CDA 4253 FPGA System Design
FPGA Architectures

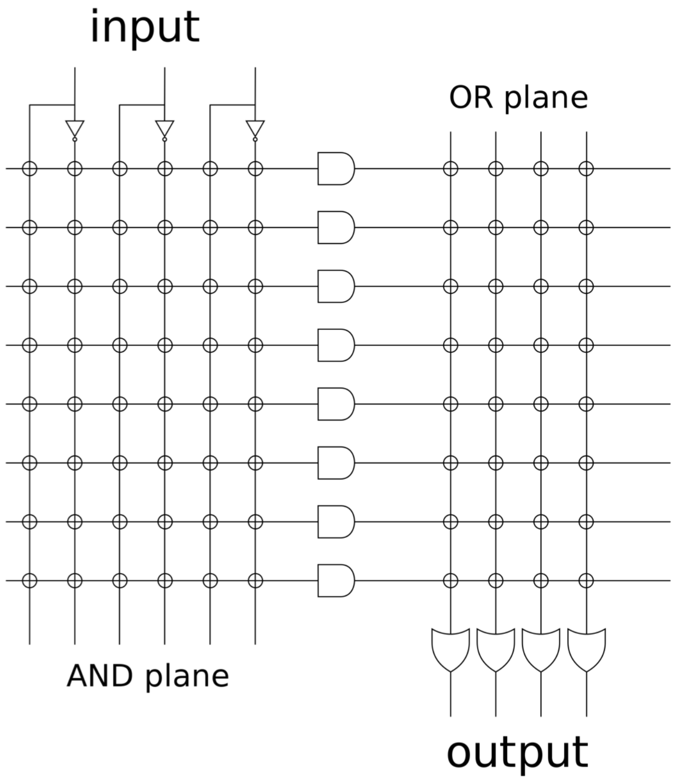
Hao Zheng
Dept of Comp Sci & Eng
U of South Florida

How to HW Reconfigurable

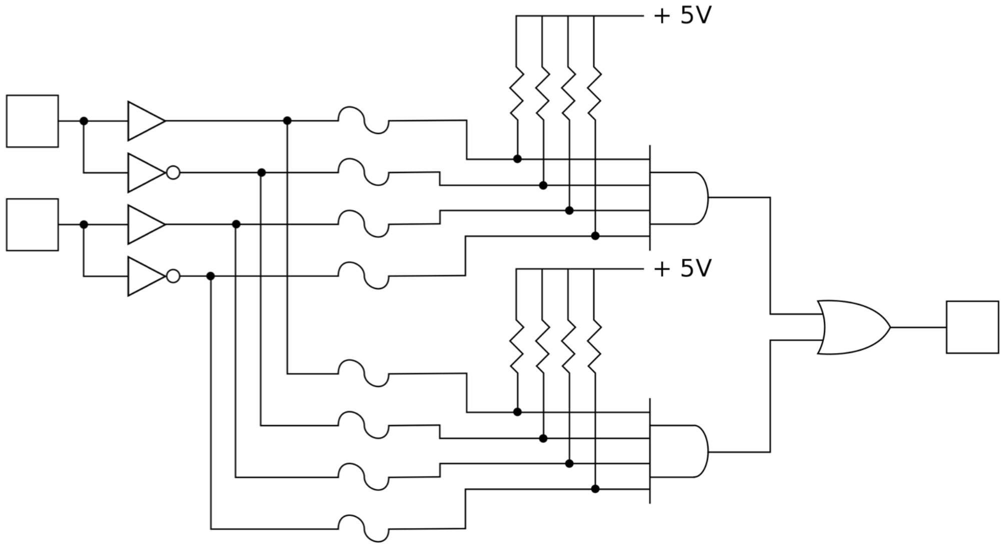
- Not SW
- Change structure
  - Change connections among components
  - Change logic functions of components

History – Simple Programmable Logic

PLA



PAL

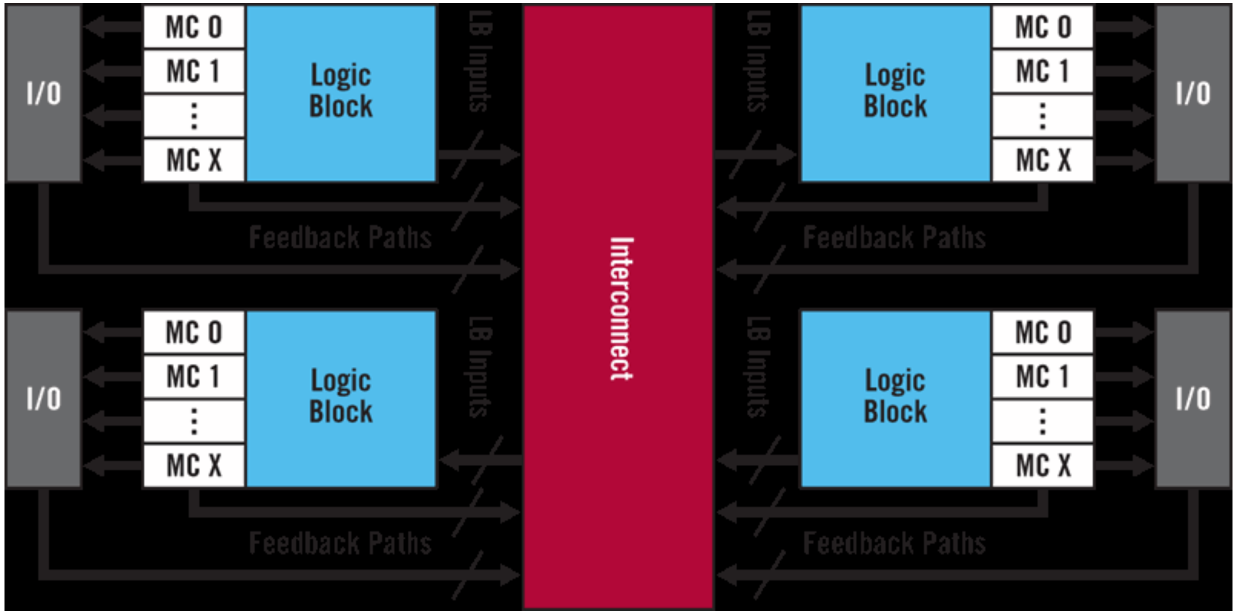


Simplified programmable logic device

Source: Wikipedia

History – Complex Programmable Logic

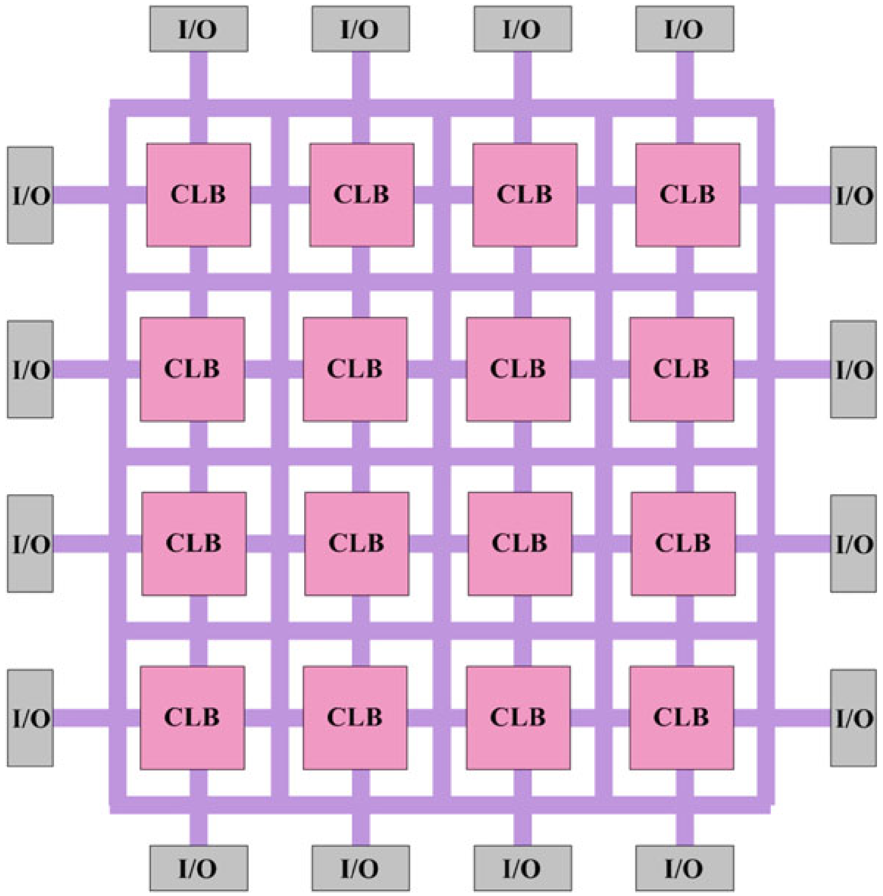
- Built on top of SPL
- Suitable for small scale applications
- Coarse-grained programmability



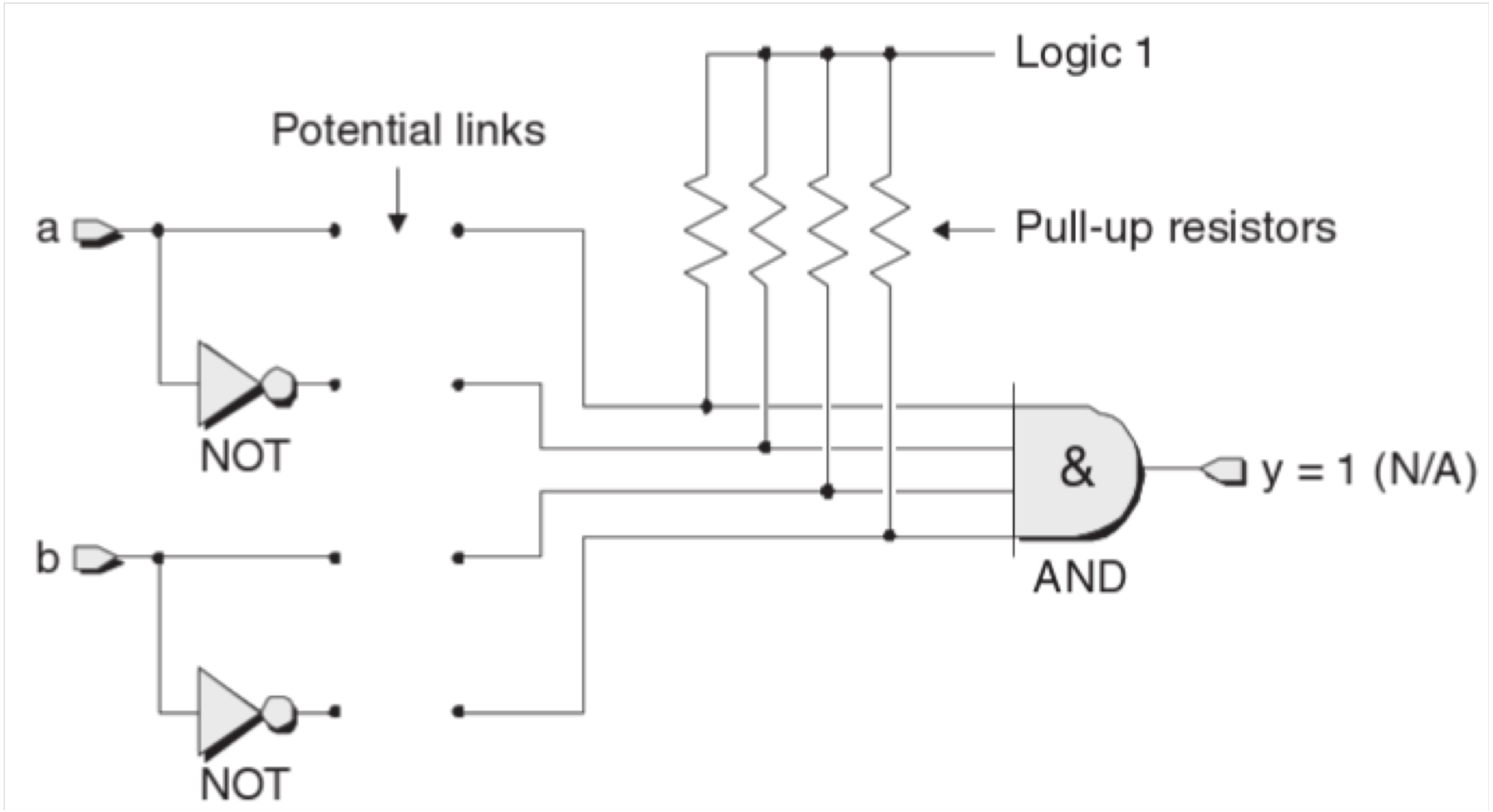
FPGAs – Generic Architecture

Also include common fixed logic blocks for higher performance:

