CPLD’s Application

- For the Applications with several and/OR planes and a few number of FF like:
  - Graphical controller
  - LAN contreoller
  - UARTs

- Advantages
  - Higher speed
  - Simpler programmability
  - Possible predictability

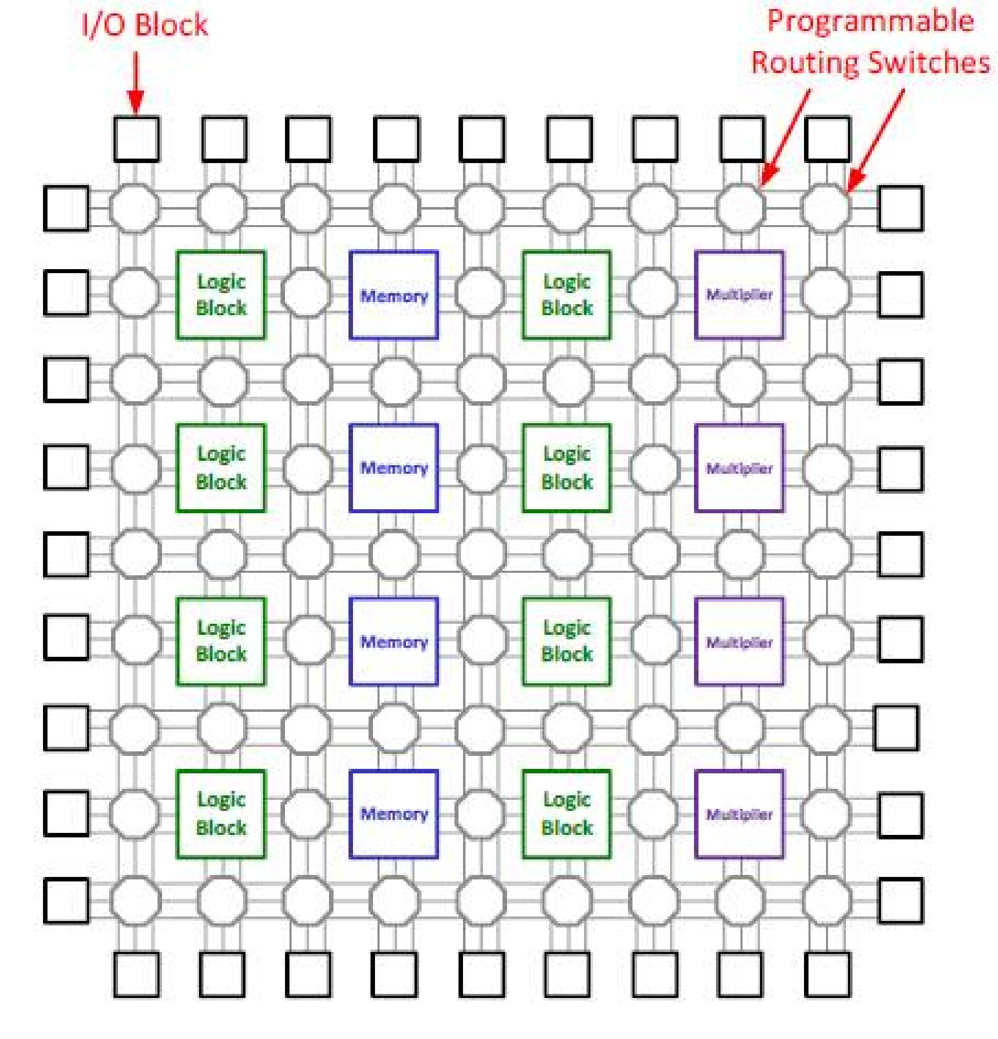
CPLD vs FPGA

1. Complexity of CPLD is between FPGA and PLD.
2. CPLD featured in common PLD:-
   1. Non-volatile configuration memory – does not need an external configuration PROM.
   2. Routing constraints. Not for large and deeply layered logic.

3. CPLD featured in common FPGA:-
   1. Large number of gates available.
   2. Can include complicated feedback path.

FPGA Architecture

- FPGAs consists of 3 main resources
- **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

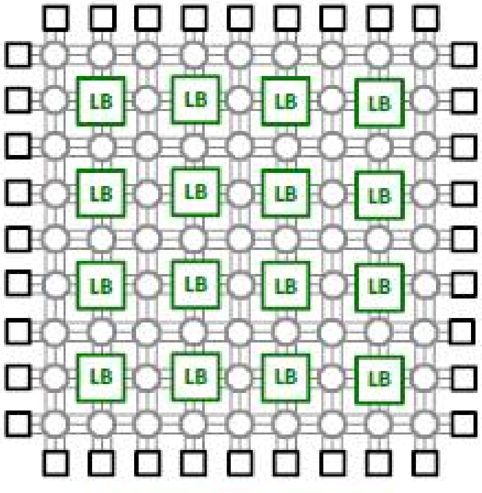


FPGA

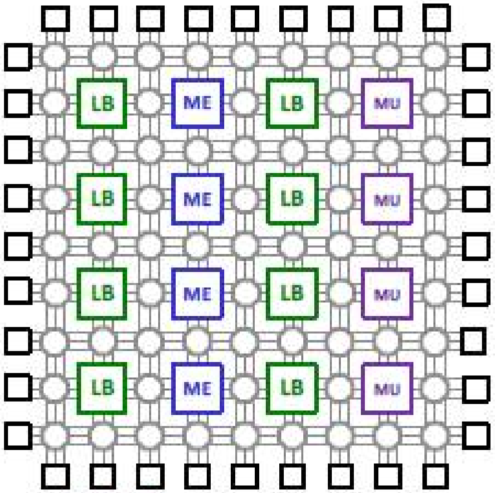
- Pre-fabricated silicon devices that can be electrically programmed to become any kind of digital circuit or system .
- A very large array of programmable logic blocks surrounded by programmable interconnects
- Contains logic blocks instead of AND/OR planes (multi-level logic of arbitrary depth)
- Can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

FPGA 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!



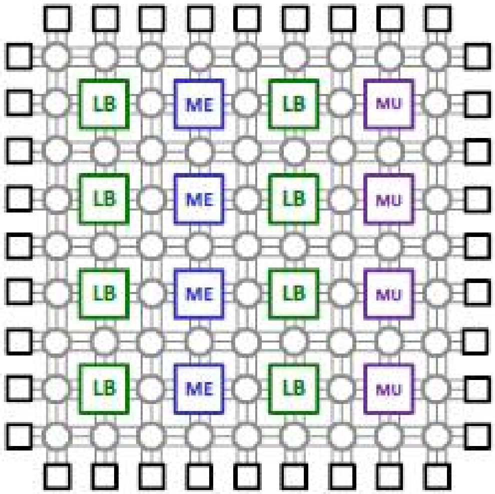
Homogeneous

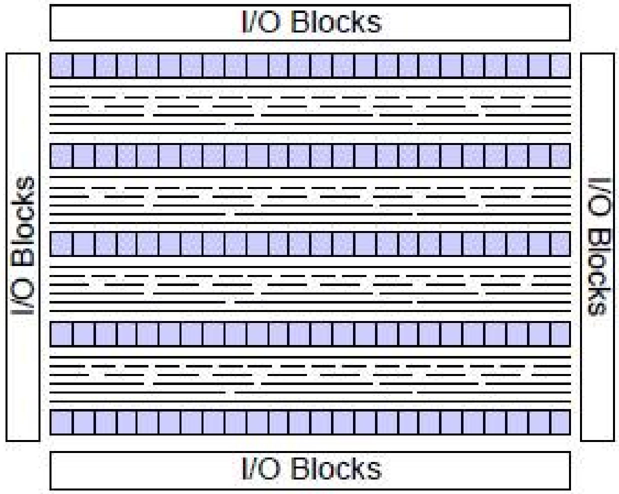


Heterogeneous

FPGA Categories (Floor Plan)

