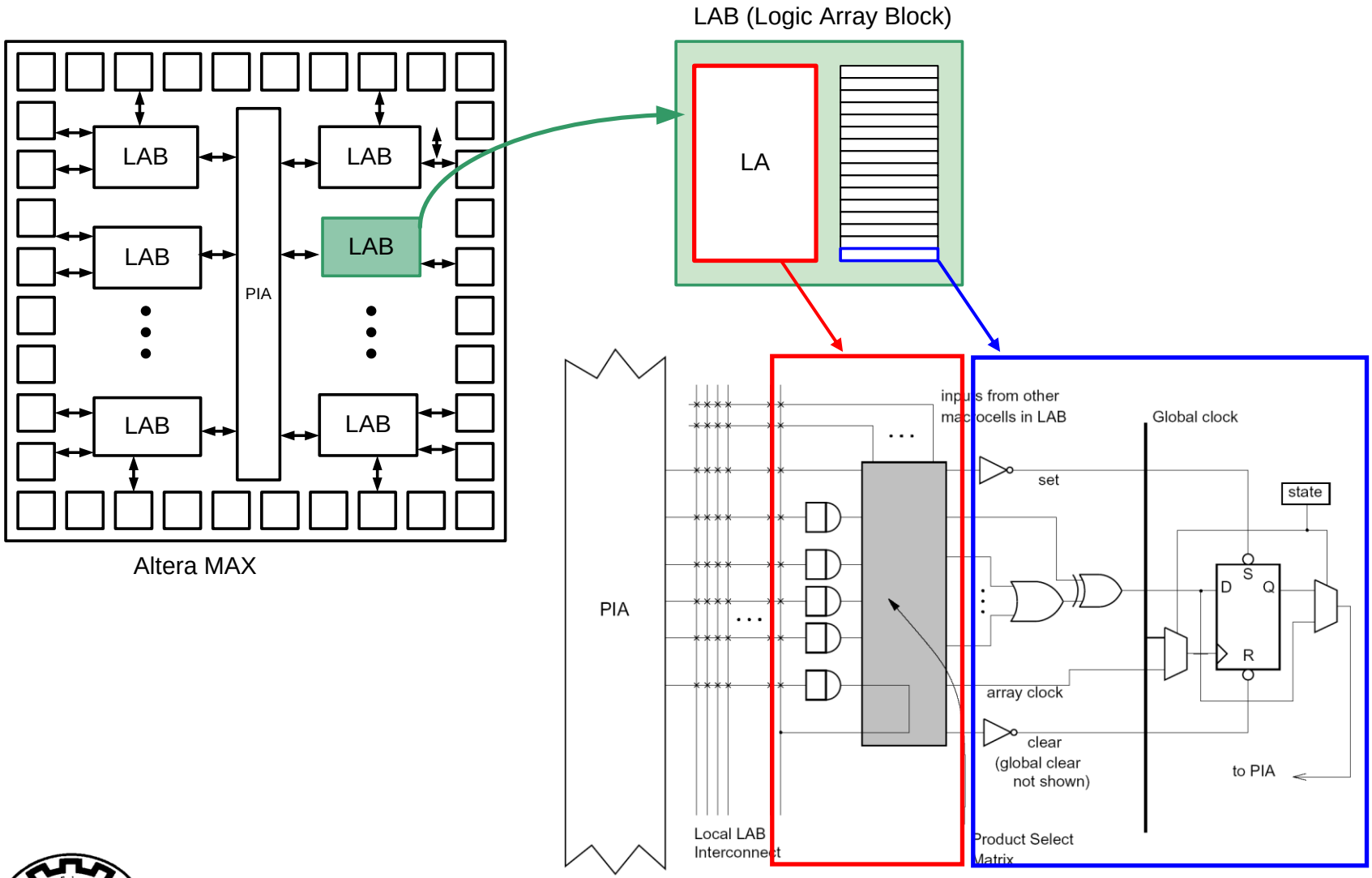
CPLD: Altera MAX7000

- Comprises:
  - Several Logic Array Blocks (LAB), a set of 16 macrocells
  - Programmable Interconnect Array (PIA)
    - Consists of set of wires that span the entire device
    - Makes connections between macrocells and chip's input/output pins
- In total consists of 32 to 512 macrocells
- Four dedicated input pins
  - For global clock or FF resets



Altera MAX 7000

CPLD: Altera MAX7000



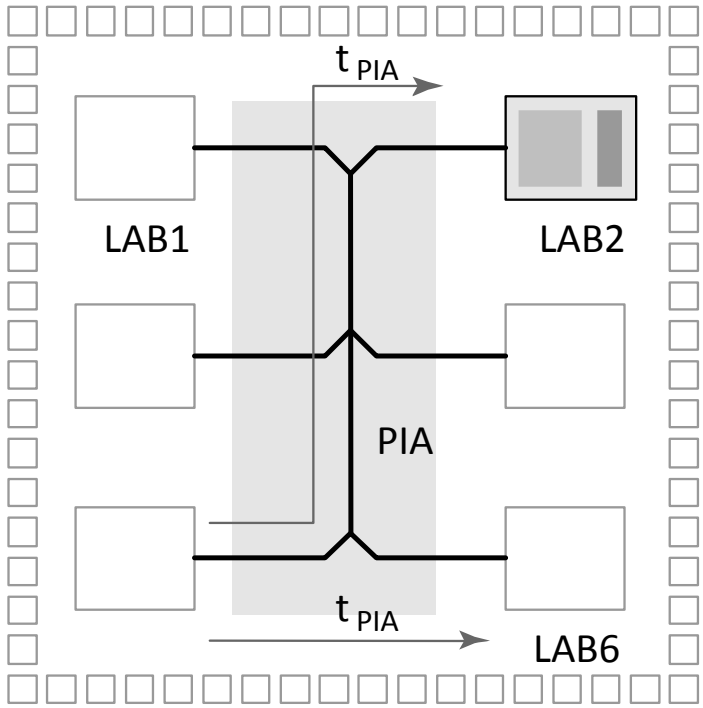
CPLD: Altera MAX7000

- Comprises:
  - Wide programmable AND array followed by
  - A narrow fixed OR array

- OR gate can be fed from:
  - Any of the five product terms within the macrocell
  - or up to 15 extra product terms from other macrocells in the same LAB

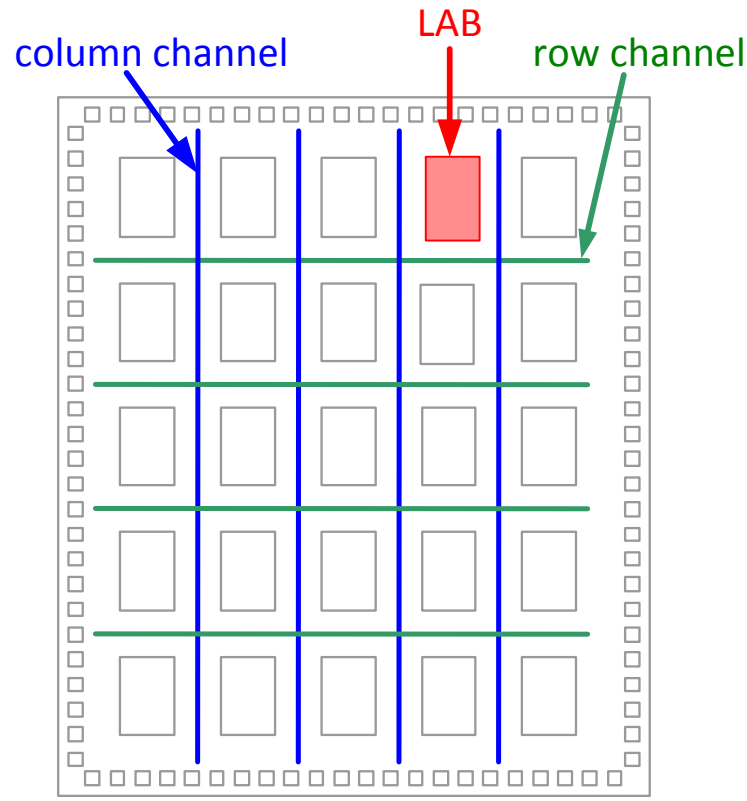
  ⇒ more flexibility

CPLD: Altera MAX7000 Interconnect Architecture



Array-based (MAX 3000, 7000)

- ❑ Fixed routing delay b/w blocks
- ❑ Simple and predictable delay
- ❑ Not scalable to large # of macrocells

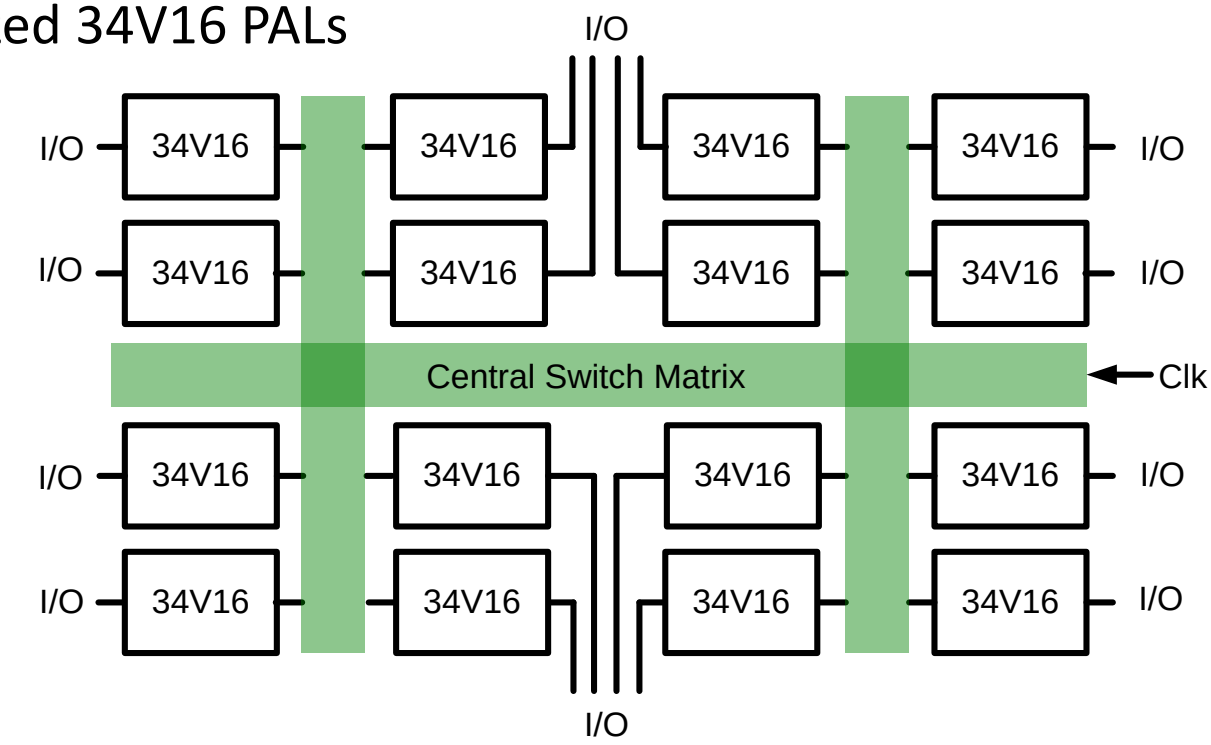


Mesh-based (MAX 9000, 10K)

- ❑ LABs can connect to row and column channels
- ❑ Suitable for large # of macrocells (512)

Advanced Micro Devices (AMD) CPLDs:

- Mach family (Mach 1 to Mach 5) all EEPROM-based technology
  - Mach 1, 2: Multiple 22V16 PALs
  - Mach 3, 4, 5: Several optimized 34V16 PALs
- **Mach 4:**
  - Consists of:
    - 6 to 16 PAL (2K-5K gates)
    - Central switch matrix
  - In-circuit programmable

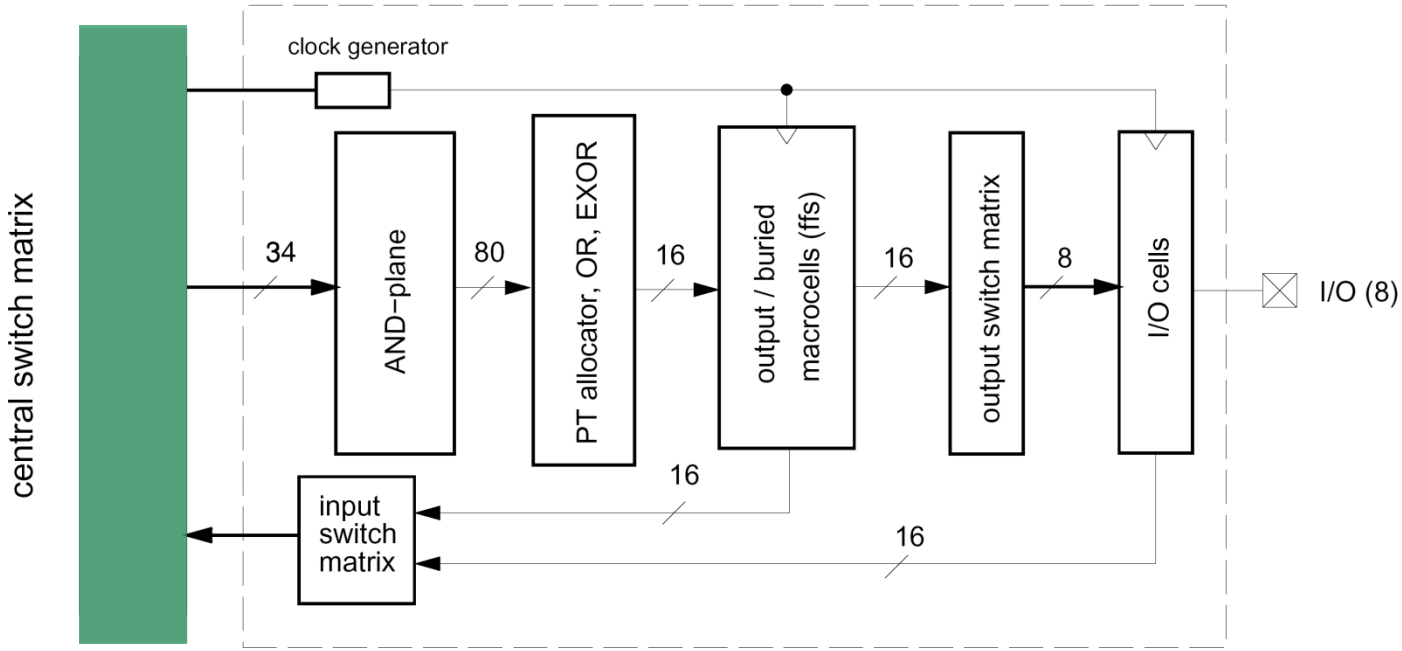


All connections b/w PALs and even inside a PAL routed through the central switch matrix

AMD Mach 4 PAL Block:

- 34V16 (34 maximum inputs, volatile, max 16 outputs)
- In addition to a normal PAL, it consists of:
  - product term allocator b/w AND plane and macrocells, which distributes product terms to whichever OR-gate required
  - Output switch matrix b/w OR gates and I/O

Any macrocell can drive any of the I/O pins (more flexibility)



CPLD Applications:

- Circuits that can exploit wide AND/OR gates and do not need large number of flip-flops
  - Graphic controllers
  - LAN controllers
  - UARTs
  - Cache control

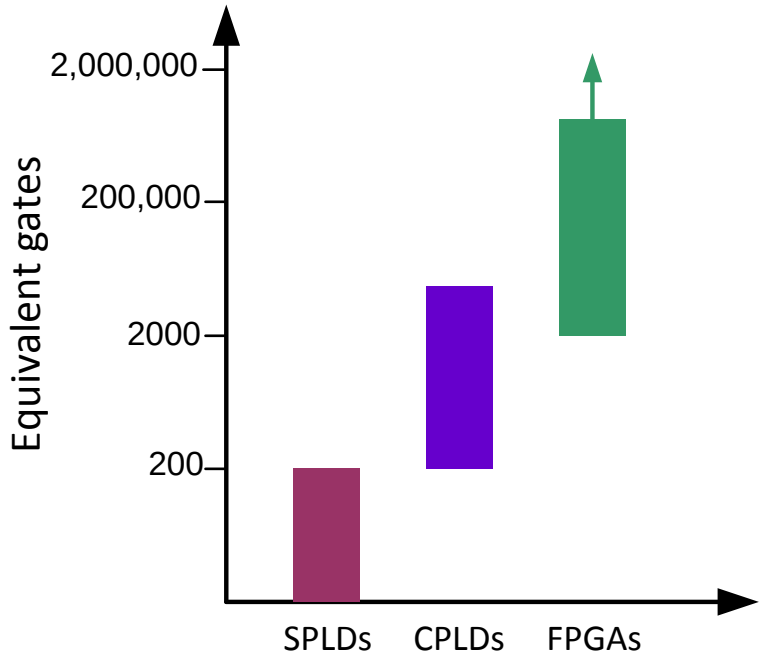
- **Advantages:**
  - Easy to re-program even in-system
  - Predictability of circuit implementation
  - High-speed implementation

Circuit Size Metric

❑ Size Metric:

- ➢ How many basic gates can be built on the circuit
- ➢ Common measure: number of two-input NAND gates

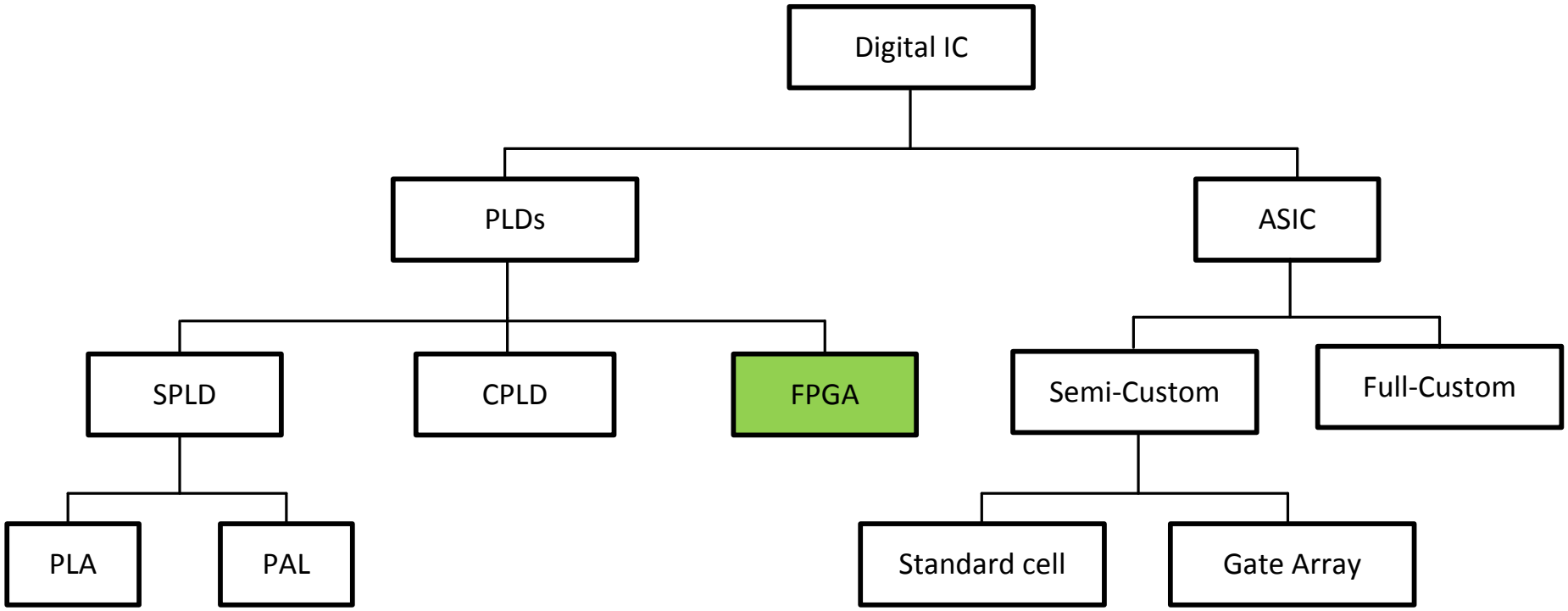
Device	Size	Design Type
SPLD	~ 200 gates	Small
CPLD	~ 10,000 gates	Moderate
FPGA	~ 1,000,000 gates	Large



Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- **Field-Programmable Gate Array (FPGAs)**
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

Programmable Logic Designs (PLDs)



FPGA

❑ FPGA: (Field-Programmable Gate Array)

- ➢ Pre-fabricated silicon devices that can be electrically programmed to become any kind of digital circuit or system
- ➢ A very large array of programmable logic blocks surrounded by programmable interconnects
- ➢ Contains logic blocks instead of AND/OR planes (multi-level logic of arbitrary depth)
- ➢ Can be programmed by the end-user to implement specific applications
- ➢ Capacity up to multi-millions gates
- ➢ Clock frequency up to 500MHz

FPGA

❑ Three ages of FPGAs

Period	Age	Comments
1984 - 1991	Invention	• Technology is limited, FPGAs are much smaller than the application problem size • Design automation is secondary • Architecture efficiency is key
1991 - 1999	Expansion	• FPGA size approaches the problem size • Ease-of-design becomes critical
2000 - 2007	Accumulation	• FPGAs are larger than the typical problem size • Logic capacity limited by I/O bandwidth

FPGA Applications

- **Popular applications:**
  - Prototyping a design before the final fabrication (using single FPGA)
  - Emulation of entire large hardware systems (using multiple FPGAs)
  - Configured as custom computing machines
    - Using programmable parts to “execute” software rather than software compilation on a CPU
  - On-site hardware reconfiguration
  - Low-cost applications
  - DSP, logic emulation, network components, etc…

FPGA History

- First SRAM-based FPGA by Wahlstorm 1967
- First modern-era FPGA by Xilinx 1984
  - 64 logic blocks
  - 58 input/outputs
- Today:
  - Four main manufacturers (Altera, Xilinx, Actel, Lattice)
    - Over 300,000 logic blocks
    - Over 1100 input/outputs

FPGA Structure

- FPGAs consists of 3 main resources:

FPGA Fabrics

1. Logic Blocks

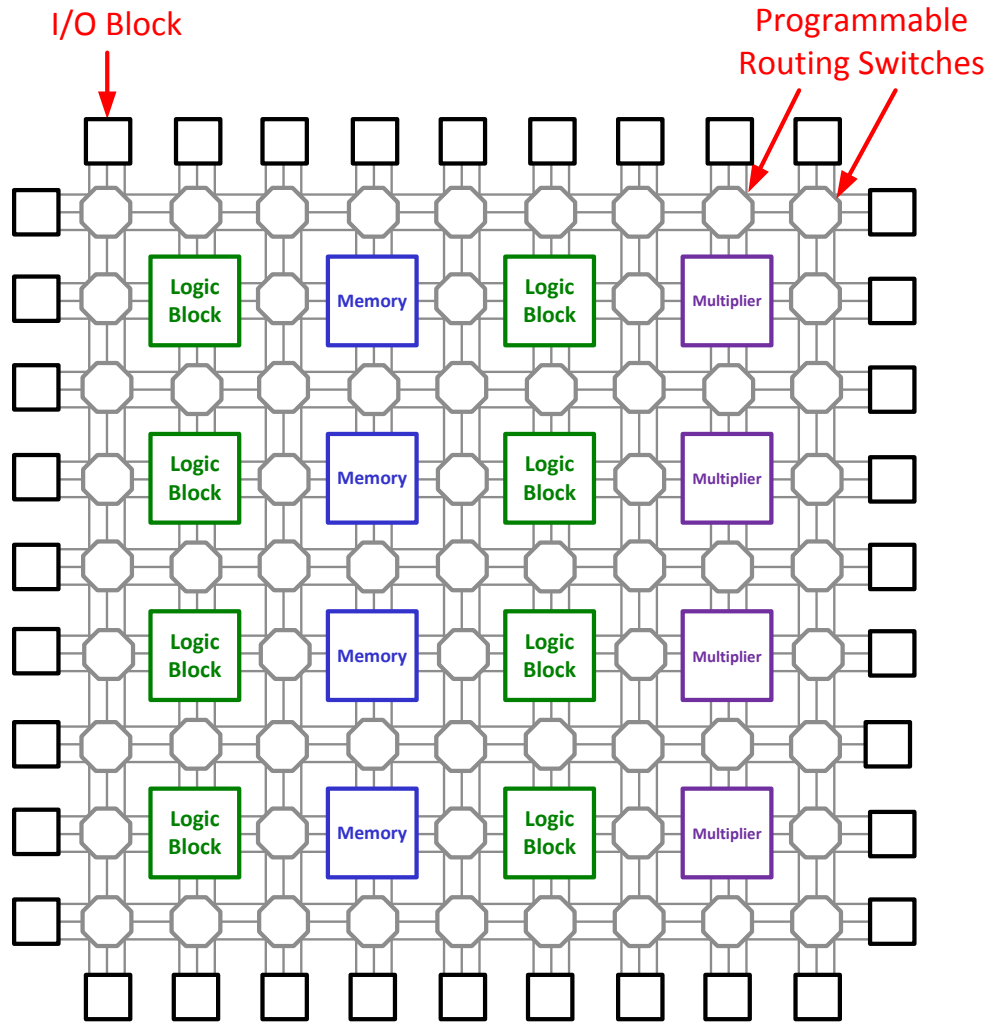
- General logic blocks
- Memory blocks
- Multiplier blocks

2. Program. Routing Switches

- Programmable horizontal/vertical routing channels
- Connecting blocks together and I/O

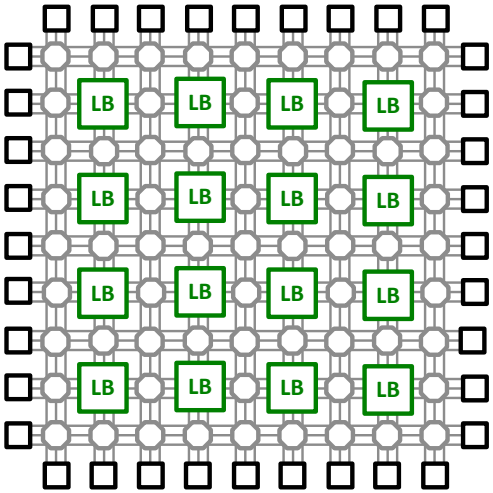
3. I/O Blocks

- Connecting the chip to the outside

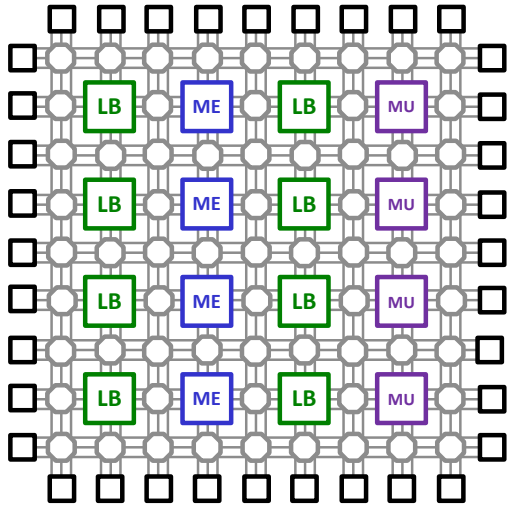


FPGA Categories (Structure)

- There are two main categories of FPGAs in terms of their structure:
  - **Homogeneous:**
    - Employs only one type of logic block
  - **Heterogeneous:**
    - Employs mixture of different blocks such as dedicated memory/multiplier
    - Very efficient for specific functions
    - Might go waste if not used!



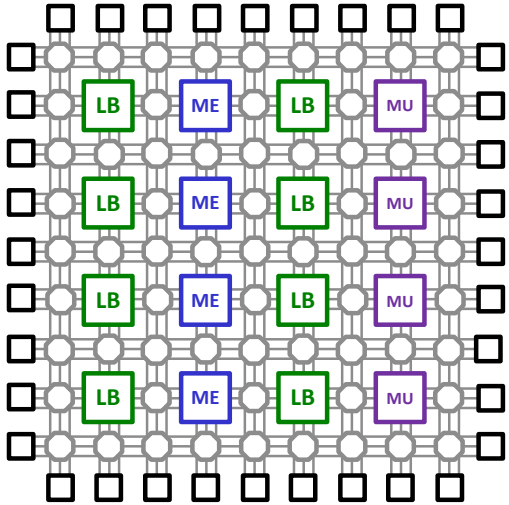
Homogeneous



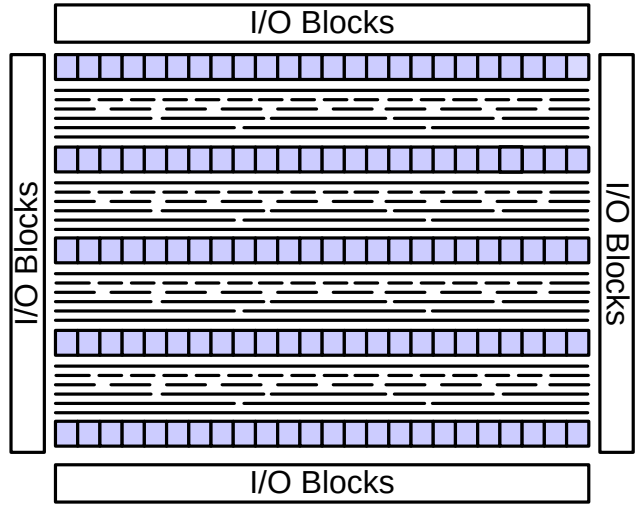
Heterogeneous

FPGA Categories (Floor Plan)

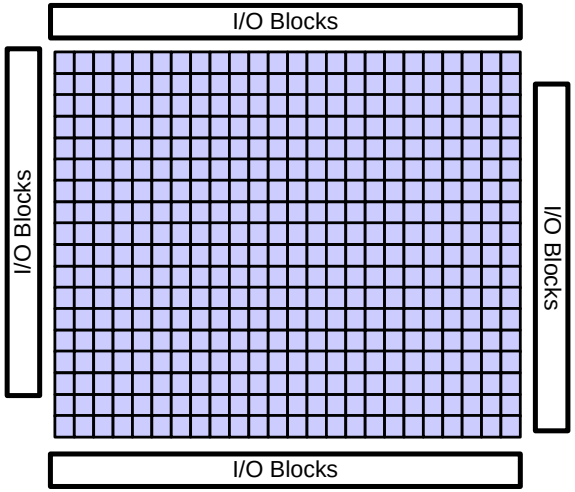
Symmetrical Array



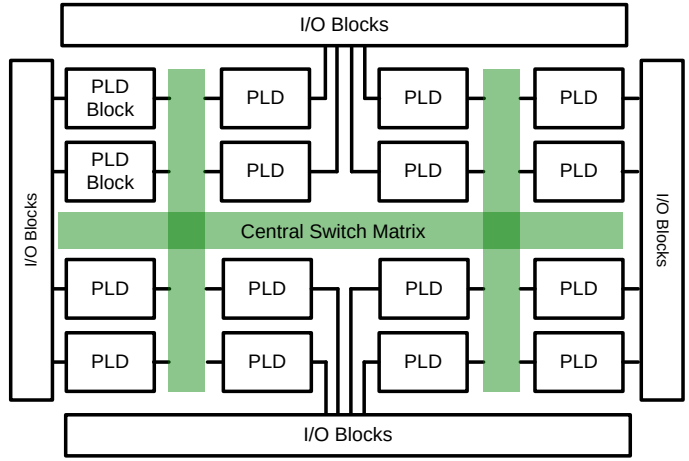
Row-Based



Sea-of-Gates



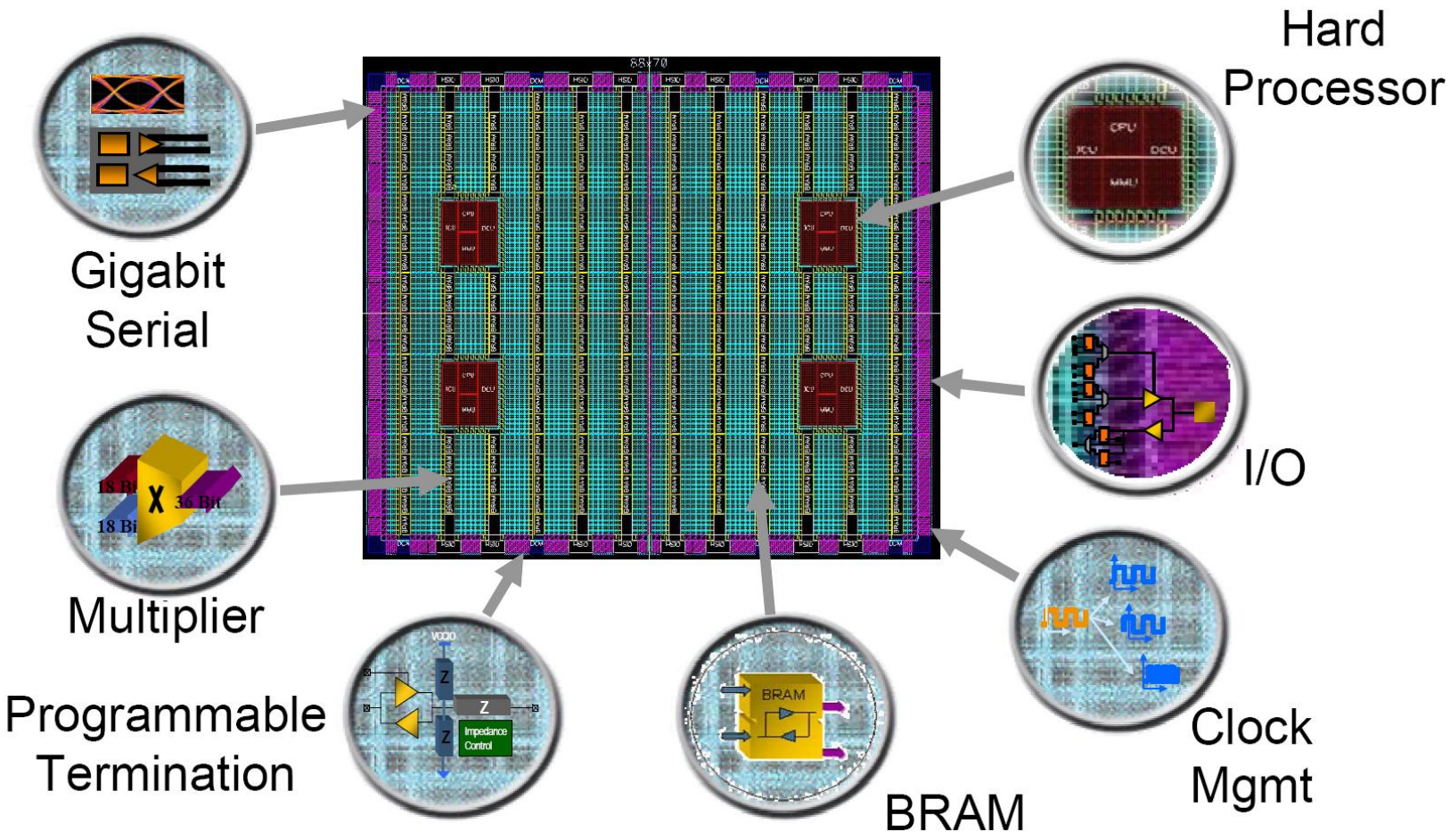
Hierarchical PLD



FPGA Categories (Architecture)

- There are three main categories of FPGAs in terms of their architecture:
  - **Fine-grained: (early stages)**
    - Logic Block (LB) consists of logic gates plus a register
  - **Coarse-grained: (more efficient)**
    - LB consists of logic gates, MUXs
    - Multi-bit ALU
    - Multi-bit registers
  - **Platform FPGAs:**
    - Sophisticated logic blocks
    - CPU (PowerPC) to run some functions in software
    - PCI bus
    - RAM, PLL
    - Very fast Gbps transceivers for high-speed serial off-chip communication

Modern Commercial FPGAs

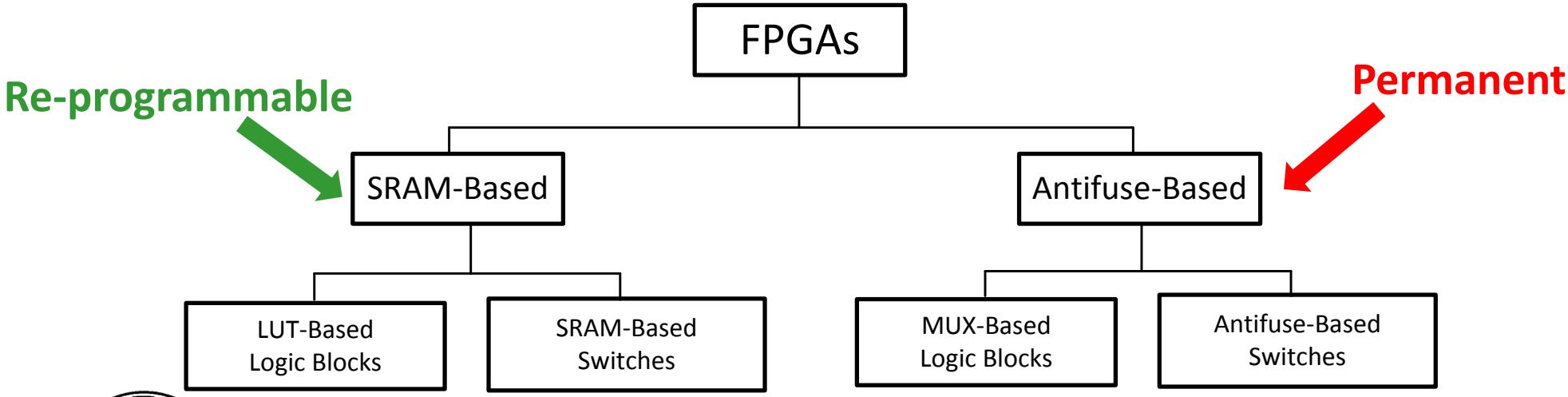


Modern Commercial FPGAs

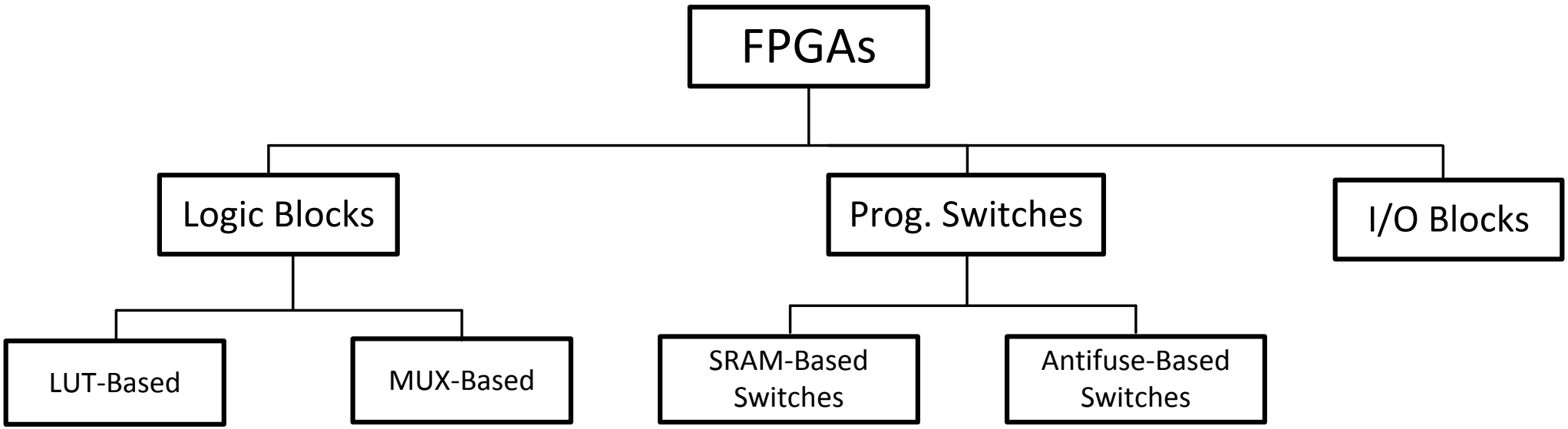
- The concept of coupling microprocessors with FPGAs in heterogeneous platforms was considerably attractive.
- In this programmable platform, microprocessors implement the control-dominated aspects of DSP systems and FPGAs implement the data-dominated aspects.
- With FPGAs, the user is given full freedom to define the architecture which best suits the application.