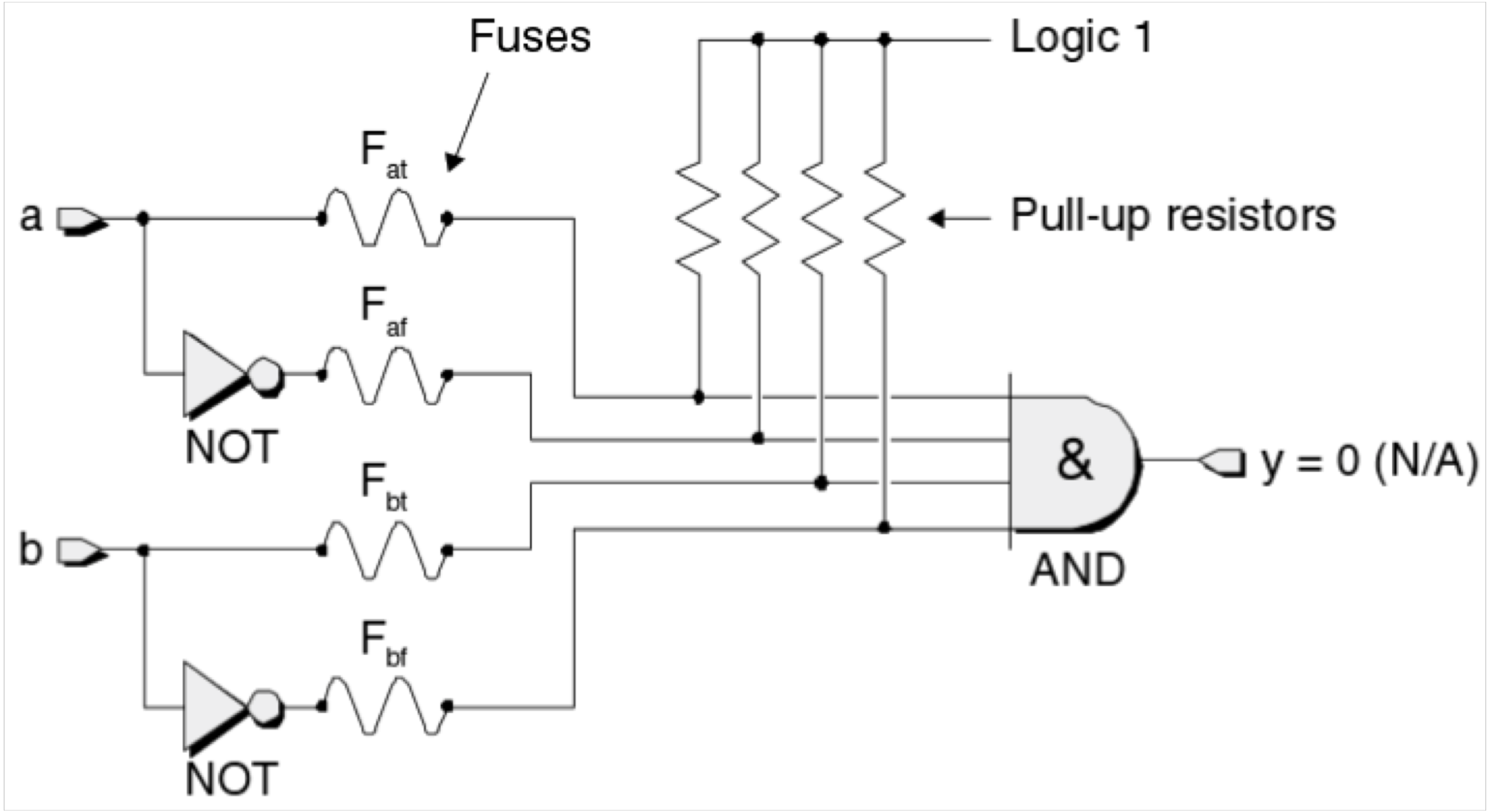
- On-chip mem.
- DSP/Multiplier
- Fast arithmetic logic
- Microprocessors
- Communication logic