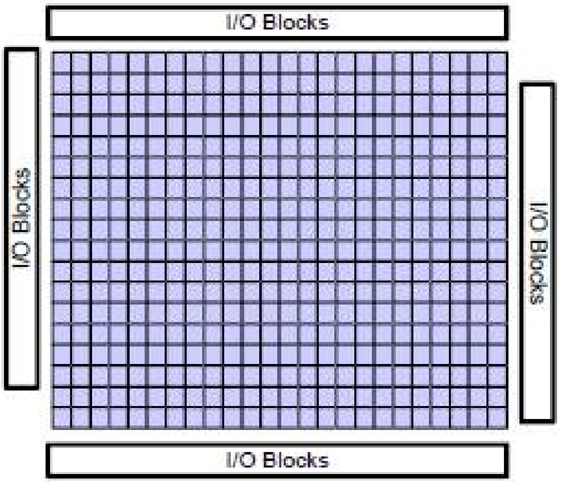
Symmetrical Array

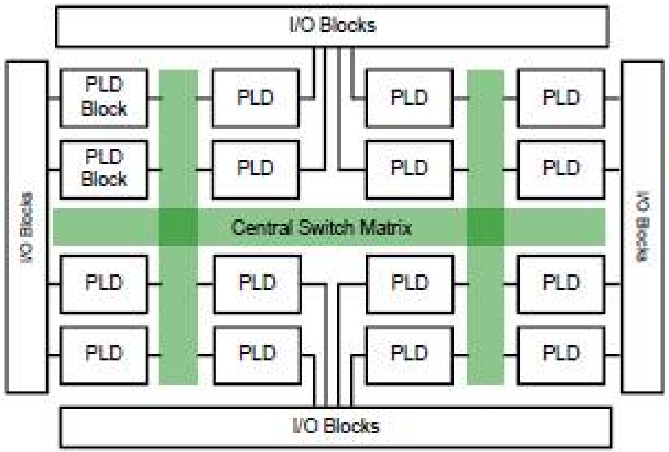




Row-Based

Sea-of-Gates



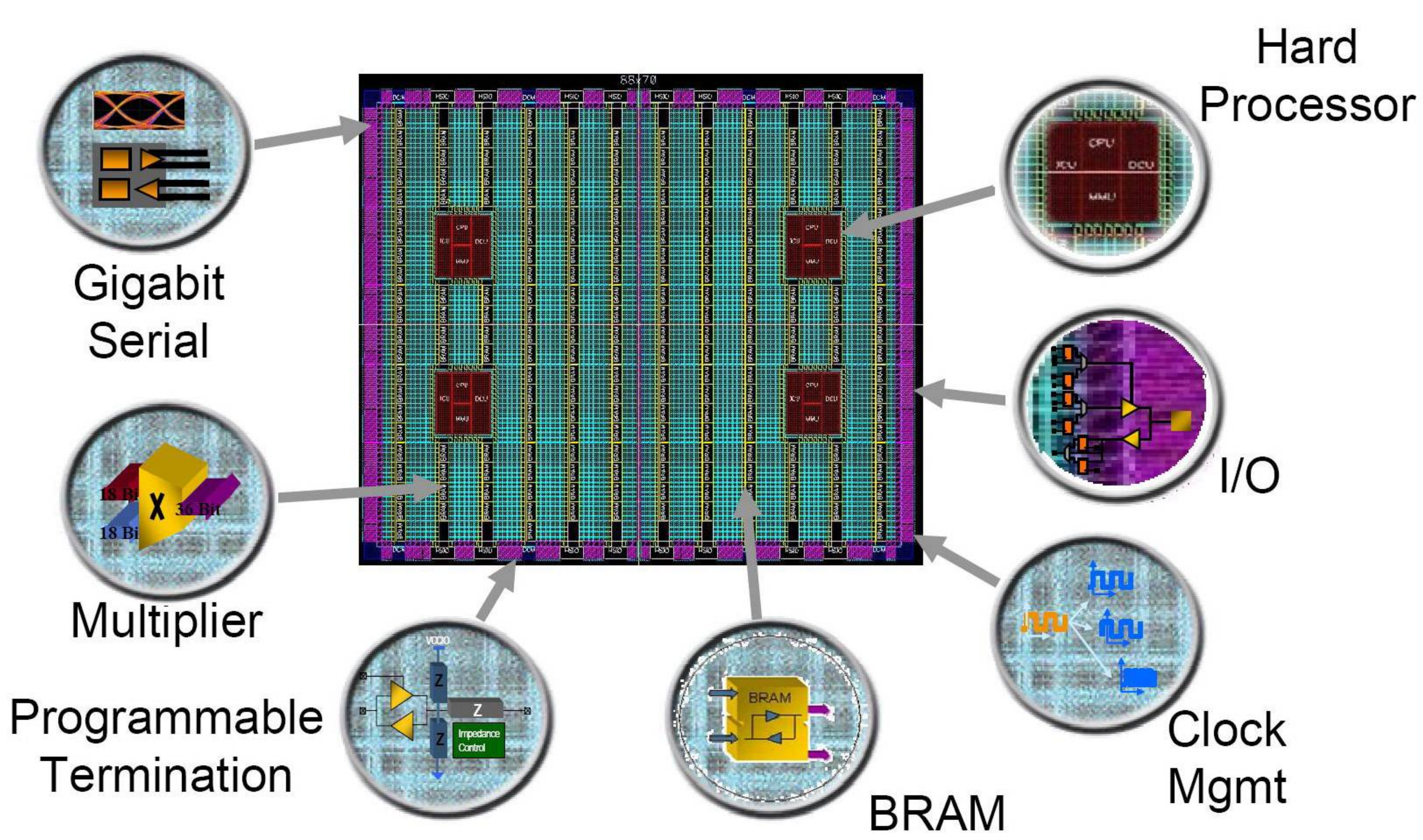


Hierarchical PLD

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

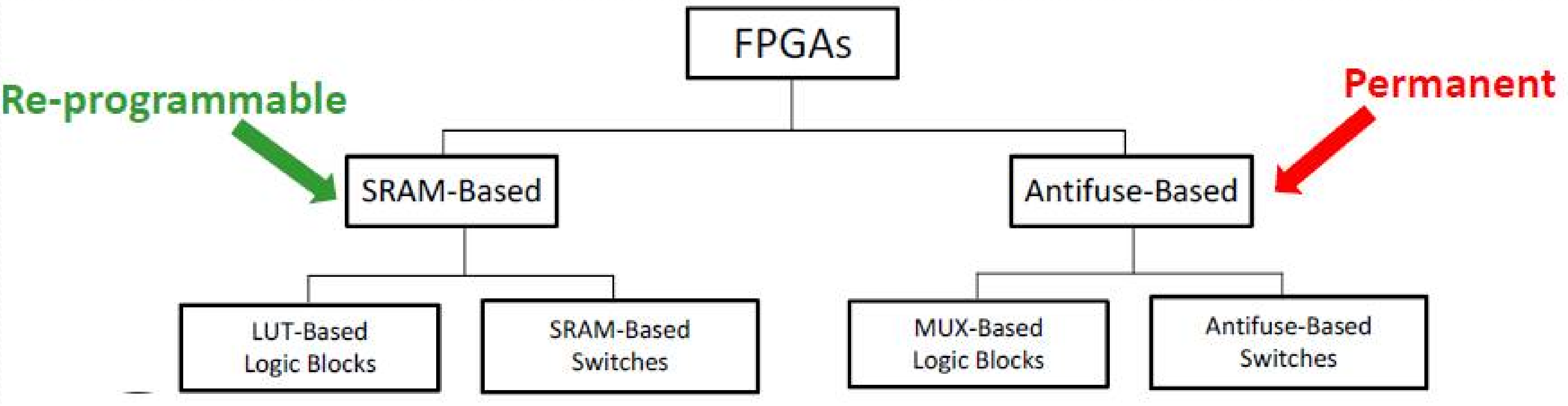
Modern Commercial FPGAs



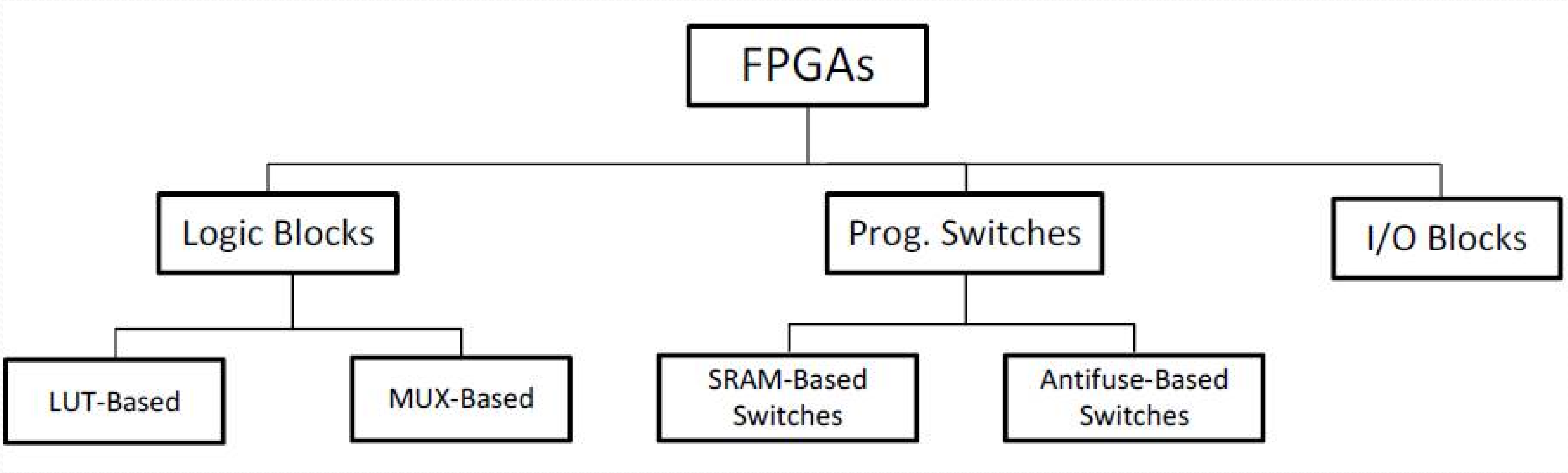
FPGA Categories (Fabrics)

- In general FPGAs are categorized to two general categories
- SRAM based FPGAs (Xilinx, Altera)(Reconfigurable and reprogrammable)
  - LUTs are used for implementing logical blocks
  - SRAM cells are used for reprogrammable switches