FPGA Categories (Fabrics)

- There are two main categories of FPGAs in terms of their fabrics:
  - **SRAM-based FPGAs (Xilinx, Altera) [Re-programmable, Re-configurable]**
    - Using Lookup Tables (LUTs) to implement logic blocks
    - Using SRAM-cells to implement programmable switches
  - **Antifuse-based FPGAs (Actel, Lattice, Xilinx, QuickLogic, Cypress) [Permanent]**
    - Using multiplexers (MUXs) to implement logic blocks
    - Using antifuses to implement programmable switches



FPGA Categories (Another View)



Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- **Field-Programmable Gate Array (FPGAs)**
  - **Logic Blocks**
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

Logic Block

- The logic block is the most important element of an FPGA, which provides the basic computation and storage elements used in digital logic systems

- Logic blocks are used to implement logic functions

- A logic block has a small number of inputs and outputs

- The logic block of an FPGA is considerably more complex than a standard CMOS gate b/c:
  - A CMOS gate implements only one chosen logic function
  - An FPGA logic block must be configurable enough to implement a number of different functions

Logic Block Design

❑ Transistors as the basic logic block (fine-grained)

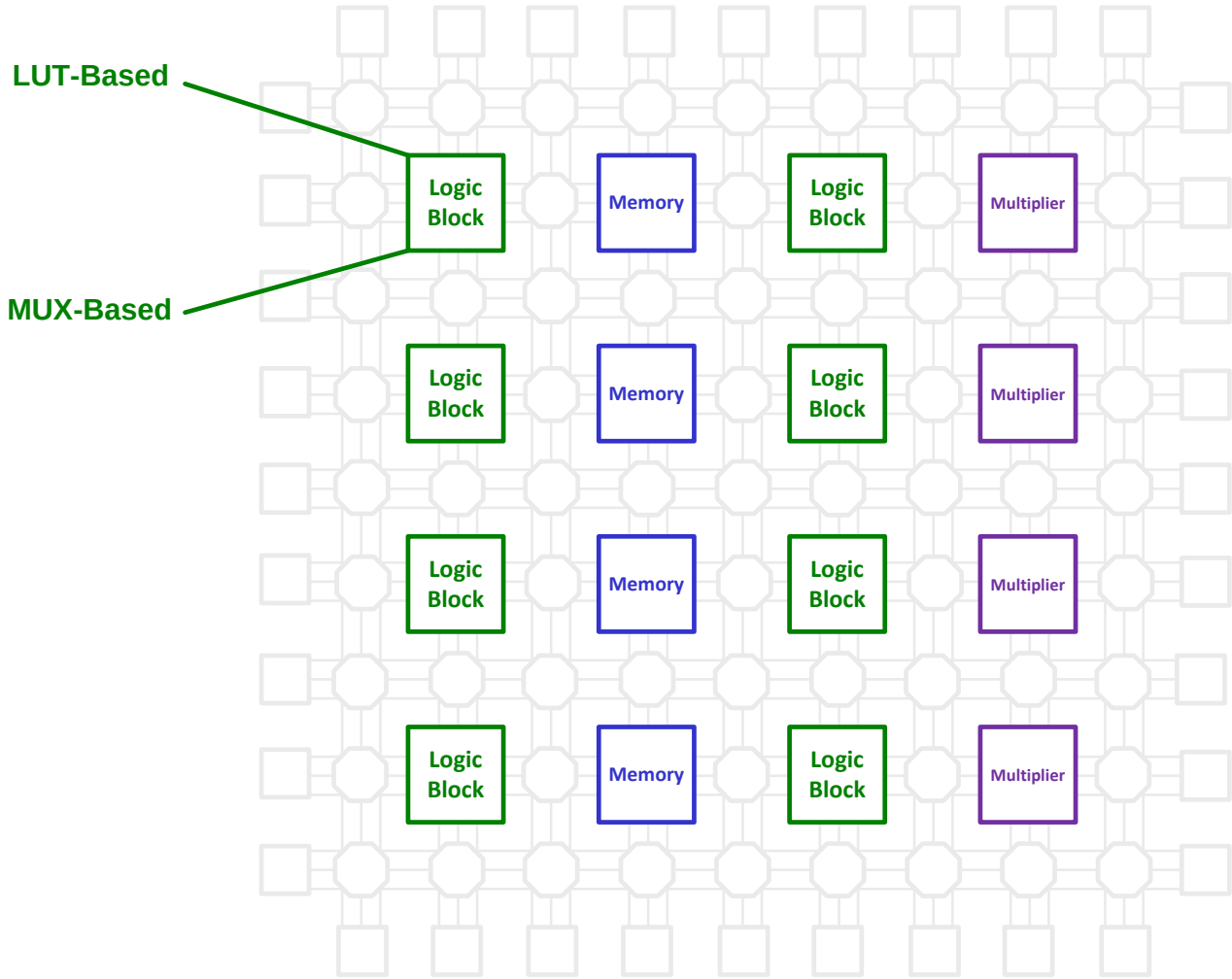
- ➢ Build gates & storage elements from it
- ➢ Tried in Crosspoint
- ❖ **Drawbacks:**
  - ➢ Requires huge amount of Prog. interconnects to create a typical logic function
  - ➢ Low area-efficiency (b/c Prog. switches are area intensive)
  - ➢ Low performance (b/c each routing hop is slow)
  - ➢ high power consumption (higher interconnects capacitance to charge and discharge)

❑ Processors as the basic logic block (coarse-grained)

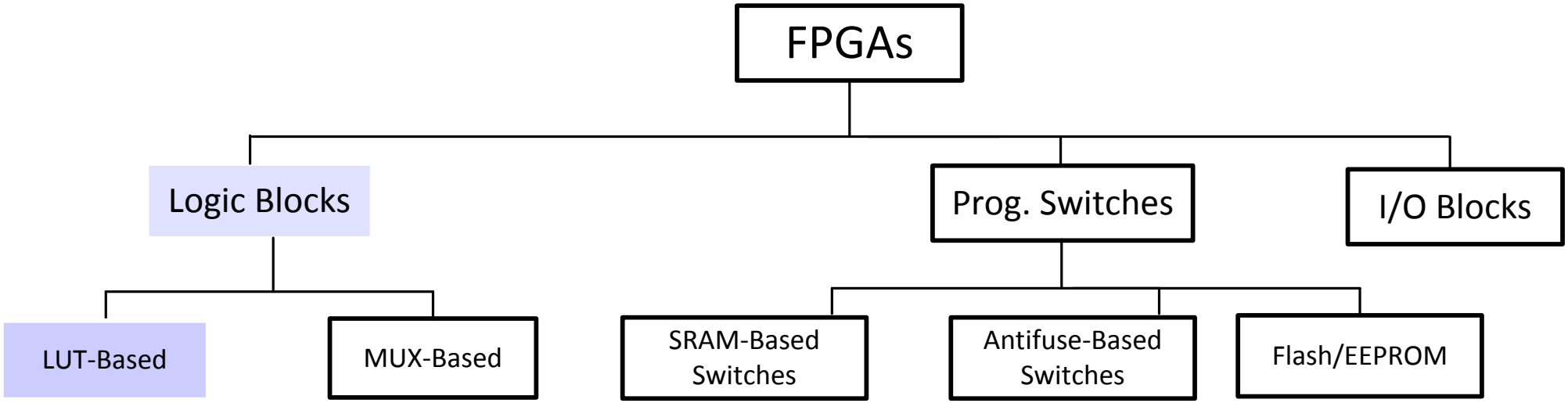
- ❖ **Drawbacks:**
  - ➢ Incredibly inefficient for implementing simple functions
  - ➢ Less performance than customized hardware

→ Logic blocks should be designed as something in between

FPGA Categories



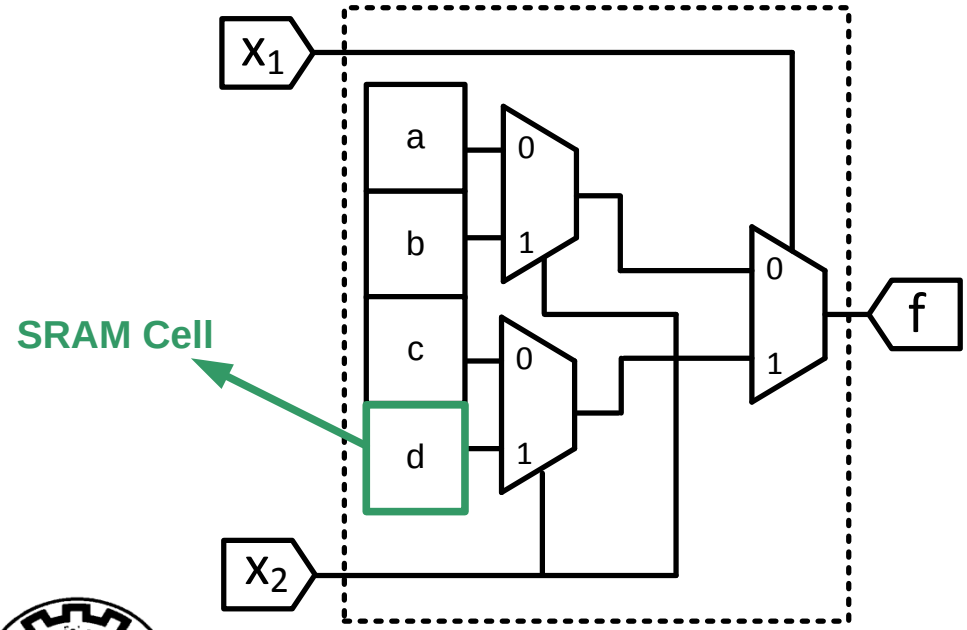
Logic Blocks (LUT-Based)



LUT-Based Logic Block (Used in SRAM-Based FPGAs)

- Lookup Table (LUT)
  - Uses a set of 1-bit storage elements to implement logic functions

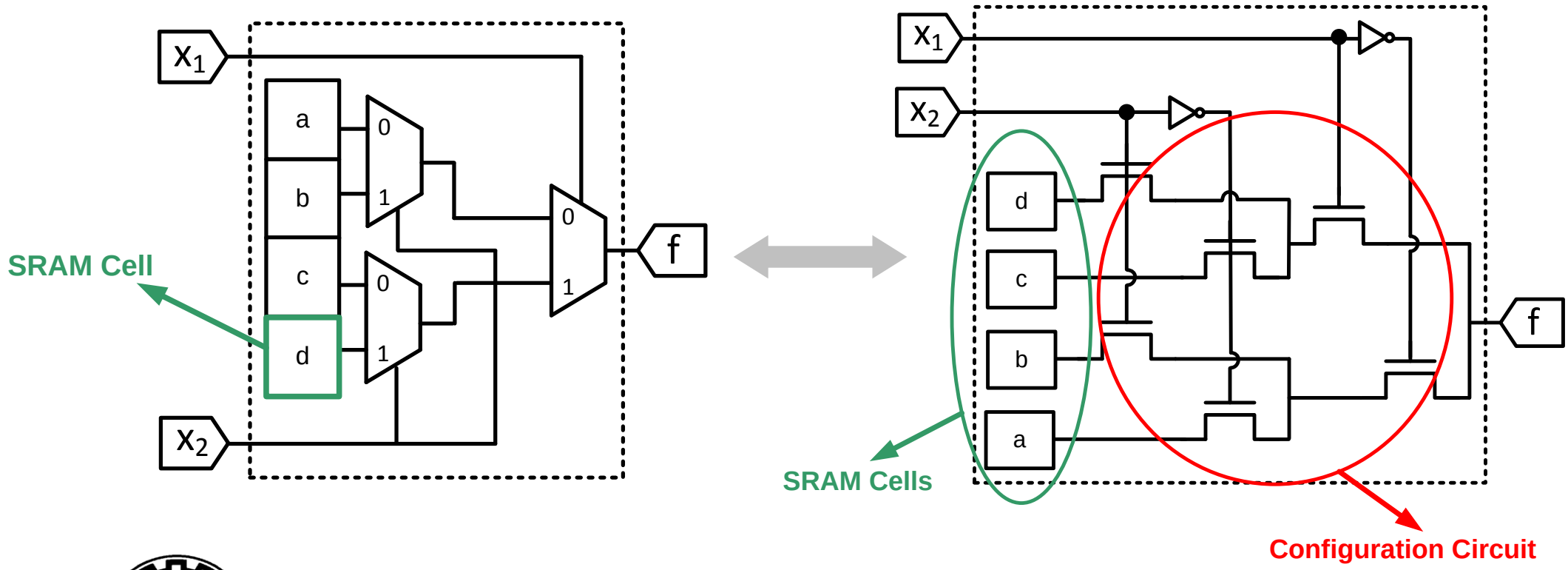
- Example:
  - A 2-input LUT
    - Capable of implementing any logic function of two variables



x_1	x_2	f
0	0	a
0	1	b
1	0	c
1	1	d

LUT-Based Logic Block

- Lookup Table (LUT) consists of:
  - Memory (SRAM Cells)
  - Configuration circuit that selects the proper memory bit

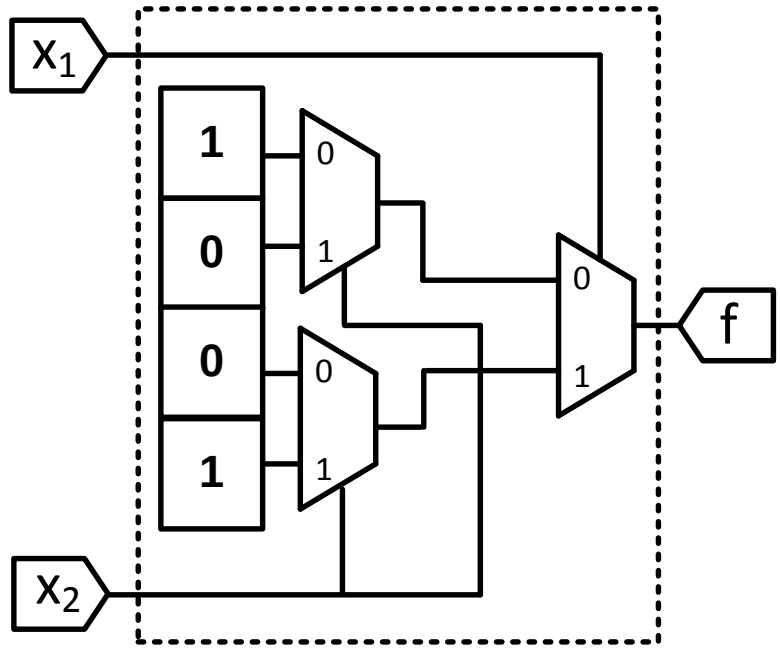


LUT-Based Logic Block

❖ Example:

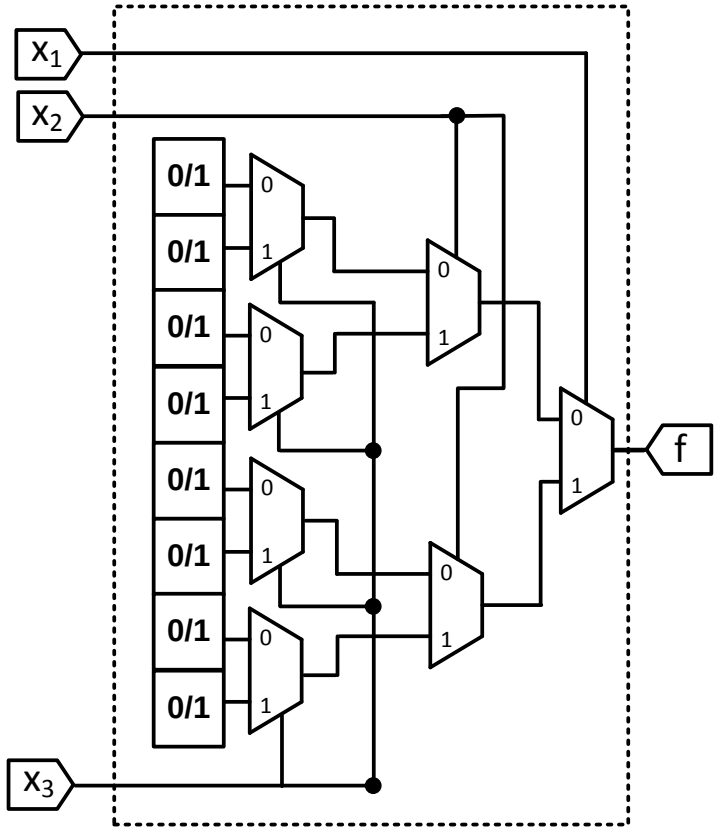
➢ $f = \bar{x}_1\bar{x}_2 + x_1x_2$

x_1	x_2	f
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**



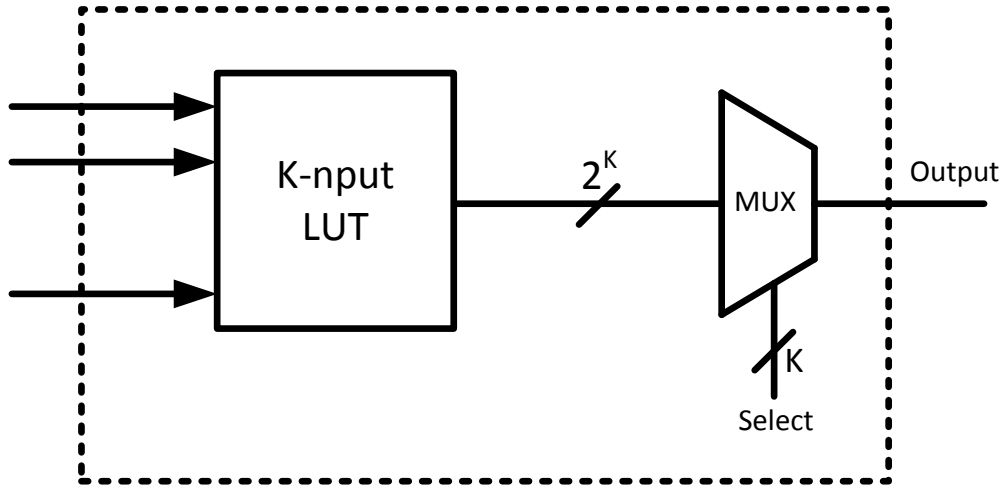
LUT-Based Logic Block

- Example:
  - A 3-input LUT
    - Capable of implementing any logic function of three variables



LUT-Based Logic Block

- In general: (A K-input LUT)
  - Capable of implementing any logic function of K variables
  - Can implement 2^{2K} different logic functions
  - The logic in LUT can be easily changed by changing the bits stored in the SRAM cells



- A typical logic block in commercial FPGAs has 4-6 inputs (6-input LUTs)

LUT-Based Logic Block

- A typical logic block in commercial FPGAs has 4-6 inputs
  - 4-input LUTs:
    - Xilinx XC4000
    - Xilinx Virtex family up to and including Virtex 4
    - Altera FLEX, Cyclone, Stratix I
  - Fracturable 6-input LUTs: (a.k.a **Adaptive Logic Module (ALM)**)
    - Xilinx Virtex 5
    - Altera Stratix II

LUT-Based Logic Block

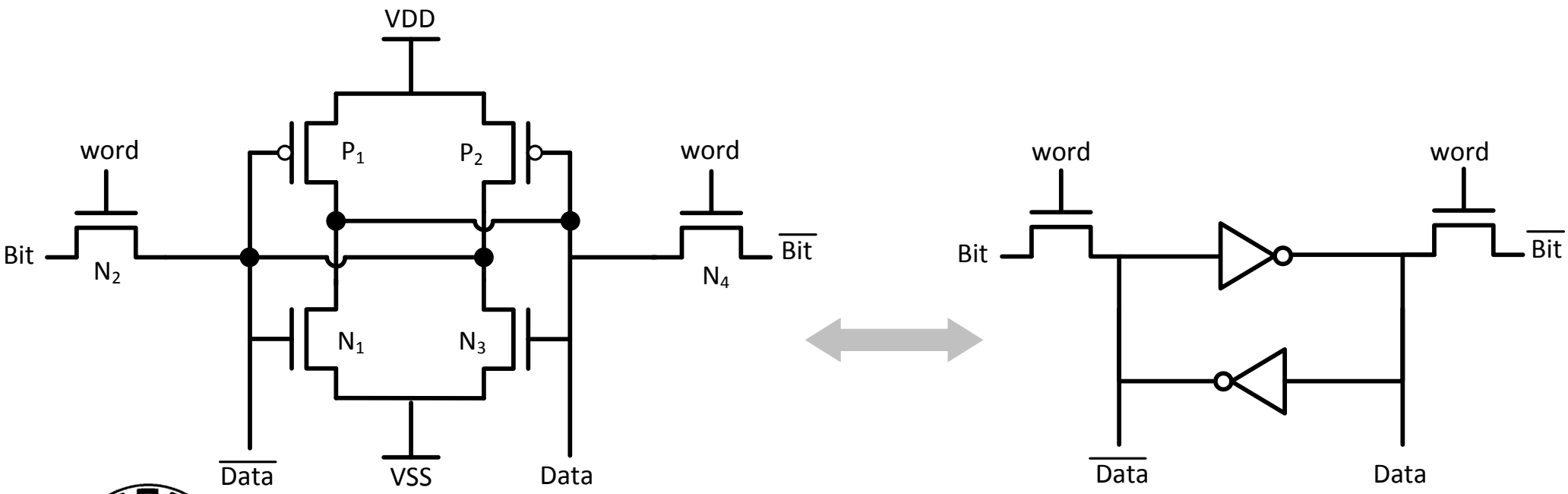
- Storage cells in the LUT are SRAM cells that are "volatile"
  - Lose their values when the power supply turns off
    - ⟹ Therefore, FPGA has to be re-programmed again

- Often a small memory chip, programmable read only memory (PROM) is used to hold their contents permanently

- LUT values are loaded automatically from the PROM when power is applied to the chip.

SRAM Cell used in LUT-based FPGAs

- The value is stored in the middle four transistors
- These four transistors form a pair of inverters connected in a loop
  - "word=0" ⟹ SRAM cell stores the value
  - "word=1" ⟹ Read/Write is performed



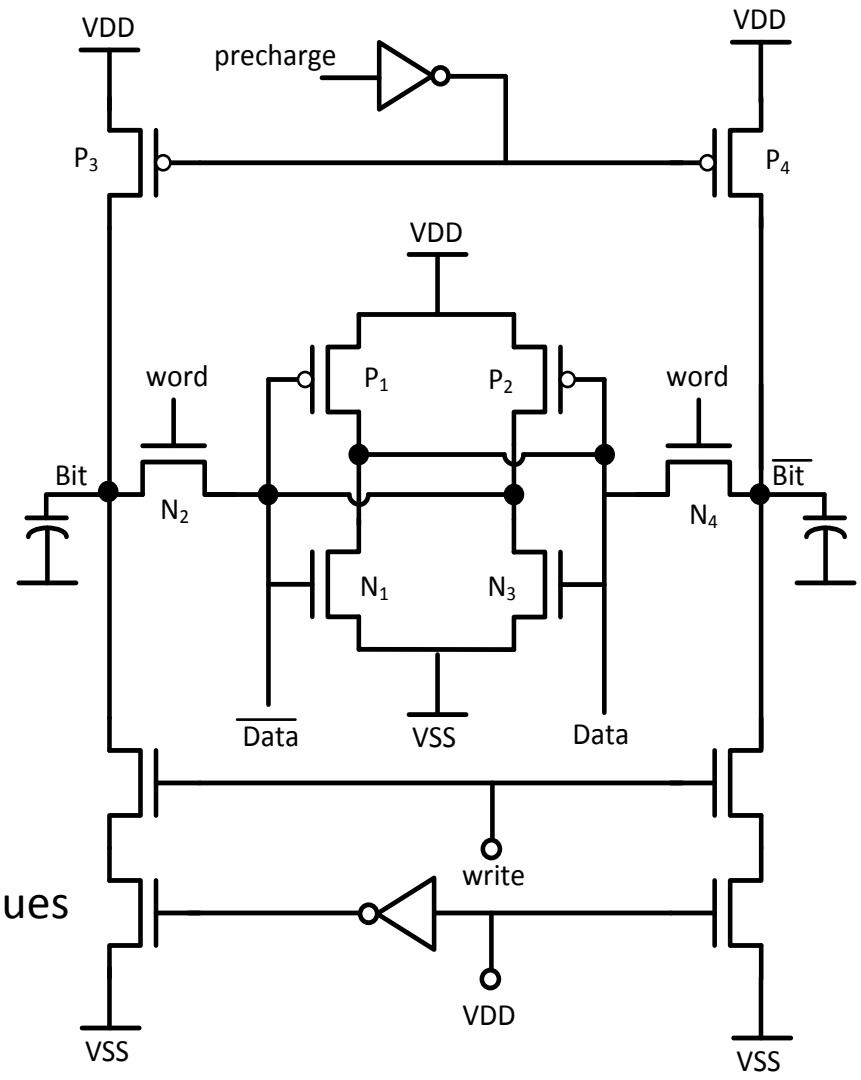
SRAM Cell Read/Write Operation

- **Read Operation:**
  - 1) Bit & $\overline{\text{Bit}}$ are precharged to VDD
    (Data=0 & $\overline{\text{Data}}$=1 or Data=1 & $\overline{\text{Data}}$=0)
  - 2) Then "word=1"
    - if Data=0 ⟹ Bit discharges through N_2 & N_1
    - if Data=1 ⟹ $\overline{\text{Bit}}$ discharges through N_4 & N_3

Read Stability Condition: $W_{N_1,N_3} > W_{N_2,N_4}$

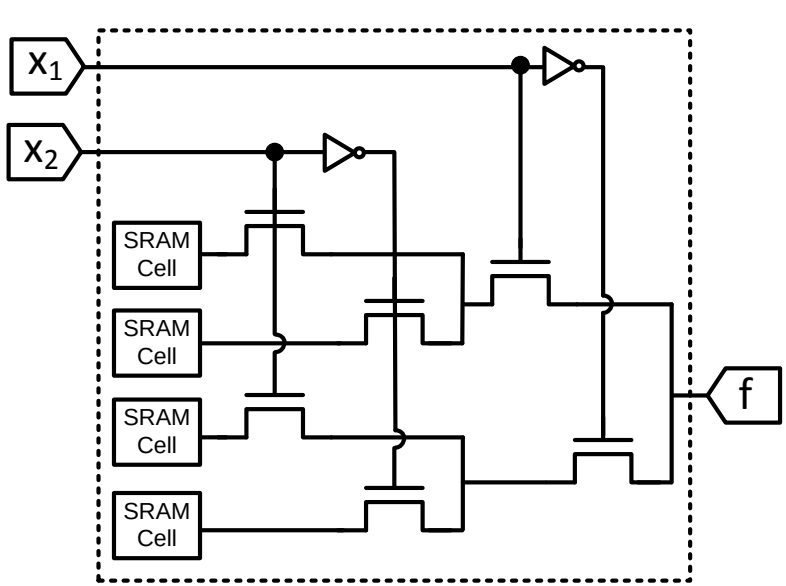
- **Write Operation**
  - 1) Bit & $\overline{\text{Bit}}$ are set to the desired values
    (e.g., Bit = 1 and $\overline{\text{Bit}}$=0 if "1" is to be written)
  - 2) Then "word" is set to "1"
    Charge sharing forces the inverter to switch values

Write Stability Condition: $W_{N_2,N_4} > W_{P_1,P_2}$

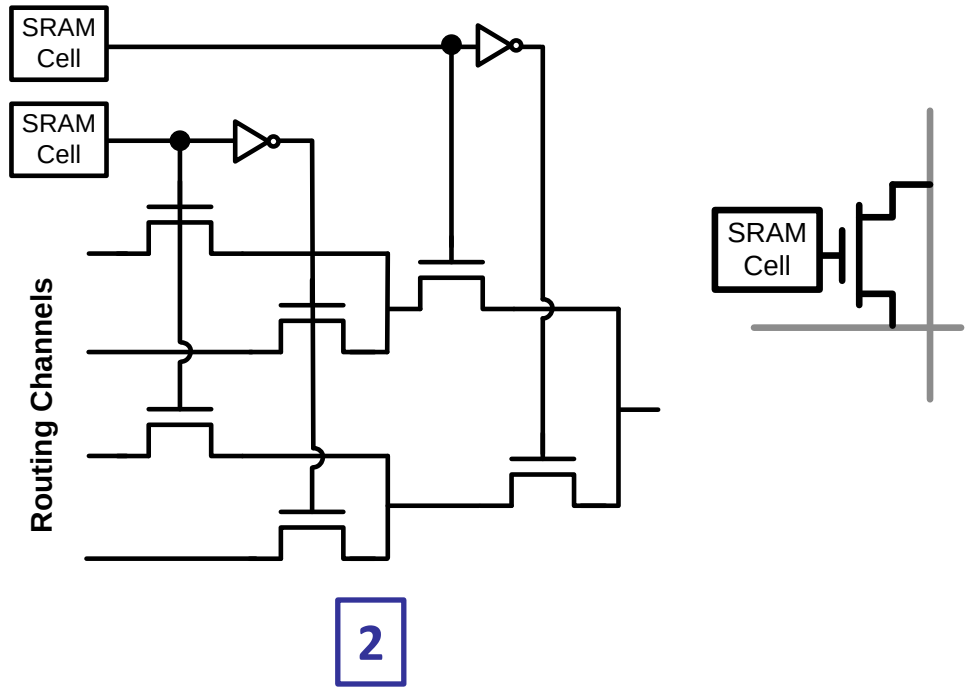


SRAM Cell

- **Two primary uses:**
  1. To store data in LUTs to implement logic functions
     - Uses only one side of the cell (e.g., Bit)
  2. To set the select lines in the programmable interconnects



1



2

LUT-Based Logic Block

❑ LUT-based logic blocks in most commercial FPGAs have some additional elements for efficient implementation (better than their LUT-based realizations)