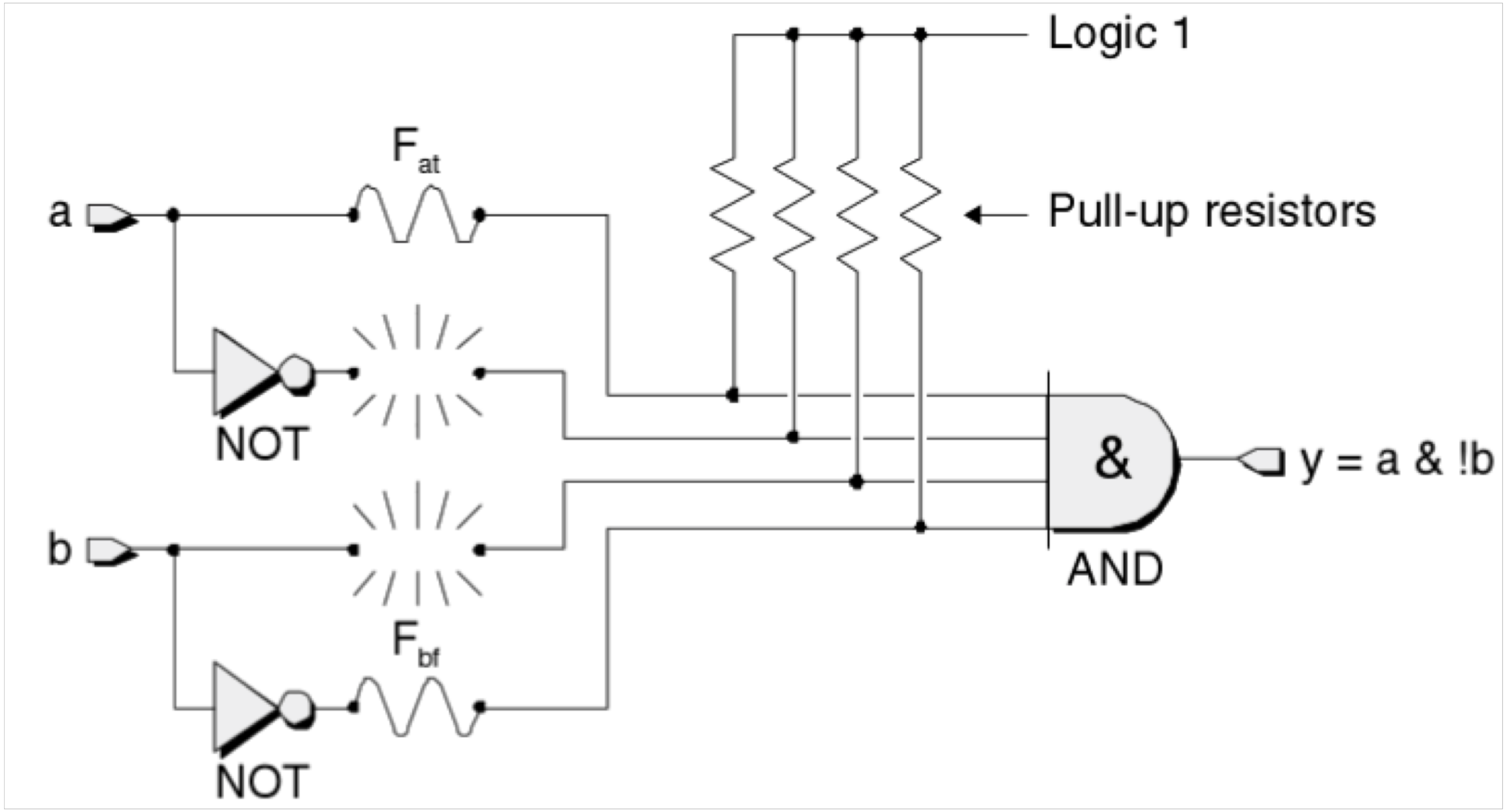
Programming Technologies



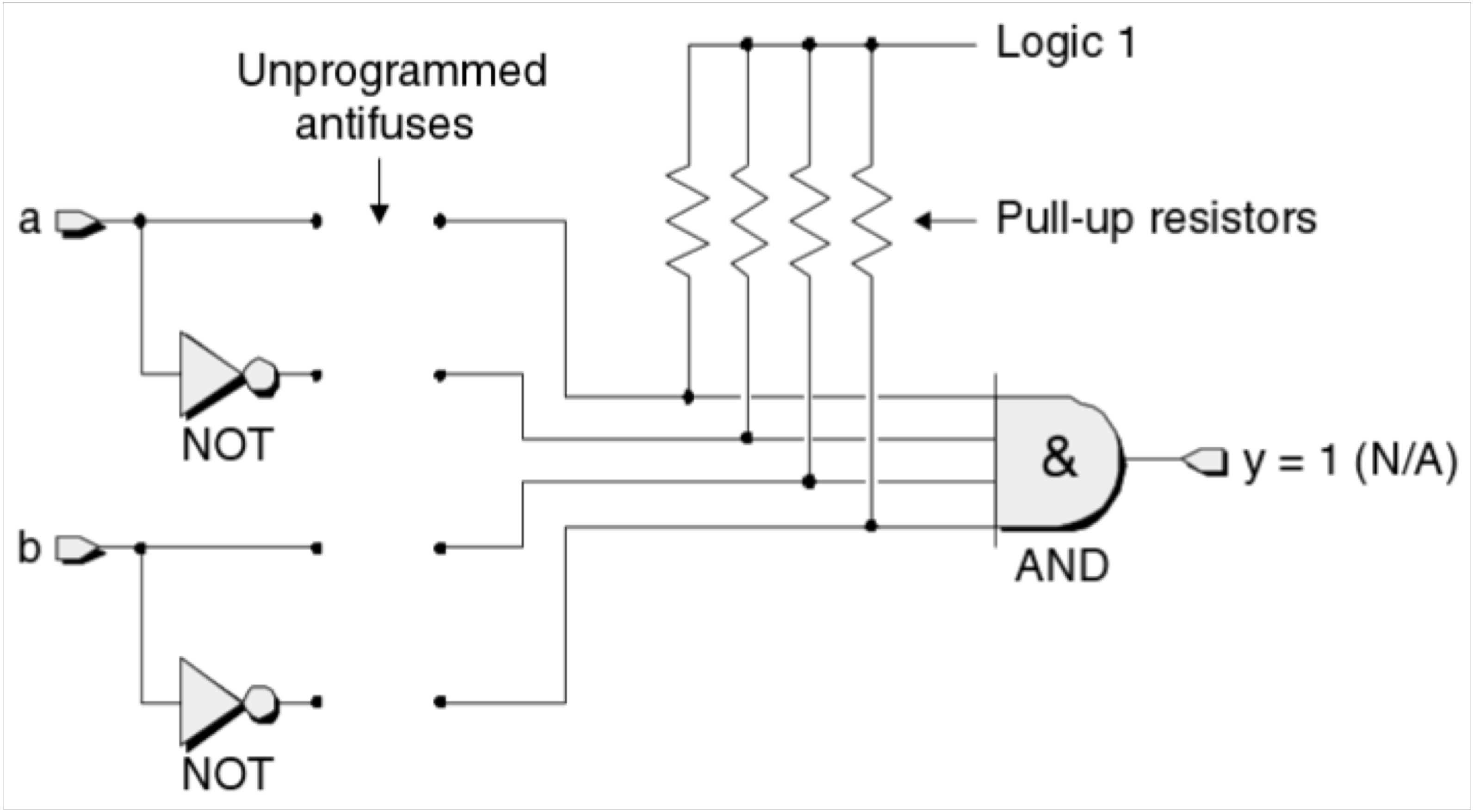
Programming Technologies: Fuses



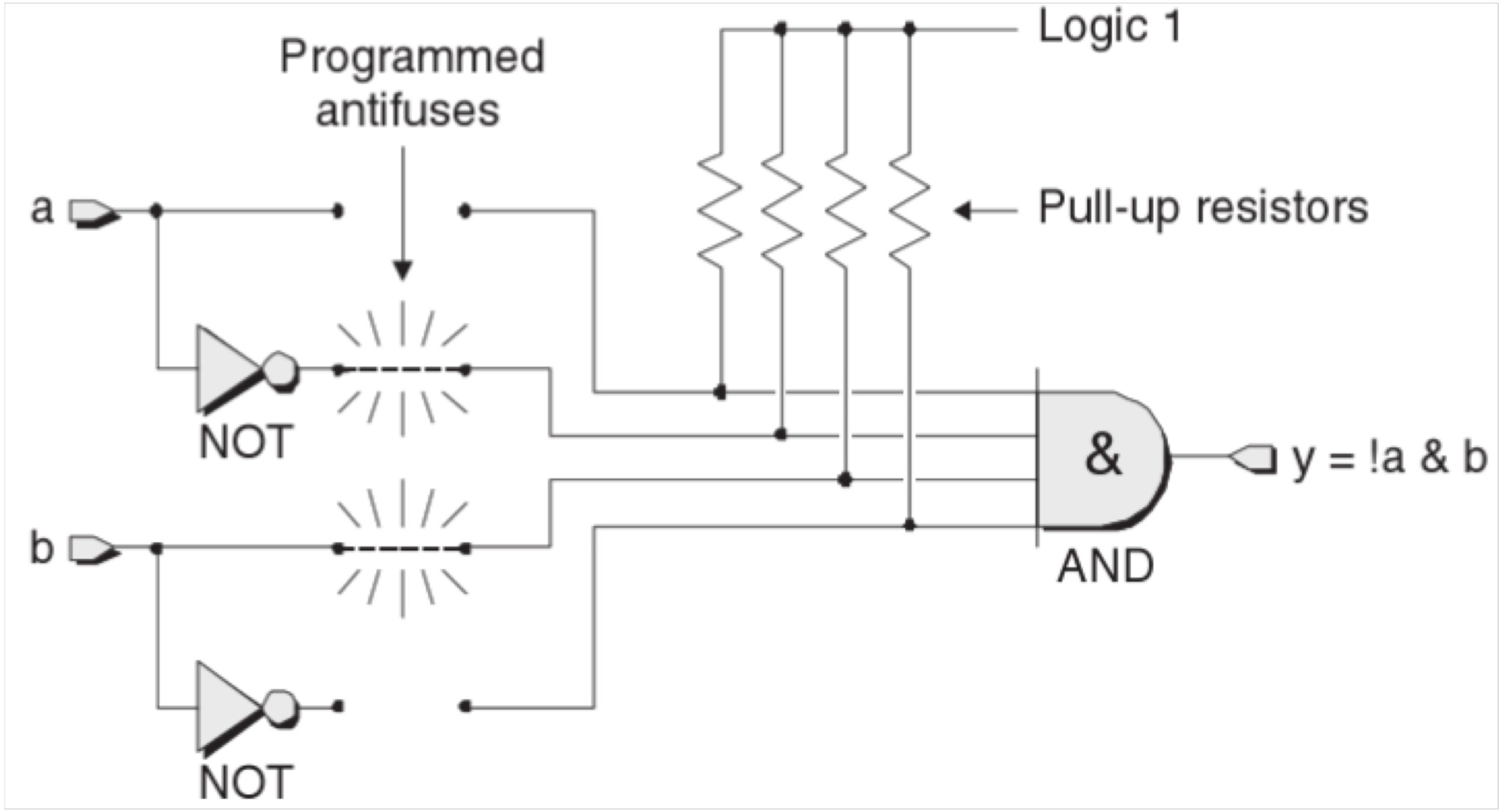
Programming Technologies: Fuses



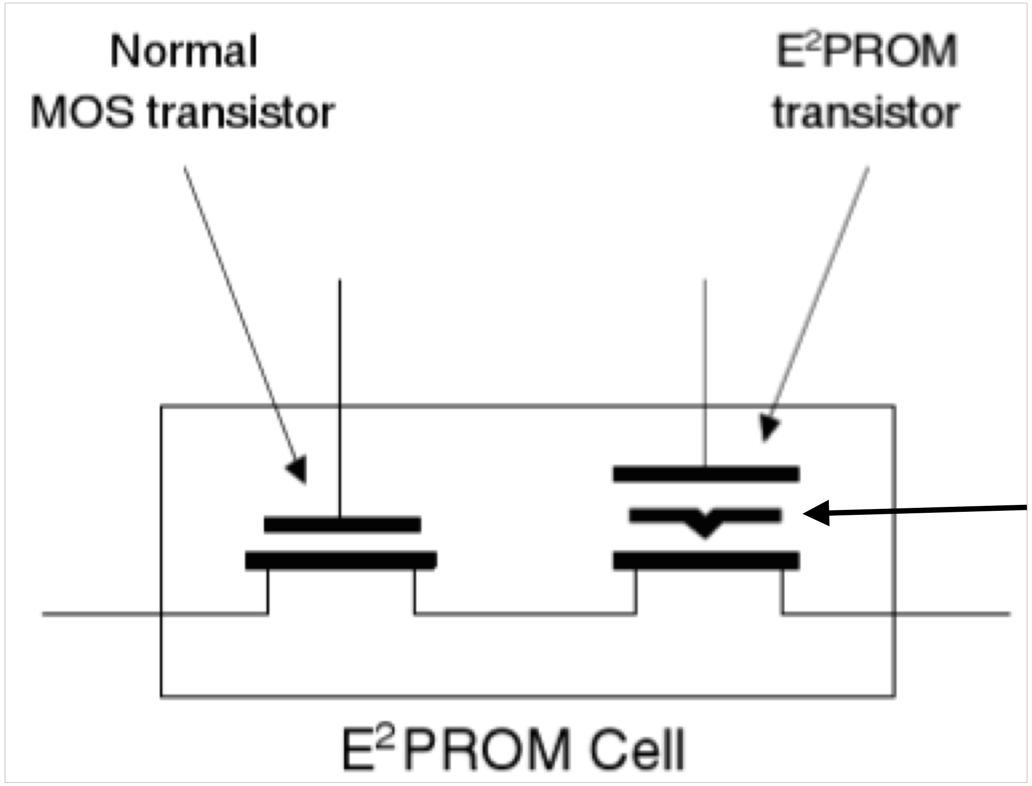
Programming Technologies: Anti-fuses



Programming Technologies: Anti-fuses

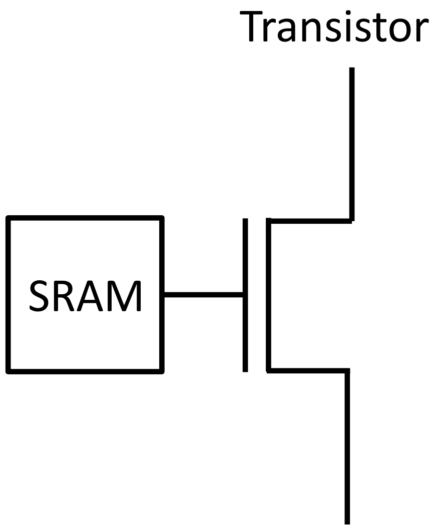


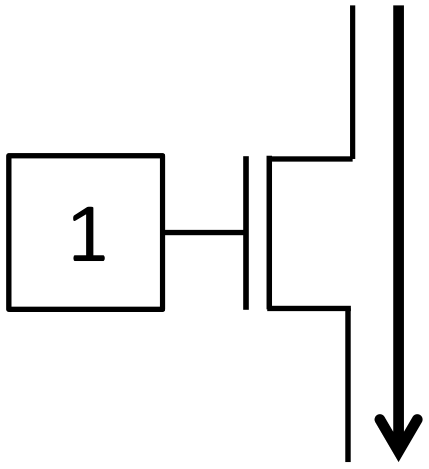
Programming Technologies: FLASH

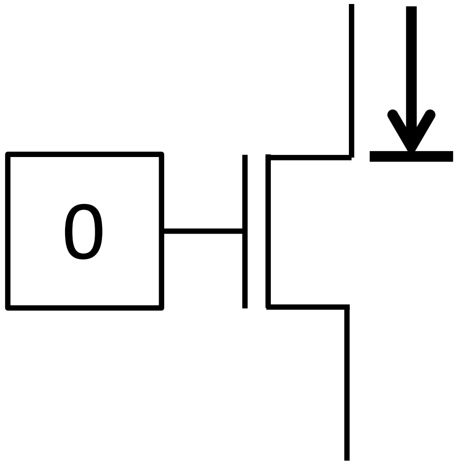


floating gate

Programming Technologies: SRAM



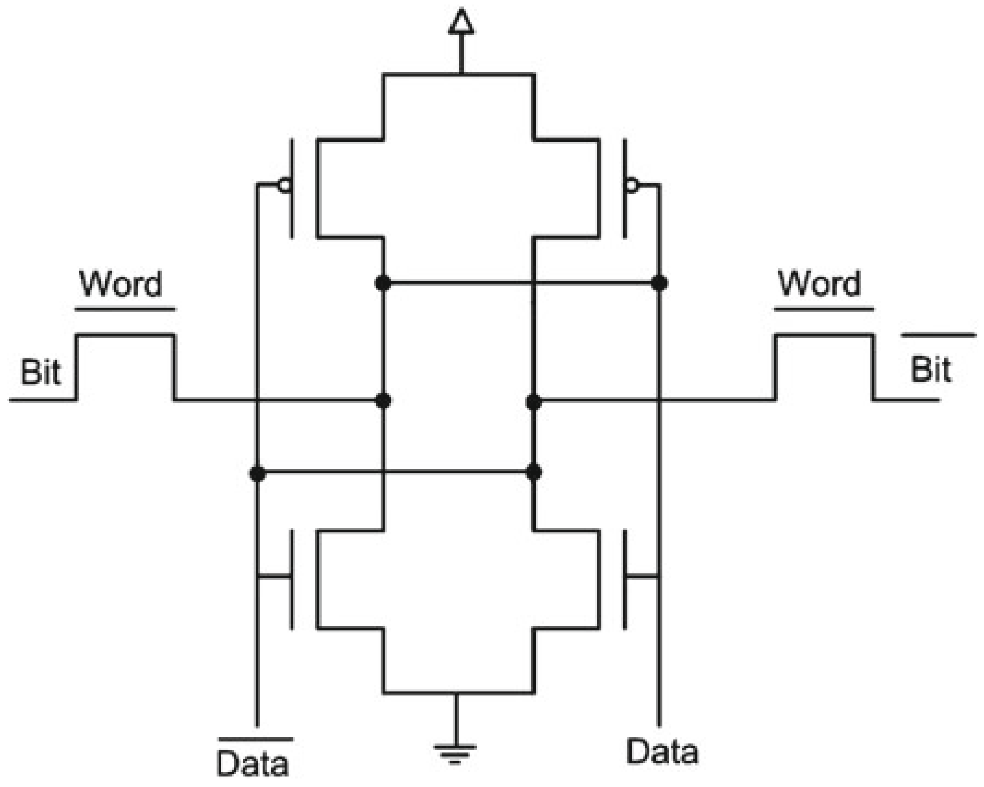


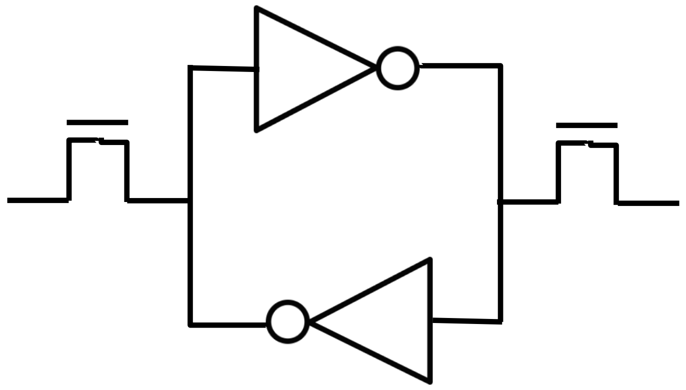


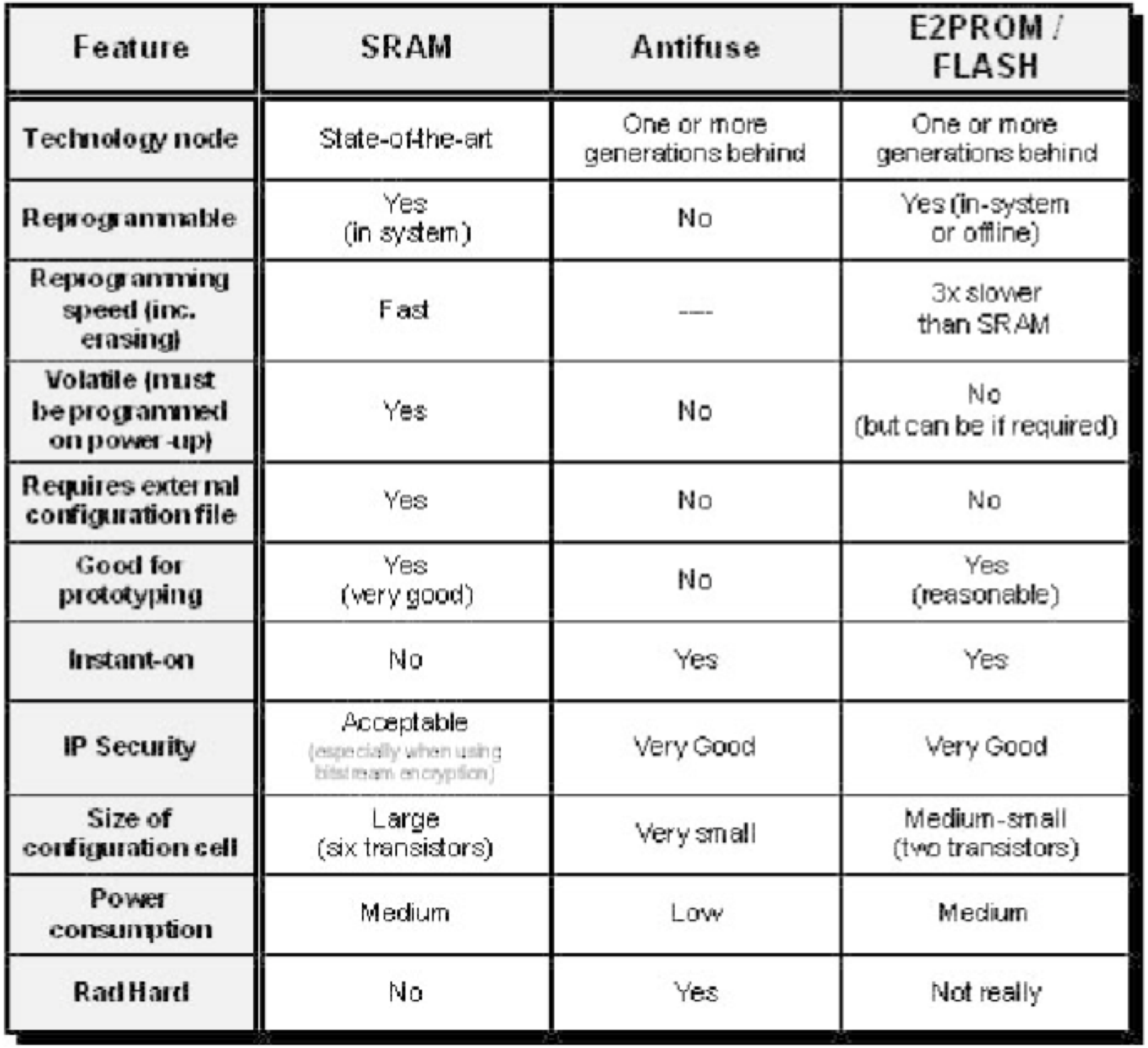
Open

Closed

Static RAM Cell



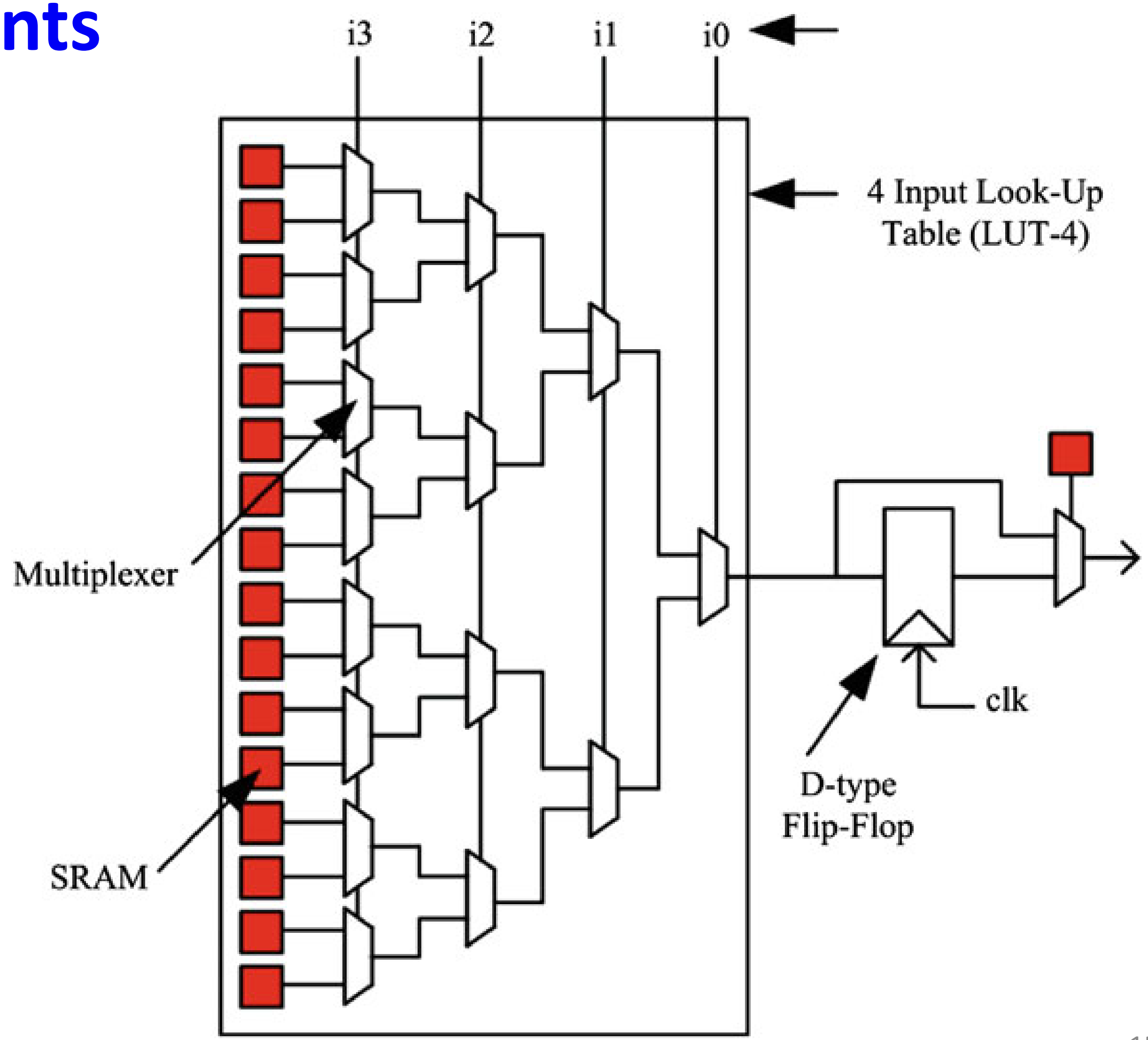




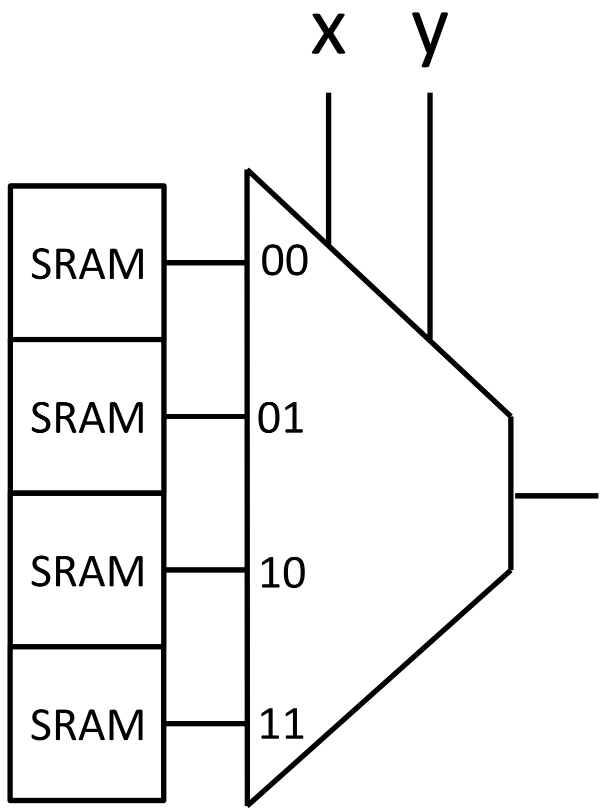