  - Anti-fuse based FPGA(Xilinx, Actel, Cypress, QuickLogic, Lattice(Permanent progrmming)
    - Using Mux for implementing logical blocks.
    - Anti-fuse are used for switches



FPGA Categories (Another View)

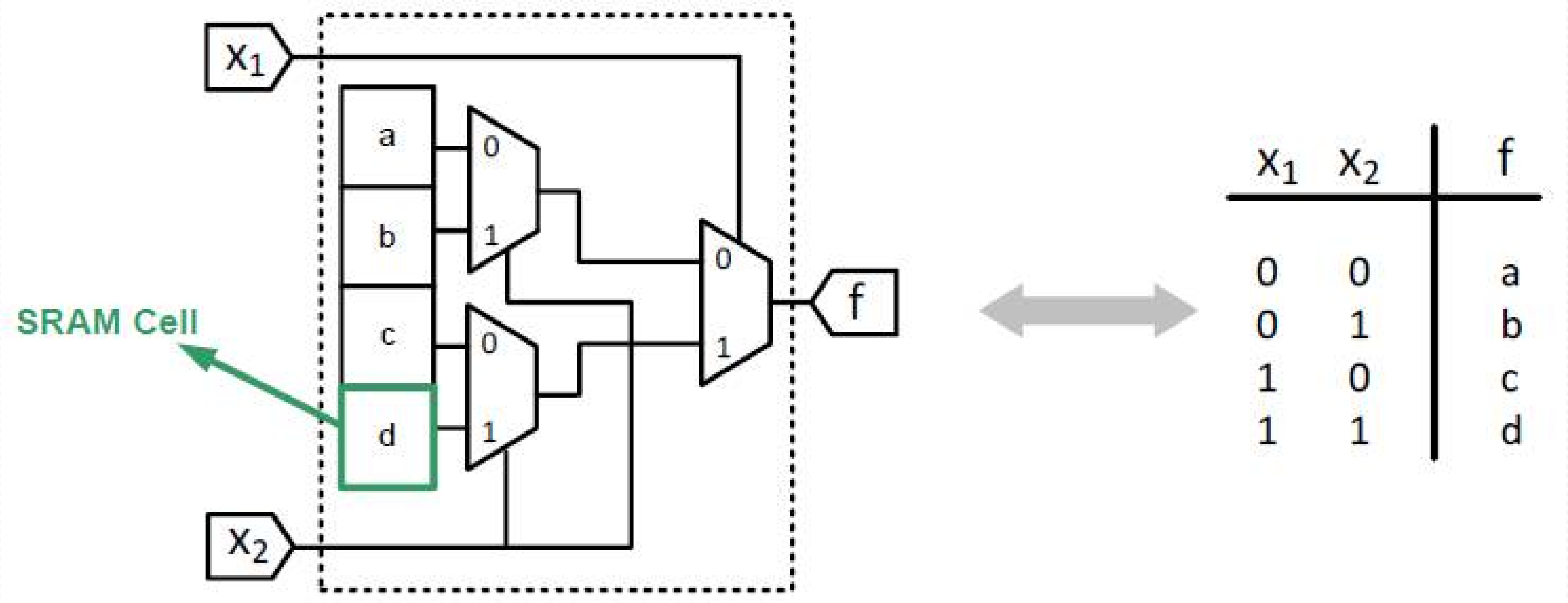


Logic Block

- 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 because:
  - A CMOS gate implements only one chosen logic function
  - An FPGA logic block must be configurable enough to implement a number of different functions

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

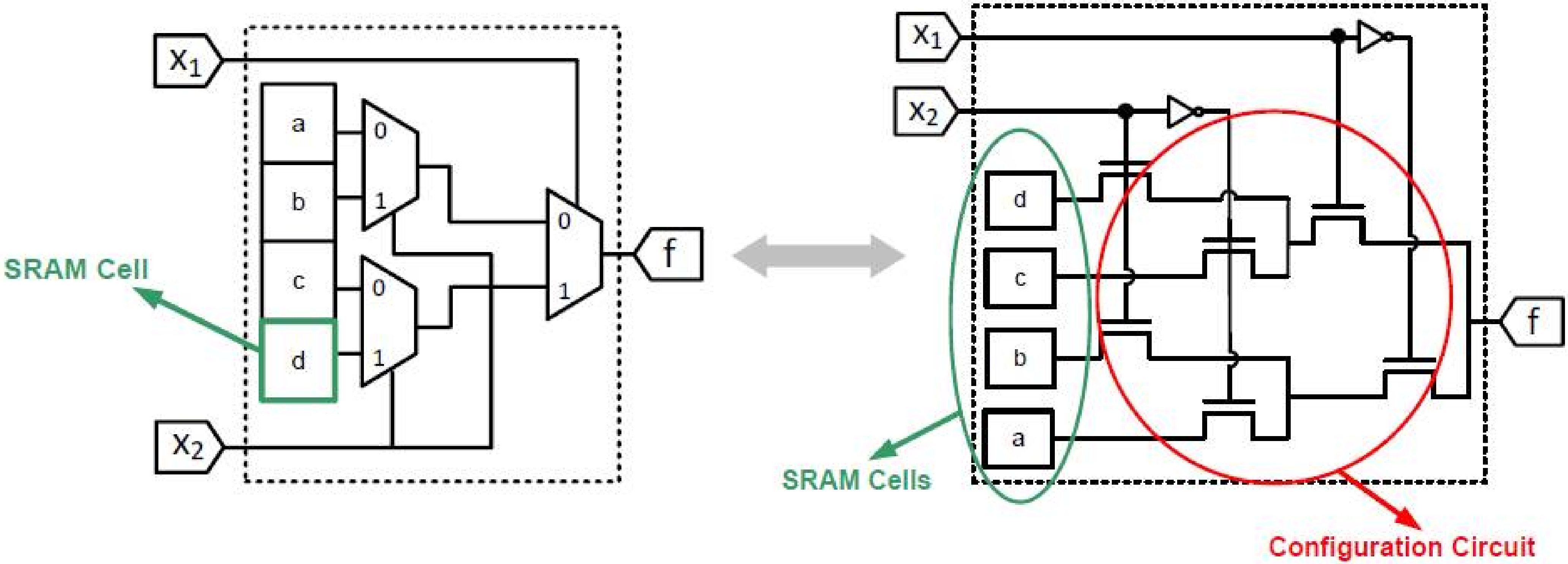
- [Lookup Table] (LUT)
  - Using one bit memory cells to implement logical functions
- Example: two input LUT
  - Every two input function could be implemented using this structure



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

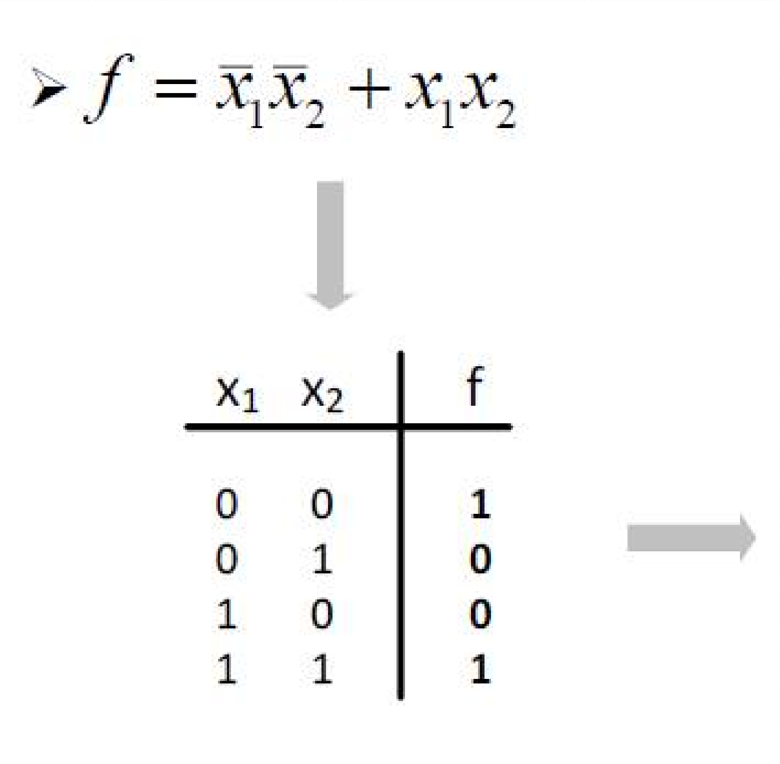
LUT-Based Logic Block

- LUT blocks consists the following circuits
  - Memory SRAM cells
  - Reconfigurable circuit to choose the address of memory