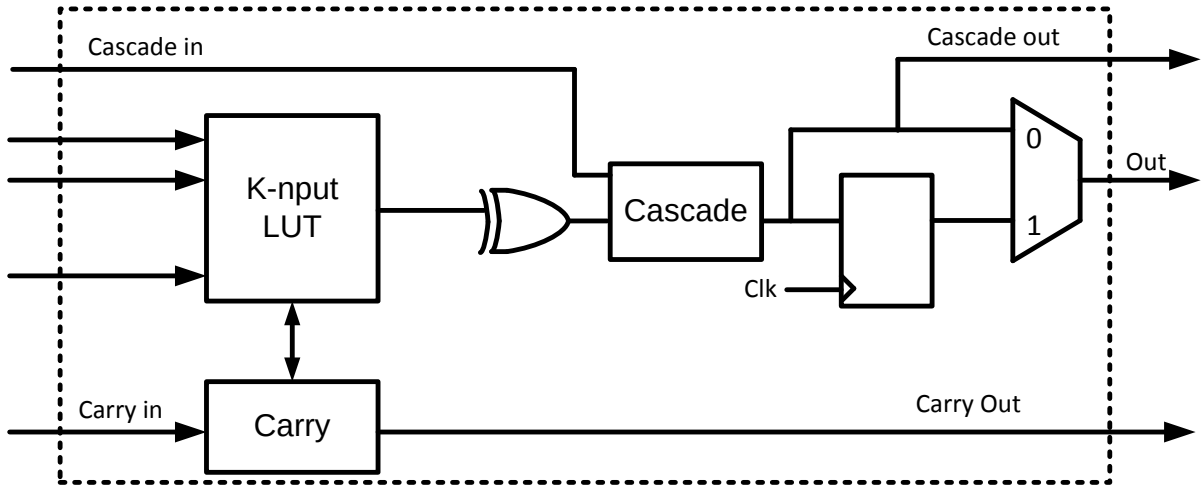
➢ **Extra elements inside LUT-based logic blocks: (Soft Logic)**

Coarse-grained

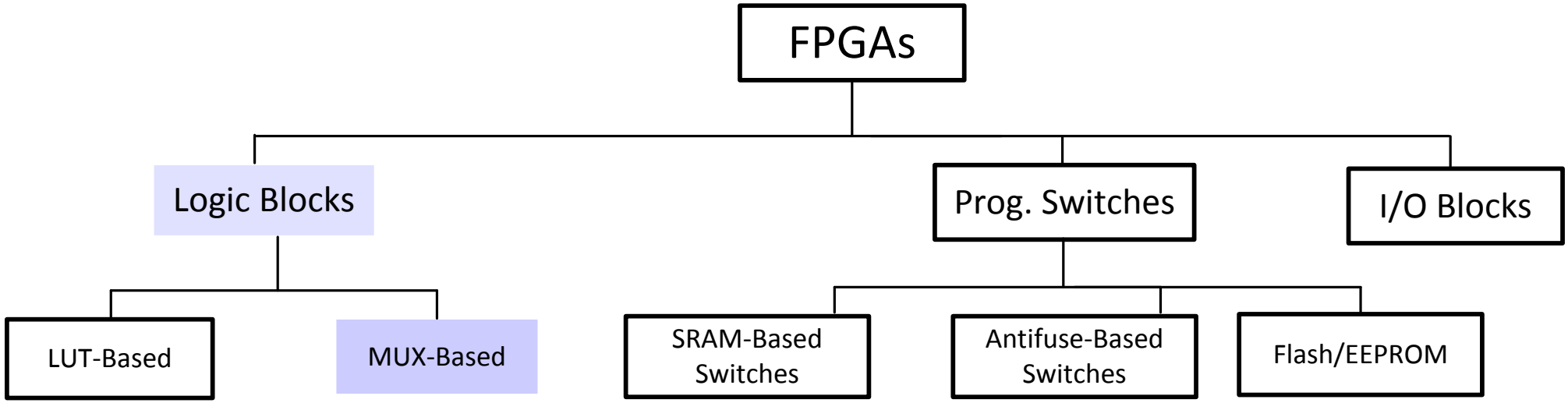
- LUT
- Flip-flops, MUXs, XOR

Fine-grained

- Blocks to support arithmetic carry, sum, and subtraction functions
- Cascade (to implement wide AND and larger functions)



Logic Blocks (MUX-Based)



MUX-Based Logic Block (Used in Antifuse-Based FPGA)

- The logic block in **antifuse-based FPGAs** are generally based on multiplexing
- Functions can be realized using MUXs based on Shannon's expansion
- Shannon's Expansion Theorem:

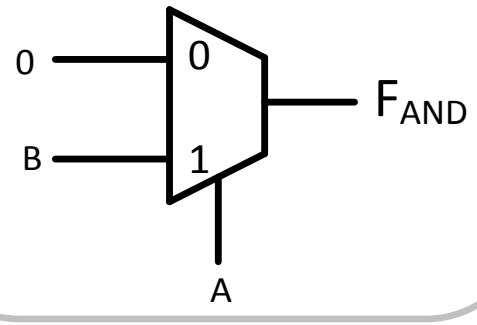
> Any logic function $f(x_1, x_2, \ldots, x_n)$ can be expanded in the form of:
>
> $$x_k . f(x_1, x_2, x_{k-1}, 1, x_{k+1}, \ldots, x_n) + x_k' . f(x_1, x_2, x_{k-1}, 0, x_{k+1}, \ldots, x_n)$$

- Example:
  - $F(A, B, C) = A'B + ABC' + A'B'C$
    $= A.F(1,B,C) + A'.F(0,B,C)$
    $= A(BC') + A'(B+B'C)$

MUX-Based Logic Block

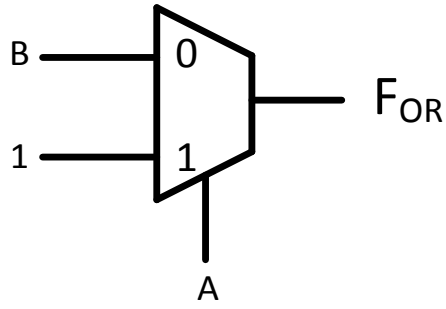
AND Gate:

$F_{AND} = A.B = A.B + A'0$



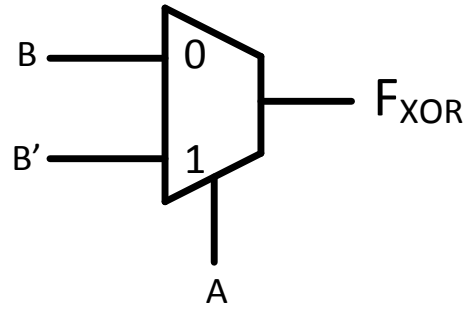
OR Gate:

$F_{OR} = A+B = A.1 + A'B$



XOR Gate:

$F_{AND} = AB'+A'B$



MUX-Based Logic Block

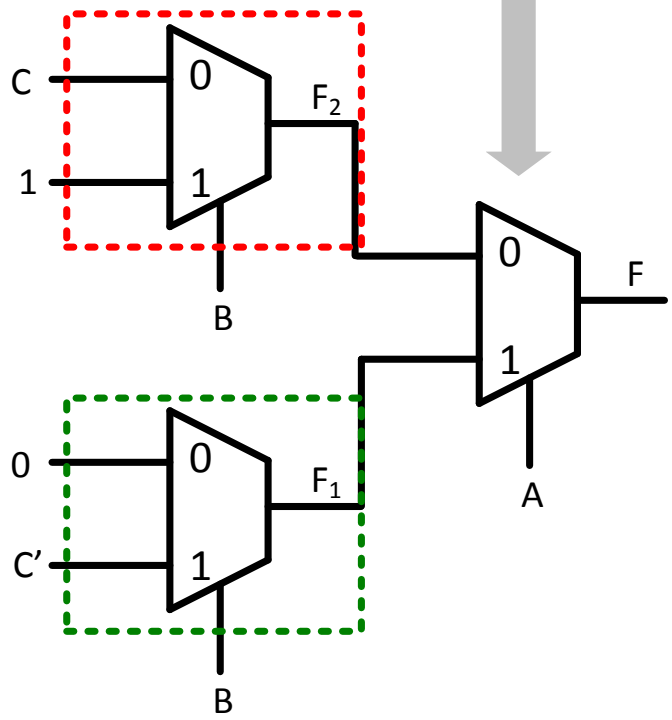
❖ Example:

➢ F(A, B, C) = A'B + ABC' + A'B'C

= A(BC') + A'(B+B'C) = A.F_1 + A'.F_2

➢ F_2 = B+B'C = B.1 + B'.C

➢ F_1 = BC' = BC' + B'.0

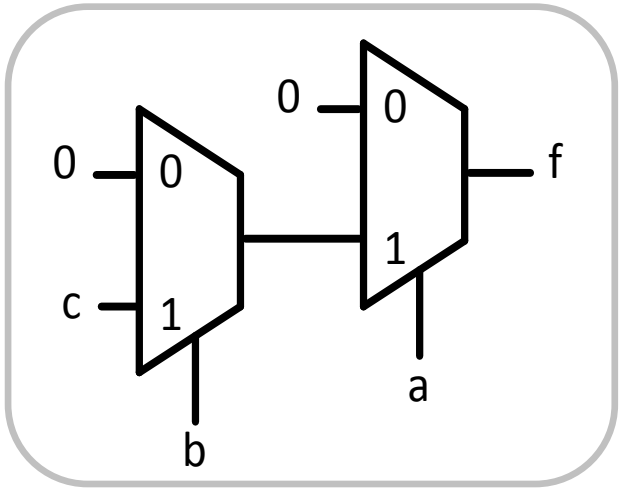


MUX-Based Logic Block

- The logic block in **antifuse-based FPGAs** are generally based on multiplexing

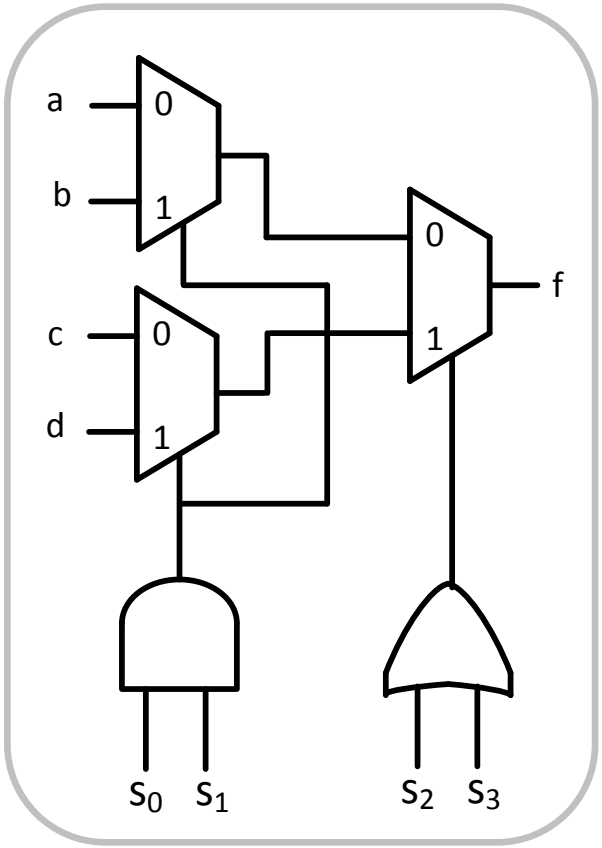
❖ Example:

- Three-input AND function
  - f = a.(b.c+b'.0)+a'(0)



MUX-Based Logic Block

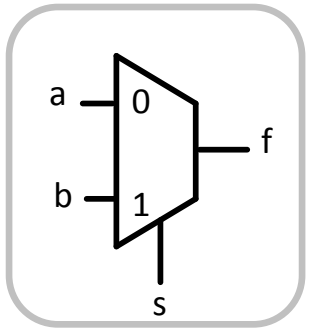
- A more complex logic block



S_0	S_1	S_2	S_3	f
0	0	0	0	a
0	0	0	1	a
0	0	1	0	a
0	0	1	1	b
0	1	0	0	c
0	1	0	1	c
0	1	1	0	c
0	1	1	1	d
1	0	0	0	c
1	0	0	1	c
1	0	1	0	c
1	0	1	1	d
1	1	0	0	c
1	1	0	1	c
1	1	1	0	c
1	1	1	1	d

MUX-Based Logic Block

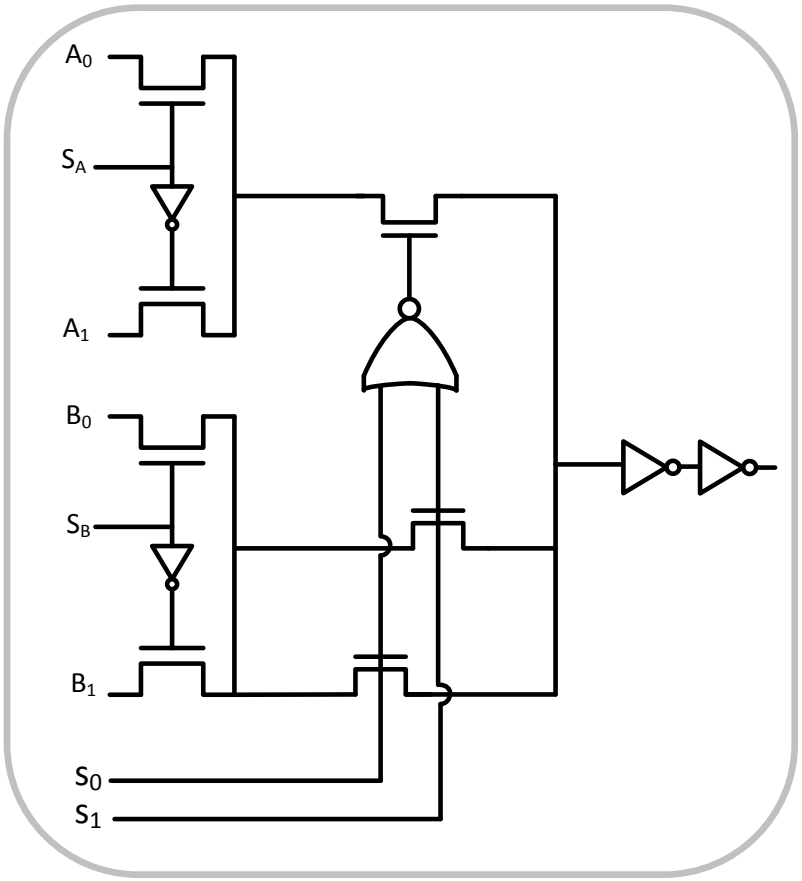
- MUX-Based configurable logic block



a	b	s	f
0	0	0	0
0	X	1	X
0	Y	1	Y
0	Y	X	XY
X	0	Y	XY’
Y	0	X	X’Y
Y	1	X	X+Y
1	0	X	X’
1	0	Y	Y’
1	1	1	1

MUX-Based Logic Block

- MUX-Based configurable logic block (can also be used to build latches/registers)



A_0	A_1	B_0	B_1	S_A	S_1	S_0	S_B	OUT
1	1	0	1	A	0	B	A	(AB)’
0	1	0	1	0	0	B	A	(AB)’
0	1	0	1	0	B	0	A	(AB)’
0	1	0	1	0	0	A	B	(AB)’
1	0	0	1	A	0	B	A	A^B
1	0	0	1	A	B	0	A	A^B
Q	0	D	0	CLR	CLK	0	CLR	Latch
Q	0	CLR	0	CLR	CLK	0	D	Latch

Comparison b/w MUX-based and LUT-based

- **LUT-based Logic Block (LB) using SRAM cells:**
  - An n-input LUT function requires 2^n SRAM cells
  - Each SRAM cell requires 8 transistors
    - e.g., a 4-input function requires 16x8=128 transistors
  - Decoding circuitry is also required
    - e.g., decoder for a 4-input LUT is a MUX with 96 transistors
  - Delay of LUT is independent of the function implemented and is dominated by the delay through the SRAM cell (same for all functions!)
  - SRAM consumes power even when its inputs do not change. The stored charge in the SRAM cell dissipates slowly.
  - LUT-based LB is considerably more expensive than a static CMOS gate.
  - Easier implementation through loading configuration bits

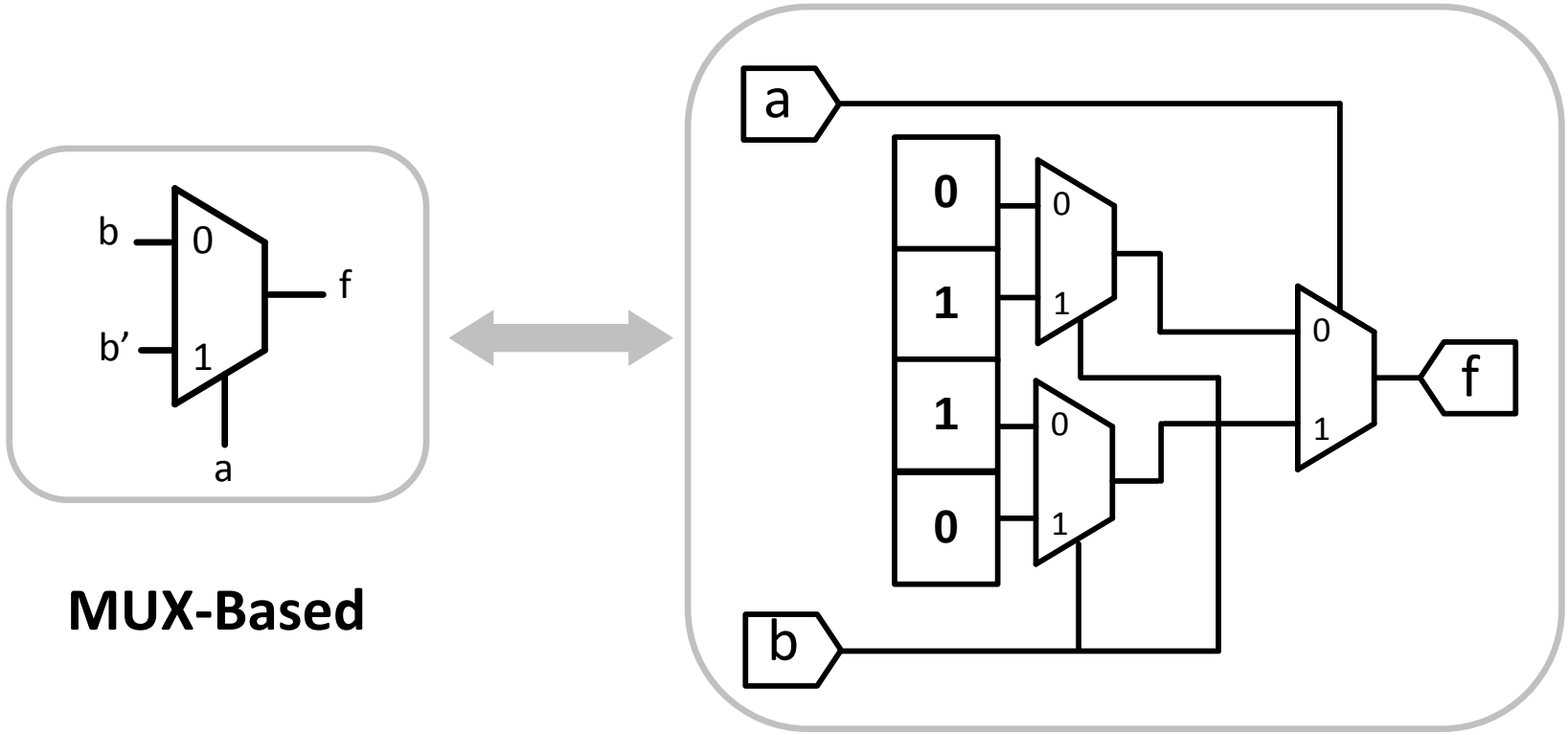
Comparison b/w MUX-based and LUT-based

- **MUX-based LB using Static CMOS:**
  - Number of transistors a function of number of inputs and the function
    - An n-input NAND requires 2n transistors
    - An n-input XOR is more complicated
  - The delay of a static gate depends on the number of inputs, function, and the transistor sizes
  - MUX-based implementation consumes no power while the inputs are stable (ignoring the leakage power)
  - Synthesizer has a hard time figuring out how to implement a certain function into the given MUX structure

Comparison b/w MUX-based and LUT-based

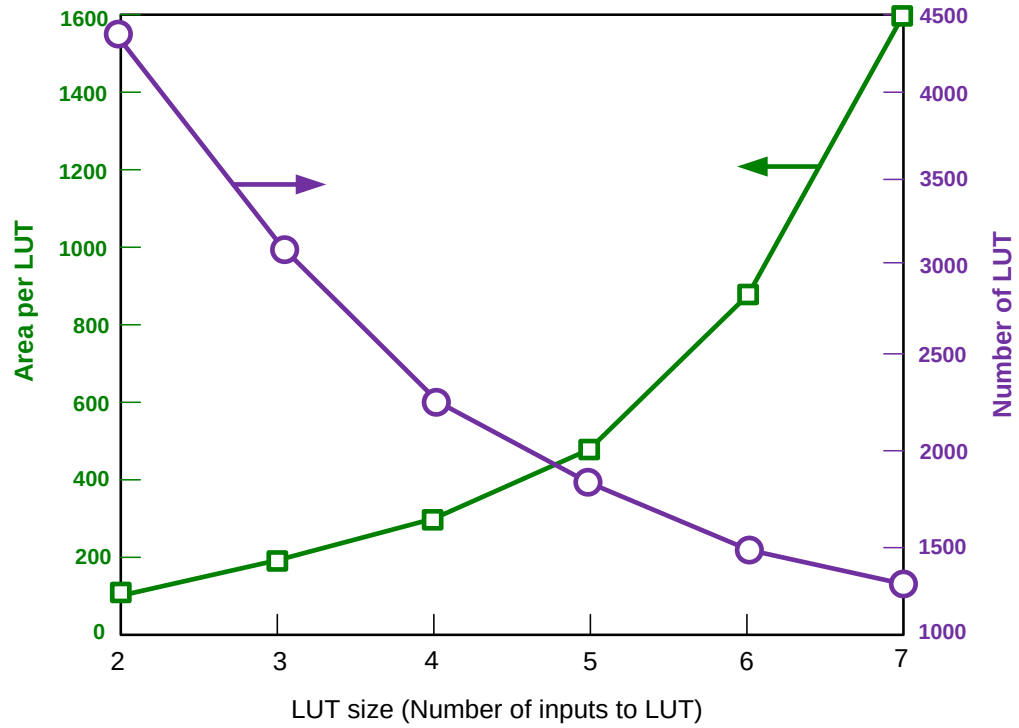
❖ **Example:**

➢ Implementation of an XOR in two cases:



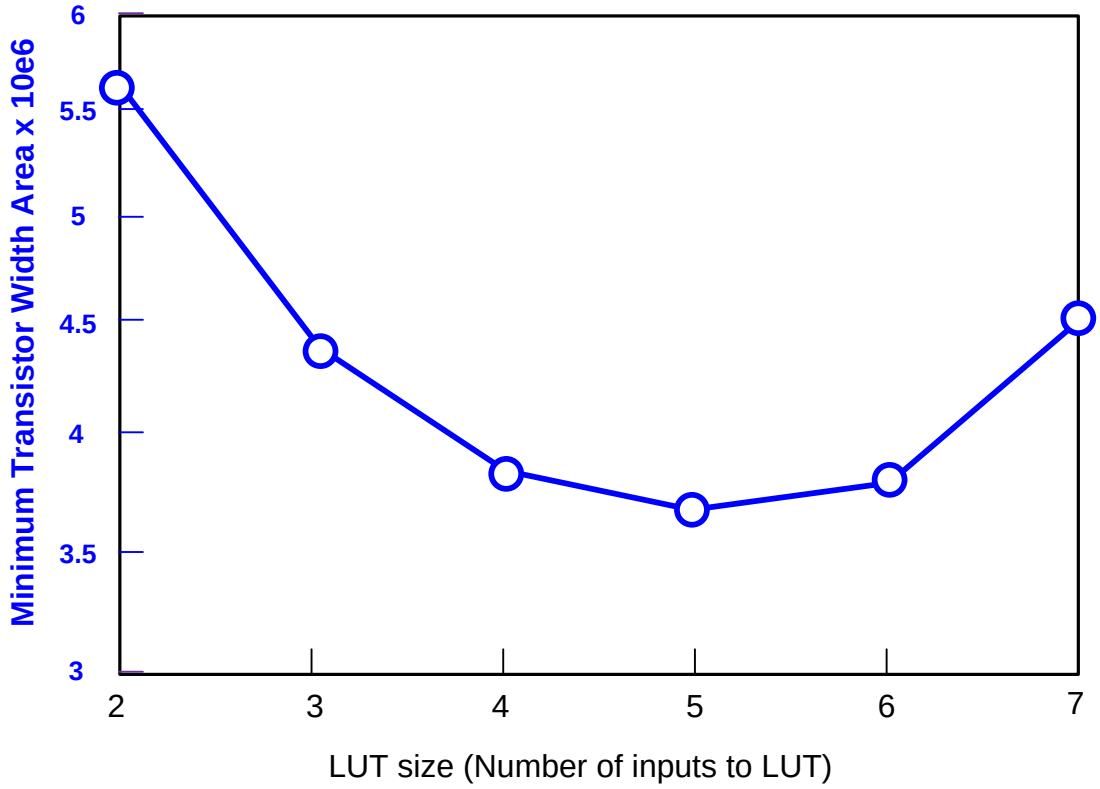
Logic Block Design: Area Trade-off

- As the functionality of a logic block (LB) increases:
  - Fewer LBs are needed to implement a given design (**good**)
  - Its size and the amount of routing increases (**bad**)
    - Number of bits in a K-input LUT is 2^K (exponential area increase with K)



Logic Block Design: Area Trade-off

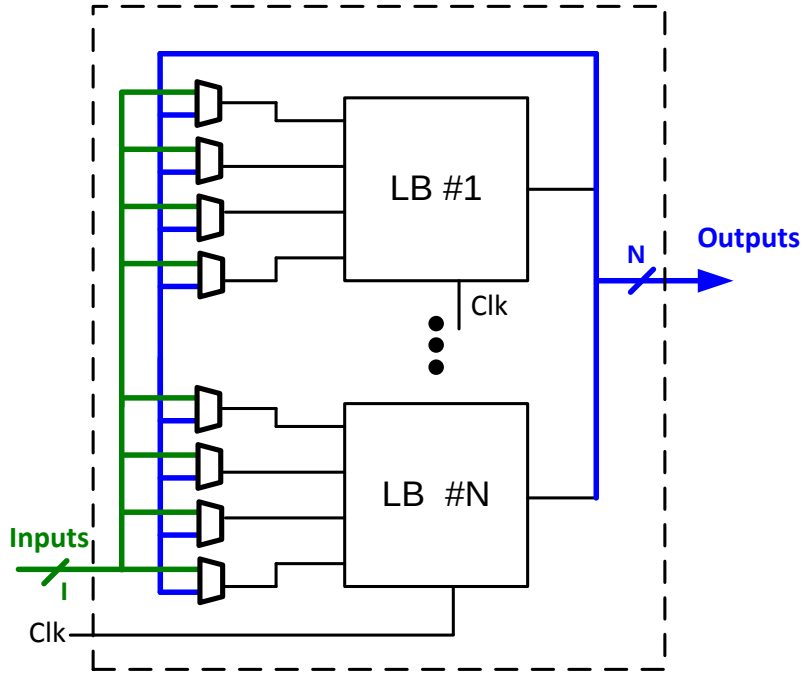
- Total area as a function of LUT size: (product of two previous curves)



4 to 6-input LUT size is optimal in terms of area!

Logic Block Design: Granularity

- An alternative is to change the granularity of each logic block
- It means to integrate a few logic blocks in a cluster (Clusters of LUTs)
- Logic blocks in a cluster are programmably connected together by a local interconnect structure
- This idea is used in most current commercial FPGAs



Logic Block Design: Granularity

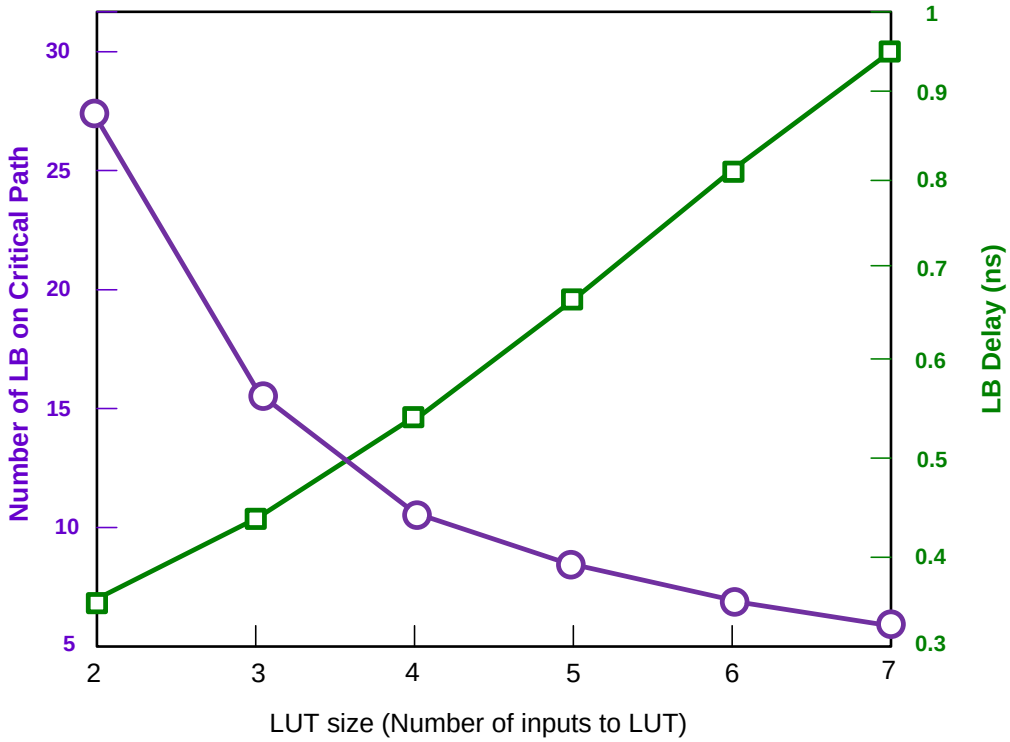
- In this approach the size of the logic and internal routing grows quadratically as opposed to the exponential growth for the LUT size

- More area per logic block with less area increase

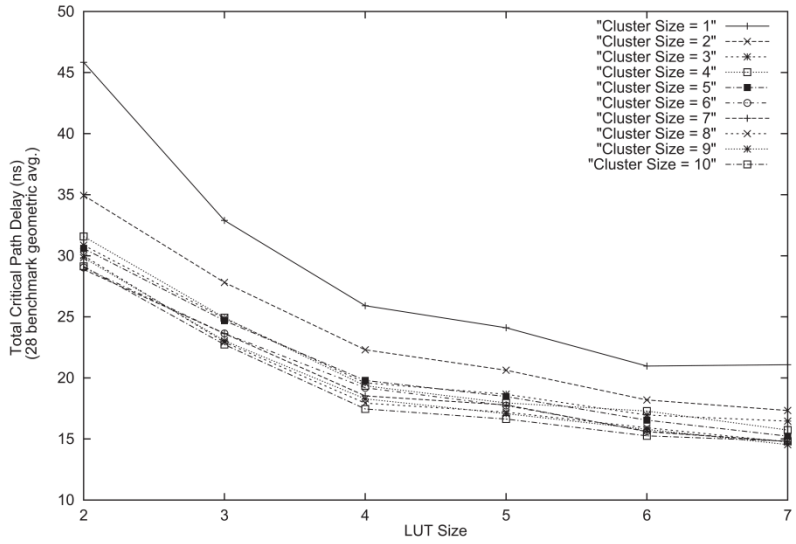
- There is also pin saving as follows:
  - Number of pins needed for N basic logic block with K-input LUT: KN
  - Number of pins needed for a cluster of N K-input LUTs: K(N+1)/2
    - Thus, there are fewer inputs to the cluster from the external inter-cluster routing than the total number of inputs to the basic logic blocks inside the cluster

Logic Block Design: Speed Trade-off

- As the functionality of a logic block (LB) increases:
  - Fewer LBs are used on the critical path (**good**)
  - Less inter-logic routing ⟶ less delay ⟶ higher speed performance
  - The internal delay of each LB increases (**bad**)



Logic Block Design: Speed Trade-off



- **Observations:**
  - Increasing the cluster size, decreases the critical path (up to 3 significant)
  - Higher LUT size results in less delay on the critical path
    - Not too different after LUT size of 4-5
- Optimal point considering both the area and speed optimizations:

LUT size: 4-input or 5-input

Cluster size: 2-4

Logic Block Design in Heterogeneous FPGAs

- If there is a dedicated specific-purpose hard circuit on the FPGA for a function, it has superior area, speed and power consumption over its implementation in general purpose logic blocks.

- For instance, a Flip-Flop (FF) can be built using LUTs and gates but it can also be explicitly designed or customized inside a logic block, much more efficient.

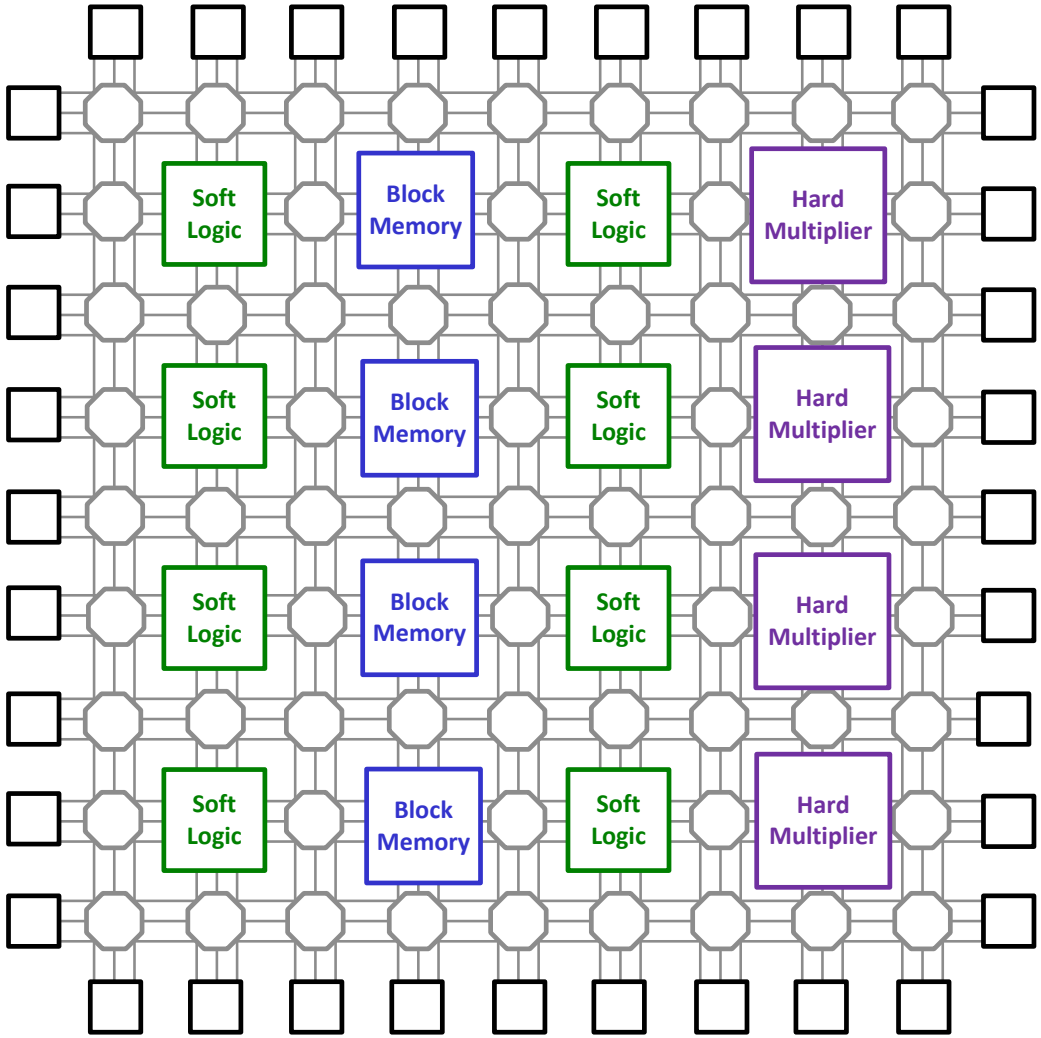
- In all commercial heterogeneous FPGAs, various dedicated blocks are designated to improve area and speed efficiency.