Feature	SRAM	Antifuse	E2PROM / FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	----	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very Good	Very Good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium
Rad Hard	No	Yes	Not really

Basic Logic Elements (BLEs)

Basic component that can be programmed to logic functions and provide storage.



Lookup Tables (LUTs)



<u>Commercial FPGAs</u>

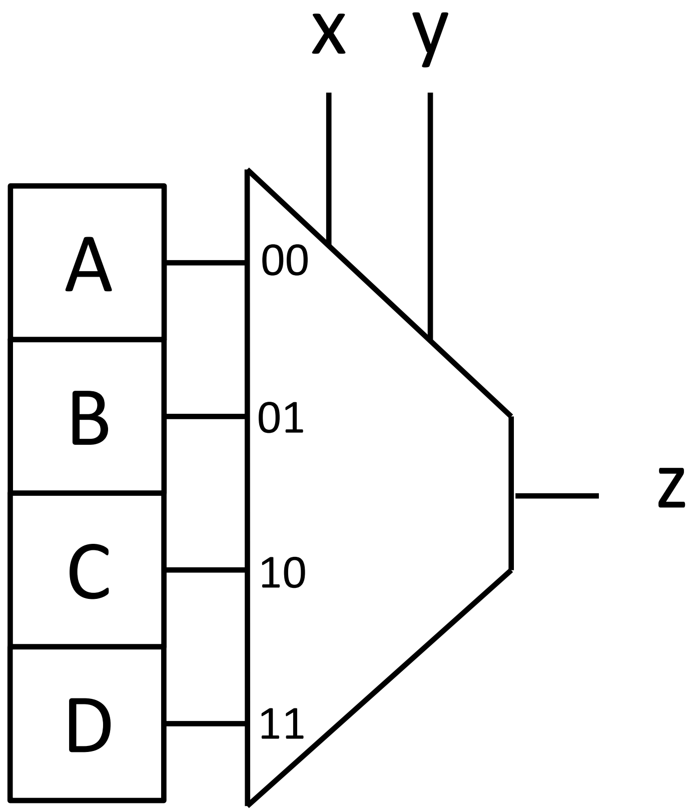
- Xilinx: 6-LUT
- Altera: 6-LUT
- Microsemi: 4-LUT

For x-input LUT, it can be programmed into one of

$$2^{2^{x}}$$

functions.

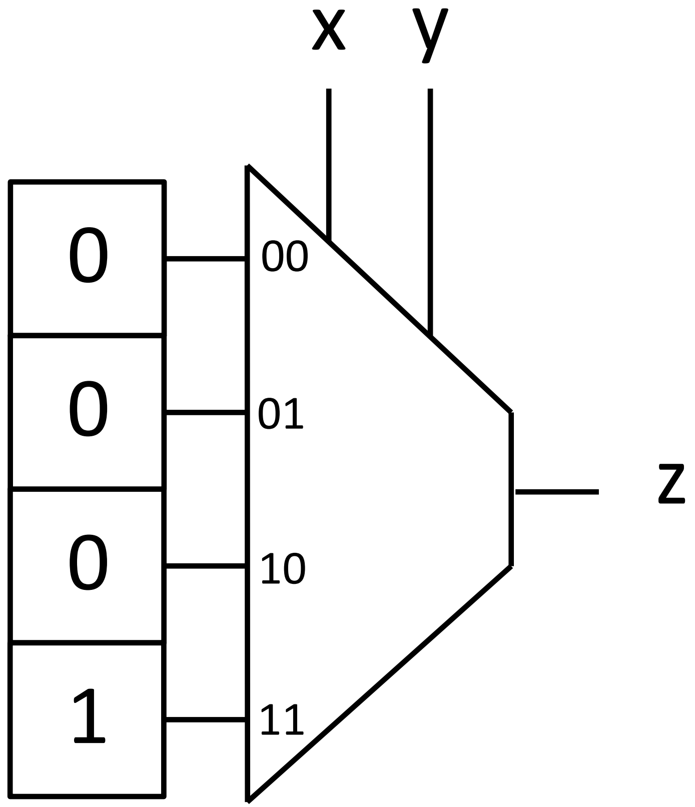
LUT = Programmable Truth Table



x	y	z
0	0	A
0	1	B
1	0	C
1	1	D

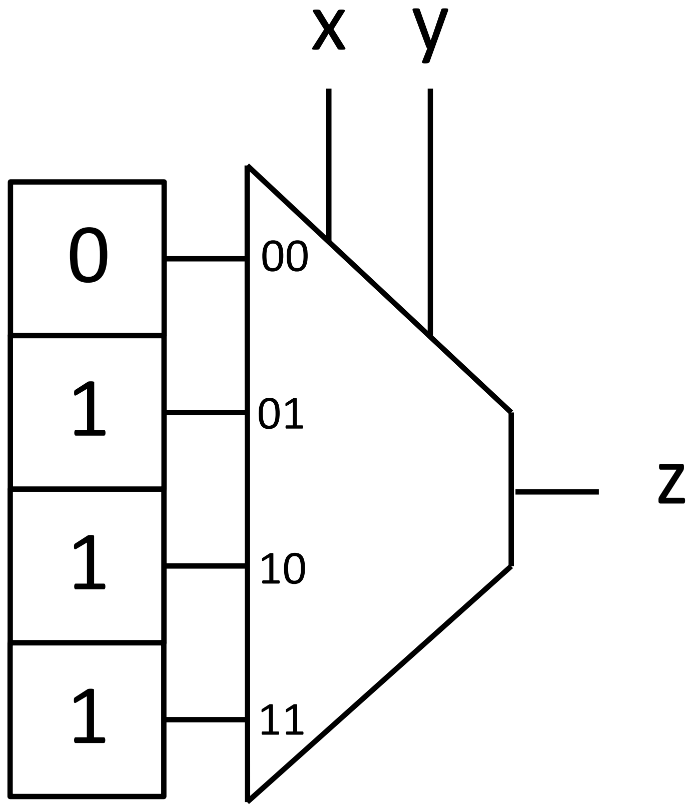
Also called function generator.

AND



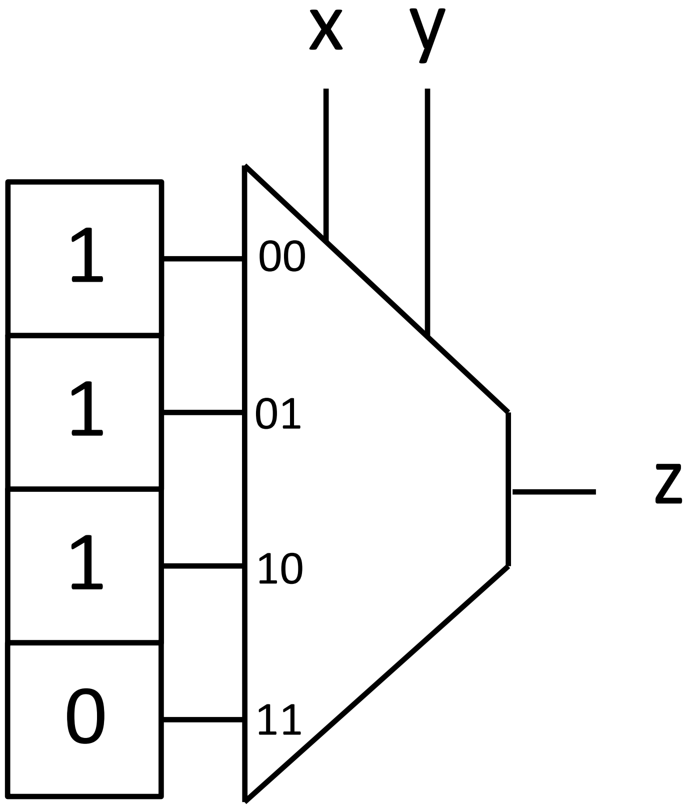
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

OR



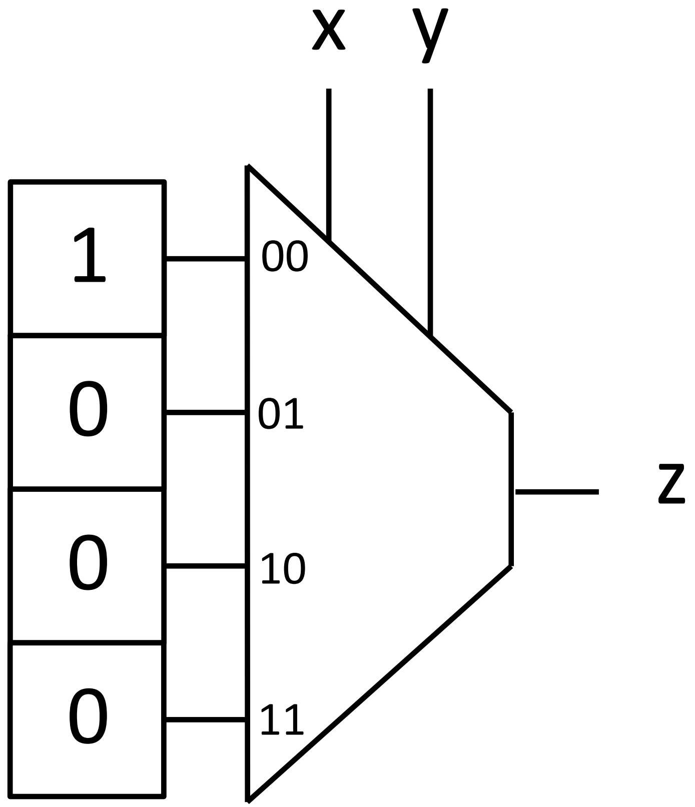
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