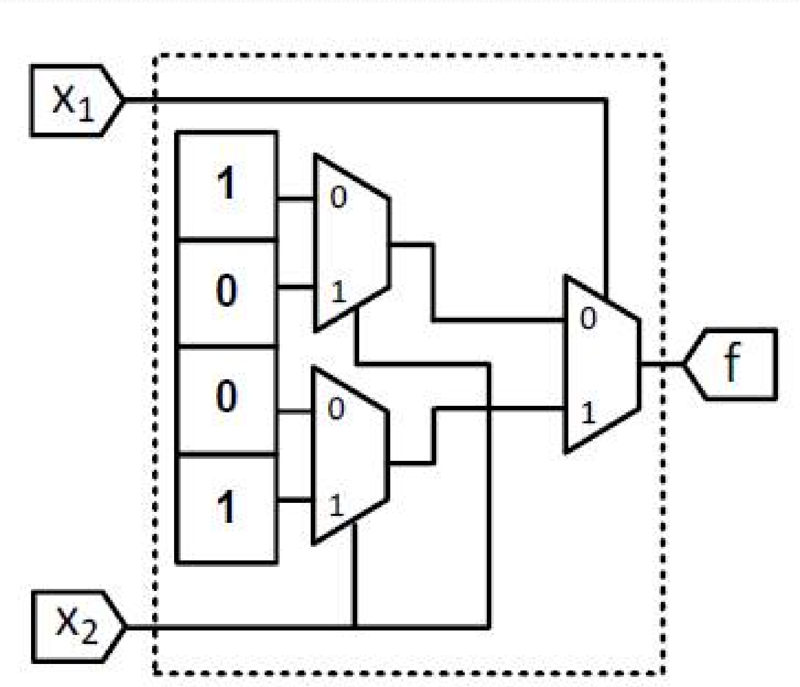
LUT-Based Logic Block

- Example



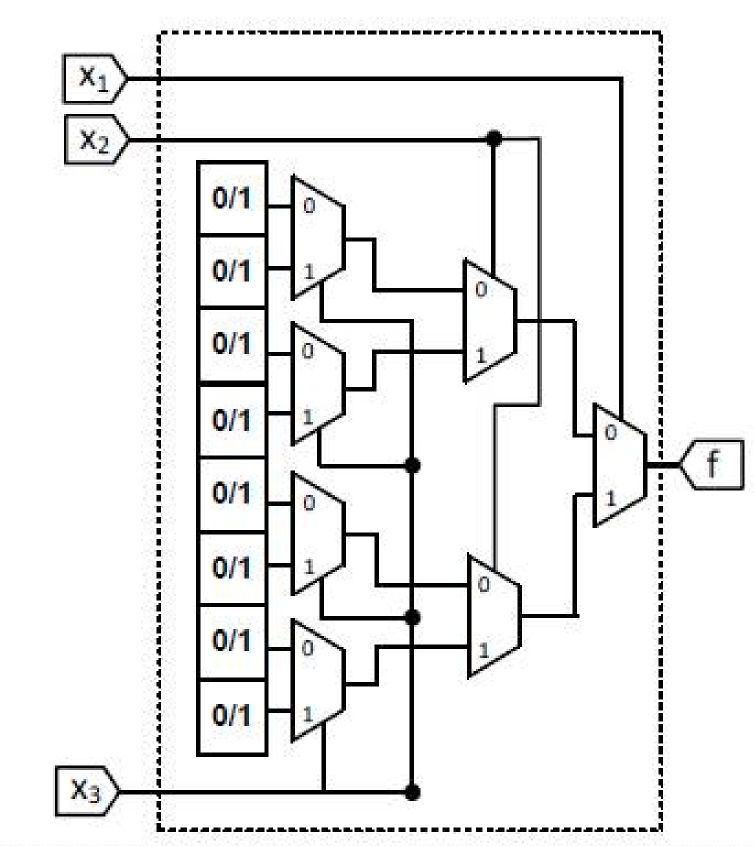
➢ $f = \bar{x}_1\bar{x}_2 + x_1x_2$

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



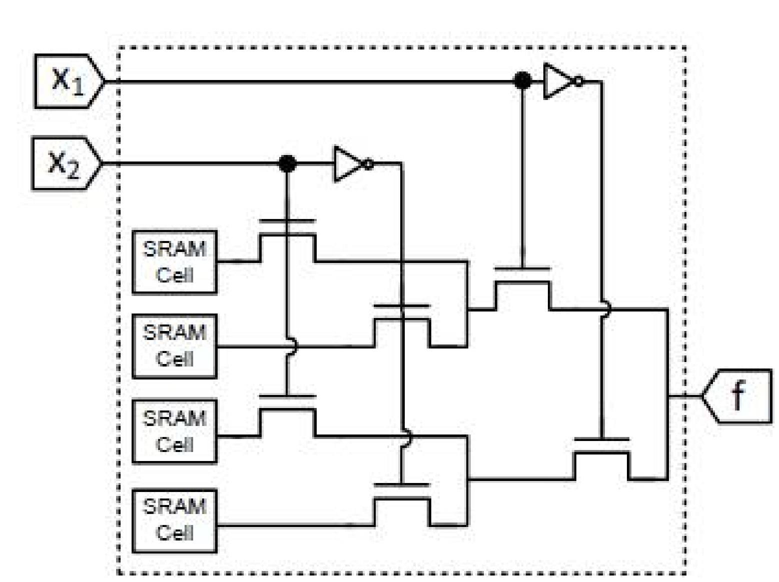
LUT-Based Logic Block

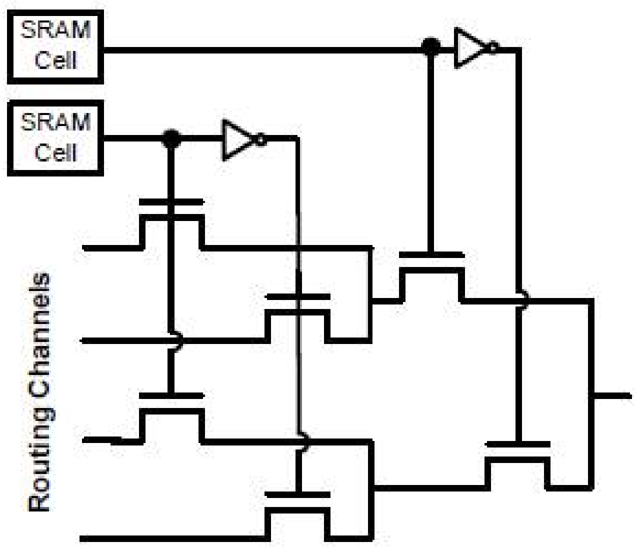
- Example: 3 inputs LUT
  - Can be used for implementing any 3 inputs function



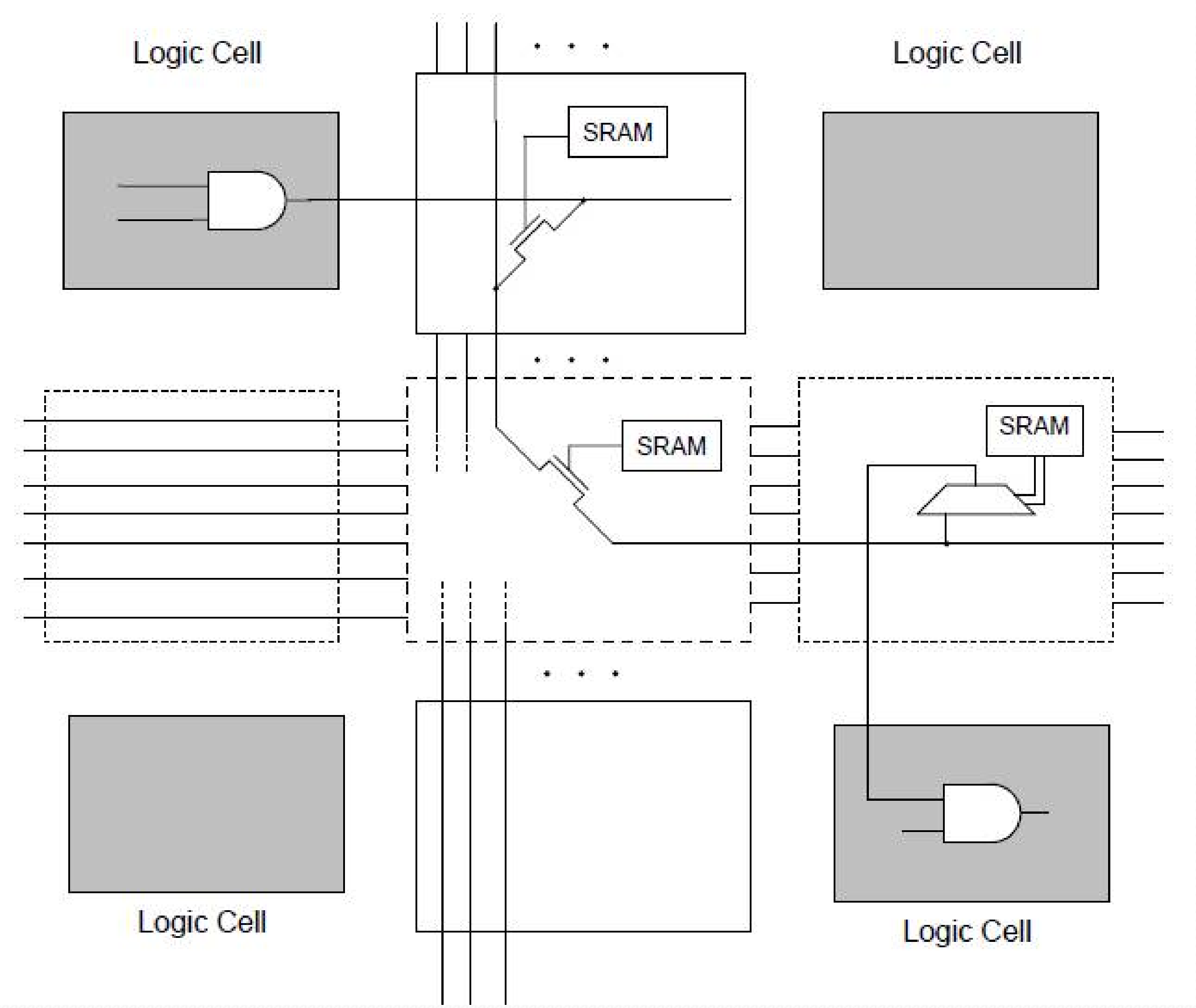
SRAM cells

- 1. For storing data in LUT for implementing logical function.
- 2. For select signal in programmable interconnect could be used.