- What kind of specific functions should be included?

Logic Block Design in Heterogeneous FPGAs

- Heterogeneity may exist in two levels:
  - **Extra elements inside general purpose logic blocks: (Soft Logic)**
    - Flip-flops
    - MUXs
    - XOR
    - Blocks to support arithmetic carry, sum, and subtraction functions
  - **Different types of blocks: (Hard Logic)**
    - Multi-bit block RAMs (first used in FLEX 10K)
    - Multiply-accumulation (MAC) blocks (e.g., in Startix I, II, III)
    - Hard multiplier blocks (e.g., in Xilinx Virtex families)

Logic Block Design in Heterogeneous FPGAs



Soft Logic in Heterogeneous FPGAs

- Carry logic modules are dedicated blocks, provided to help implement faster addition operations

- The carry over is passed b/w internal LUTs via dedicated routing
  - General routing is avoided to achieve less signal delay

- Normally an XOR gate is also included in the carry chain to generate the SUM to build an adder

Memory Blocks in Heterogeneous FPGAs

- First appeared in Altera FLEX 10K
- Flexibility of being configured in various aspect ratios is crucial
  - b/c different applications need different block sizes and aspect ratios
  - e.g., in Flex 10K a 2K memory in (1x2048), (2x1024), (256x8)
- Covers a significant fraction of the FPGA die area
- More important in larger systems
- Most complementary FPGAs employ dual-port memory blocks

Computation-Oriented Blocks in Heterogeneous FPGAs

- **Most common: Hard multiplier**
  - e.g., Virtex II contains 18x18 2's complement multipliers
  - Startix I contains a single 36x36 multiplier (can also be broken into eight 9x9 multipliers and an adder to sum the results)
- If multipliers are not used by an application their blocks are wasted
- In order to avoid waste of resources:
  - Multiple sub-families of a device with different ratios of soft logic to hard logic are created (so choose the one that fits the best)
  - For example, Virtex 4/5 have three sub-families:
    - More soft logic and memory
    - Focus on arithmetic unit
    - Focus on high-speed interface

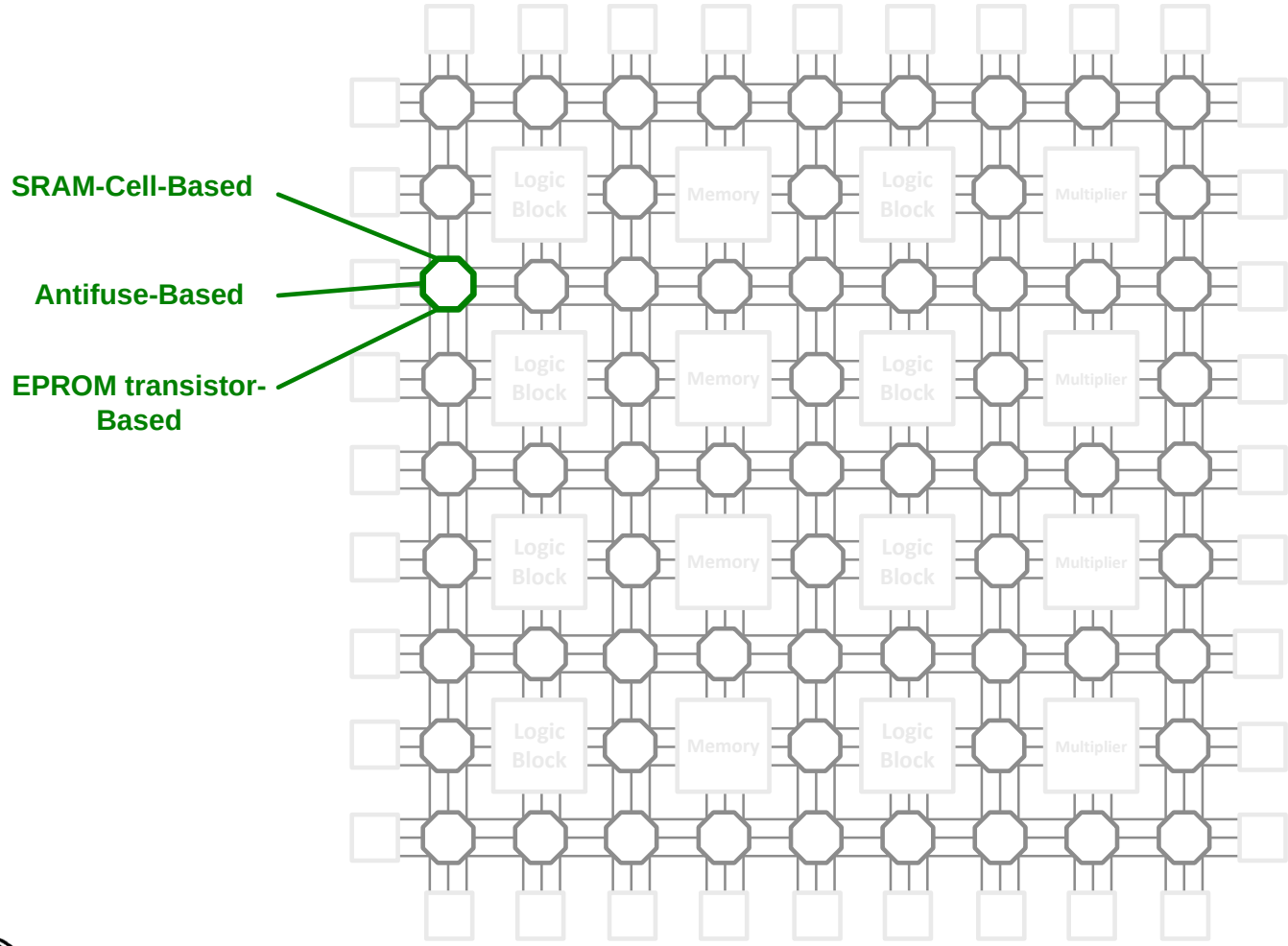
Microprocessors in Heterogeneous FPGAs

- Microprocessors are vital in many digital systems
- Often used in conjunction with FPGA logic
- It is a great idea to integrate it with FPGAs on a single die
- For example:
  - Xilinx Virtex II Pro FPGAs have 1, 2, or 4 IBM power PC cores integrated with Virtex II logic fabric
  - Virtex 4, 5 subfamilies also support power PC cores on the die

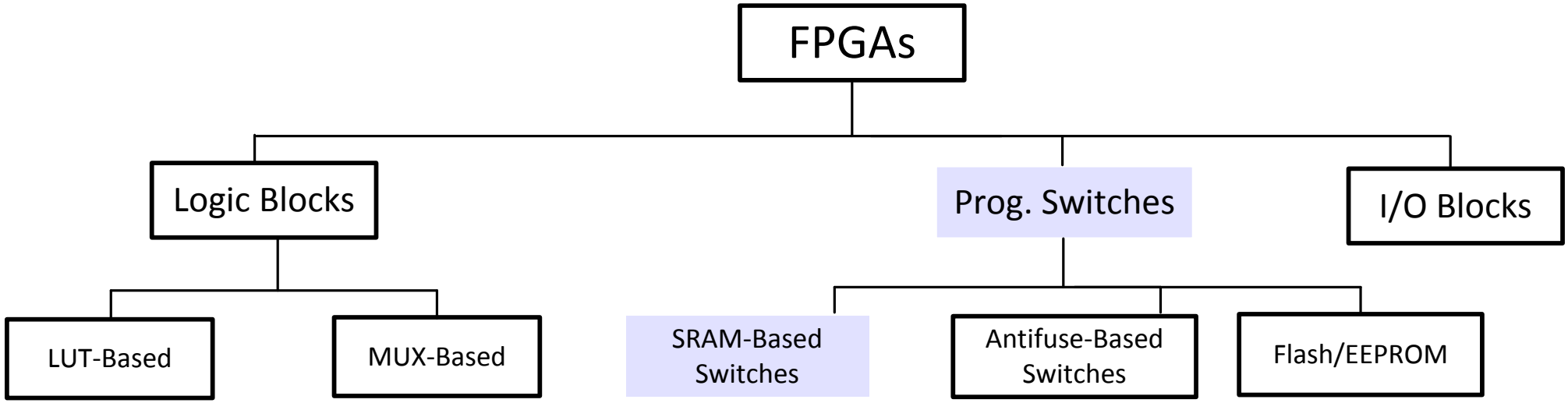
Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- **Field-Programmable Gate Array (FPGAs)**
  - Logic Blocks
  - **Programmable Routing Switches**
  - I/O Pads
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

Programmable Switches

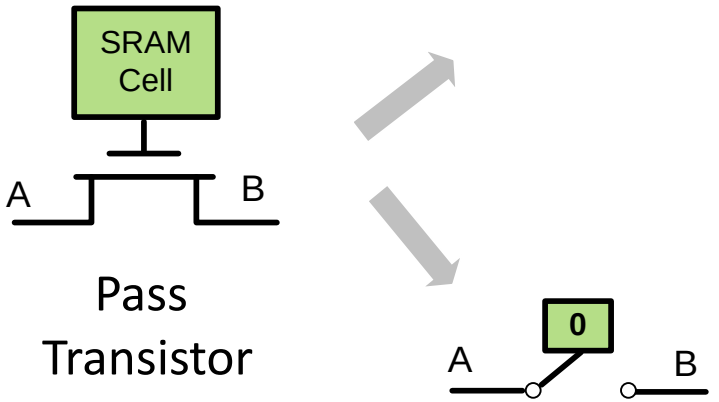


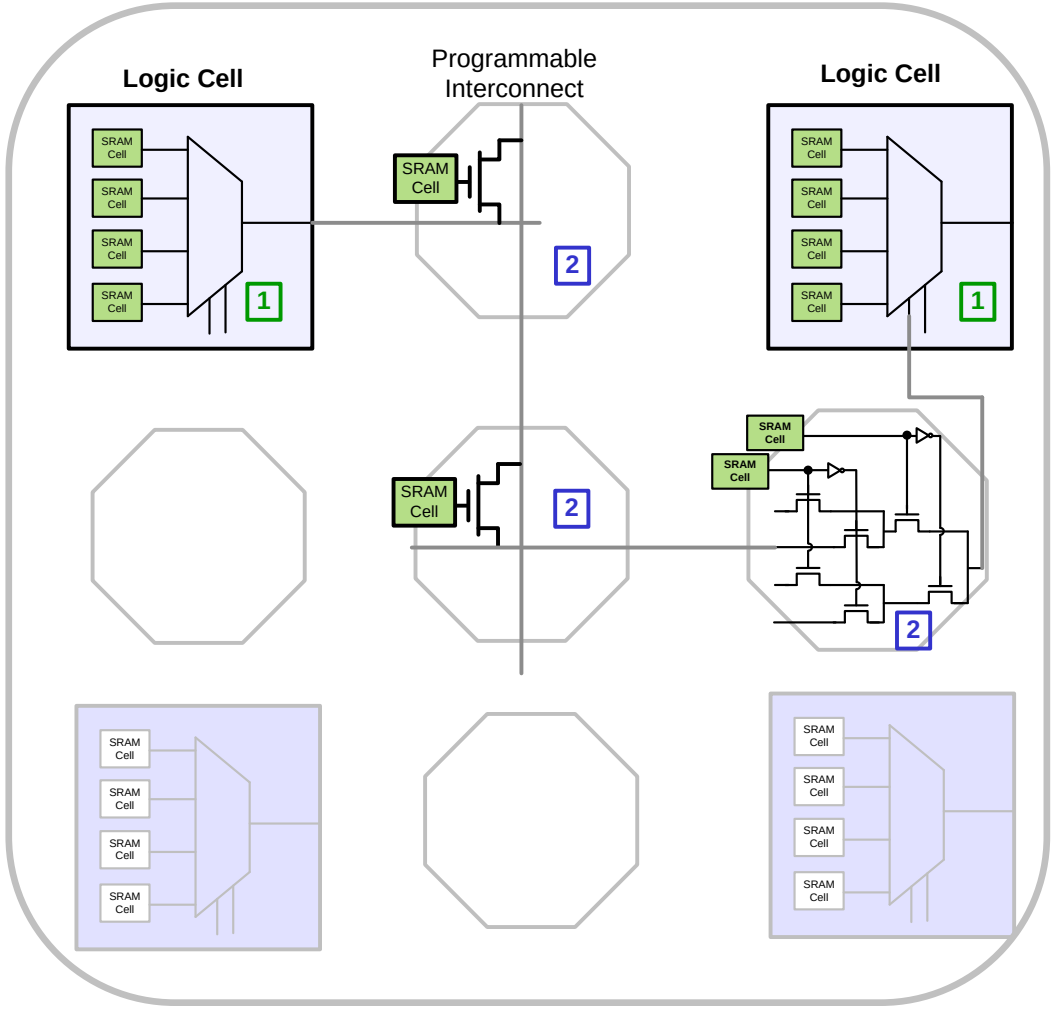
SRAM-Based Programmable Switches



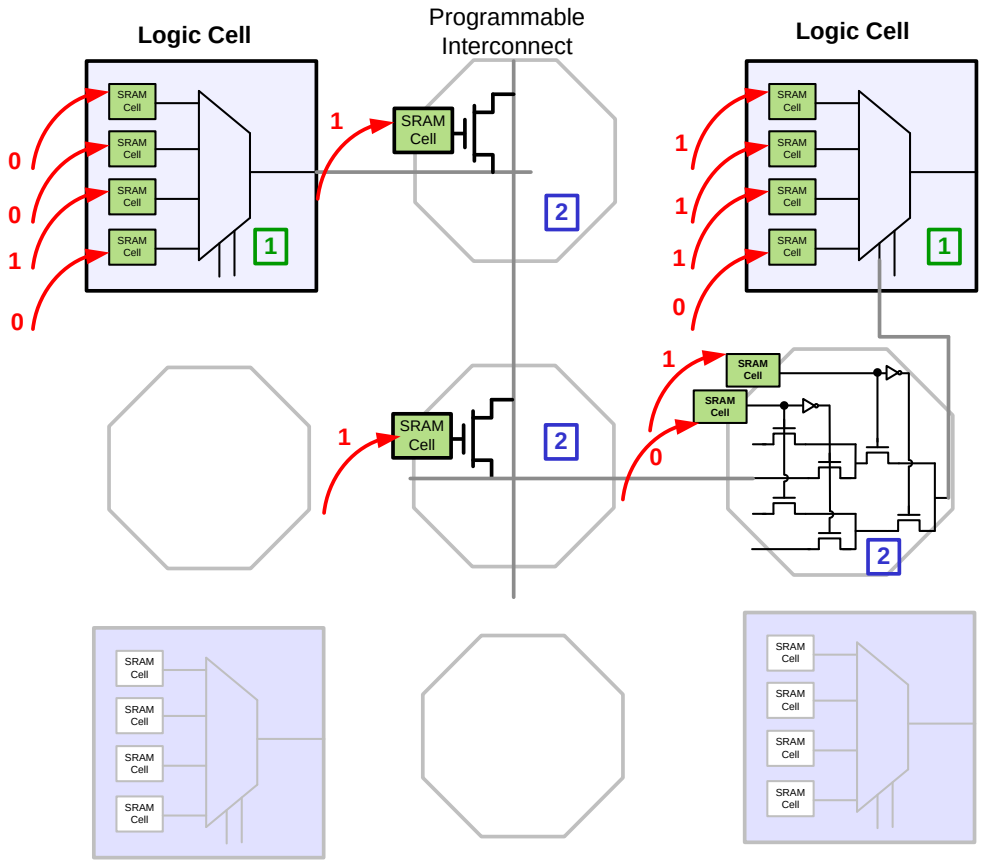
SRAM-Based Programmable Switches

- SRAM Cell is used both in logic blocks and the Prog. Interconnections:





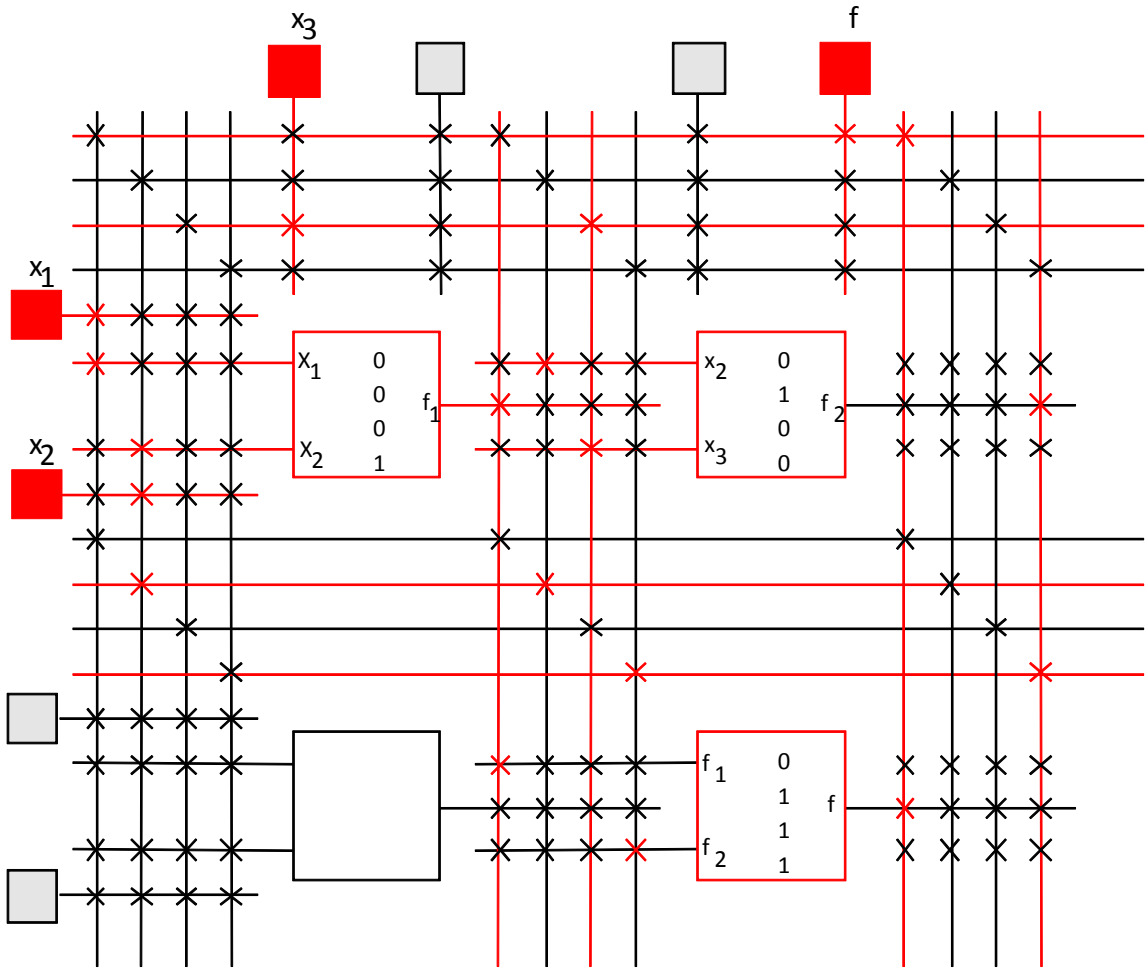
SRAM-Based Programmable Switches



When programming, configuration bits are loaded into SRAM cells both in the LUTs and interconnection switches

Programming FPGAs

- Programming an FPGA by configuring Logic Blocks & Routing



Configuration of SRAM-based FPGA

- SRAM-based FPGAs are reconfigured by changing the content of the SRAM cells in LUTs and programmable interconnect

- A few pins are dedicated for configuration

- Two ways of configuration:
  1. Download the configuration bits directly from PC using a download cable
     - Good for prototyping and debugging mode
     - Not reliable in the production mode
  2. Store configuration bits in PROMs on the PCB with the FPGA
     - Upon power-up they are loaded into the FPGA

FPGA Interconnect Design

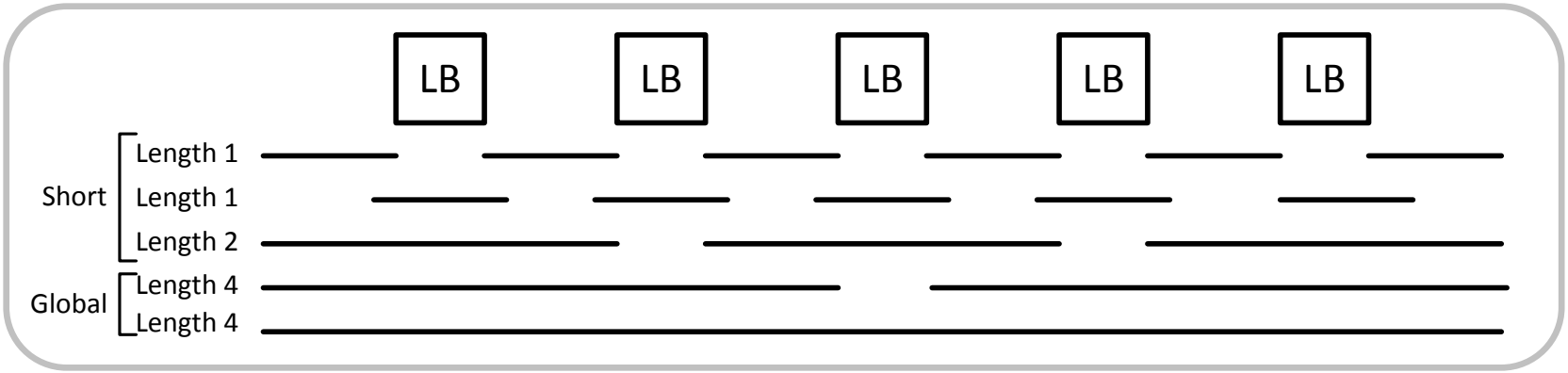
- Interconnect design is really important b/c the most area in an SRAM-based FPGA is consumed by the routing switches.

- Interconnects are organized in wiring channels or “routing channels”

- A typical FPGA has many different kinds of interconnect to be fully customized for different delay/speed requirements:
  - Short wires
  - Global wires
  - General purpose wires
  - Clock distribution network

FPGA Interconnect Design

- In order to make all required connections b/w logic blocks efficiently, FPGA routing channels have wires of a variety of lengths (segmentation)
- **Segmentation:**
  - **Short wires:** Connect only local logic blocks (e.g., the carry chain in LBs)
    - Do not take up much area and have small delay
  - **Global wires:** Designed for long-distance communication
    - May have built-in electrical repeaters to reduce delay



SRAM-Based Programmable Interconnect

- Interconnect design in SRAM-Based FPGAs is tricky b/c the circuitry can introduce significant delay and cost a large silicon area.
- Two options:
  - Pass transistor
  - Three-state buffer (larger but provides amplification)

Pass Transistor

Tri-state Buffer

SRAM-Based Programmable Interconnect

- These elements introduce delay to the interconnect
- **Objective:** reduce the delay, How?
  1. Increase the width of the transistors
     - Less delay (**good**)
     - More silicon area (**bad**)
  2. Increase the wire width
     - Less resistance (**good**)
     - More capacitance (**bad**)
- How much we should increase the width?

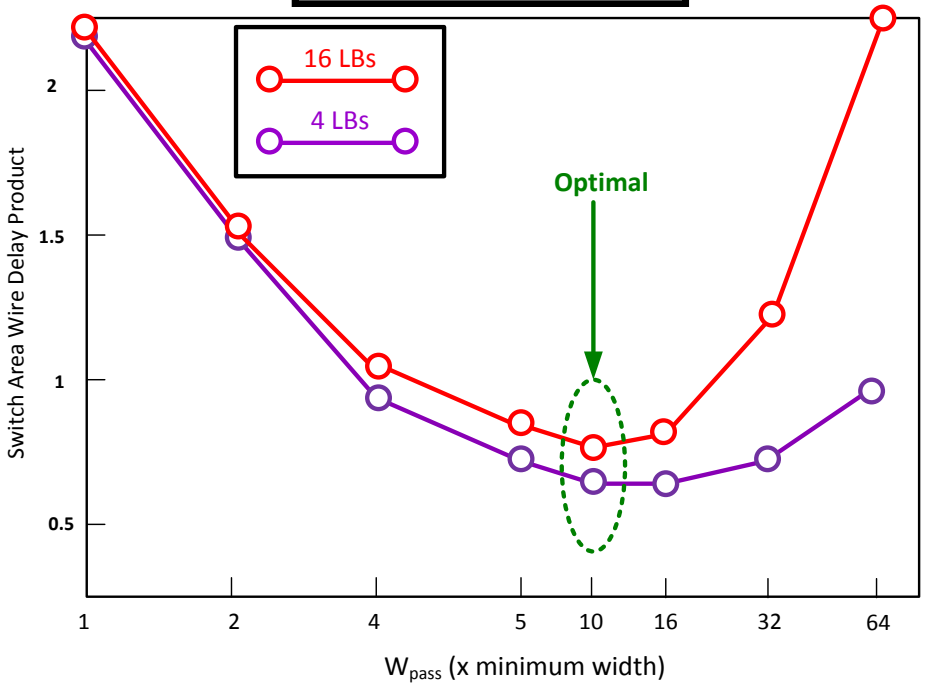
Define a metric: Area-delay product

(To consider both restrictions)

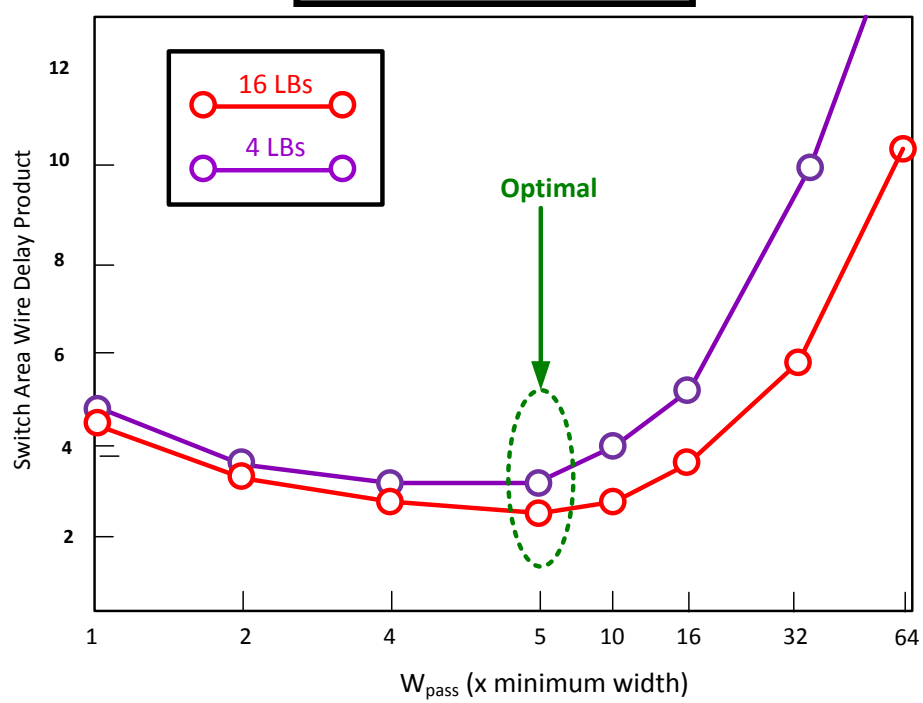
SRAM-Based Programmable Interconnect

- Consider 4 and 16 logic blocks

Pass Transistor



Tri-state Buffer



The tri-state buffer requires smaller transistors (b/c it provides amplification)

SRAM-Based Programmable Switches

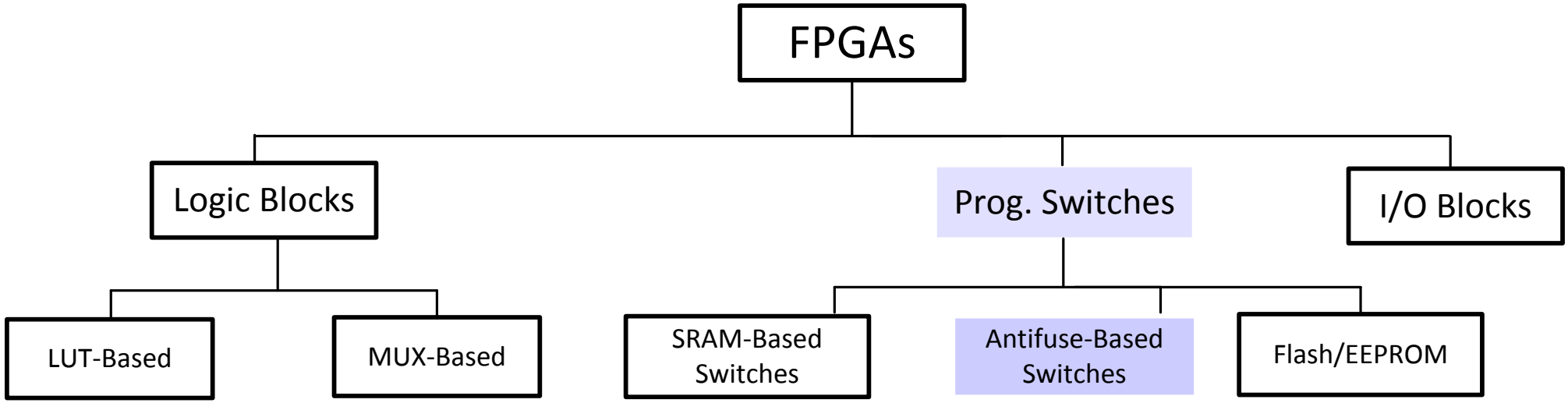
❑ Advantages:

- ➢ Re-programmability (infinite number of times)
- ➢ Use of standard CMOS fabrication process technology
  - ▪ Use of the latest CMOS technology
  - ▪ Benefits from increased integration, higher speed, lower dynamic power

❑ Drawbacks:

- ➢ **Size**: SRAM cell requires 6 transistors
- ➢ **Volatility**: an external device (like an EPROM) is needed to permanently store the configuration bits when the device is powered down (extra cost)
- ➢ **Non-ideal pass transistors**: SRAM cells rely on pass transistors that have large on-resistance and capacitance load
- ➢ **Reliability**: the bits in the SRAM are susceptible to theft

Antifuse-Based Programmable Switches



Antifuse-Based Programmable Switches

- The programmable element is an antifuse
- Programmed by applying a voltage across it
  - Normal condition: high resistance link
  - When programmed (blown): low resistance (20-100 Ohm)
    - Permanently programmed (unlike SRAM)

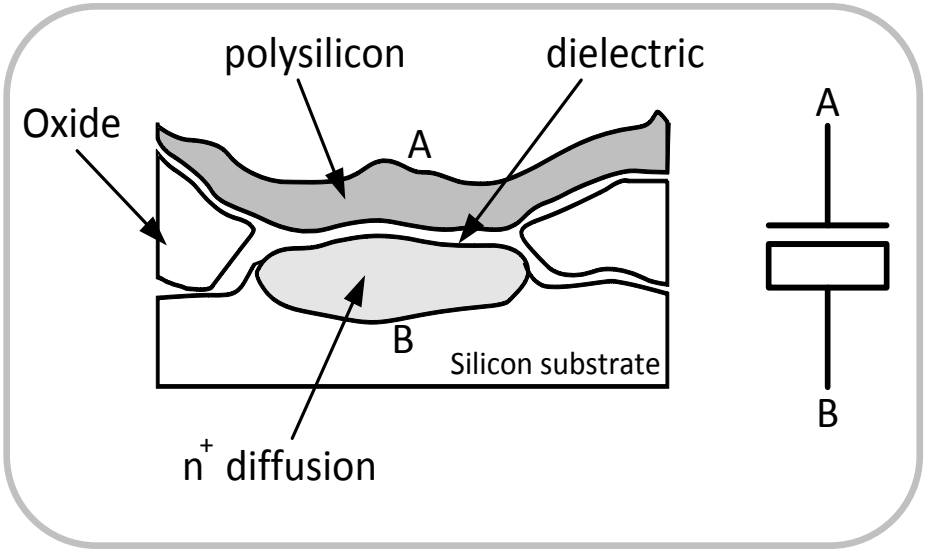
A high voltage blows the antifuse so it conducts

- Why antifuse and not fuse?
  - Well, interconnect networks are sparsely populated, which means that most of them are not connected
  - So antifuse is used, which is an open circuit by default

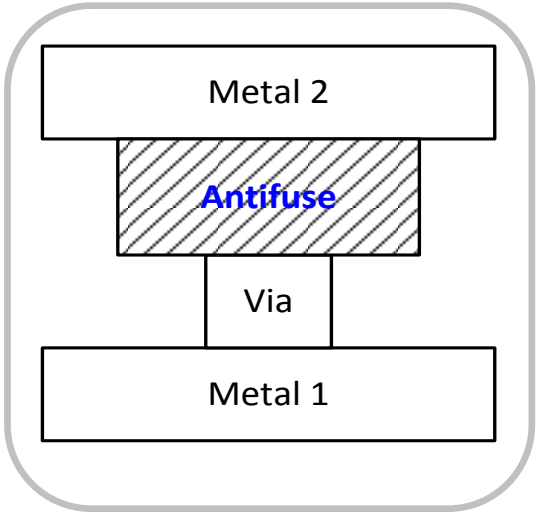
Antifuse-Based Programmable Switches

- **Two general structures:**

Poly to Diffusion (Actel)

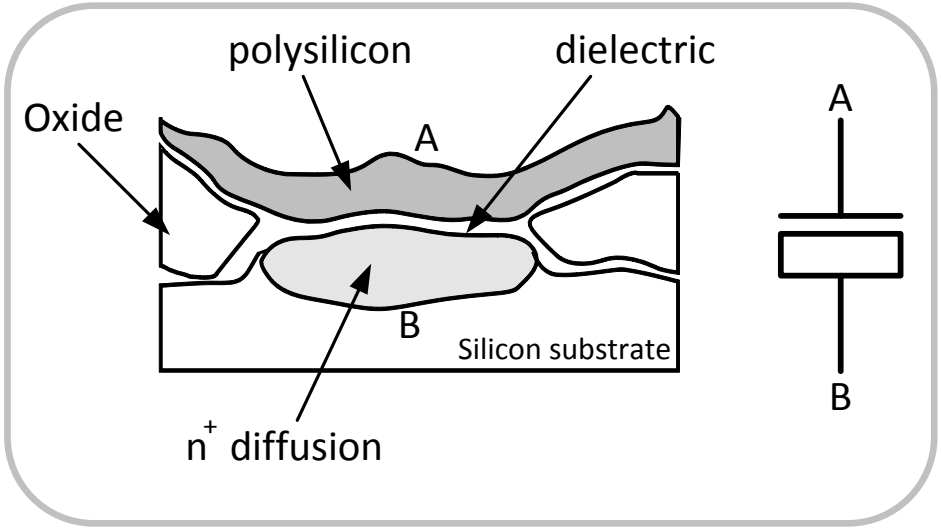


Metal to Metal (Via Link)