NAND



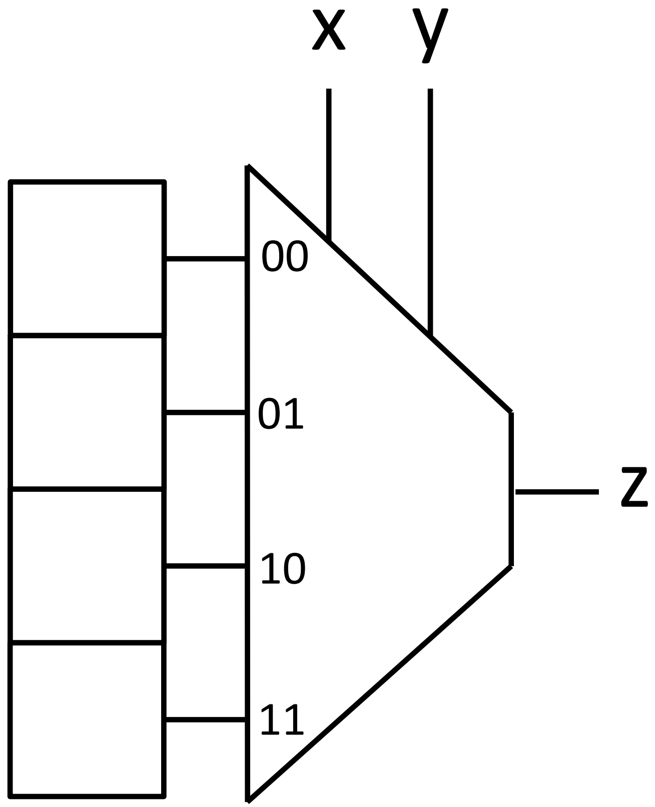
x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

NOR

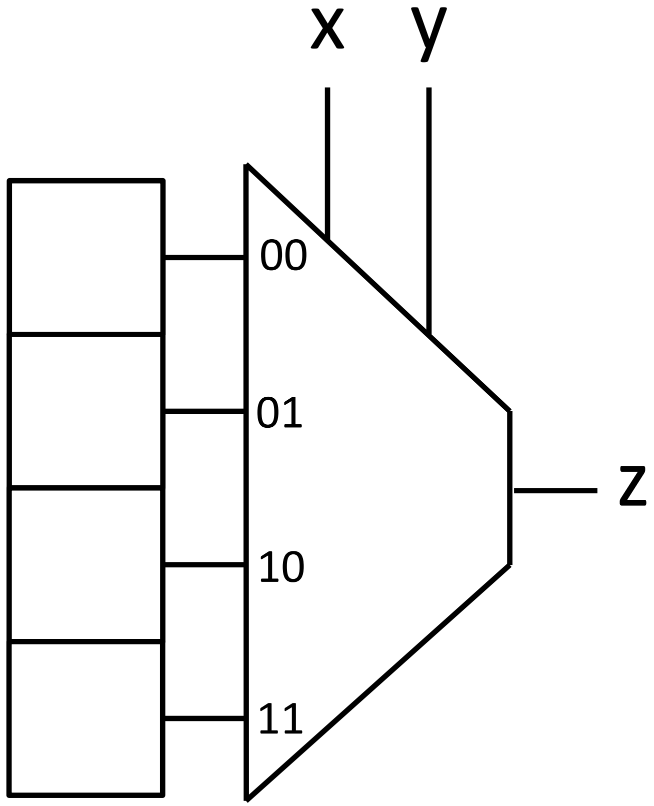


x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

XOR

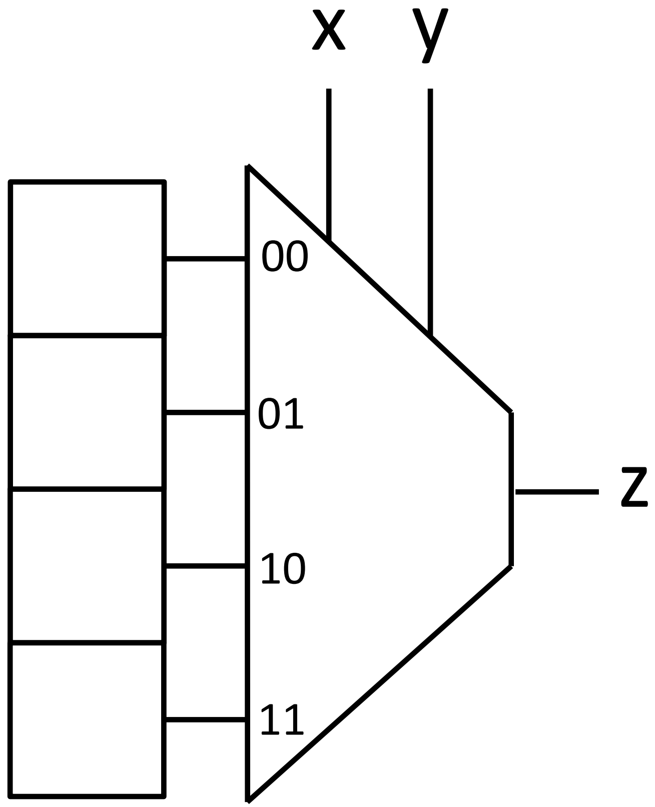


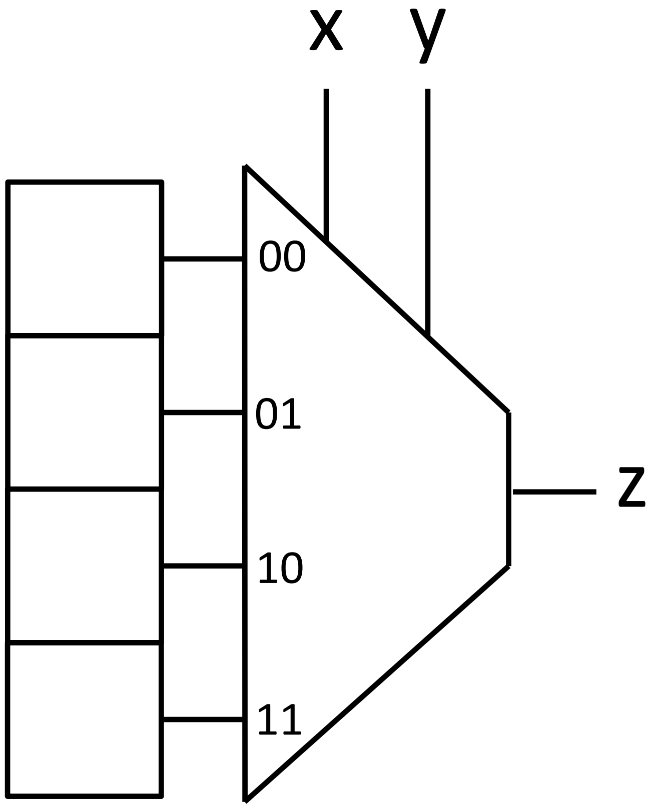
XNOR



$z = \overline{y}$

$z = \overline{y} + x$





Features of LUTs

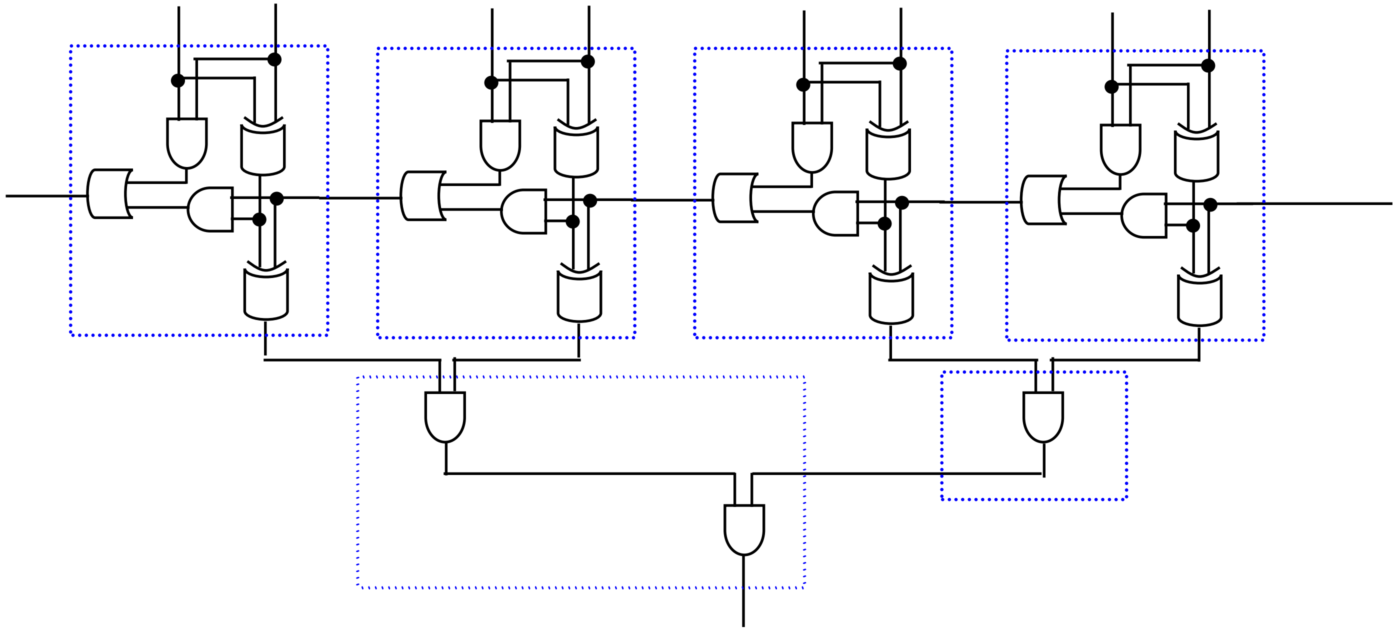
- A LUT is a piece of RAM.
  - Can be configured as distributed RAM in Xilinx.
  - Can be configured as shift registers.
- A *n*-LUT can implement any *n*-input logic functions.
  - Logic minimization should reduce the number of inputs, not logical operators.
- All logic functions implemented by a *n*-LUT have the same propagation delay.

Look-up-tables (LUTs)

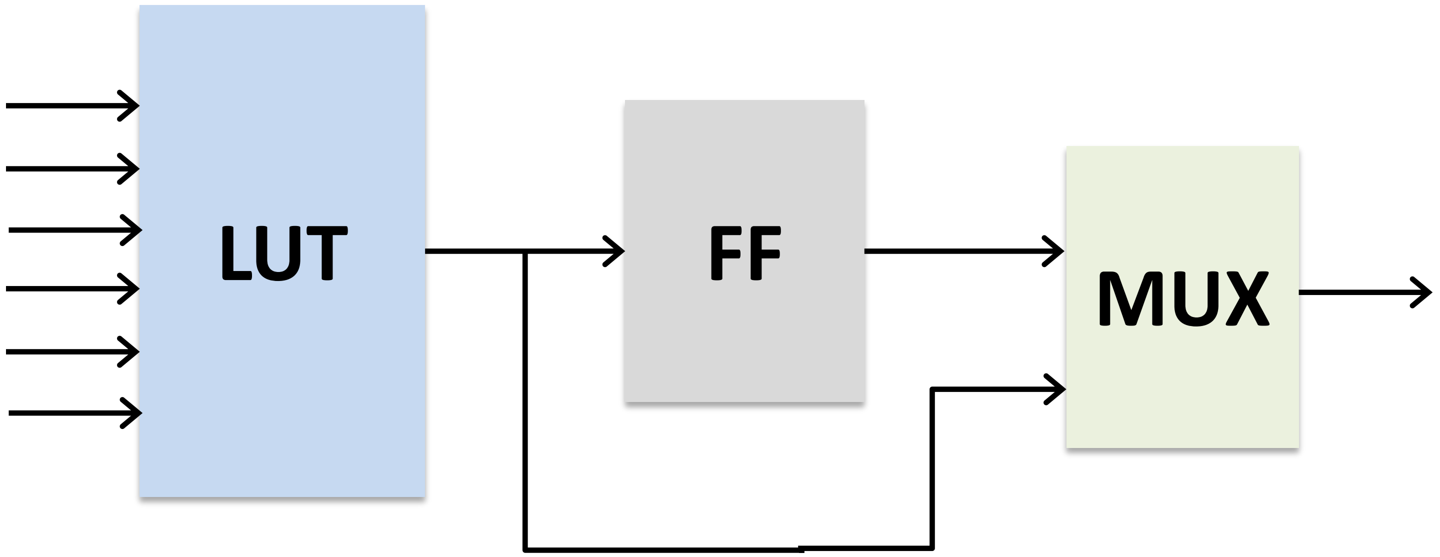
- Why aren't FPGAs just a big LUT?
  - Size of truth table grows exponentially based on # of inputs
    - 3 inputs = 8 rows, 4 inputs = 16 rows, 5 inputs = 32 rows, etc.
  - Same number of rows in truth table and LUT
  - LUTs grow exponentially based on # of inputs
- Number of SRAM bits in a LUT = $2^i * o$
  - i = # of inputs, o = # of outputs
  - Example: 64 input combinational logic with 1 output would require 2^{64} SRAM bits
    - 1.84×10^{19} SRAM bits required.
- Large LUT → long latency
- Clearly, not feasible to use large LUTs
  - So, how do FPGAs implement logic with many inputs?