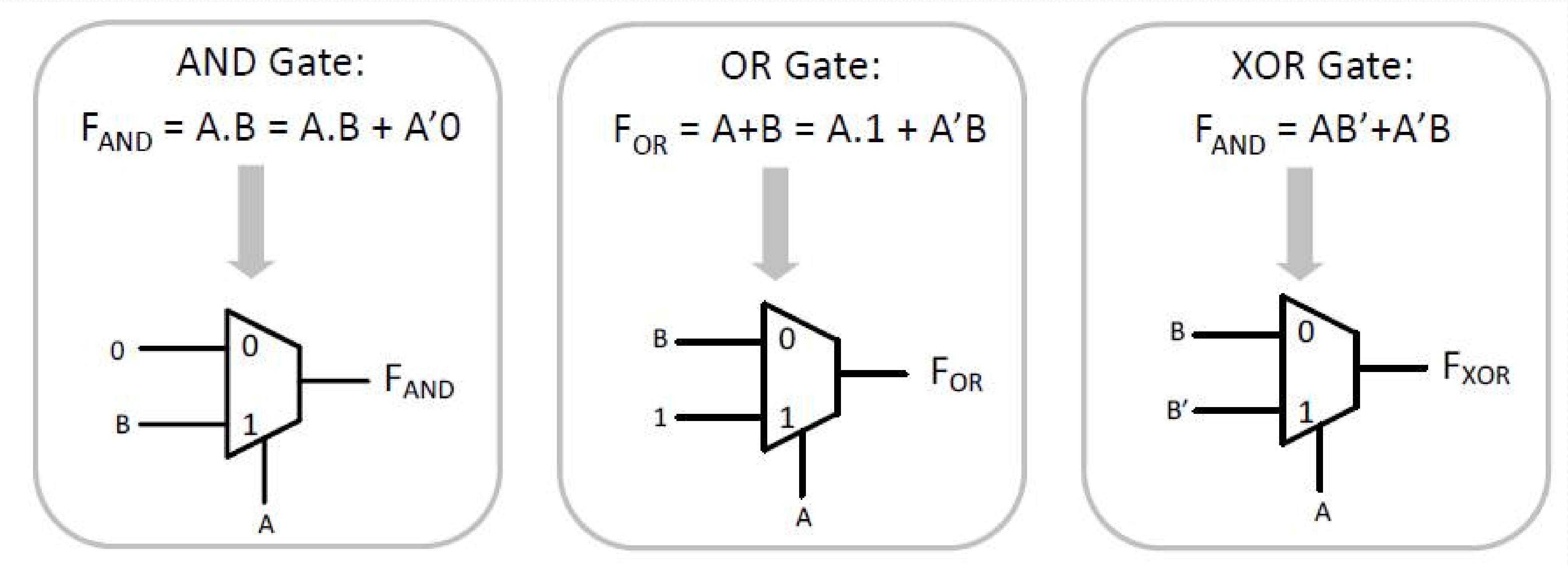




SRAM programming Technology



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

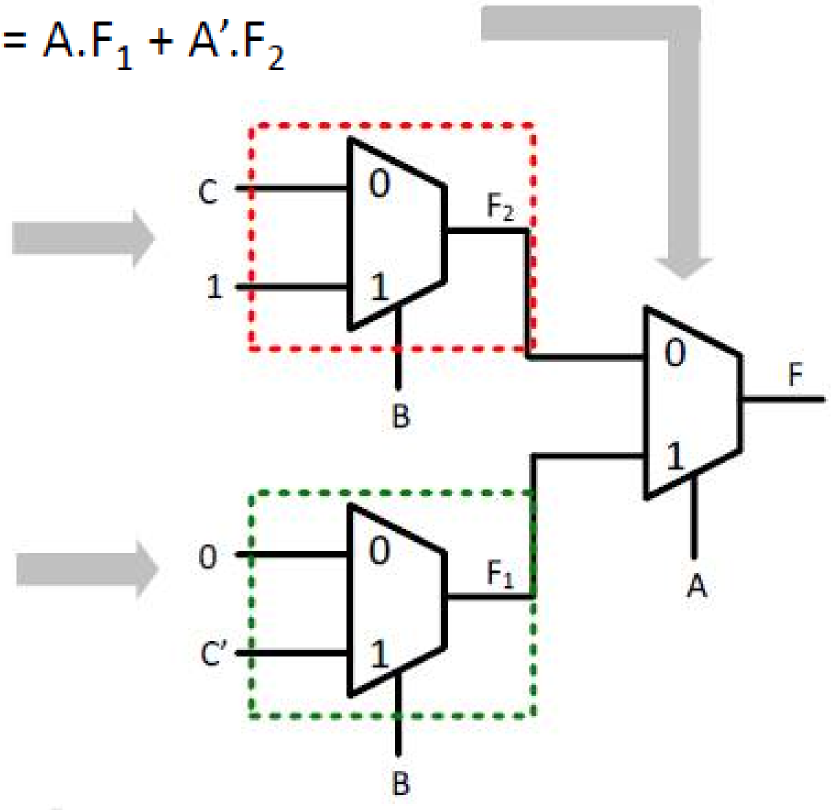
- Example

➢ $F(A, B, C) = A'B + ABC' + A'B'C$

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

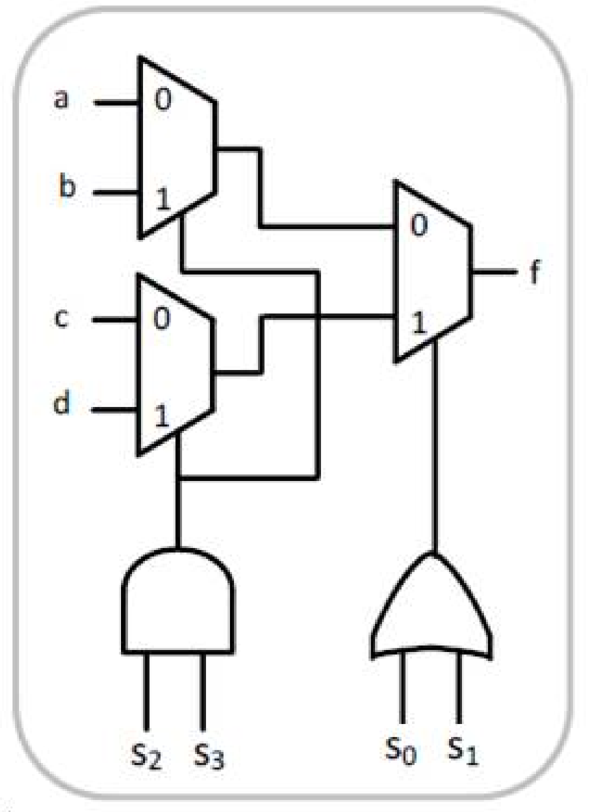
➢ $F_2 = B+B'C = B.1 + B'.C$

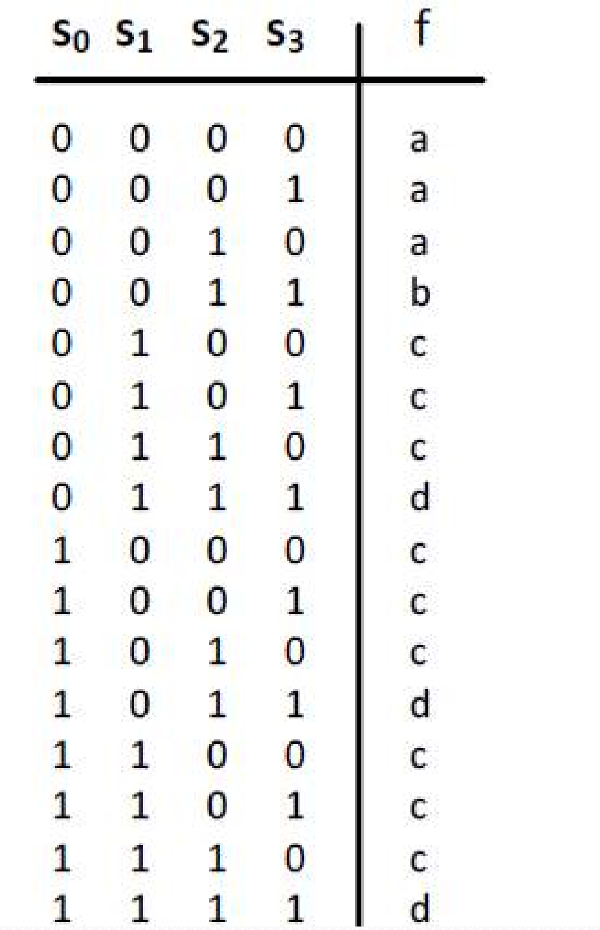
➢ $F_1 = BC' = BC' + B'.0$



MUX_based Logic Block

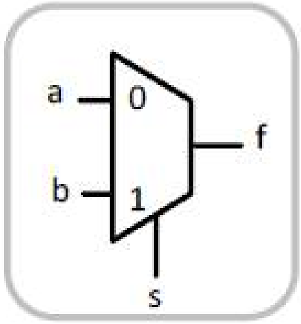
- Example





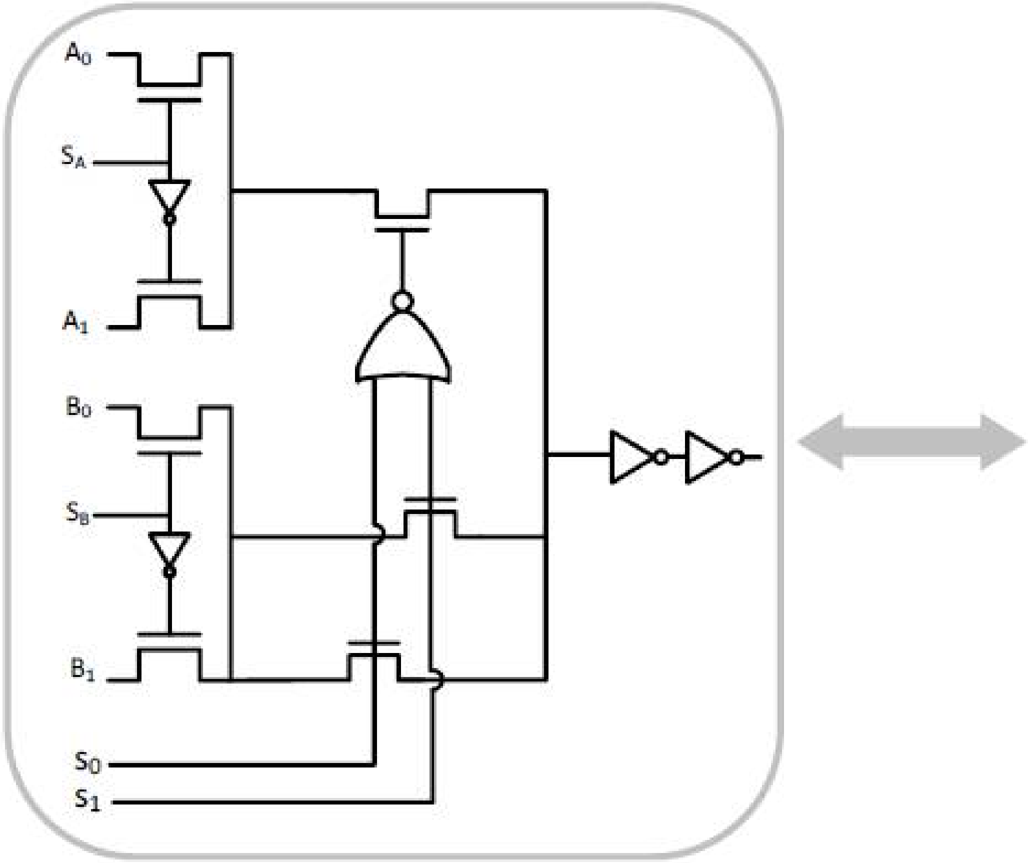
S_0	S_1	S_2	S_3	f
0	0	0	0	a
0	0	0	1	a