Antifuse: Poly-to-Diffusion (Actel)

- Three-layer sandwich structure: (called **PLICE**)
  - **Top layer**: polysilicon (conductor)
  - **Middle layer**: dielectric (insulator)
    - Isolates top and bottom (un-prog.)
    - Low-resistance link (programmed)
    - Amorphous silicon or silicon oxide
  - **Bottom layer**: n^+ diffusion (conductor)



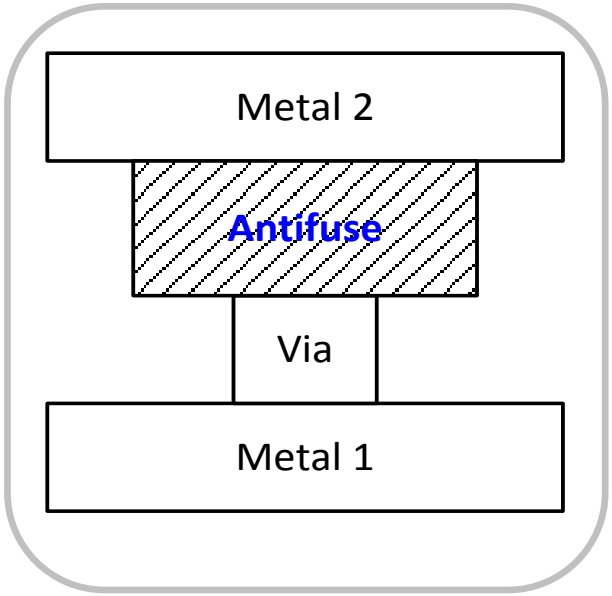
Antifuse

- Each antifuse in the FPGA has to be programmed separately

A high voltage/current breaks down/melts the insulator and it conducts (Permanent Link)

Antifuse: Metal-to-Metal (QuickLogic)

- Three-layer sandwich structure: (called **ViaLink**)
  - **Top layer**: Metal (conductor)
  - **Middle layer**: Thin amorphous Si (insulator)
    - Isolates top and bottom (un-prog.)
  - **Bottom layer**: Metal (conductor)



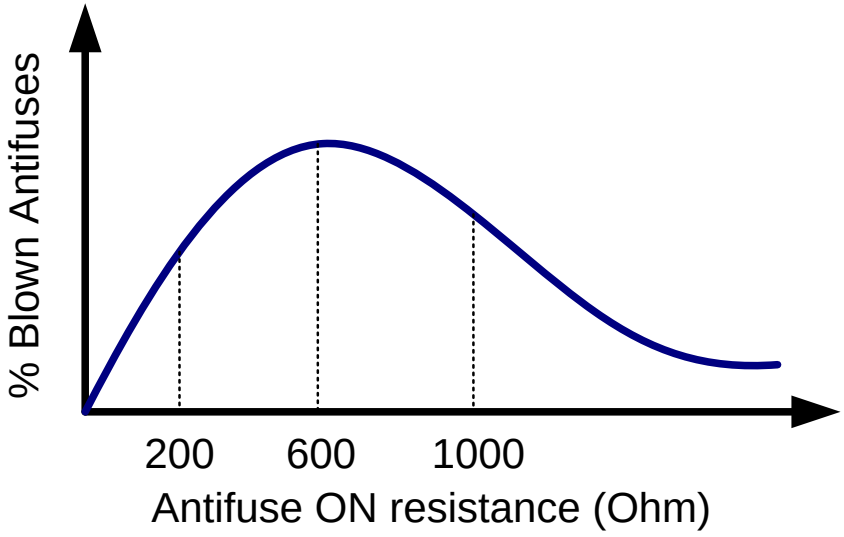
Antifuse

- **Advantages:**
  - Direct metal to metal eliminating connection b/w poly & diffusion thus reducing parasitic capacitance and interconnect space requirement
  - Lower resistance

Antifuse-Based Programmable Switches

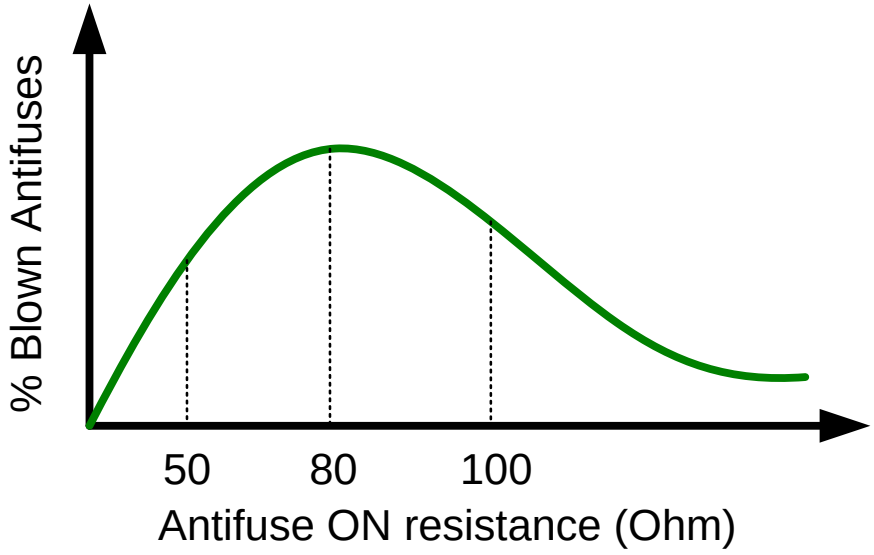
❑ Comparison of the ON resistance

Poly to Diffusion (Actel)



PLICE

Metal to Metal (QuickLogic)



ViaLink

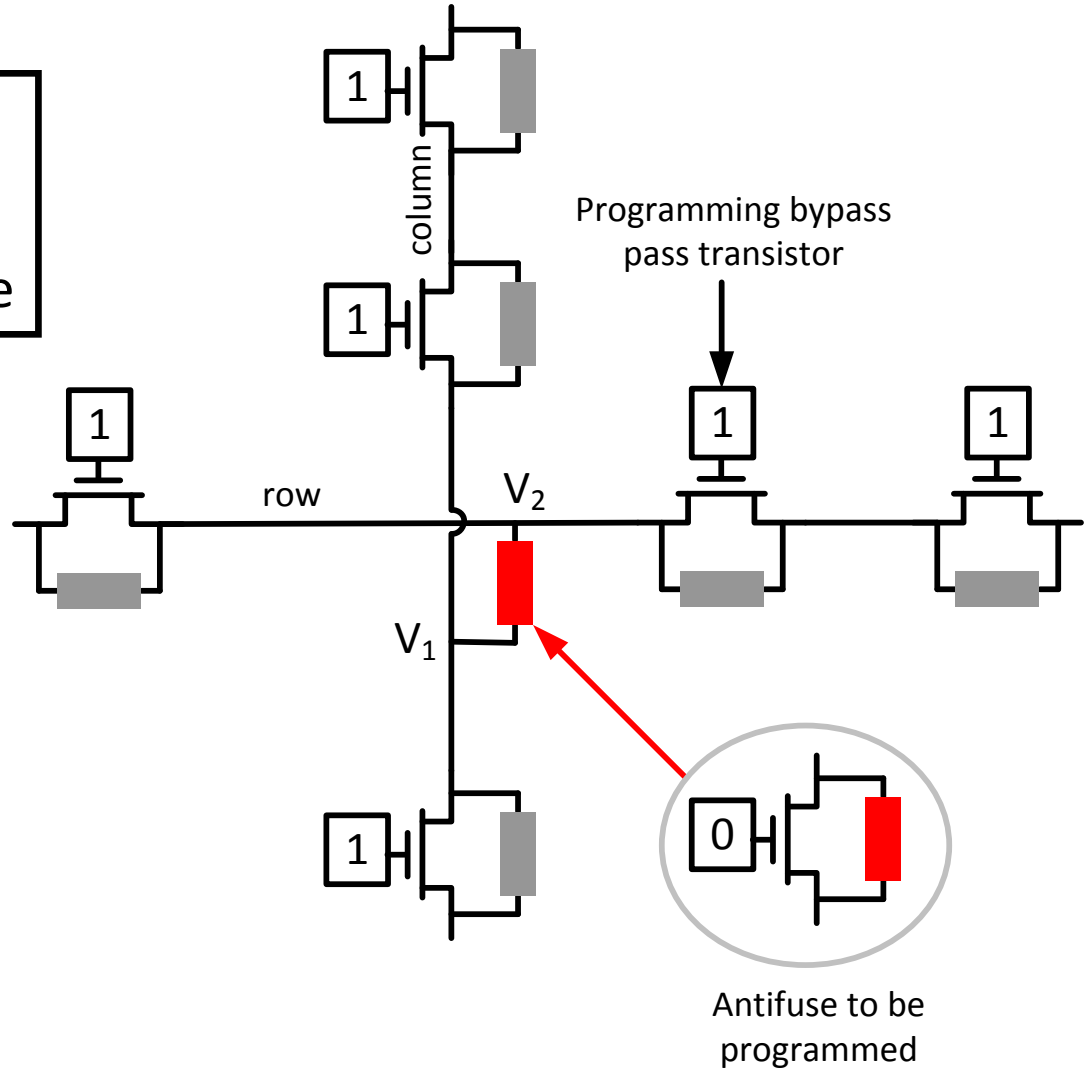
Antifuse-Based Programmable Switches

- An antifuse slows down the interconnect path less than a pass transistor in a SRAM-based FPGA.
- To be able to program every antifuse, each antifuse is connected in parallel with a pass transistor
- The pass transistor allows the antifuse to be bypassed during programming
- Gates of the pass transistors are controlled to select the appropriate row & column for the desired antifuse
- Voltage is applied across row/column so that only the desired antifuse receives the voltage.
- FPGA has circuitry that allows each antifuse to be separately programmed

To program an antifuse-based FPGA, chip is plugged into a socket on a special programming box that generates the programming voltage.

Antifuse-Based Programmable Switches

The voltage is applied across rows/columns so that only the desired antifuse receives the voltage



Antifuse-Based Programmable Switches

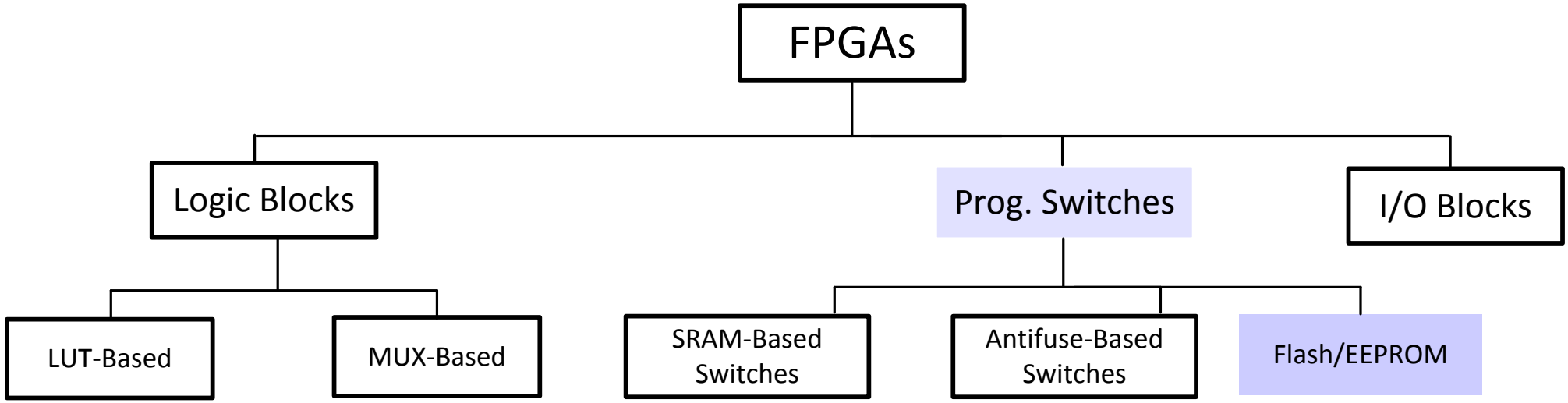
❑ Advantages:

- ➢ Requires no silicon area (low area) ⟹ more switches per device
- ➢ Lower resistance and parasitic capacitance than other technologies
- ➢ Non-volatility means
  - ▪ Instant operation
  - ▪ No need for additional on-chip memory (as opposed to SRAM-based)

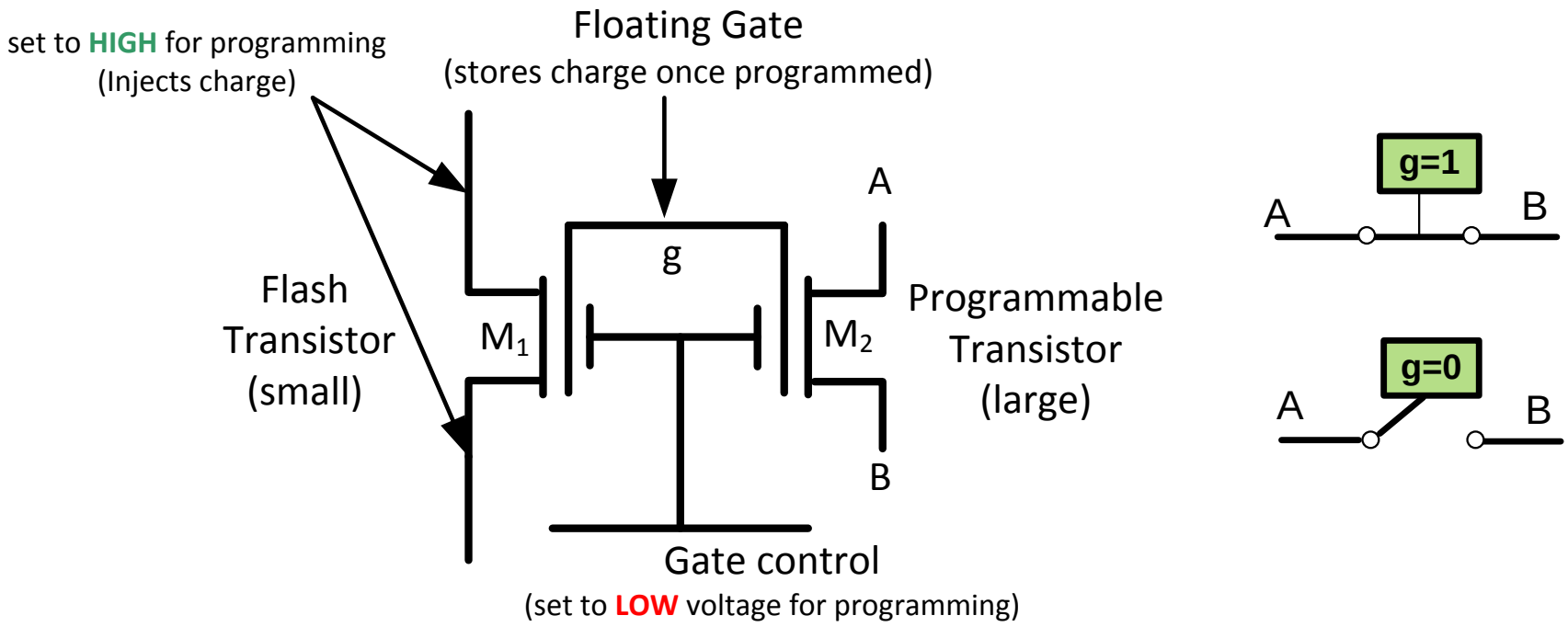
❑ Drawbacks:

- ➢ Requires non-standard CMOS process
  - ▪ Behind SRAM-based tech. in manufacturing
- ➢ Scaling challenges for antifuse
  - ▪ Hard to realize in deep sub-micron
- ➢ Not re-programmable

EEPROM/Flash-Based Programmable Switches



EEPROM/Flash-Based Programmable Switches

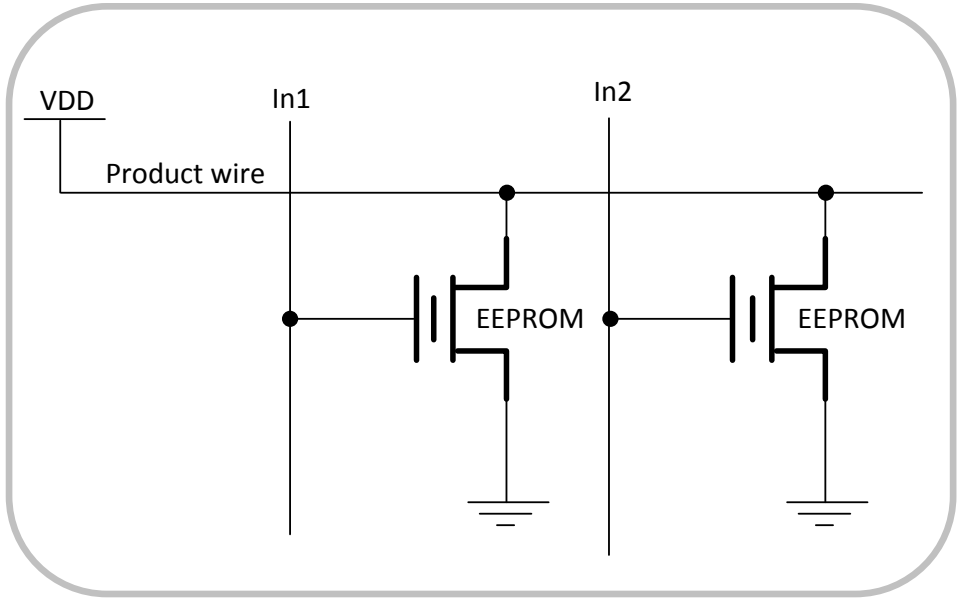


- Flash memory is a high-quality programmable read-only memory
- Has a floating gate structure, where a low-leakage capacitor holds a voltage that controls a transistor gate
- This memory cell can be used to control programming transistors

EEPROM/Flash-Based Programmable Switches

- An EEPROM transistor is also used as a programmable switch for CPLDs by placing the transistor between two wires in a way that facilitates implementation of wired-AND functions.

- An input to the AND plane can drive a product wire to ‘0’



EEPROM/Flash-Based Programmable Switches

❑ Advantages:

- ➢ Non-volatile ⟹ Does not lose information when the device is powered off
  (Thus no extra memory/flash is required)
- ➢ Improved area efficiency (less transistors needed compared to SRAM-cell)
- ➢ Re-programmable

❑ Drawbacks:

- ➢ Tricky floating-gate design
  - ▪ source-drain voltage should be low enough to prevent charge injection into the floating gate
- ➢ Can NOT be reprogrammed infinite number of times!
  - ▪ b/c of charge build-up in the oxide (e.g., Actel ProASIC3 are rated for 500 times)
- ➢ Uses non-standard CMOS process
- ➢ High resistance and capacitance due to the use of transistor-based switches

Programmable Switches

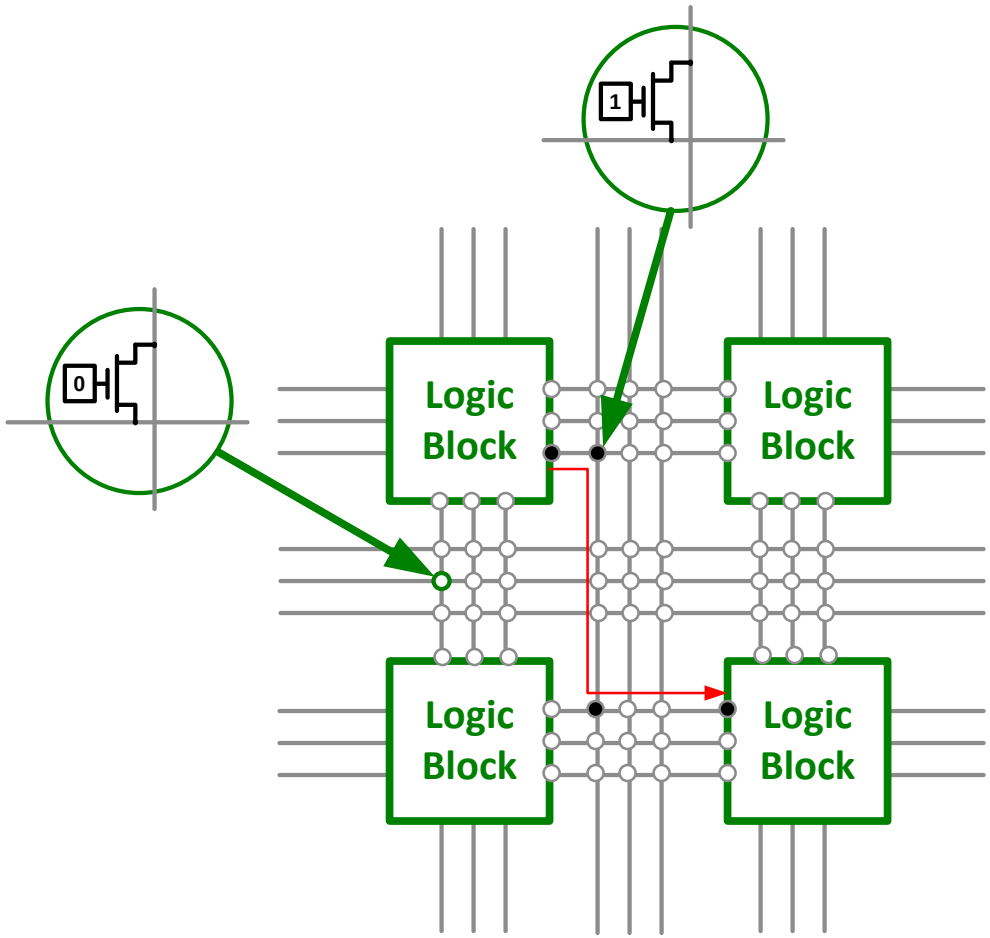
- So there are three technologies for switches:
  - SRAM cell
  - Antifuse
  - Flash-based
- The ideal technology is the one that is:
  - Non-volatile
  - Reprogrammable infinite number of times
  - Based on standard cell CMOS process
  - Offer low on resistance and capacitance
- Recent trend by Xilinx, Altera and Lattice:
  - On-chip flash memory for storage of configuration bits
  - SRAM-based interconnect switches

Comparison Between All Technologies

Manufacturer	SRAM	Flash/EEPROM	Antifuse
Volatile	Yes	No	No
Re-Programmable	Yes	Yes	No
Area	High (6 transistors)	Moderate(1 transistor)	Low(0 transistor)
Manufacturing Process	Standard CMOS	Flash Process(EECMOS)	Antifuse (CMOS+)
In-system Programmable	Yes	Yes	No
Switch Resistance	500-1000 Ohm	500-1000 Ohm	20-100 Ohm
Switch Capacitance	1-2fF	1-2fF	<1 fF
Yield	100%	100%	>90%

Routing Channels

❑ Interconnect wiring is grouped into routing channels, each of which contains a complete grid of horizontal and vertical wires.



Routing Channels

- FPGA wiring with programmable interconnect is slower than typical wiring in a custom chip b/c:
  - Pass transistor on an interconnect is not a perfect on-switch
  - Programmable interconnect is slower than a pair of wires permanently connected by a via
  - FPGA wires are generally longer than would be necessary for a custom chip

Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- **Field-Programmable Gate Array (FPGAs)**
  - Logic Blocks
  - Programmable Routing Switches
  - **I/O Pads**
- Commercial FPGA Products
- Application Specific Integrated Circuits (ASICs)

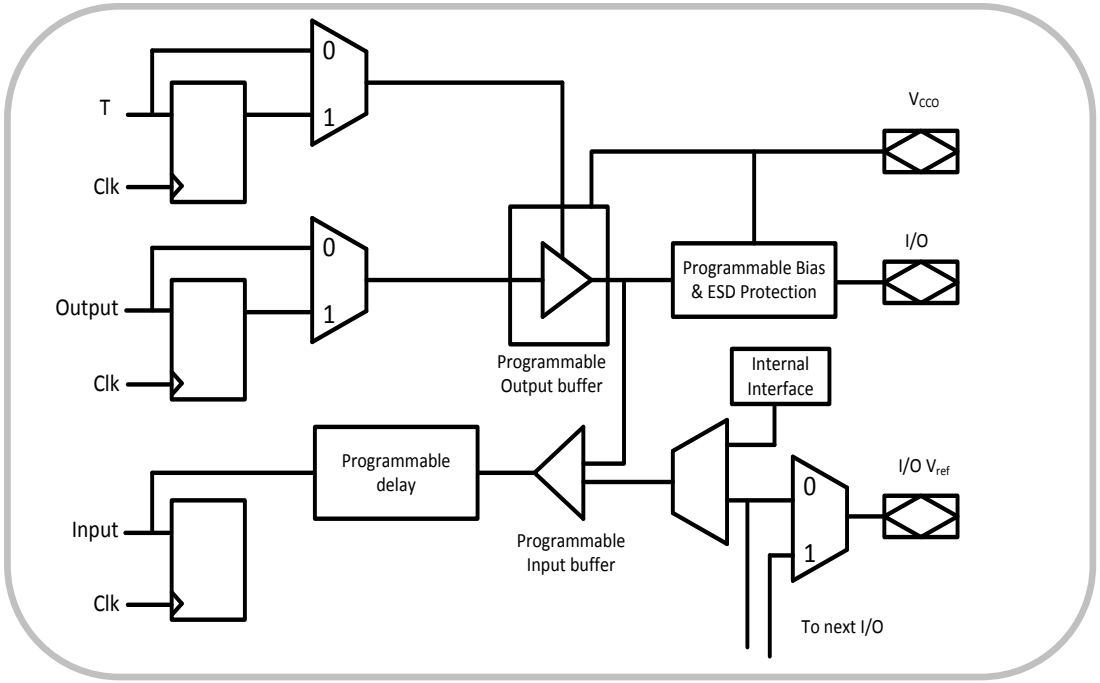
FPGA Chip I/O

- I/O pins on a chip connect it to the outside world and perform some basic functions
  - Input pins provide electrostatic discharge (ESD) protection
  - Output pins provide buffers with sufficient drive to produce adequate signals on the pins
  - Three-state pins include logic to switch b/w input and output modes
- The pins on an FPGA can be configured to act as
  - Input pin
  - Output pin
  - Tri-state pin

Xilinx Spartan II 2.5V Family I/O Pins

- Supports a wide range of I/O standards
- The I/O has three registers, one each for input, output and tri-state operation
  - Each has its own enable signal
  - They all share the same clock connection
  - Can be configured as latch or FF

The Prog. delay on the input path is to eliminate variations in hold times from pin to pin



Xilinx Spartan II 2.5V Family I/O Pins

- Supports a wide range of I/O standards, divided into eight banks
- Pads within each bank share the same reference voltage, threshold voltage and use standards that have the same V_{CCO}

I/O Standard	Input Ref. Voltage (V_{ref})	Output Source Voltage (V_{CCO})
LVTTL	N/A	3.3
LVCMOS2	N/A	2.5
PCI	N/A	3.3
GTL	0.8	N/A
HSTL Class I	0.75	1.5
SSTL3 Class I/II	1.5	3.3
CTT	1.5	3.3
AGP-2X	1.32	3.3

Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- **Commercial FPGA Products**
- Application Specific Integrated Circuits (ASICs)

Commercial FPGA Products

- As of summer 2009

Manufacturer	FPGA Products	LUT/Antifuse based	Floorplan
Actel	MX, SX, eX, Axcelerator	Antifuse-based	Row-Based
QuickLogic	PASIC, QuickRAM, Eclipse (Plus/II)	Antifuse-based	Symmetrical array
Lattice	ECP2/M, SC	Antifuse-based	Symmetrical array
Atmel	AT40K, AT40KAL	LUT-Based	Hierarchical PLD
Altera	Stratix (II/III/IV), Cyclone (II/III), Arria (II), Flex 8000, 10K	LUT-based	Hierarchical PLD
Xilinx	Virtex-II Pro, Virtex-(E,4,5,6) Spartan-(II/3) (A/E), XC4000	LUT-based	Symmetrical array

Most Dominant

Commercial FPGA Products

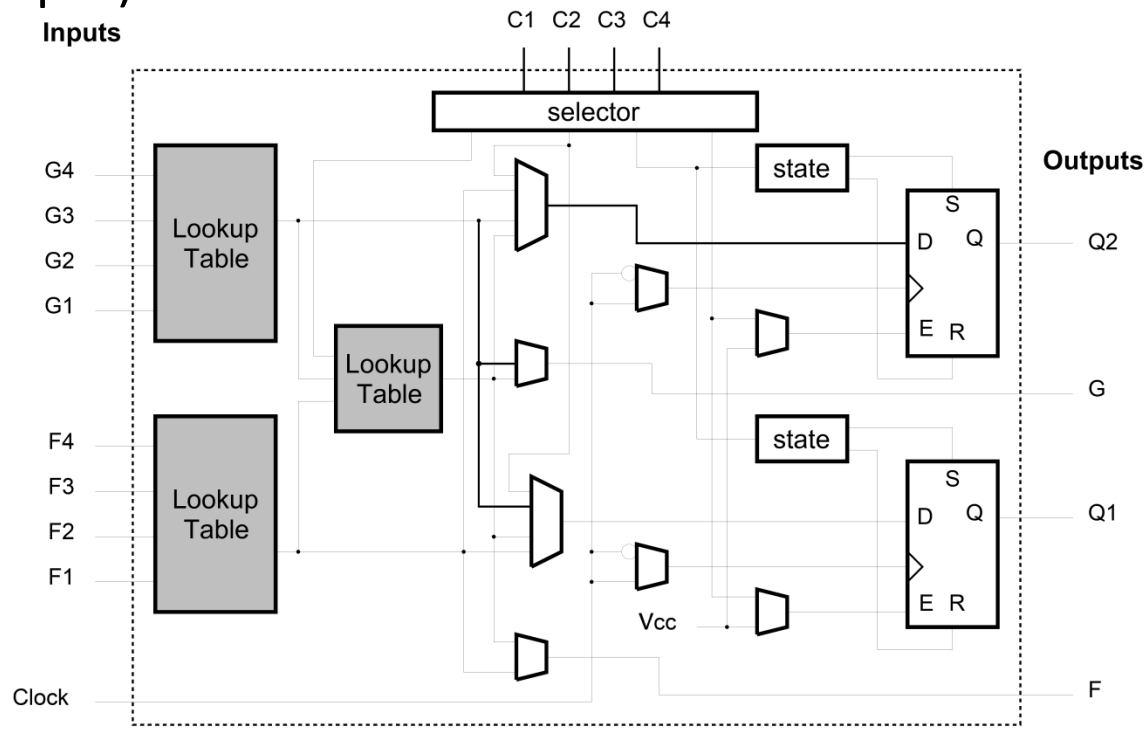
- **Xilinx:**
  - **SRAM-Based:**
    - XC2000
    - XC3000
    - XC4000
    - XC5000
    - Virtex Family (II Pro, 4, 5, 6)
    - Spartan Family
  - **Antifuse-Based:**
    - XC8100

Xilinx (XC4000 Series)

- 2,000 to 15,000 gates (XC4085 supports up to 100,000 gates)
- The building block in Xilinx FPGAs is called **Configurable Logic Block (CLB)**
- XC4000 CLB is LUT-based and consists of
  - 3 LUTs (two 4-input and one 3-input)
  - 2 Flip-Flops (FFs)
- These 3 LUTs allow implementation of:
  - Logic functions of up to 9 inputs
  - Two separate 4-input functions
- Each CLB contains circuitry that allows

Implementation of fast carry operations

(soft logic, coarse-grained)



Xilinx (XC4000 Series): Interconnect

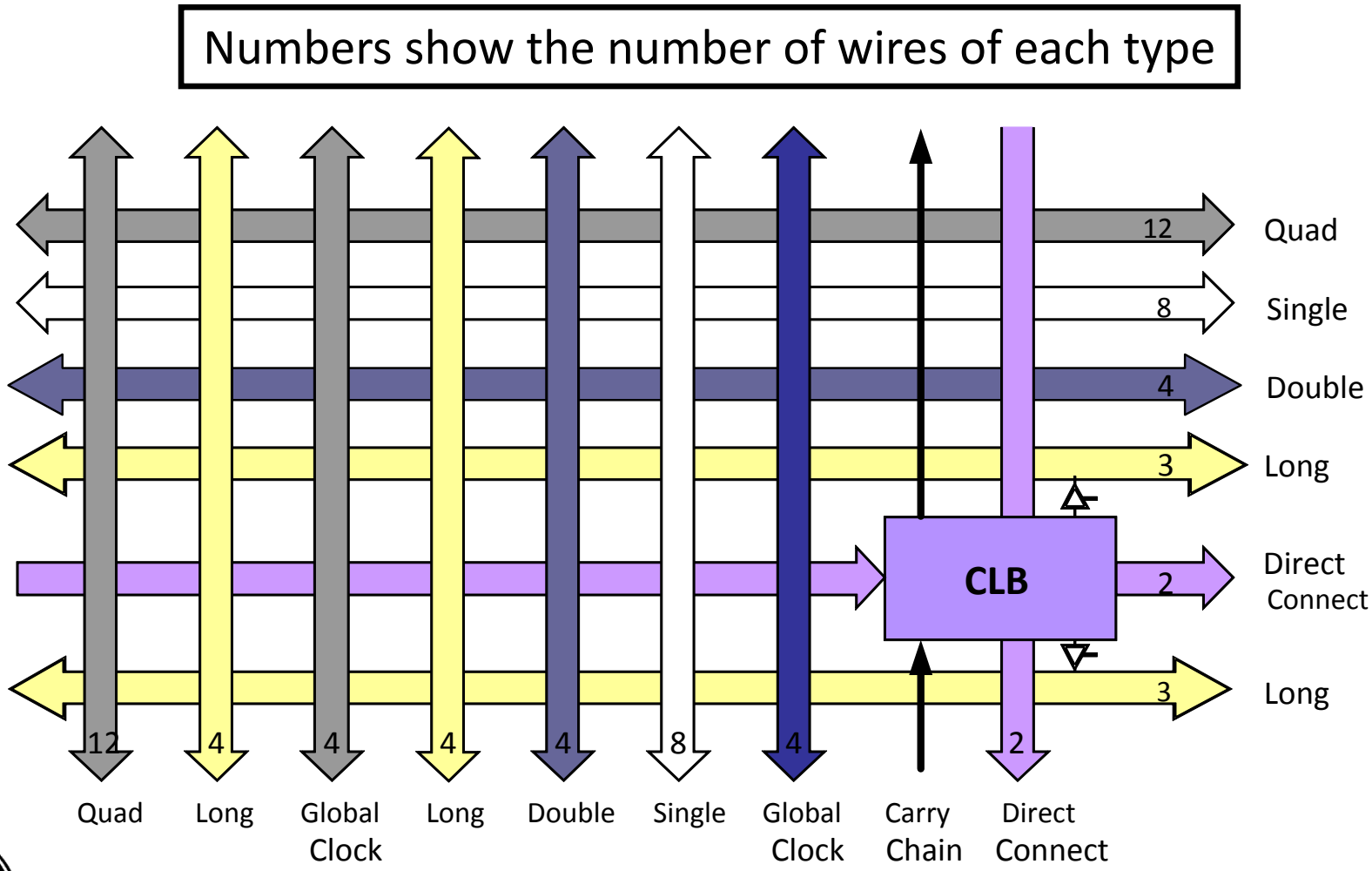
- Consists of horizontal and vertical channels