Look-up-tables (LUTs)

- Map circuits onto multiple LUTs
  - Divide circuit into smaller circuits that fit in LUTs (same # of inputs and outputs)
  - Example: 2-input LUTs



Sequential Logic

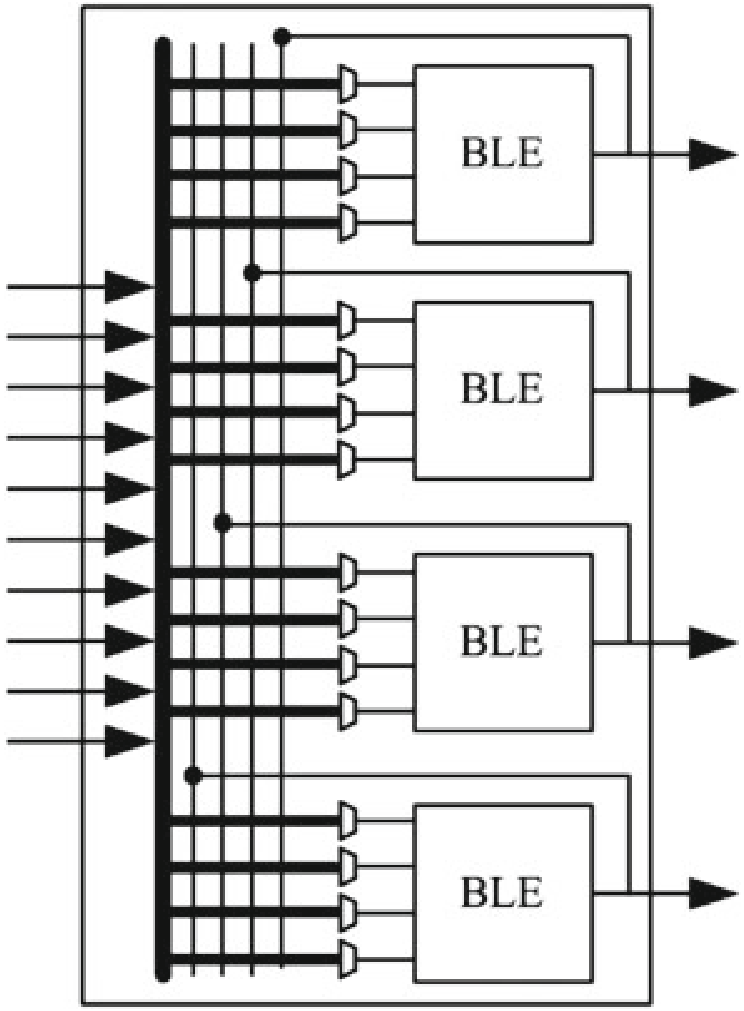


Configurable Logic Blocks

Number of BLEs are grouped with a local network in order to implement functions with a large number of inputs and multiple outputs.

More efficient to implement logic functions with common I/O.

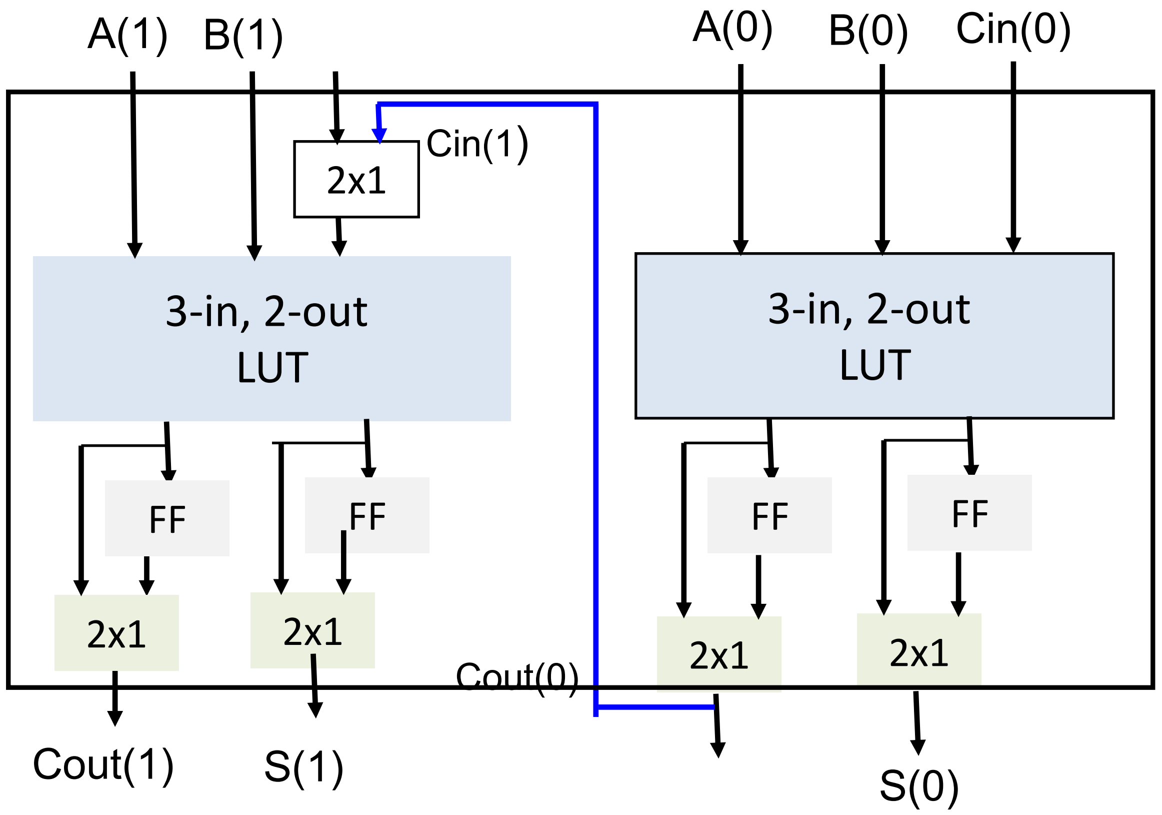
Save routing resources.



Configurable Logic Blocks (CLBs)

Example: Ripple-carry adder

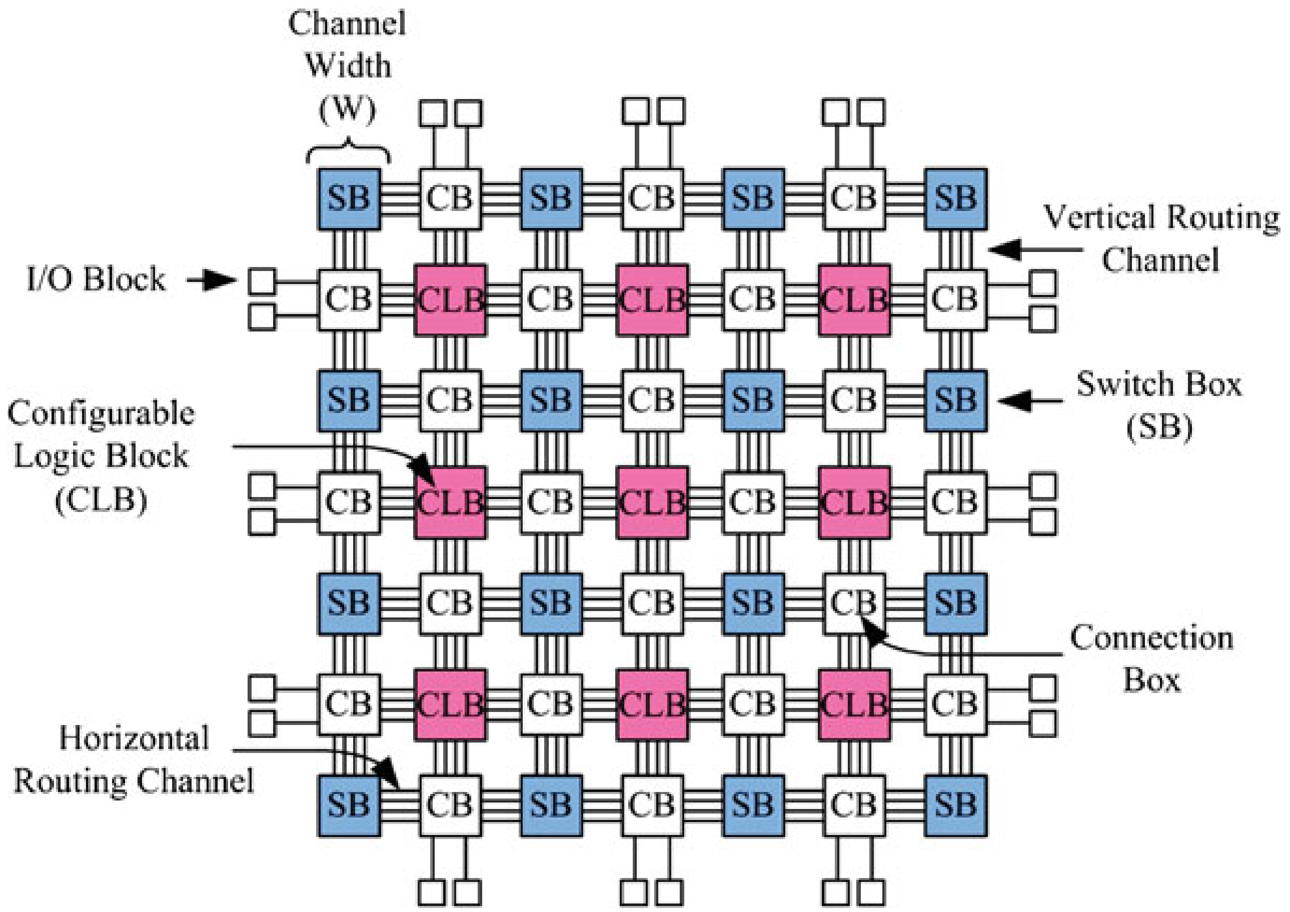
- Each LUT implements 1 full adder
- Use efficient connections between LUTs for carry signals



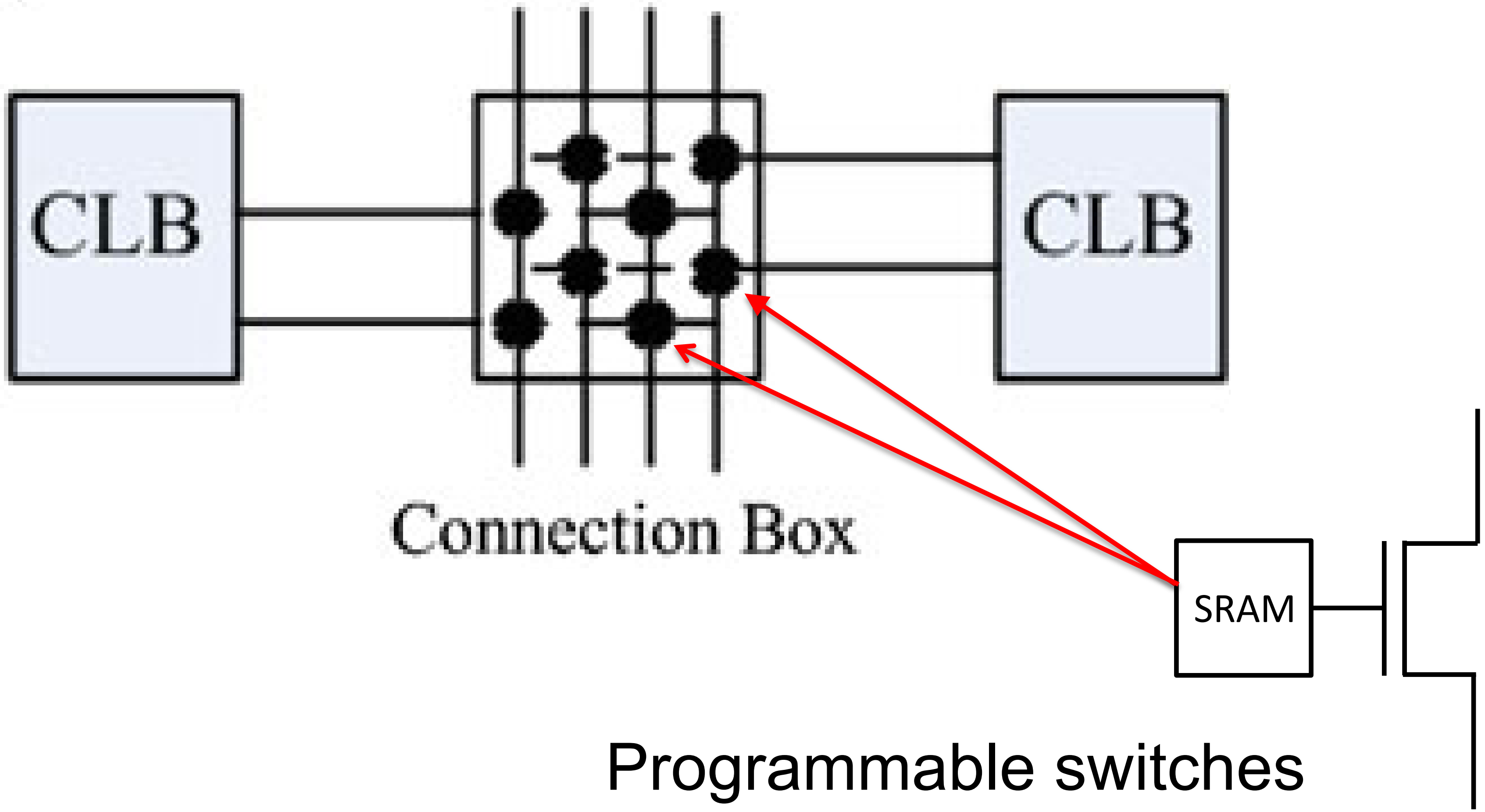
Programmable Interconnect

FPGA Routing Architectures

Must be flexible to accommodate various circuit implementations.