0	0	1	0	a
0	0	1	1	b
0	1	0	0	c
0	1	0	1	c
0	1	1	0	c
0	1	1	1	d
1	0	0	0	c
1	0	0	1	c
1	0	1	0	c
1	0	1	1	d
1	1	0	0	c
1	1	0	1	c
1	1	1	0	c
1	1	1	1	d

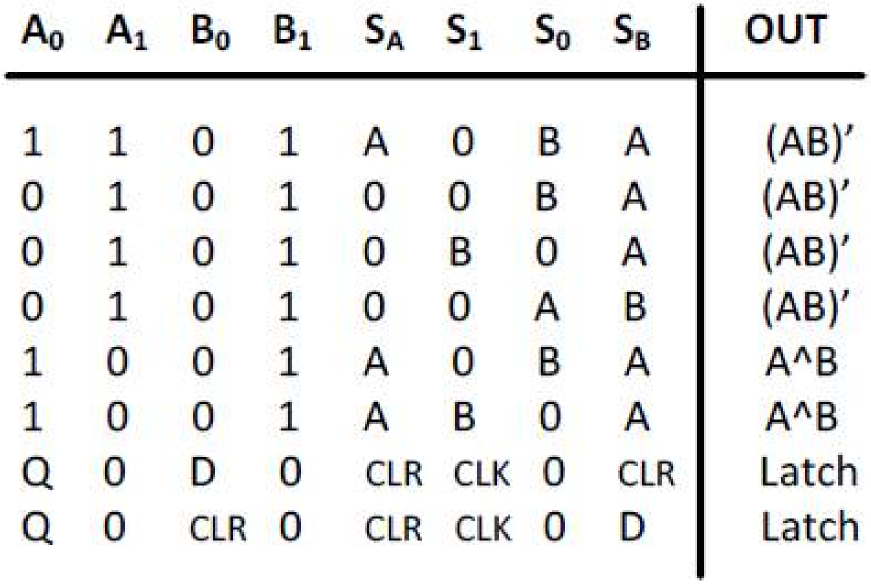
MUX_based configurable Logic Block



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 configurable Logic Block

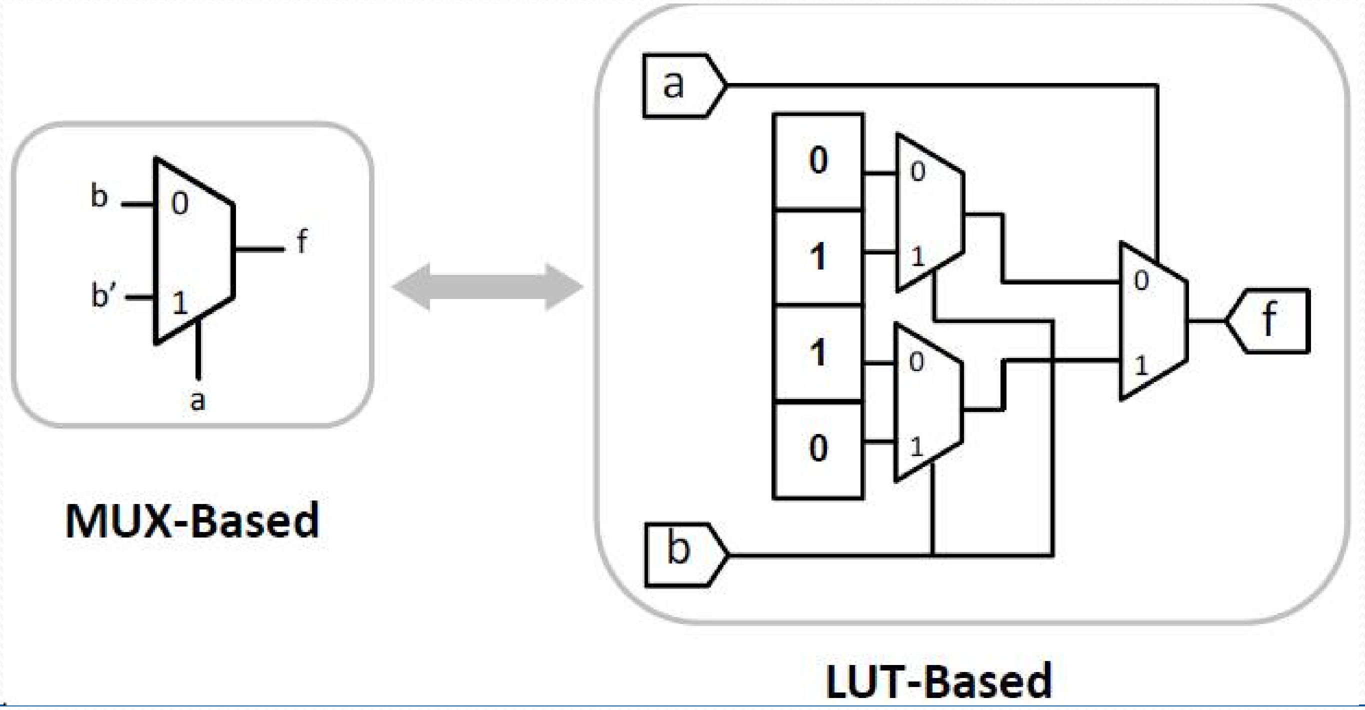




A_0	A_1	B_0	B_1	S_A	S_1	S_0	S_B	OUT
1	1	0	1	A	0	B	A	(AB)’
0	1	0	1	0	0	B	A	(AB)’
0	1	0	1	0	B	0	A	(AB)’
0	1	0	1	0	0	A	B	(AB)’
1	0	0	1	A	0	B	A	A^B
1	0	0	1	A	B	0	A	A^B
Q	0	D	0	CLR	CLK	0	CLR	Latch
Q	0	CLR	0	CLR	CLK	0	D	Latch