- Wires in each channel in XC4000 series are of different types
  - **Wire segments:** of length 1, 2, 4 (single, double, quad)
  - **Direct interconnect:** For local connections, with min delay, small fan-out
    - Effective for implementation of fast arithmetic modules
  - **Long Wires:** for global routing, high fan-out,
    - Used for time critical signals or signals distributed over long distances (Bus)
  - **Special wires:** for clock routing

LUTs in a CLB can be configured as read/write RAM cells

Xilinx (XC4000 Series): Interconnect

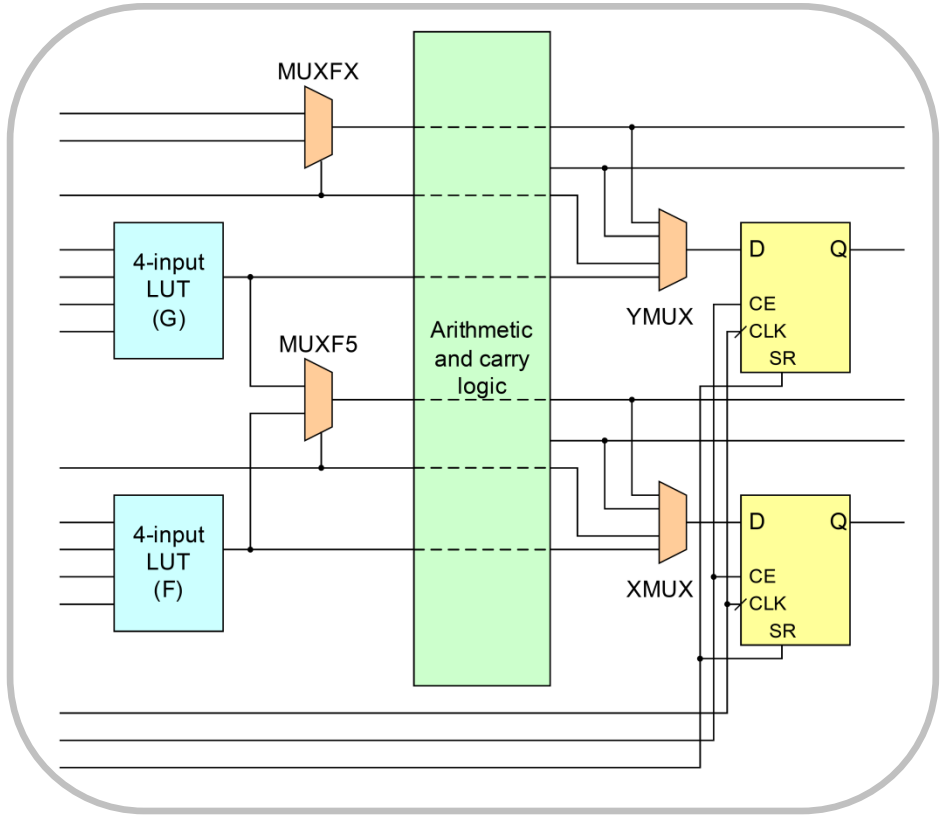
- Interconnect Architecture:



Xilinx (Virtex Series)

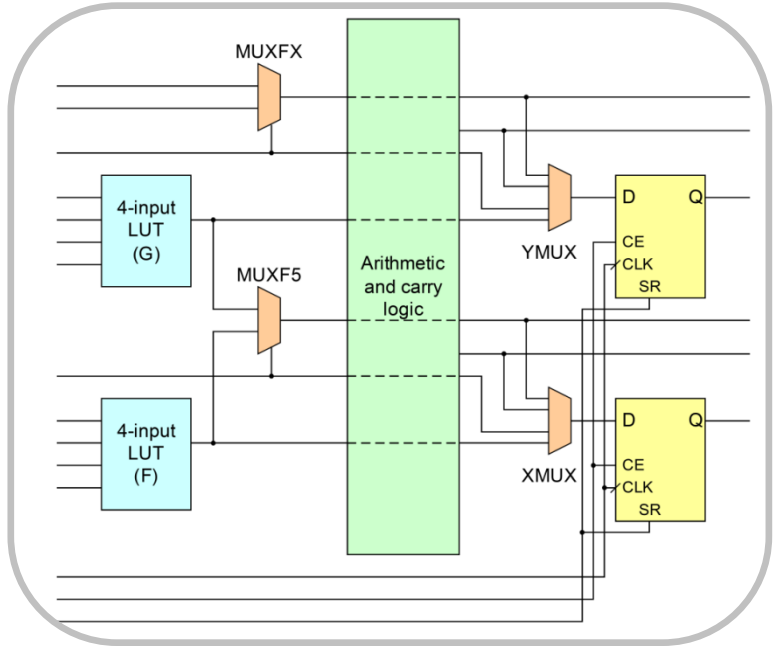
- The elementary Prog. block in Virtex/Spartan FPGAs is called **"Slice"**
- Two slices form a **Configurable Logic Block (CLB)**
- Inside each Virtex 4 slice:

Virtex-4 Slice:



Xilinx Virtex 4 Slice Architecture

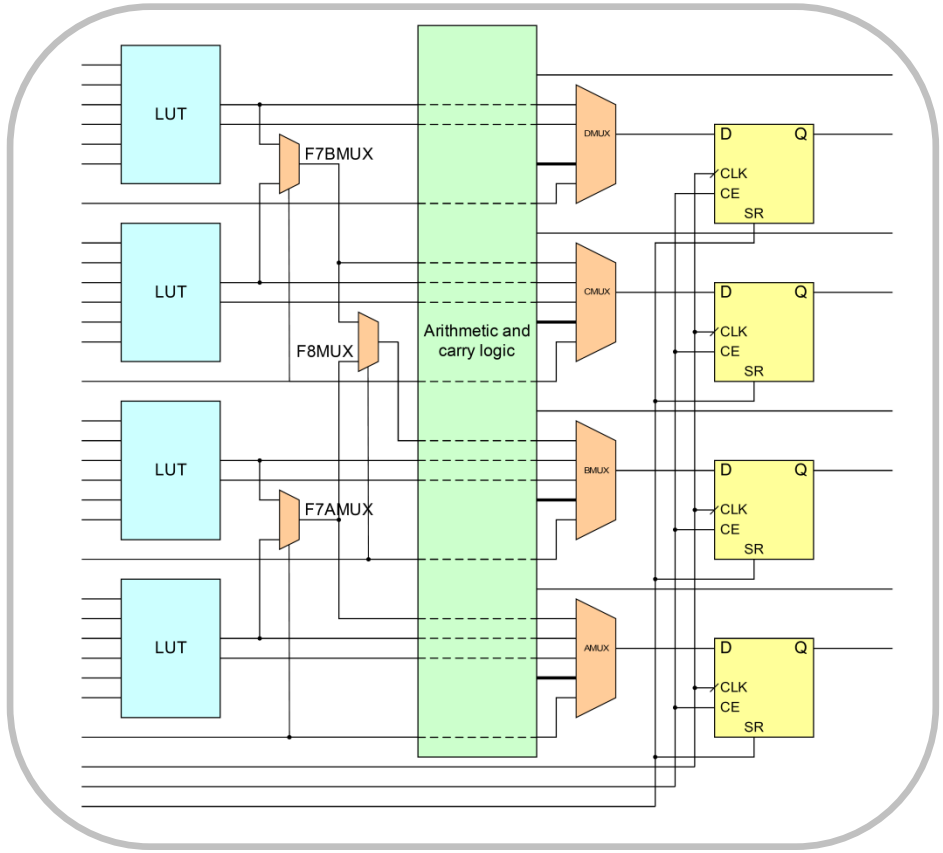
- Two 4-input LUTs (G, F)
- Two dedicated user-controlled MUXs for combinational logic
  - MUXF5 to combine outputs of G, and F to implement 5-input combinational circuit.
  - MUXFX to combine outputs of the other MUXF5 and MUXFX (from the other slices).
- Two 1-bit registers (configured as FF or latches)
- YMUX/XMUX to control the input to the registers
- Dedicated arithmetic logic
  - Two 1-bit adders
  - Carry chain
  - Two AND gates for fast multiplication



Xilinx (Virtex 5 Series)

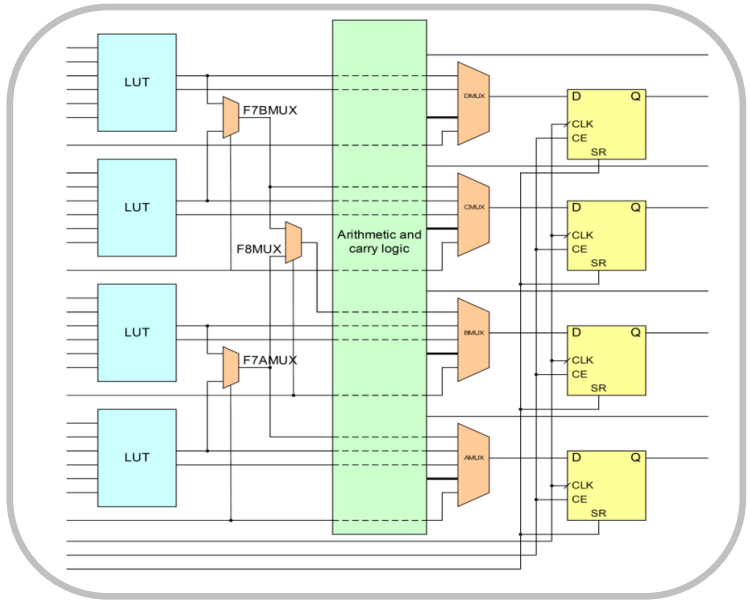
- The Virtex 5 Slice consists of four 6–input LUTs
  - As opposed to two 4-input in Virtex 4

Virtex-5 Slice:



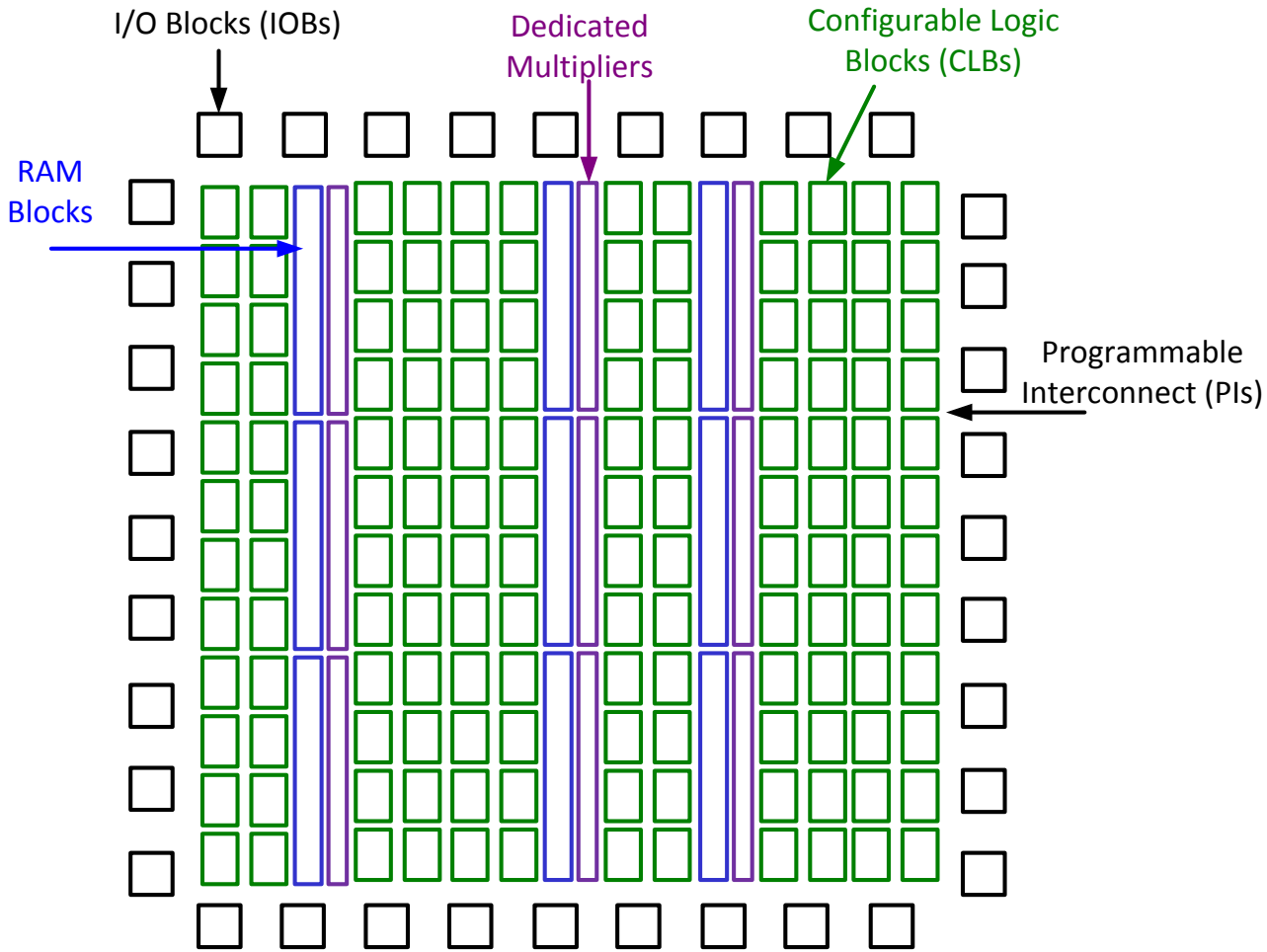
Xilinx Virtex 5 Slice Architecture

- Four LUTs that can be configured as:
  - 6-input LUTs with one output
  - 5-input LUTs with two outputs
- Three dedicated user-controlled MUXs for combinational logic
  - F7AMUX/F7BMUX to combine outputs of the LUTs to implement 7-input combinational circuits.
  - F8MUX to combine outputs of F7AMUX/F7BMUX from the other slices).
- Four 1-bit registers (configured as FF or latches)
- Dedicated arithmetic logic
  - Two 1-bit adders
  - Carry chain
  - Two AND gates for fast multiplication



Xilinx Spartan II

- Heterogeneous blocks:



Commercial FPGA Products

- **Altera:**
  - **SRAM-Based:**
    - FLEX 8000
    - FLEX 6000
    - FLEX 10000
    - Cyclone II/III
    - Stratix II, III, IV

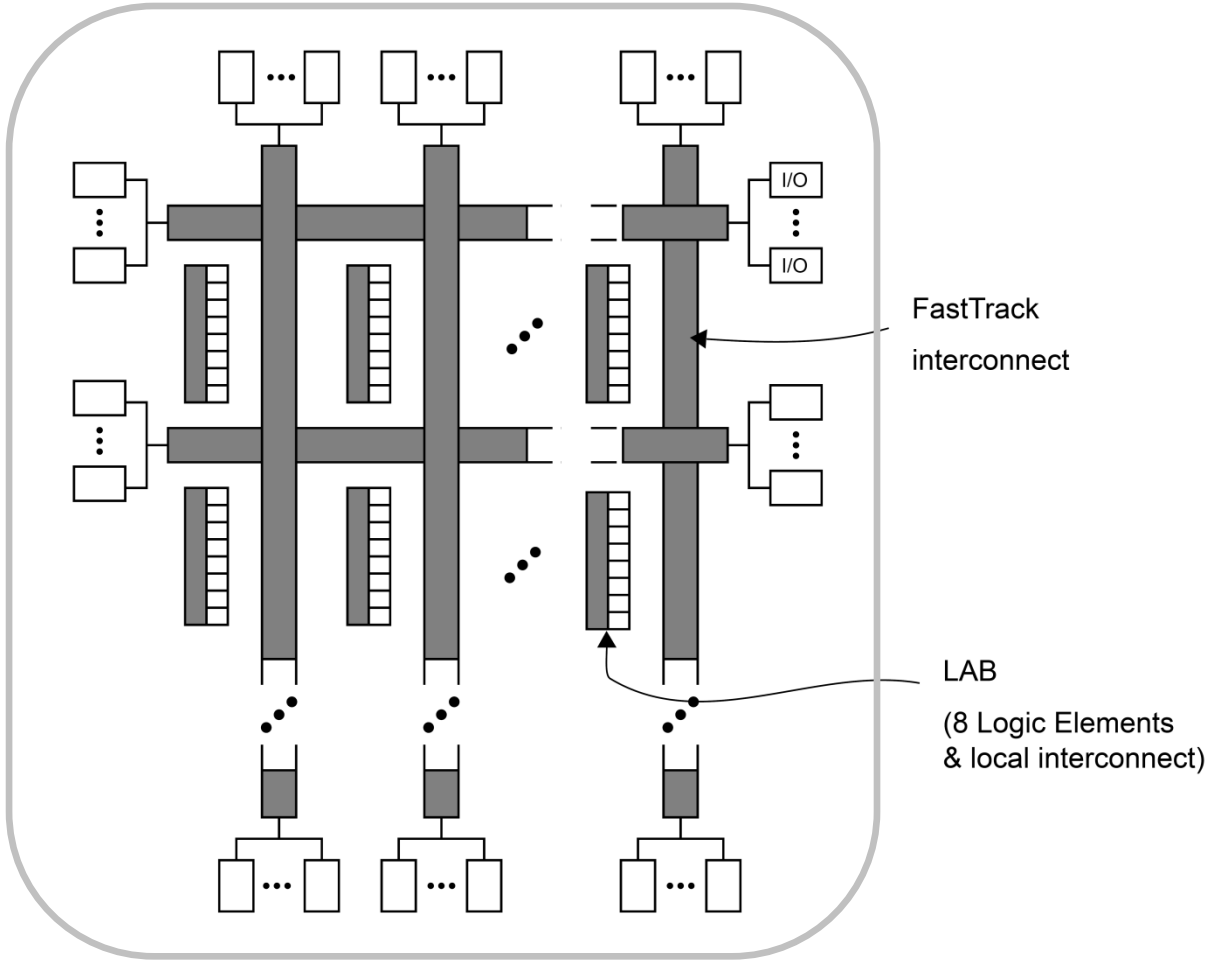
Altera (FLEX 8000 Series)

- The logic block in Altera FPGAs is called **Logic Element (LE)**

- FLEX8000 contains three main components
  1. The main building block is called **Logic Array Block (LAB)**
     - Contains eight LUT-based LEs
  2. FastTrack interconnect
     - Horizontal and vertical to connect LABs
  3. I/O pads

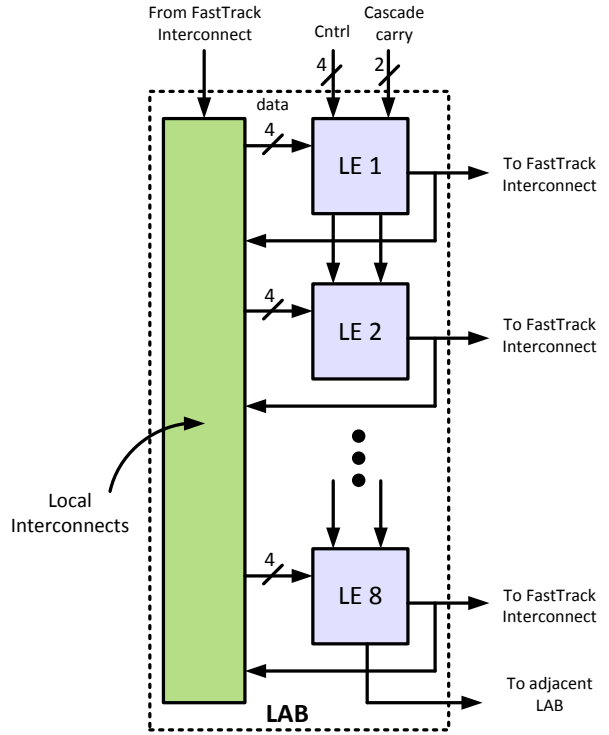
Altera (FLEX 8000 Series)

- Architecture of FLEX8000
  - LAB
  - FastTrack
  - I/O



Altera (FLEX 8000 Series)

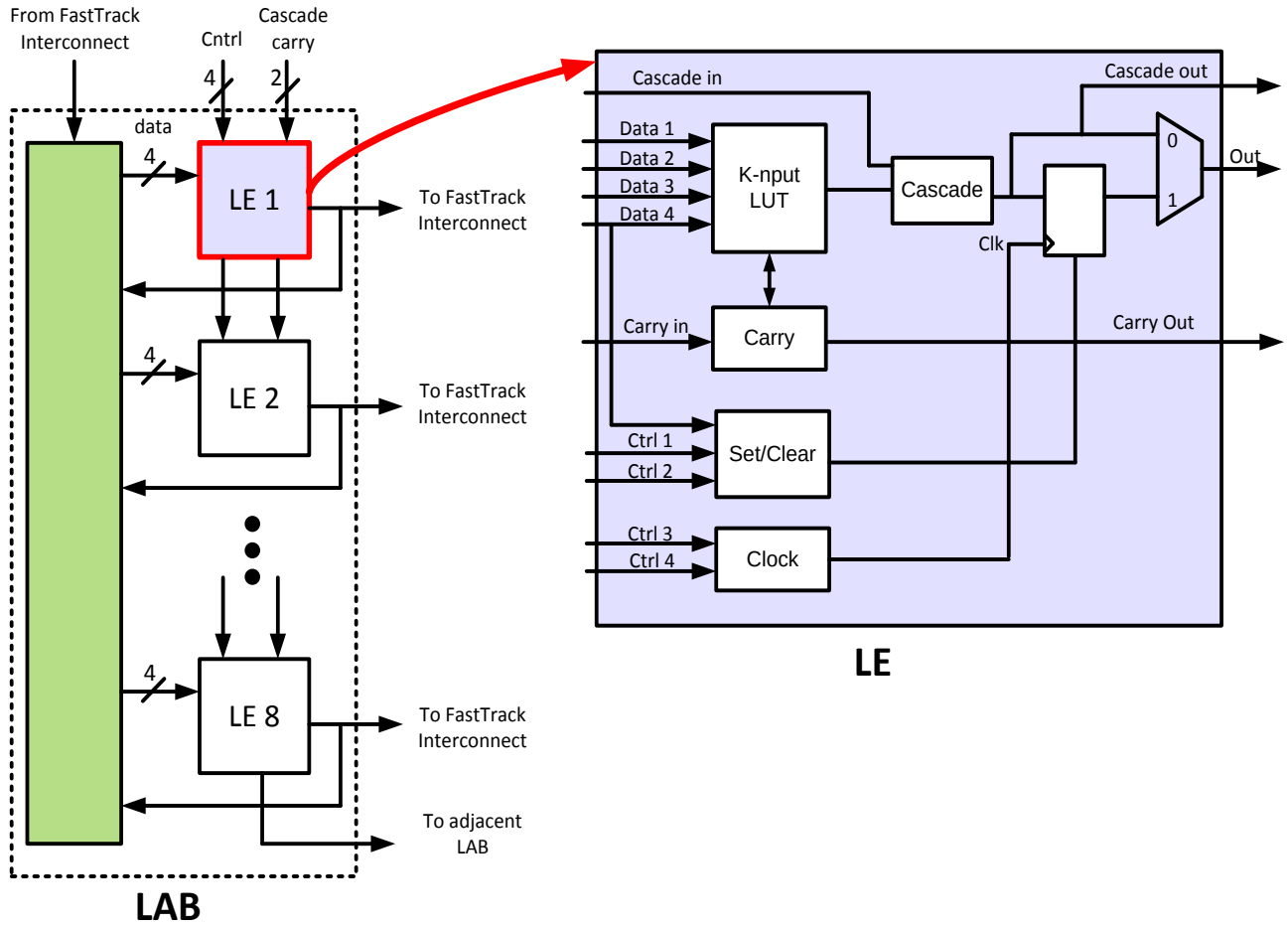
- Each FLEX8000 LAB is a group of eight LEs
- Each LAB:
  - Has a number of inputs provided from the adjacent row interconnect wires
  - Its outputs connect to the adjacent row/column wires
  - Contains local interconnects to connect any LE to another LE inside the same LAB
  - Connected to the global interconnect (fastTrack)
    - Similar to the Xilinx long lines



Altera (FLEX 8000 Series)

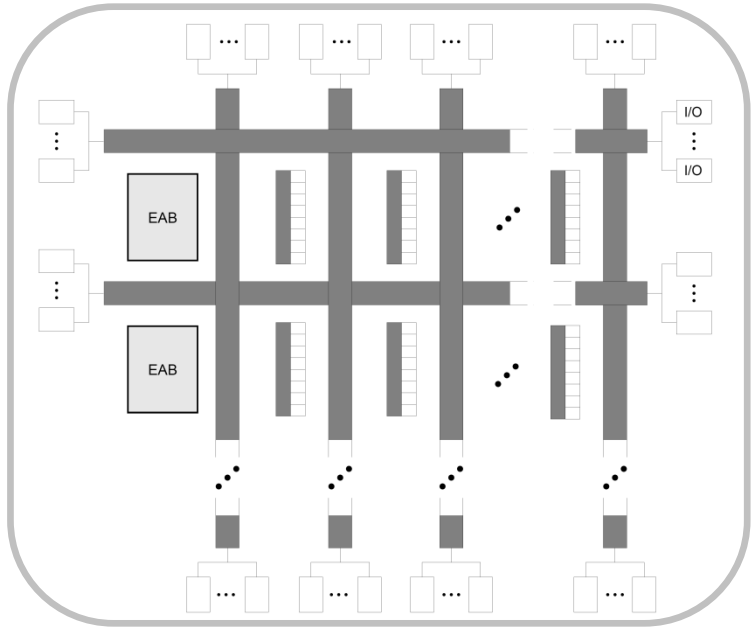
- Each FLEX8000 LE is LUT-based and consists of :
  - A 4-input LUT(to implement two 3-input functions, i.e. sum/carry fcns in a full adder)
  - A Flip-Flops (FF)
  - Carry circuitry
  - Cascade circuitry

(soft logic, coarse-grained)



Altera (FLEX 10K Series)

- Offers all the features of FLEX8000
- It also has variable-sized blocks of SRAM in each row
  - Called **Embedded Array Block (EAB)**
- Each LAB can serve as
  - An SRAM block with aspect ratios of
    - (256X8) (512X4) (1KX2) (2KX1)
  - A large multi-output LUT
    - To implement a complex circuit
    - For example a multiplier



- FLEX 10K chips are in sizes from 10K10 (10,000 gates) to 10K250 (250,000 gates)
- Chips are in various speeds, indicated by a speed grade
  - For example: 10K10-1 (faster) or 10K10-2 (slower)

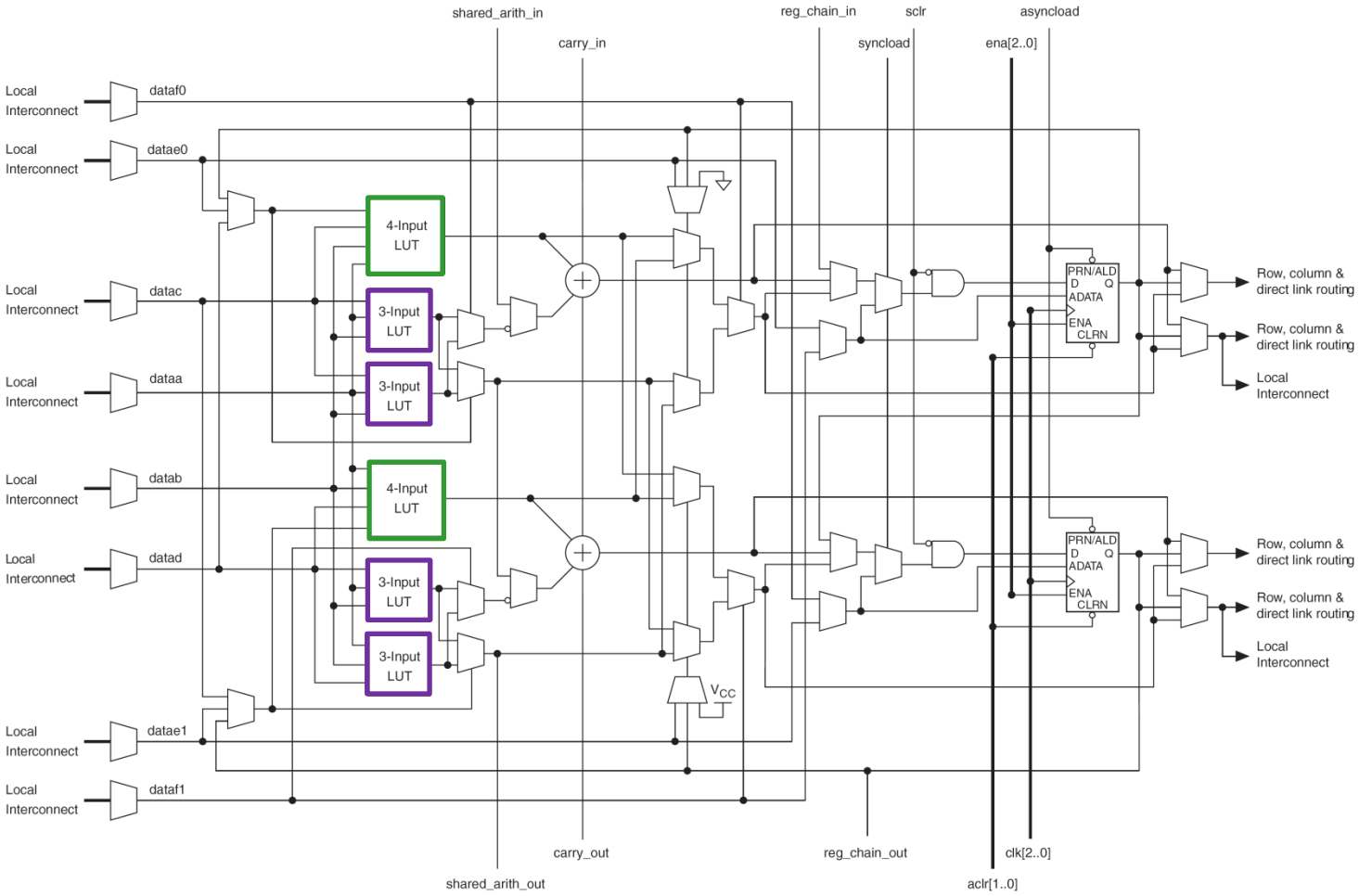
Altera (Stratix II)

- Using an Adaptive Logic Module (ALM) as its logic element.