Connection Boxes

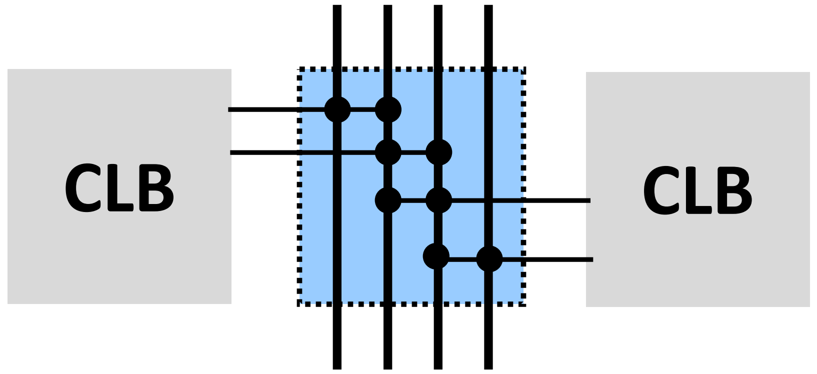


Programmable switches

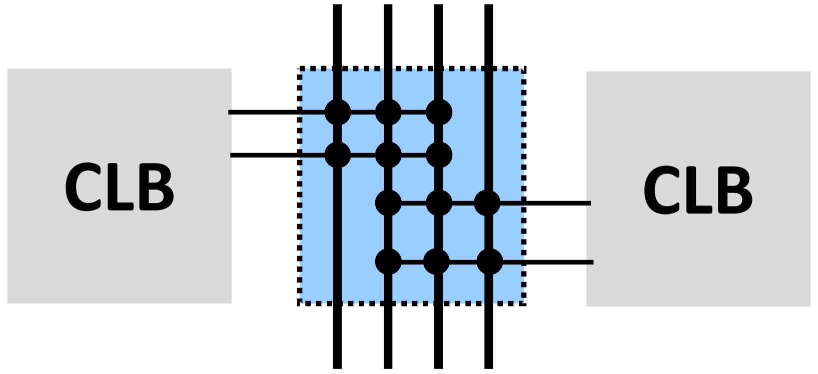
Connection Boxes

- Flexibility – the number of wires a CLB input/output can connect to

Flexibility = 2

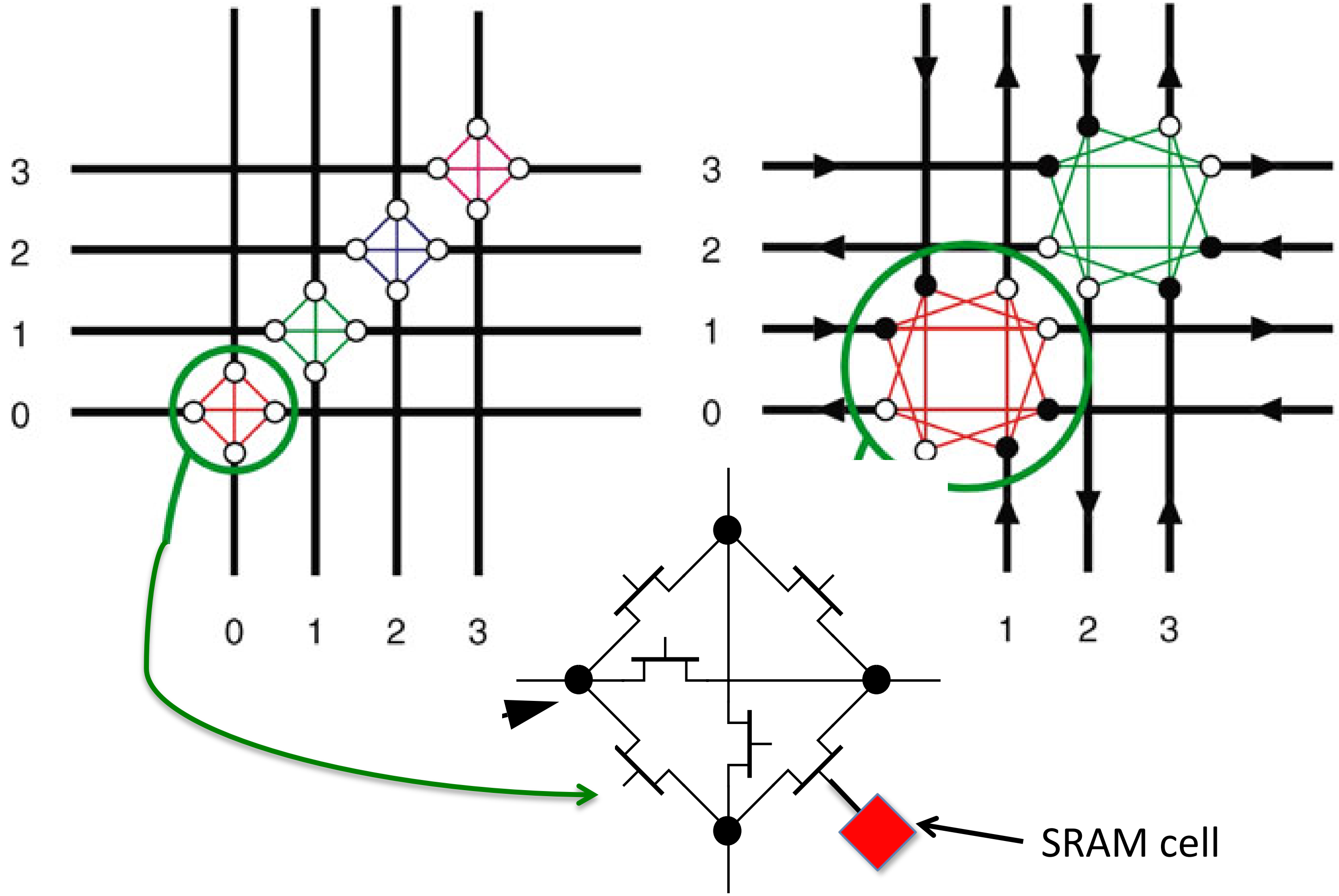


Flexibility = 3

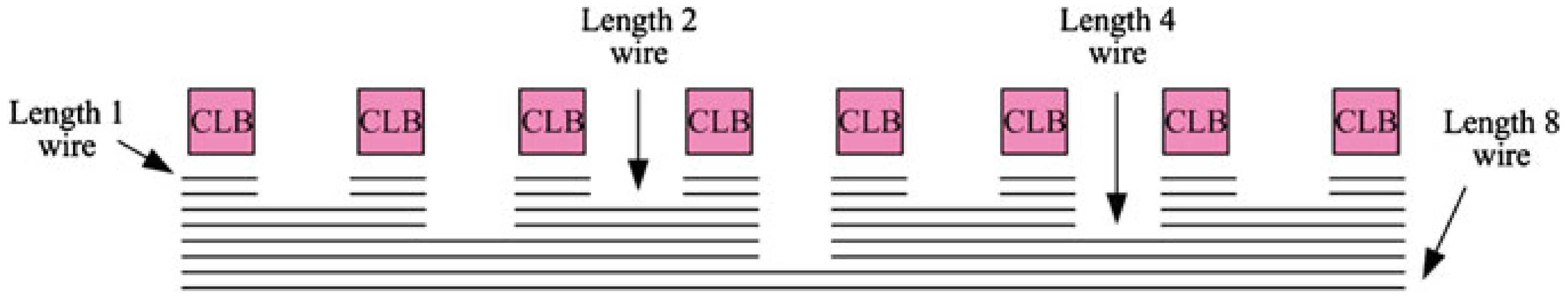


Dots represent* ***possible* *connections*

Switch Boxes

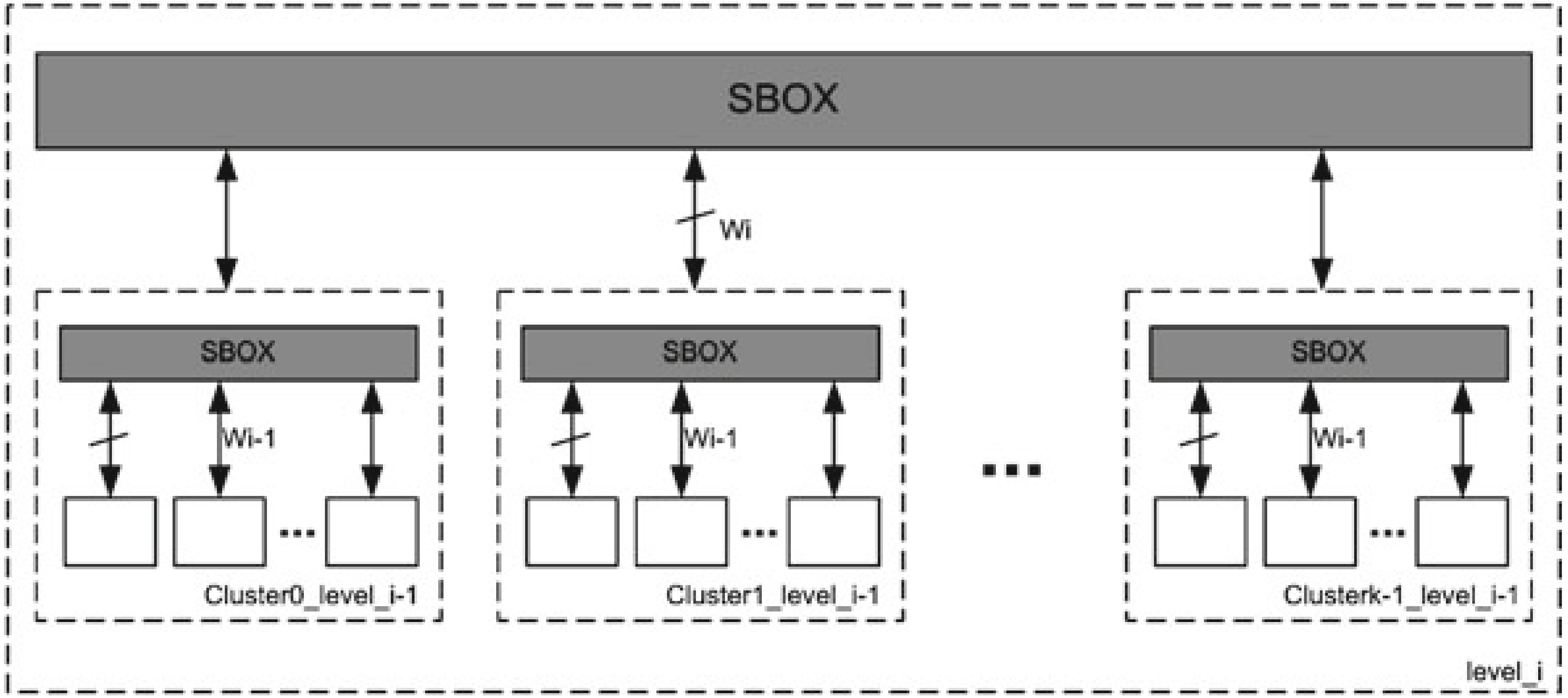


Segmented Routing



- Short wires: many, local connections.
- Long wires: few, low latency, carrying global signals
- Dedicated long wires for clock/reset signals
- Optimal routing should use minimal number of programmable connections

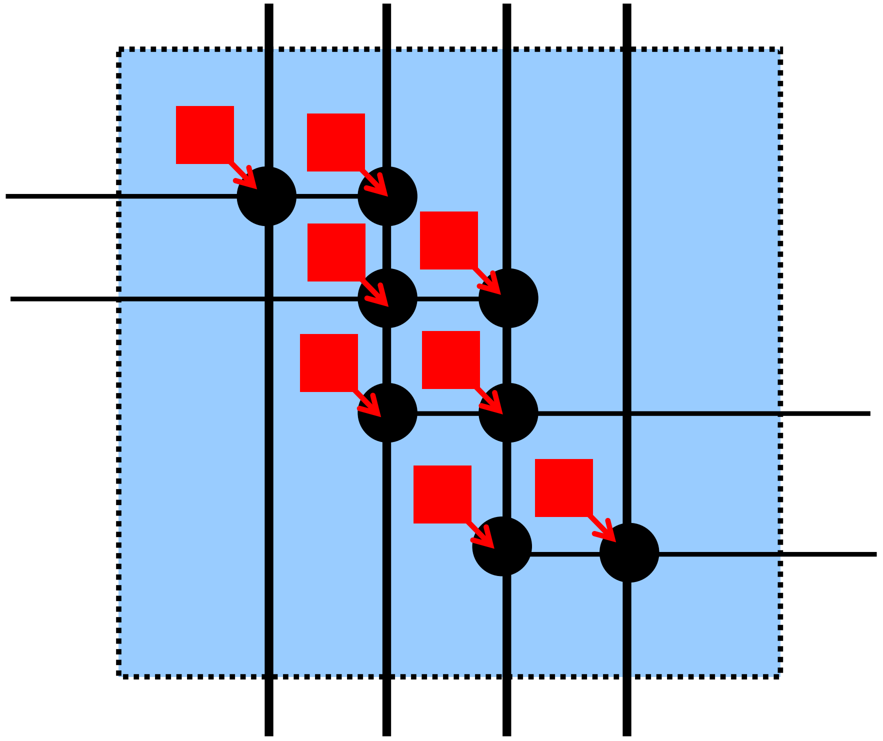
Hierarchical Routing Architecture

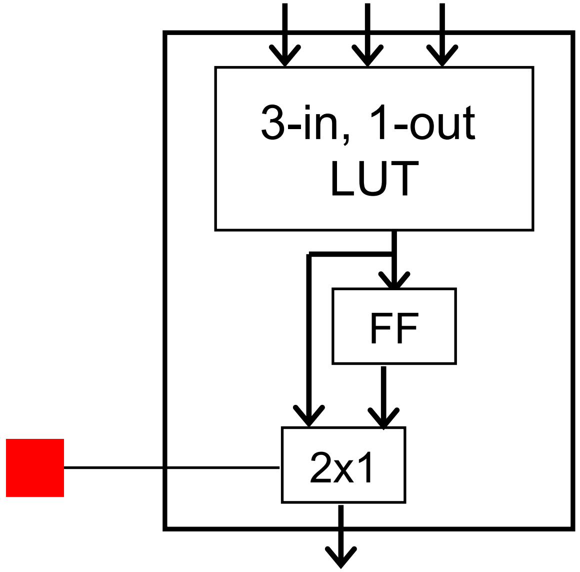


Most designs display locality of connections – hierarchical routing architecture.