Comparison (MUX_based & LUT_based)

- XOR implementation using 2 methods
  - LUT based
  - MUX based



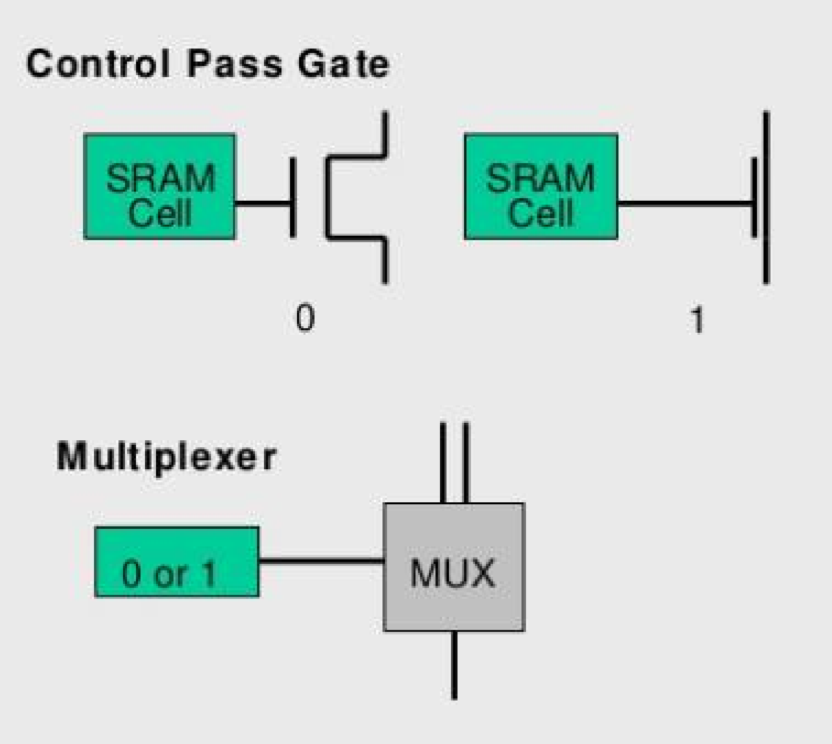
MUX-Based

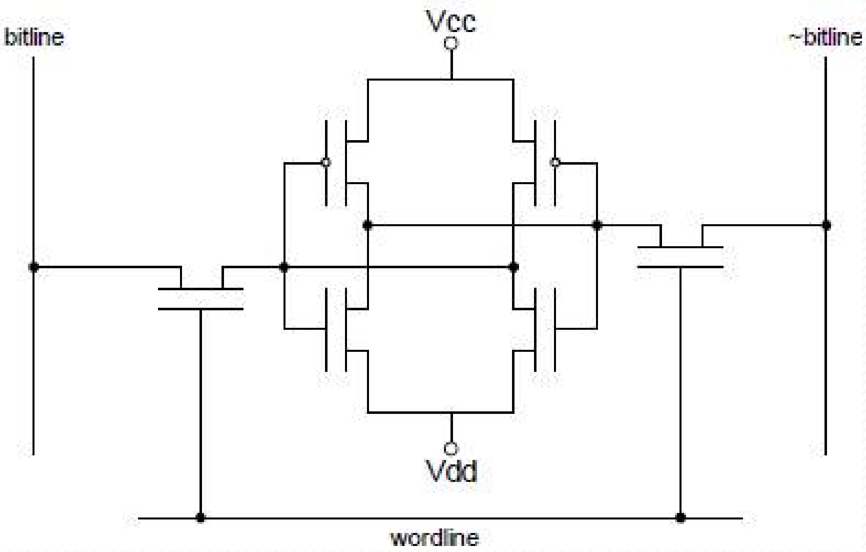
LUT-Based

Interconnects

- SRAM
- Antifuse
- EPROM

SRAM Interconnect



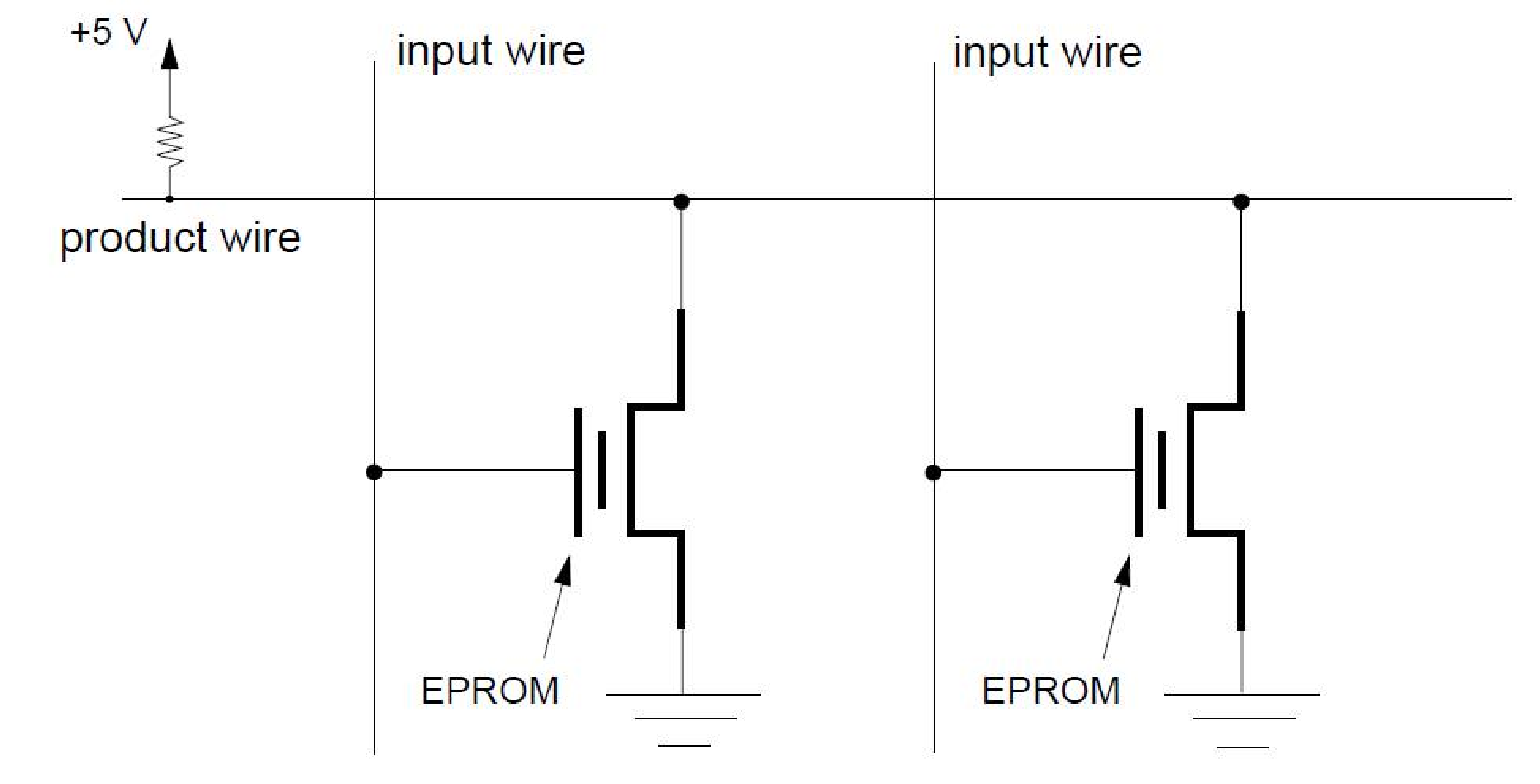


- SRAM-based FPGAs include most chips of Xilinx *Virtex and Spartan families and Altera Stratix* and *Cyclone*.