- Stratix II ALM is an 8-input structure that can implement many combinations of functions including:
  - One 6-input logic function
  - Two 4-input logic functions
  - One 5-input and one 3-input logic functions
  - Two 6-input logic functions that share the same function and four inputs

Altera (Stratix II)

- Stratix II Adaptive Logic Module (ALM) structure:

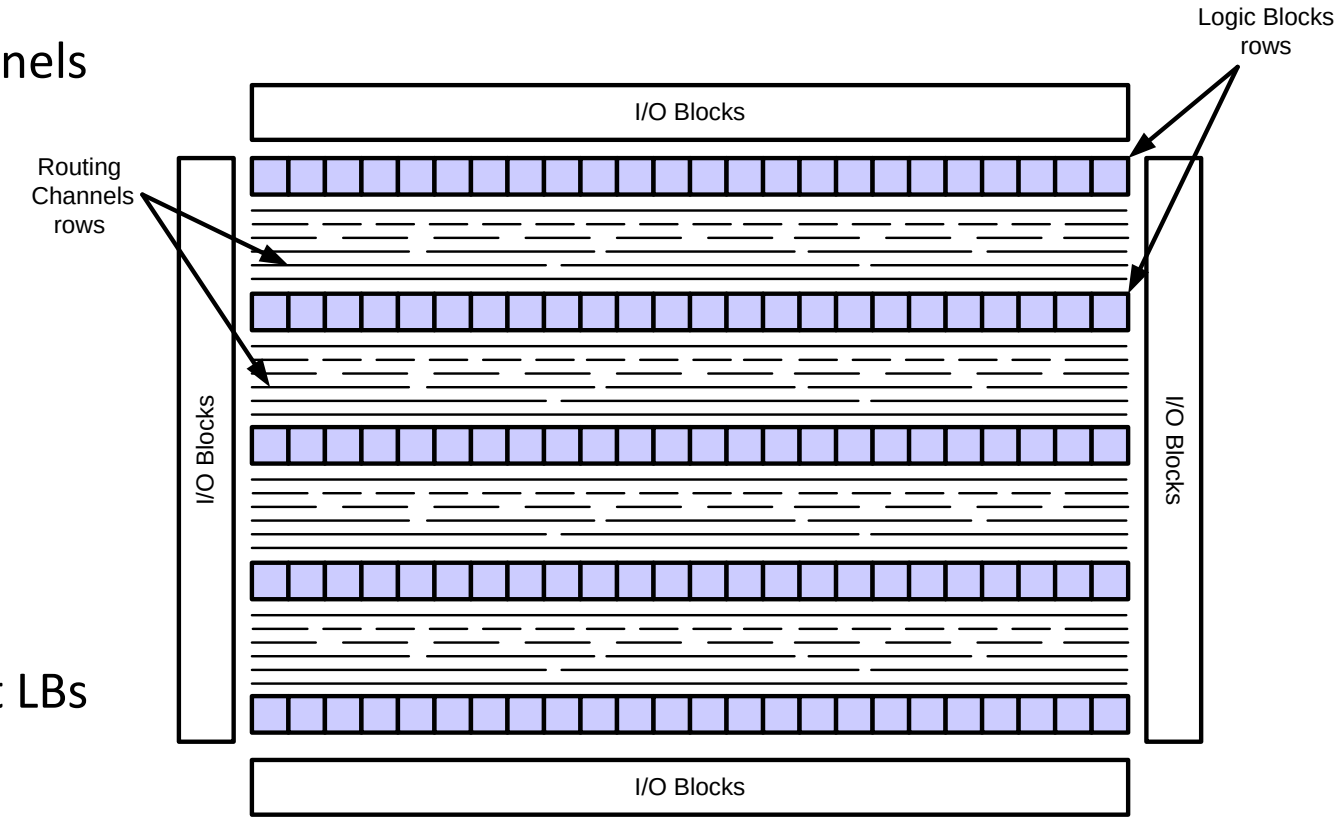


Commercial FPGA Products

- **Actel:**
  - **Antifuse-Based:**
    - Act 1
    - Act 2
    - Act 3
    - SX-A
    - Axcelerator

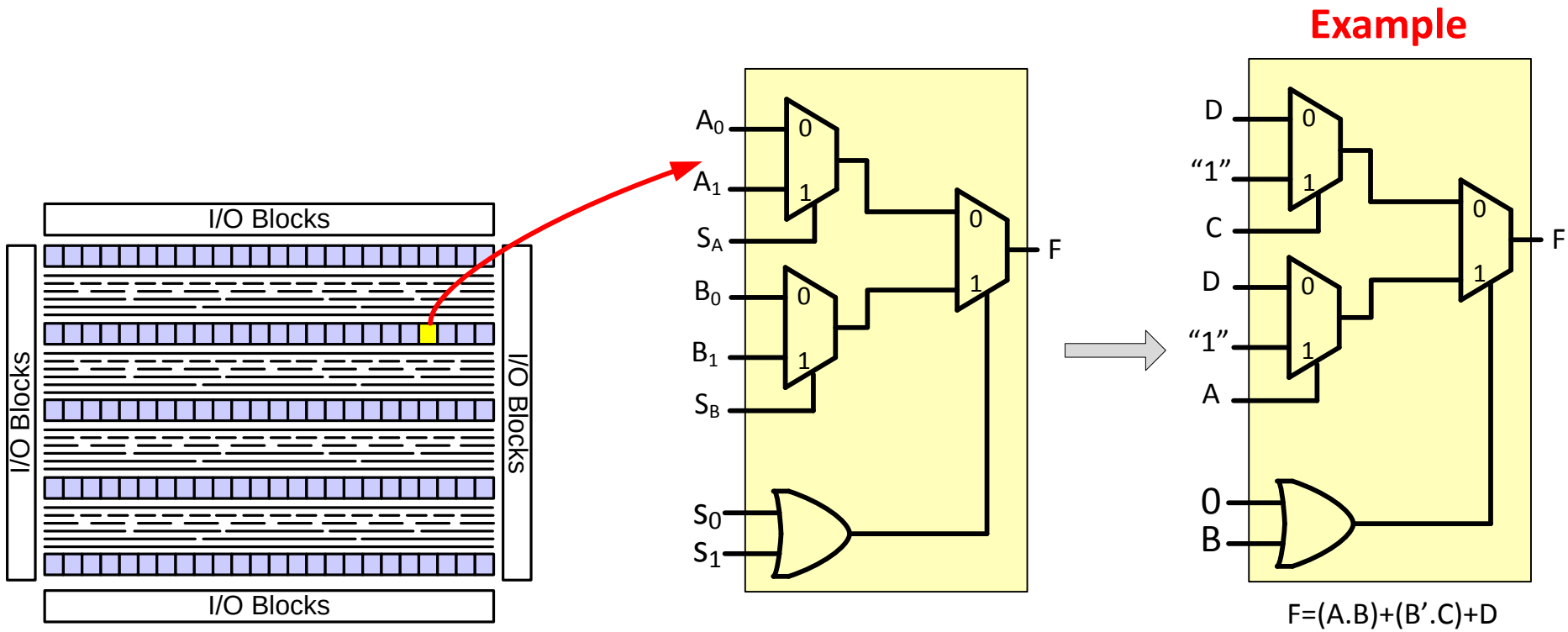
Actel FPGAs (Act 3 Series)

- A structure similar to the traditional gate arrays
- Consists of
  - Horizontal logic blocks
  - Horizontal routing channels
  - I/O blocks
- **MUX-based logic blocks**
  - MUX
  - AND/OR
- **Interconnects:**
  - Only horizontal
  - Segmented wires
  - Use antifuse to connect LBs to routing channels



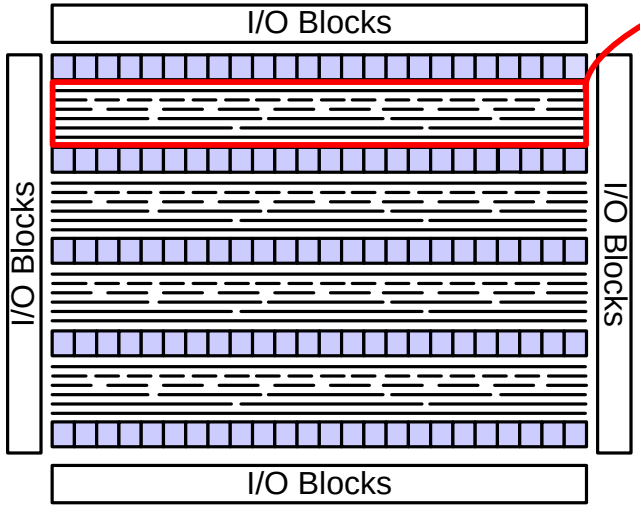
Actel FPGAs (Act 3 Series)

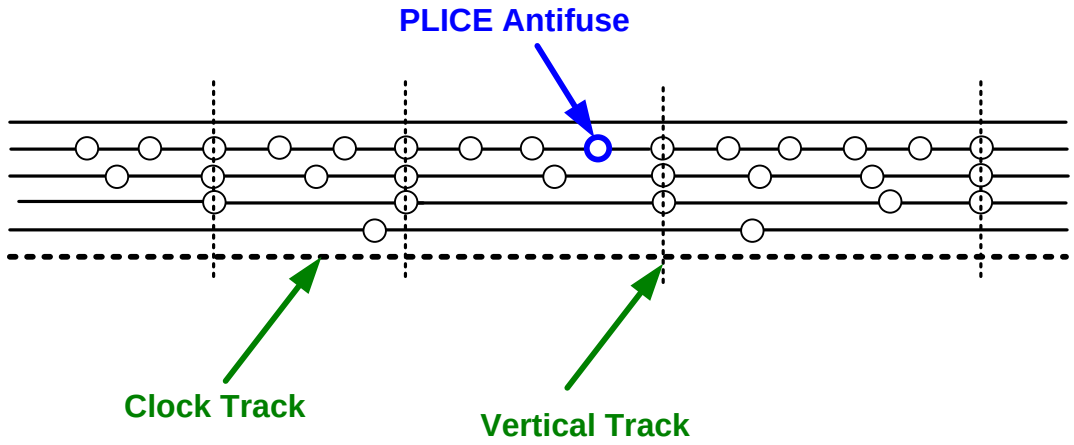
- Detailed architecture of Act 3:



Actel FPGAs (Act 3 Series)

- Detailed architecture of Act 3:

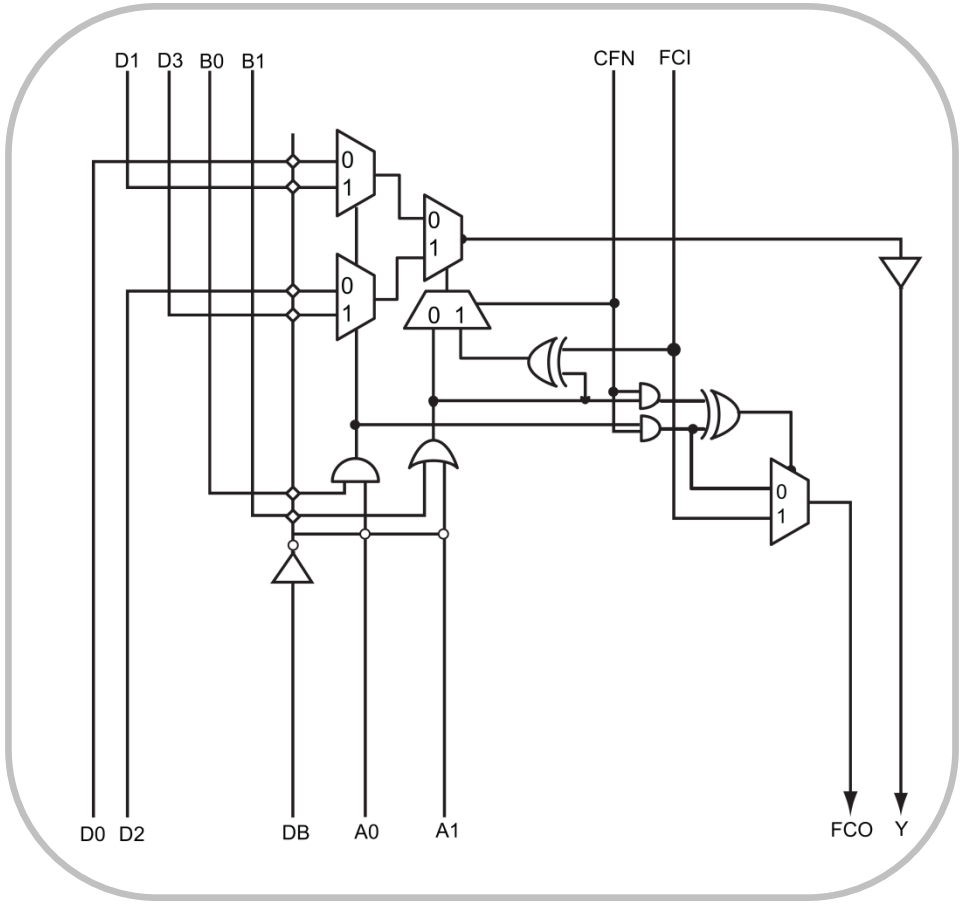




Actel Device	Number of Antifuses
A1010	112,000
A1225	250,000
A1280	750,000

Actel FPGAs (Axcelerator Series)

- Advance recent antifuse-based FPGA with 2 million equivalent gates.
- It comes with
  - Embedded SRAM Blocks
  - Chip-wide highway routing
  - Carry logic
  - PLL
- AX125 Logic Block:



Actel FPGAs (ProASIC 500K Series)

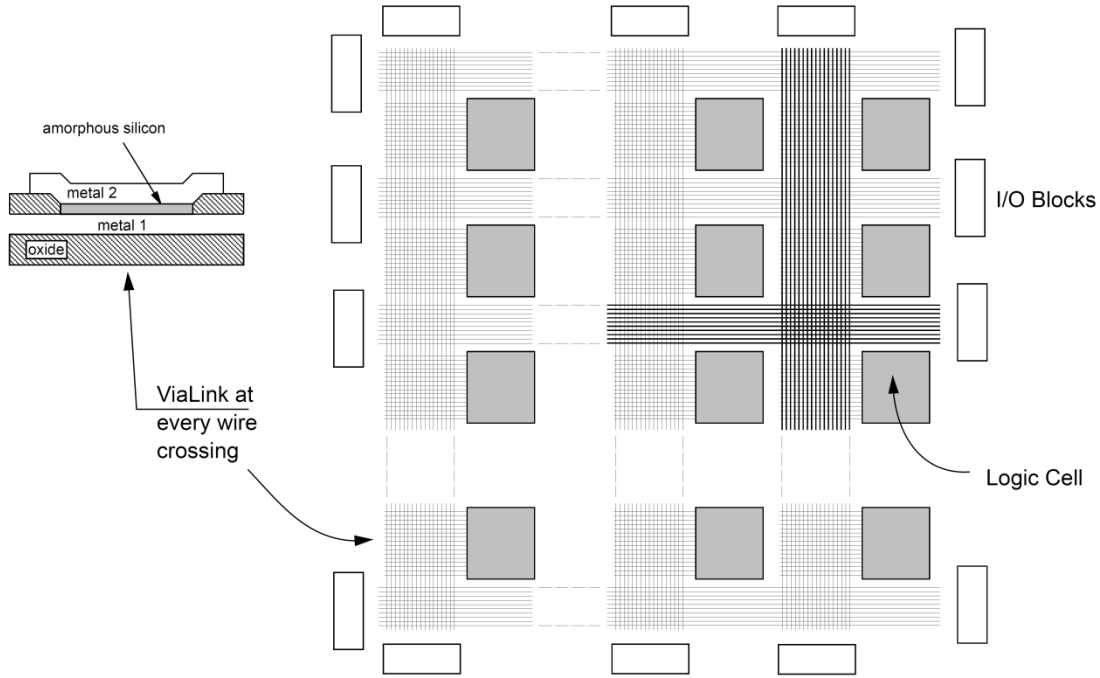
- Flash-based FPGA, using switches and MUXs for programmability to implement logic functions
- The programmed switches are used to select alternate inputs to the core logic
- It can implement any function of 3 inputs except the 3-input XOR
- The feedback paths allow the logic block to be configured as a latch
  - **in2:** clock
  - **in3:** reset

Commercial FPGA Products

- **QuickLogic:**
  - **Antifuse-Based:**
    - pASIC
    - pASIC-2

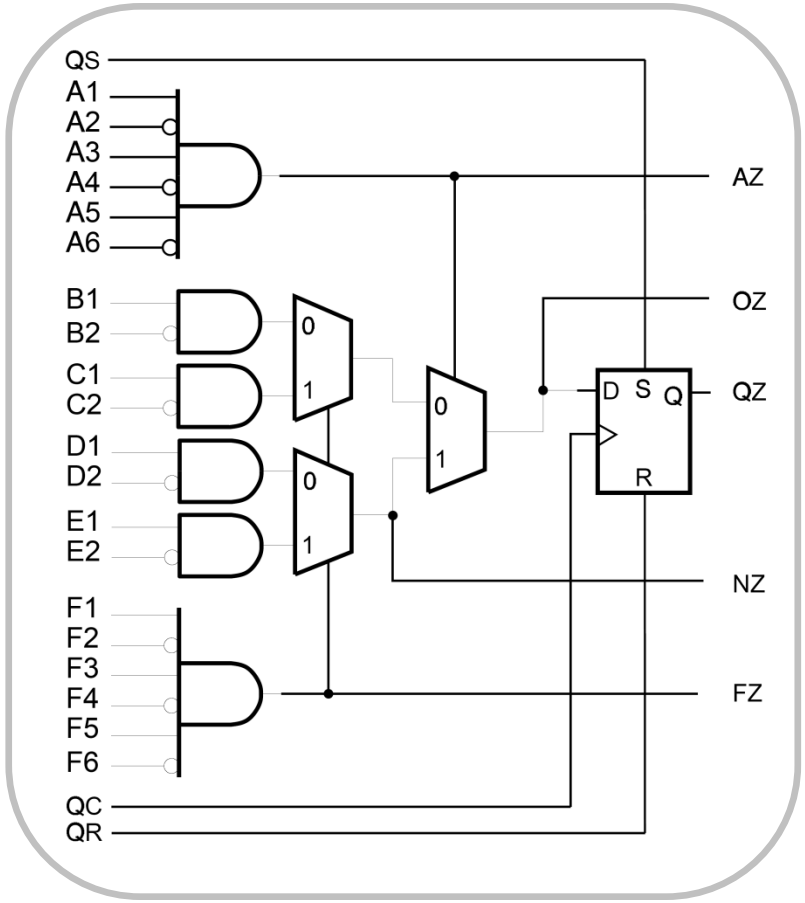
QuickLogic pASIC FPGAs

- The main competitor for Actel antifuse-based FPGAs
- Array based structure like Xilinx FPGAs
- MUX-Based logic blocks
- Interconnect consists of only long lines
  - Present at every crossing of LB pins & interconnect wires
  - Generous connectivity
- Metal-to-Metal antifuse structure
  - Called ViaLink
  - Less resistance than Actel PLICE



QuickLogic pASIC FPGAs

- Inside a pASIC Logic Block:



Commercial FPGA Products : Applications

- **Communication:**
  - Virtex 4/5/6 & Virtex II Pro **(Xilinx)**
  - Stratix II/III/IV & Stratix GX **(Altera)**

- **Consumer Electronics, Automotive &Micro Controllers:**
  - Spartan 3 **(Xilinx)**
  - Cyclone 2 **(Altera)**
  - ProASIC3/E **(Actel)**

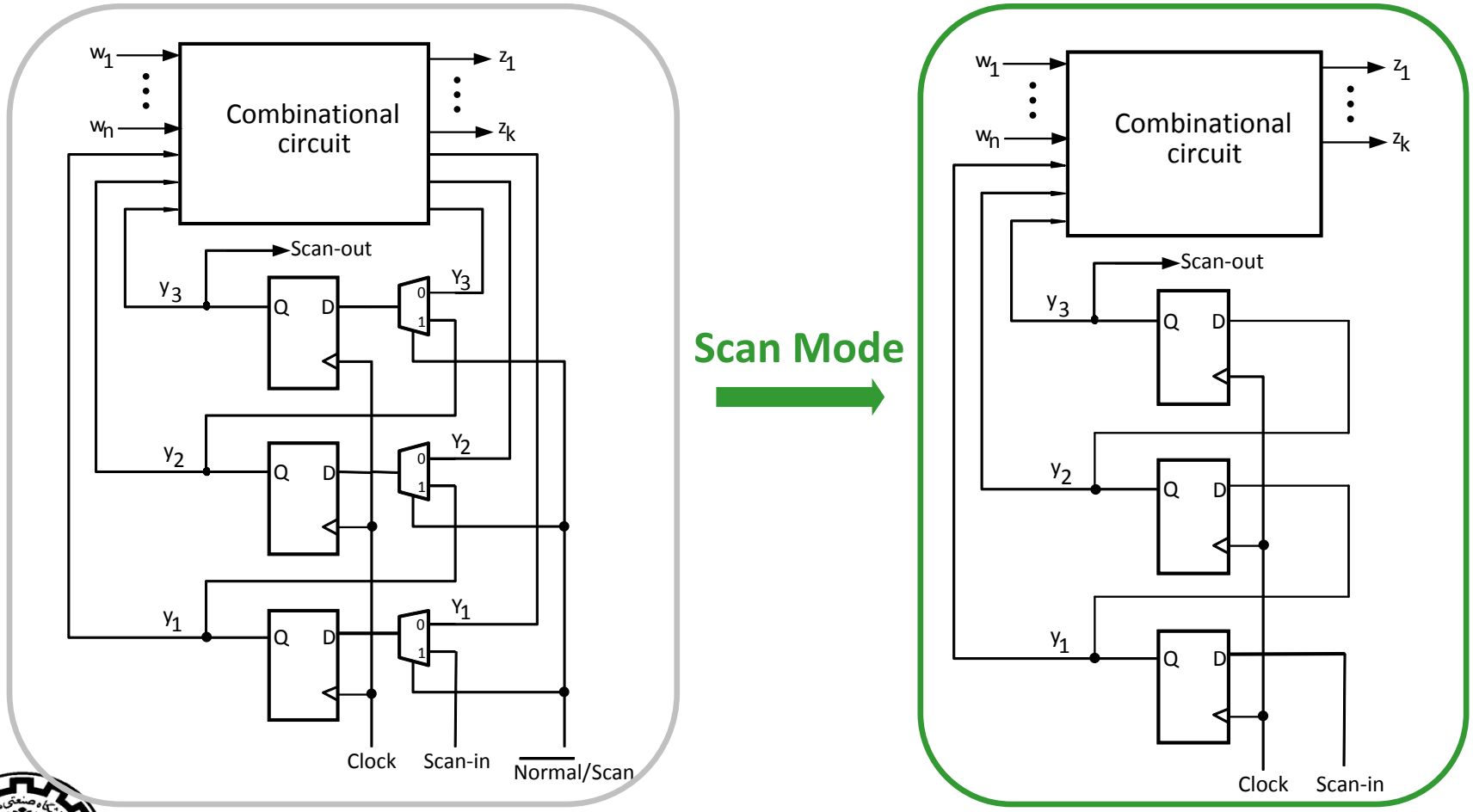
- **Aerospace & Military Applications:**
  - Axcelerator (**Actel**)

FPGA Specifications

- Number of I/O Pads
- Maximum clock frequency
- Number of equivalent gates that can be filled
- Amount of on-chip memory blocks
- Interfaces (such as PCI Express)
- On-chip CPU

FPGA Testing (Scan Chain)

- Many modern FPGAs have some **scan chains** into their testing circuitry
- Test circuitry is used to ensure that the chip/board was properly manufactured



FPGA Testing (JTAG)

- The **JTAG Standard** (Joint Test Action Group) was created to allow chips on boards to be easily tested

- It is also called “boundary scan” b/c it is designed to scan the pins at the boundary b/w the chip and the board

- JTAG is built into the pins of the chip

- During testing they are decoupled from the chip and used as a shift register

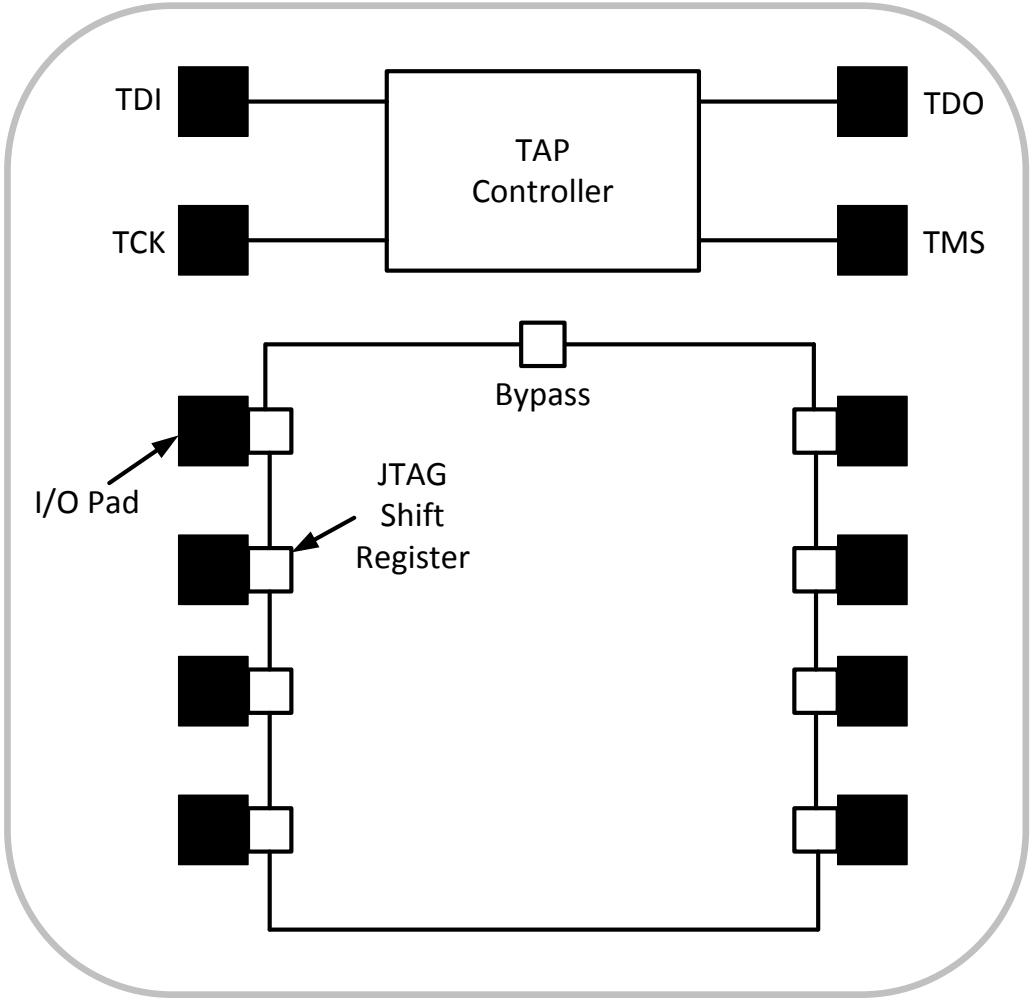
- Using the shift register, input values are placed on the chip’s pins and output values are read from the pins (controlled by test access port (TAP) block)

FPGA Testing

- JTAG has four pins:
  - TDI : Shift Register Input
  - TDO : Shift Register Output
  - TCK : Test Clock
  - TMS : Test Mode Select

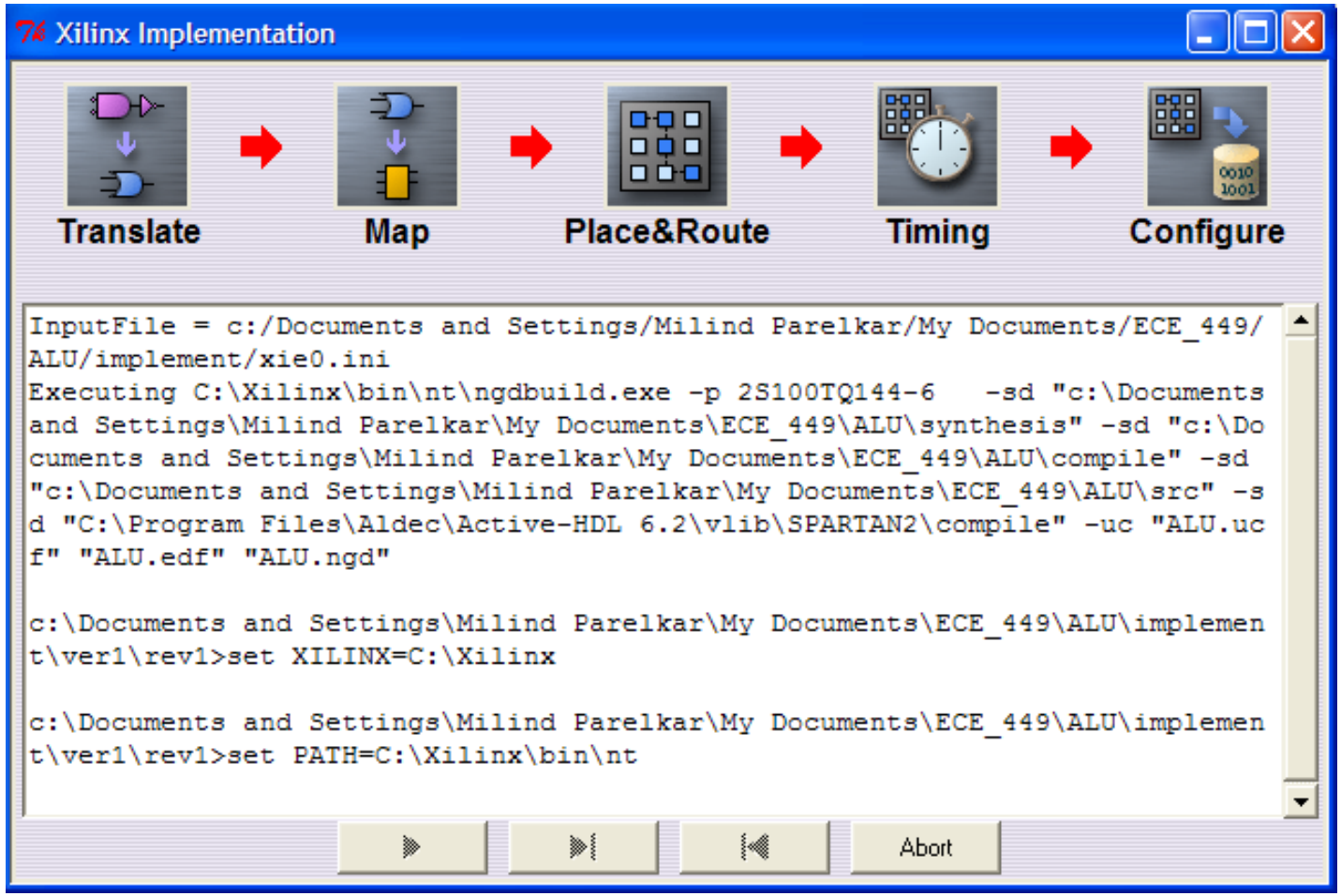
Two Important Factors in Testing:

- Controllability
- Observability



FPGA Design Flow

FPGA Design Flow

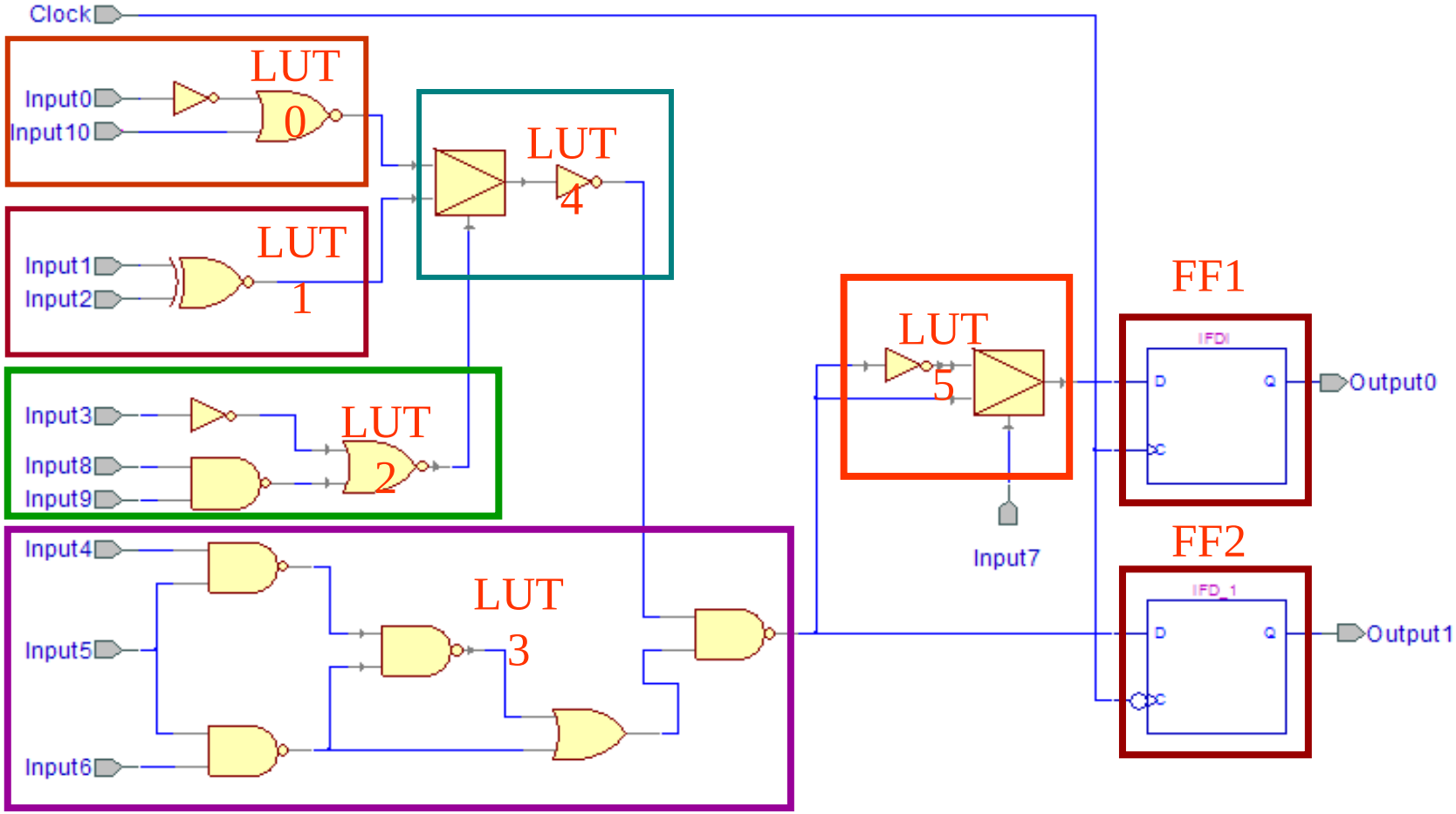


FPGA Design Flow: Mapping

- **Technology Mapping: Schematic/HDL to Physical Logic units**
- **Compile functions into basic LUT-based groups (function of target architecture)**

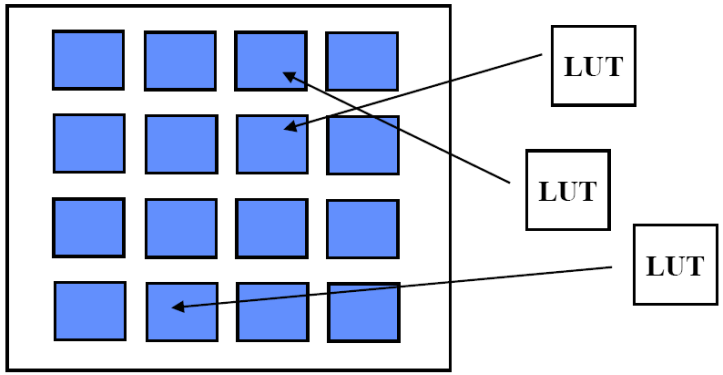

```
always @(posedge Clock or negedge Reset)
 begin
  if (! Reset)
    q <= 0;
  else
    q <= (a & b & c) | (b & d);
 end
```

FPGA Design Flow: Mapping

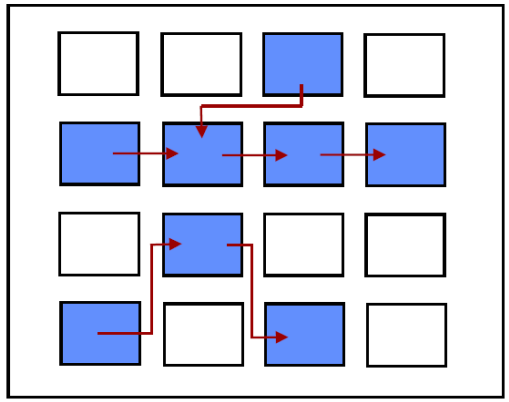


FPGA Design Flow: Placement & Routing

- **Placement** – assign logic location on a particular device



- **Routing** – iterative process to connect CLB inputs/outputs and IOBs. Optimizes critical path delay – can take hours or days for large, dense designs



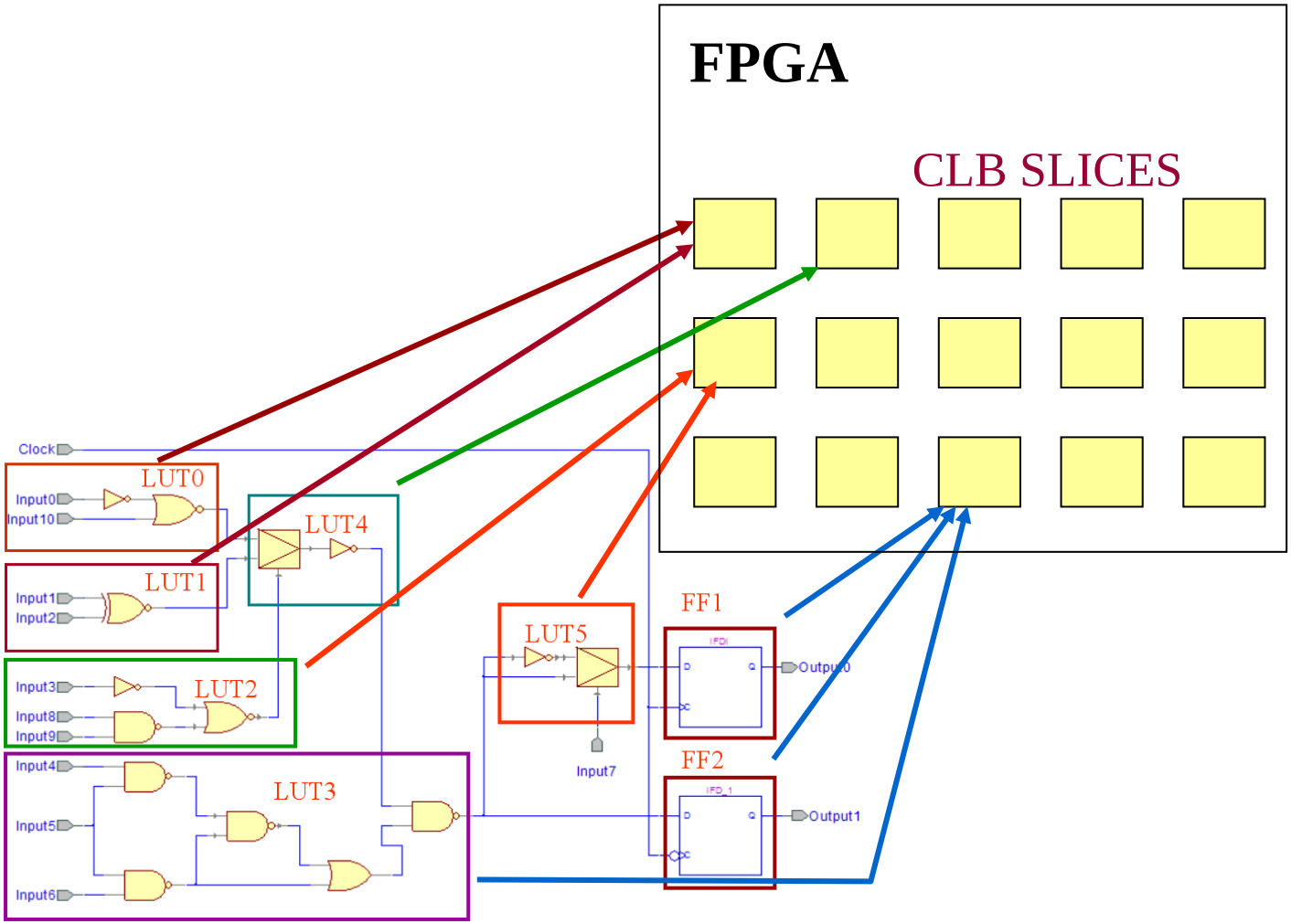
Iterate placement if timing not met

Satisfy timing? → Generate Bitstream to config device

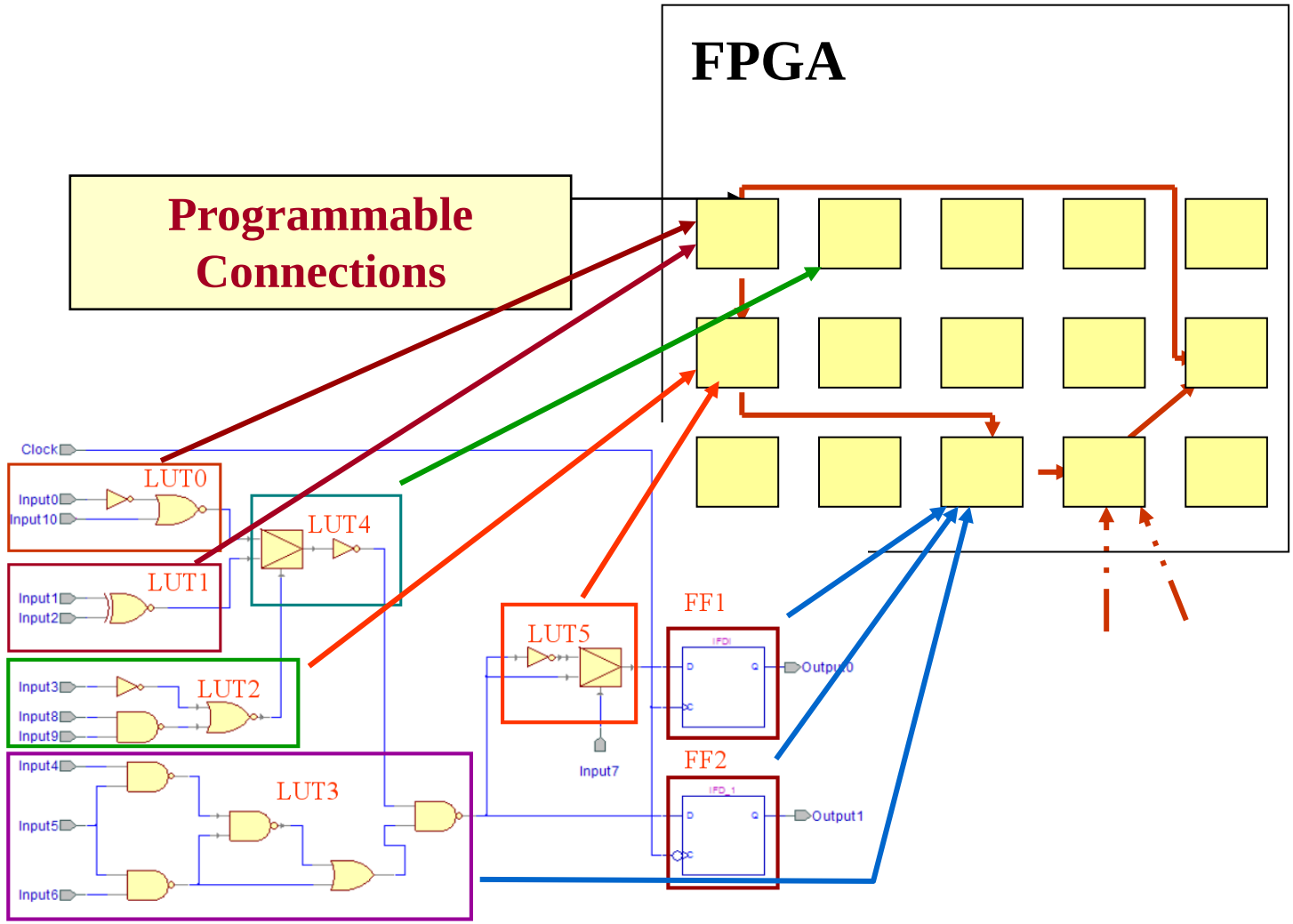
Challenge! Cannot use full chip for reasonable speeds (wires are not ideal).