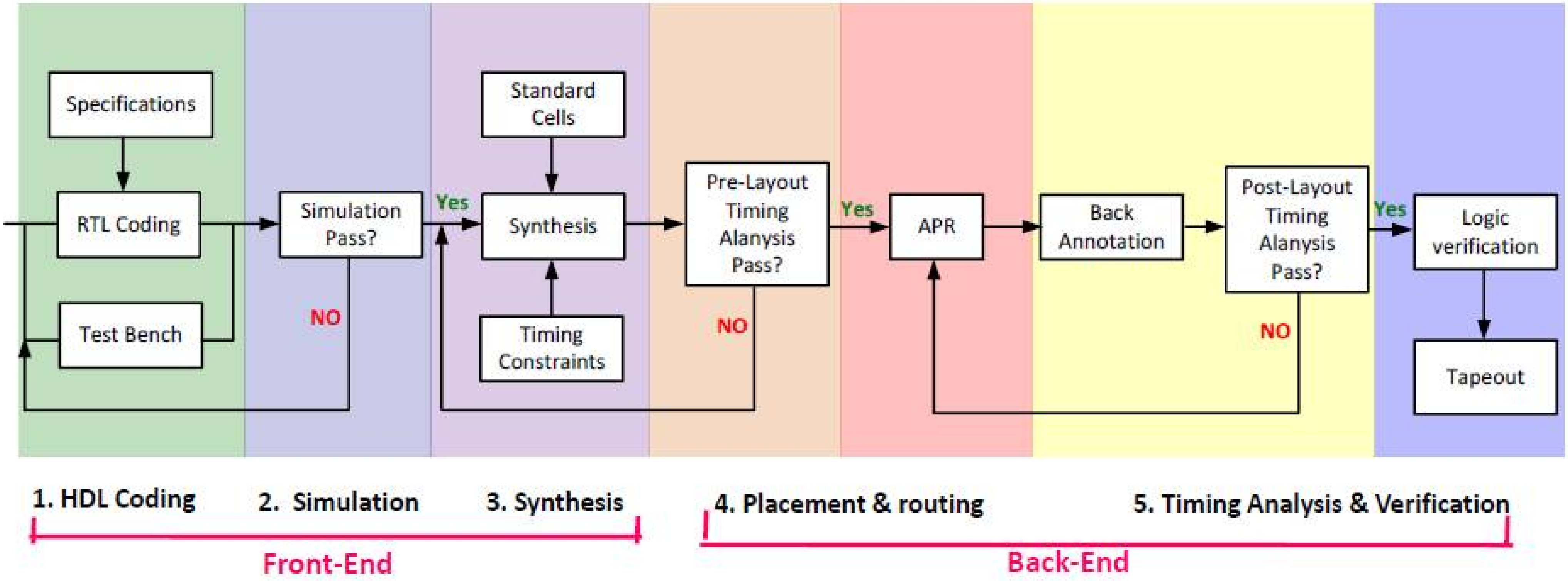
Antifuse Interconnects

- One time programmable.
- An antifuse is an electrical device that performs the opposite function to a fuse.
- One time FPGA programming

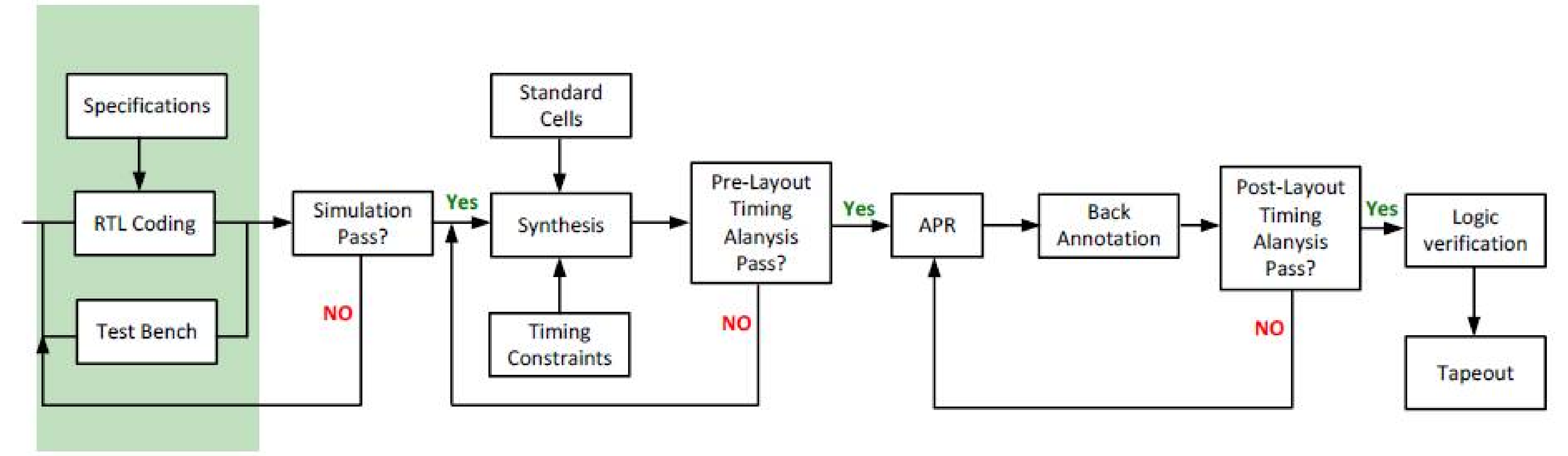
EEPROM Inteconnects



ASIC/FPGA design flowchart

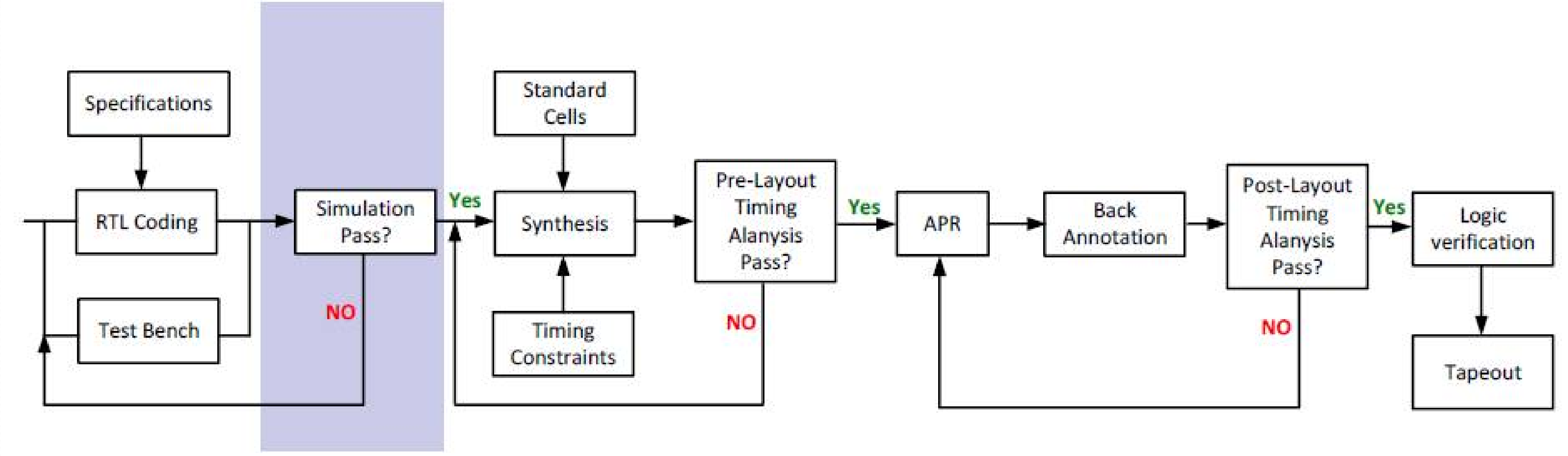


HDL coding



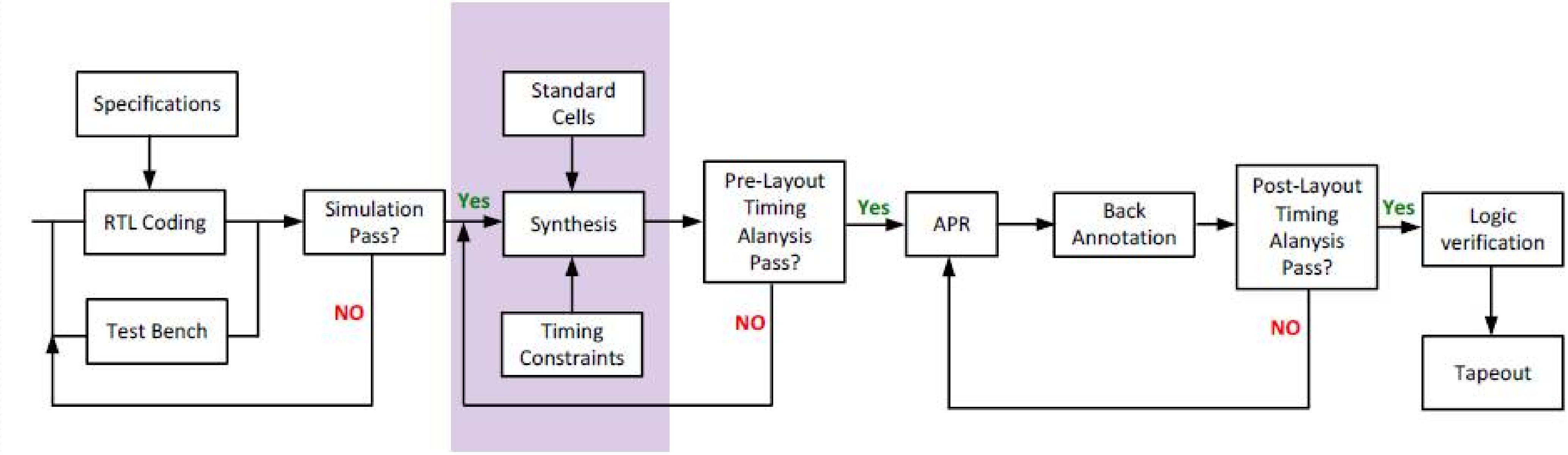
- HDL coding
- Test bench coding

VHDL Simulation



- VHDL Simulator
  - Active HDL(Aldec)
  - ISE simulator(Xilinx) یاISim
  - Modelsim(Mentor Graphics)
  - Quartus II Simulator(Altera) یا Qsim
  - VCS(Synopsys)

Synthesis



- Code analysis and converting to logic gates