Typically no more than 50% utilization.

FPGA Design Flow: Placement



FPGA Design Flow: Routing

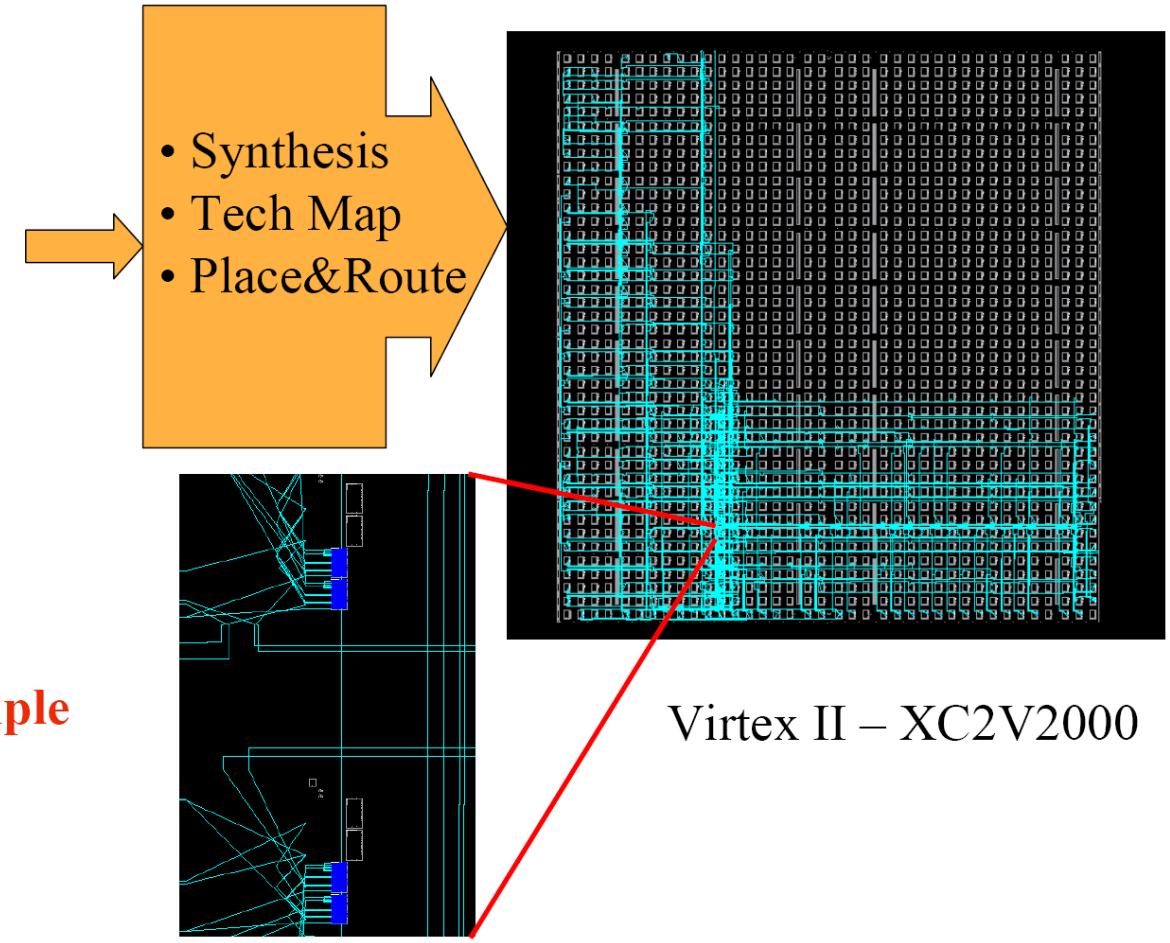


FPGA Design Flow: In a Glance

```
module adder64 (a, b, sum);
  input  [63:0] a, b;
  output [63:0] sum;

  assign sum = a + b;

endmodule
```

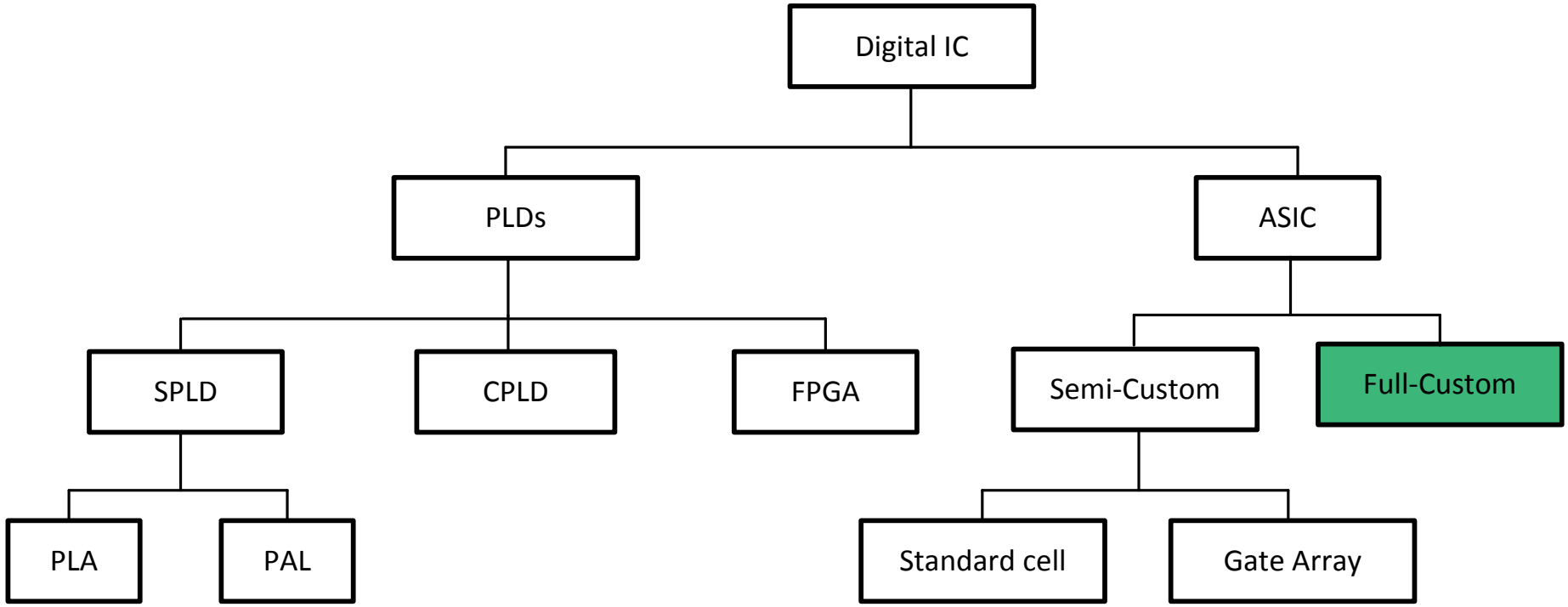


64-bit Adder Example

Outline

- Introduction
- Simple Programmable Logic Designs (SPLDs)
  - PLA
  - PAL
- Complex Programmable Logic Designs (CPLDs)
- Field-Programmable Gate Array (FPGAs)
  - Logic Blocks
  - Programmable Routing Switches
  - I/O Pads
- Commercial FPGA Products
- **Application Specific Integrated Circuits (ASICs)**

Full Custom VLSI Technology

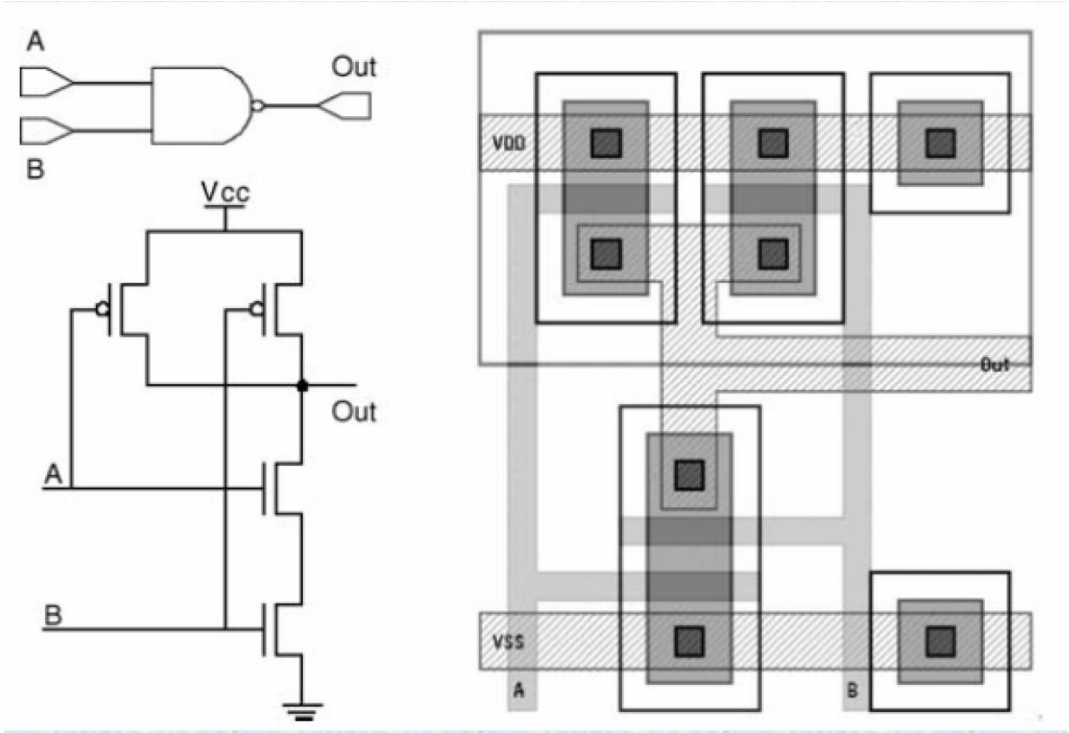


Full Custom VLSI Technology

- All layers are optimized/customized for the particular implementation:
  - Placing transistors
  - Sizing transistors
  - Routing wires

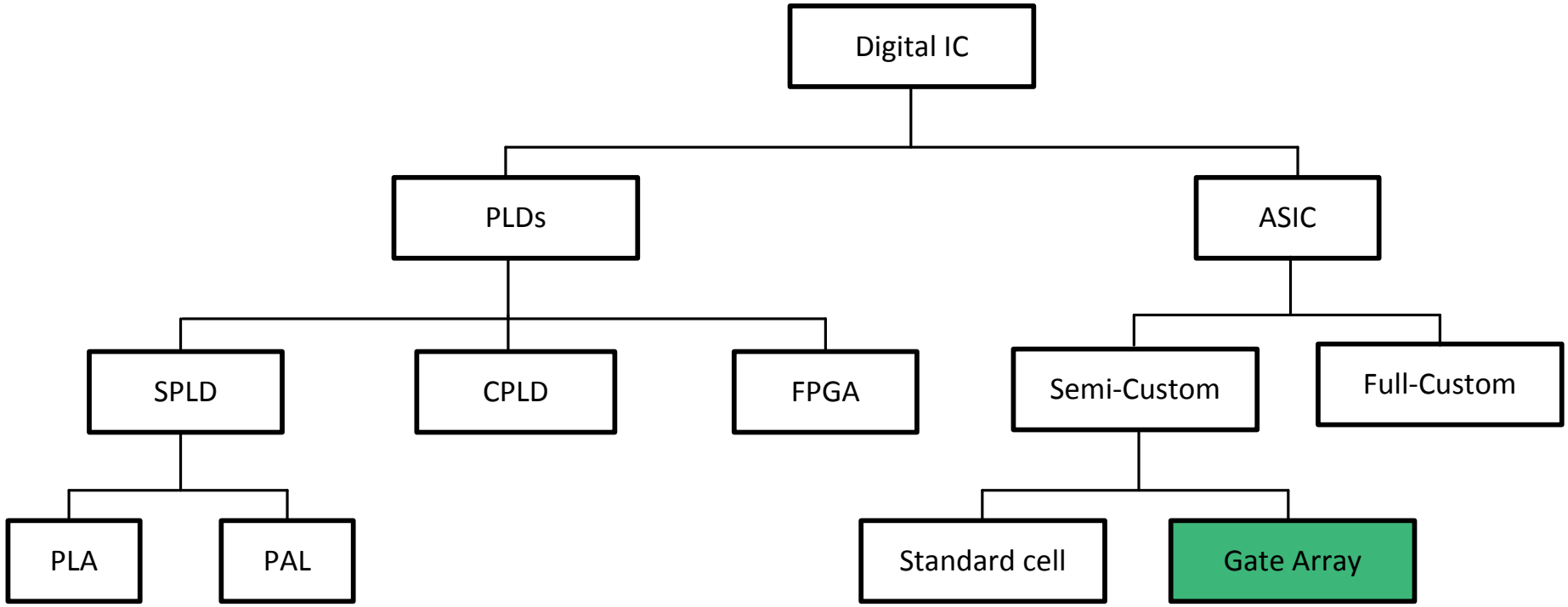
- **Benefits:**
  - Excellent performance
  - Small size
  - Low power

- **Drawbacks:**
  - High NRE cost
  - Long time-to-market



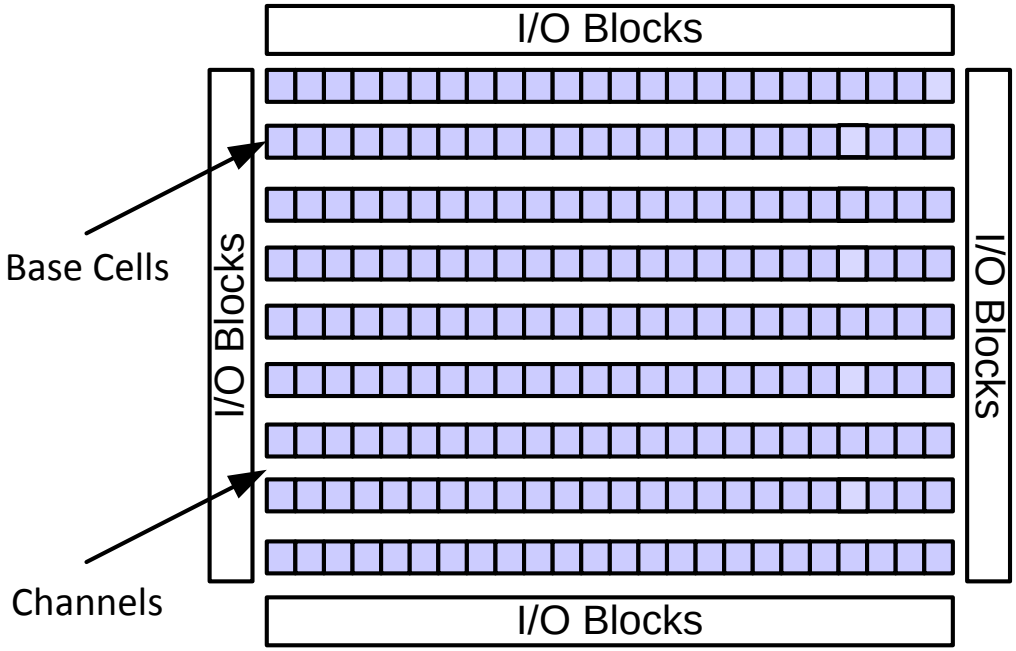
Not too common today!

Semi-Custom VLSI Technology (Gate Array)

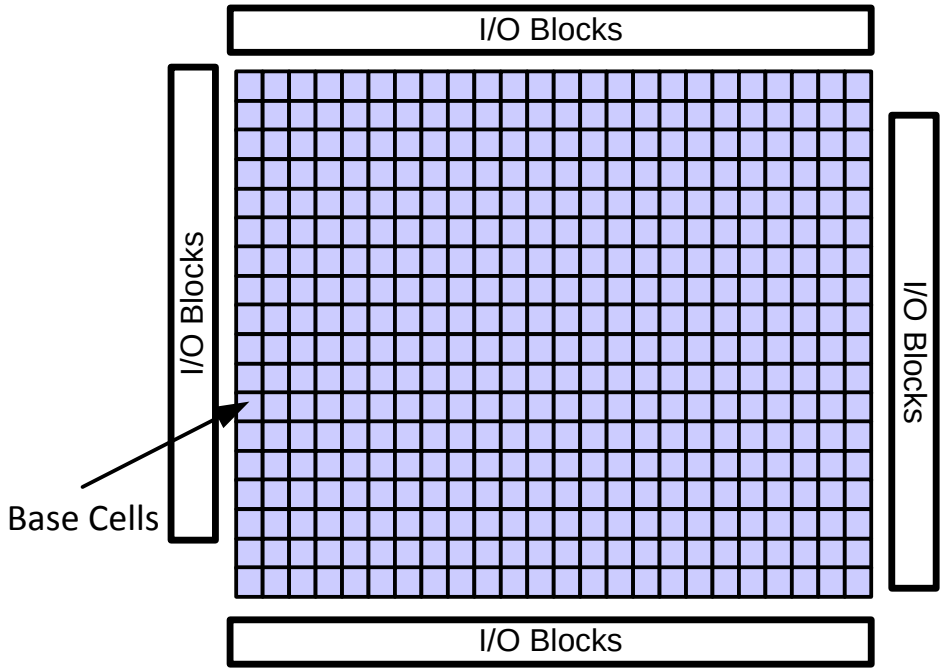


Semi-Custom VLSI Technology (Gate Array)

- Gate arrays (GAs) composed of arrays of p- and n-type transistors.
- The mapping, from transistors to gates, performed through CAD tools.

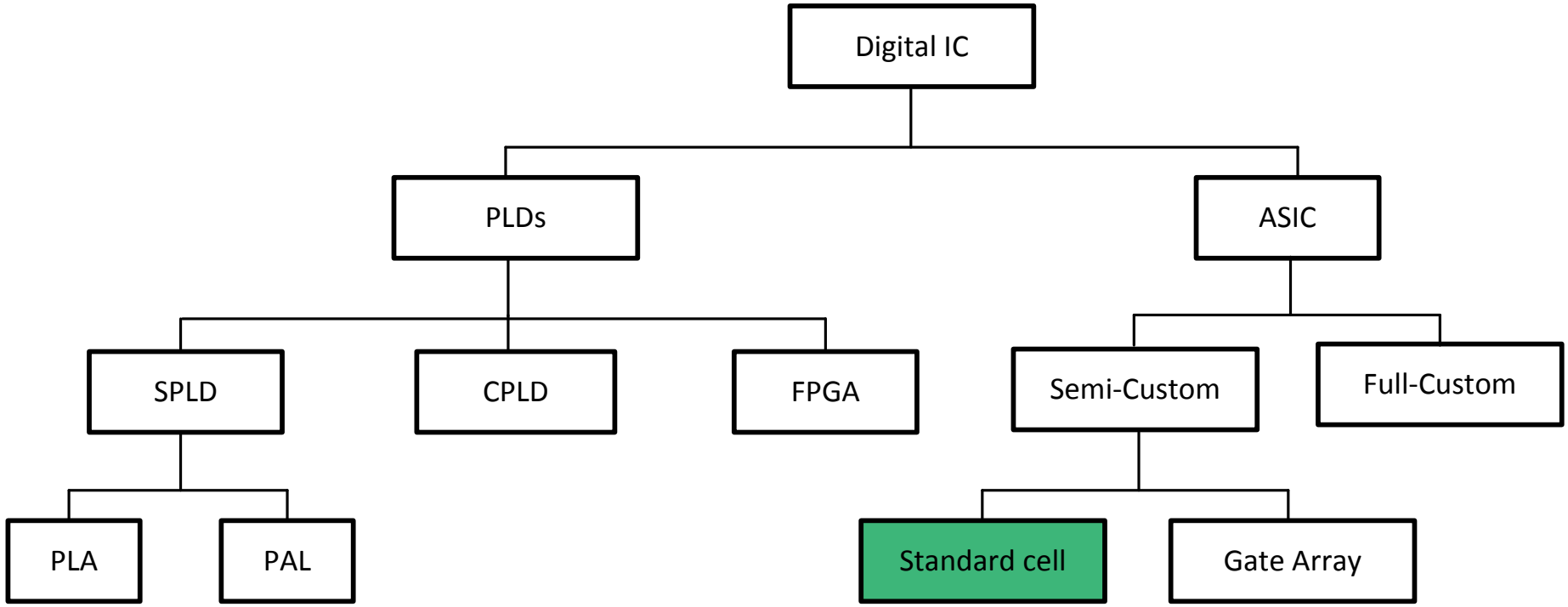


Channeled Gate Array



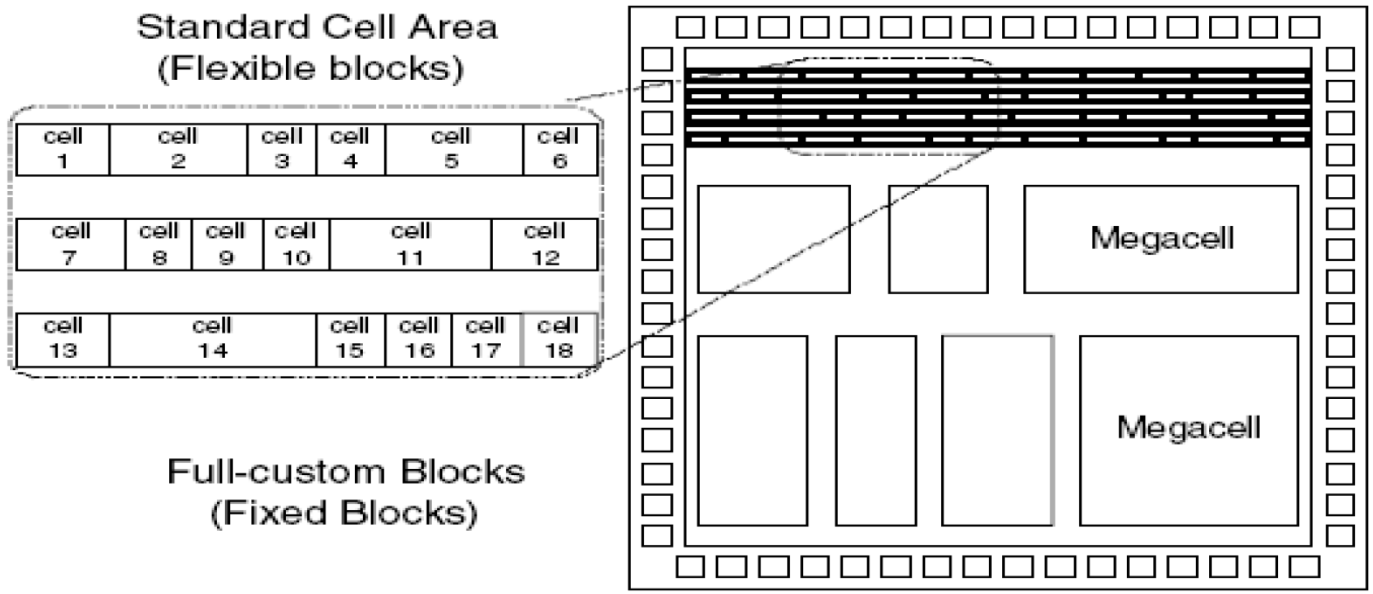
Channel-less Gate Array

Semi-Custom VLSI Technology (Standard Cell)



Standard Cell-Based ASICs

- Common logic components (e.g., gates, multiplexers, adders, …) previously designed and stored in a library for different area, speed, power requirements
- Logic components get converted to chip layouts.
- Standard-cell designs are organized, as rows of constant height cells.



FPGA vs. ASIC

❑ FPGA Advantages:

- ➢ Fast programming and testing time by the end user (instant turn-around)
- ➢ Excellent for prototyping
  - ▪ Easy to migrate from prototype to the final design
- ➢ Can be re-used for other designs
- ➢ Cheaper (in small volumes) ⟹ lower start-up costs
- ➢ Re-programmable
- ➢ Lower financial risk
- ➢ Ease of design changes/modifications
- ➢ Cheaper design tools

FPGA vs. ASIC

❑ FPGA Drawbacks:

- ➢ Slower than ASIC (2-3 times slower)
- ➢ Power hungry (up to 10 times more dynamic power)
- ➢ Use more transistors per logic function
- ➢ More area (20 to 35 times more area than a standard cell ASIC)

FPGA vs. ASIC

❑ ASIC Advantages:

- ➢ Faster
- ➢ Lower power
- ➢ Cheaper (if manufactured in large volumes)
- ➢ Use less transistors per logic function

❑ ASIC Drawbacks:

- ➢ Implements a particular design (not programmable)
- ➢ Takes several months to fabricate (long turn-around)
- ➢ More expensive design tools
- ➢ Very expensive engineering/mask cost for the first successful design

Implementation Approaches (ASIC vs. FPGA)

ASIC

Application **S**pecific
Integrated **C**ircuit

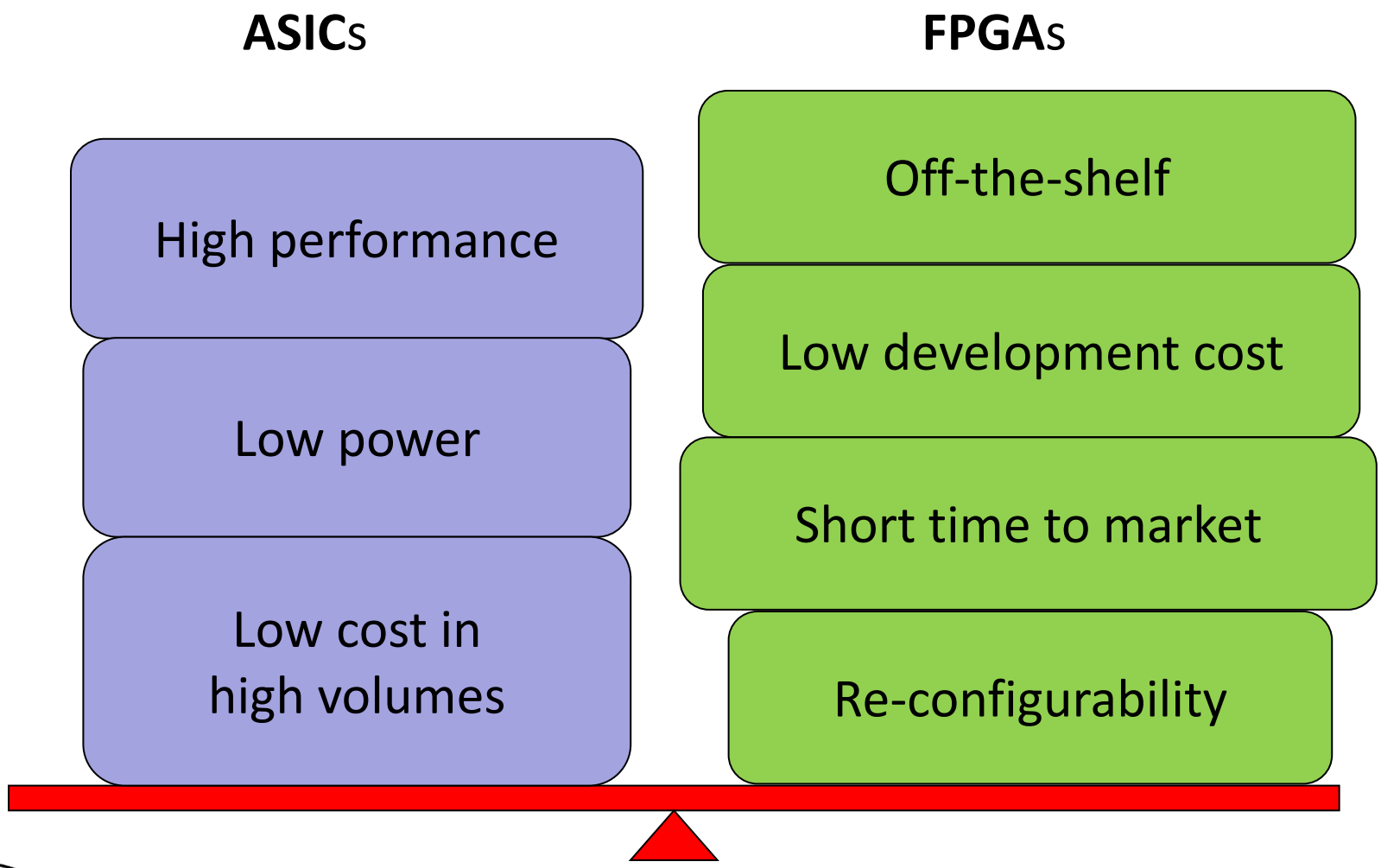
- ❑ Expensive & time consuming fabrication in semiconductor foundry
- ❑ Designed all the way from behavioral description to physical layout

FPGA

Field **P**rogrammable
Gate **A**rray

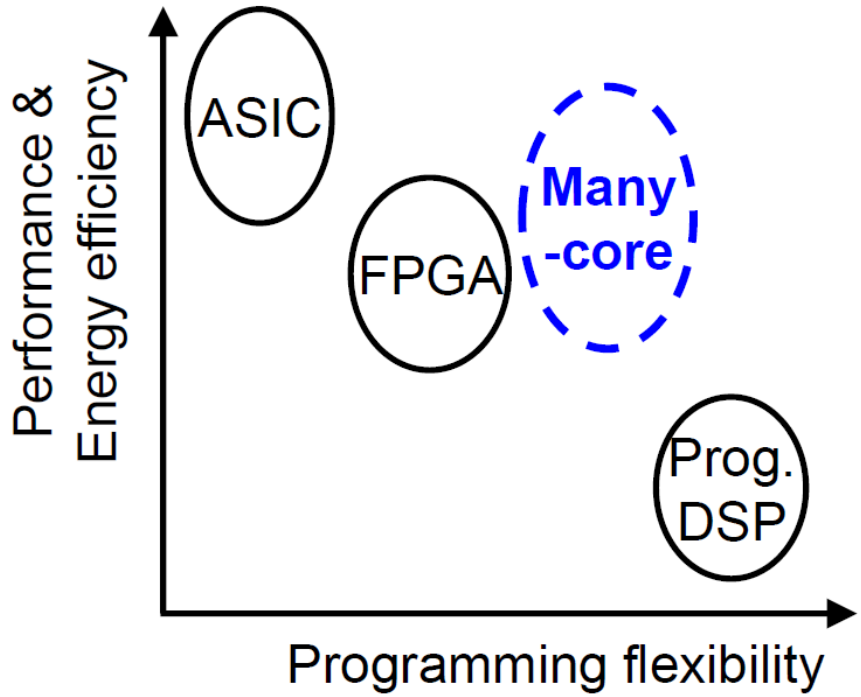
- ❑ Bought off the shelf & reconfigured by the end designers
- ❑ No physical layout design
- ❑ Design ends with a bitstream used to configure a device

Implementation Approaches (ASIC vs. FPGA)



Current Trend

- Programming flexibility
- High performance
  - Throughput
  - Latency
- High energy efficiency
- Suitable for future fabrication technologies



Target Many-core Architecture

- High performance
  - Exploit task-level parallelism in digital signal processing and multimedia
  - Large number of processors per chip to support multiple applications
- High energy efficiency
  - Voltage and frequency scaling capability per processor