Configuration

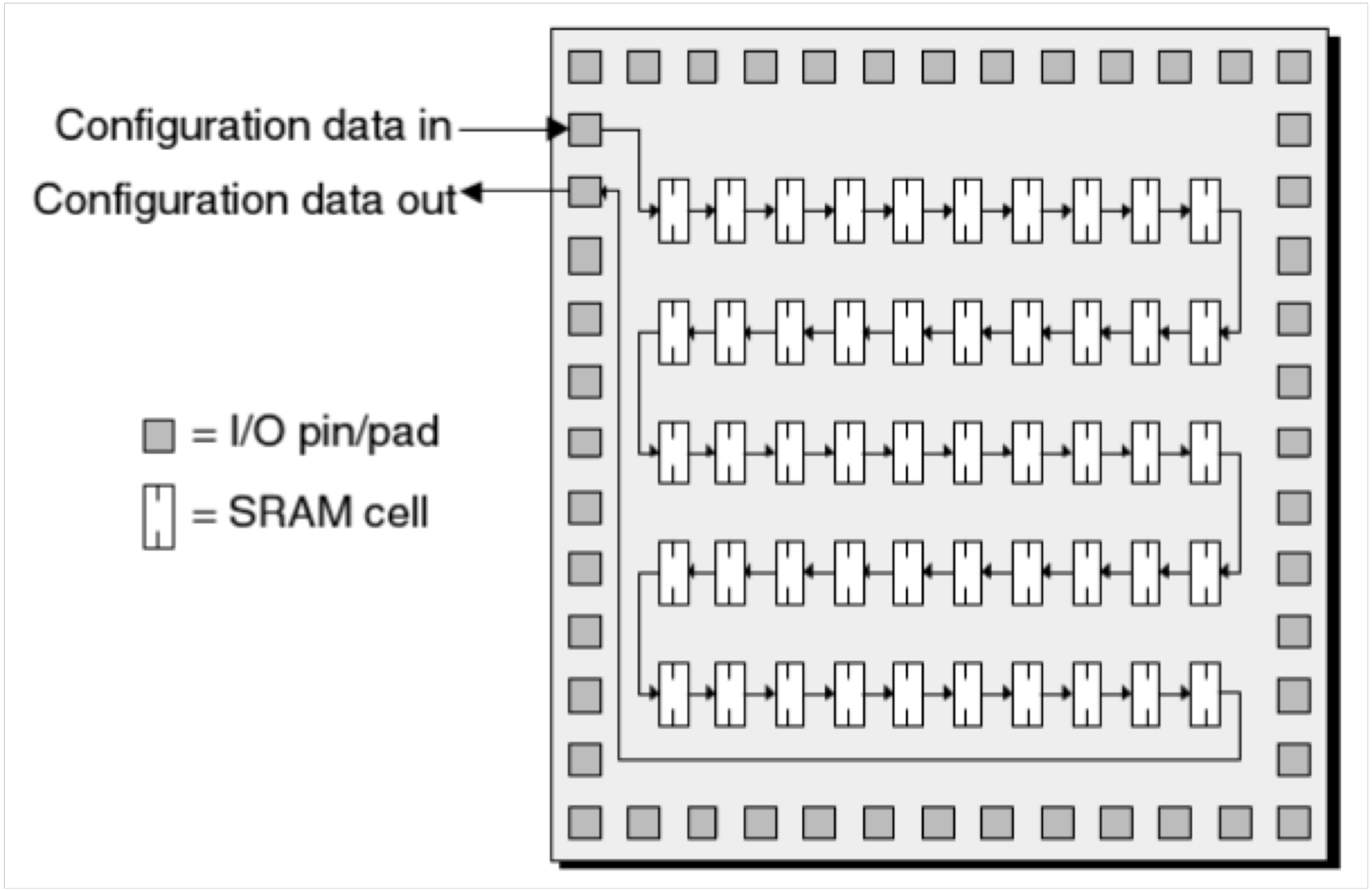
FPGA Configuration



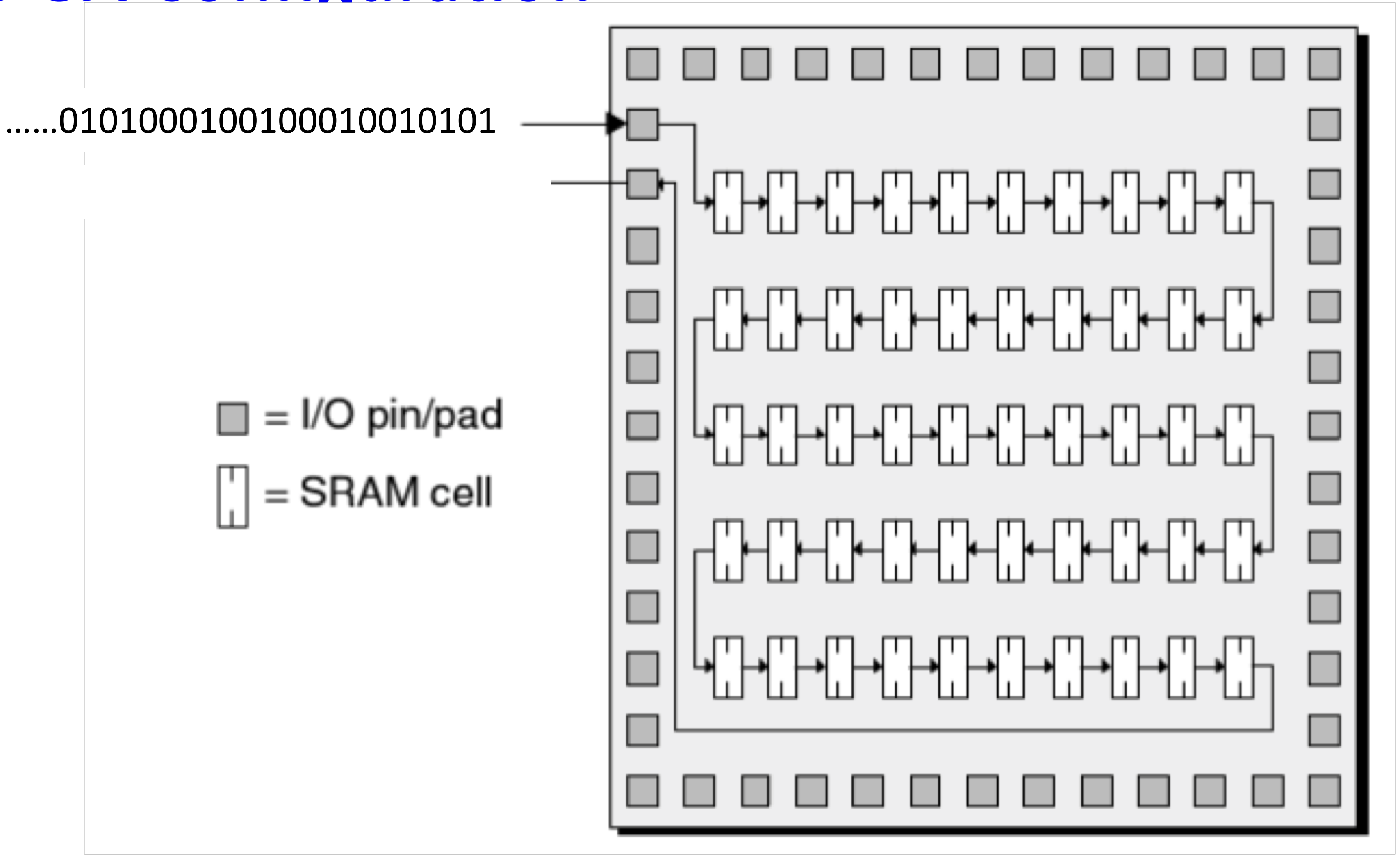


How to get a bitstream into FPGA?

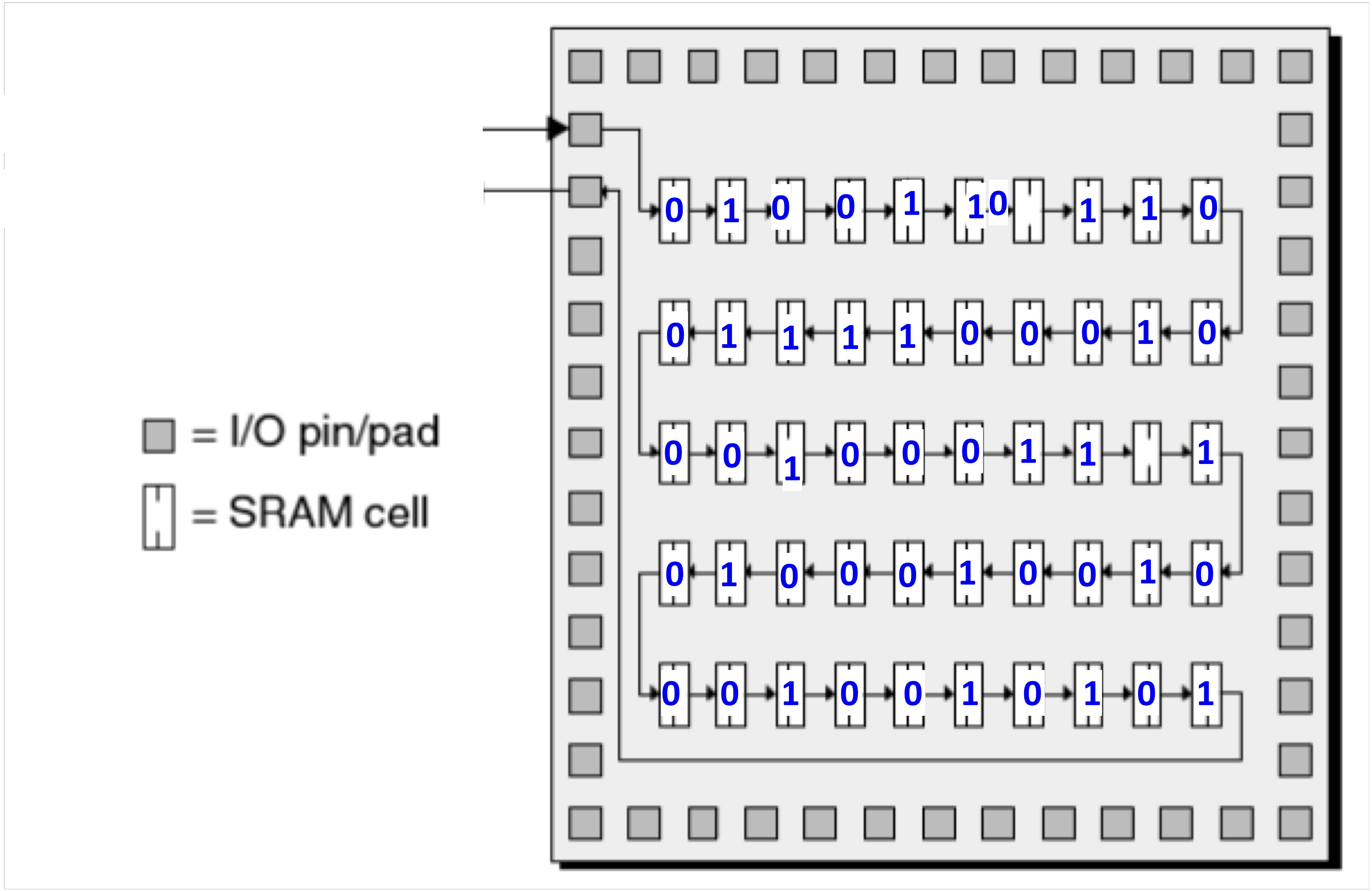
FPGA Configuration



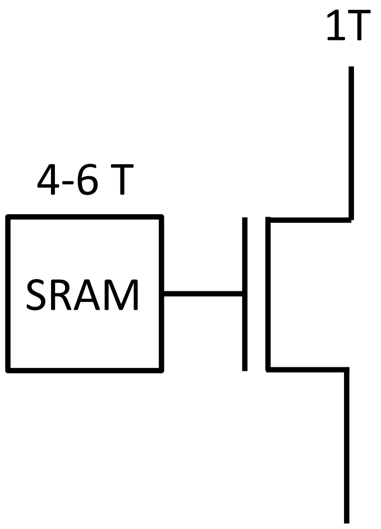
FPGA Configuration

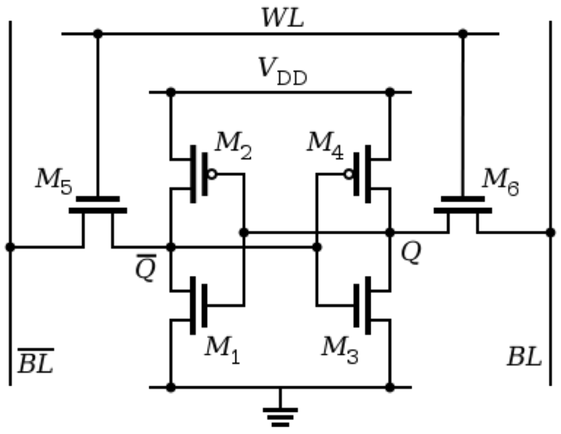


FPGA Configuration – After

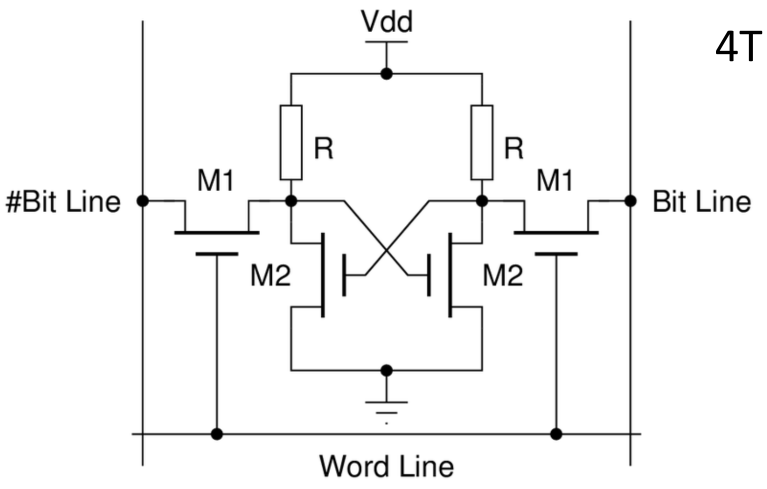


Configuration Comes at a Cost





6T SRAM

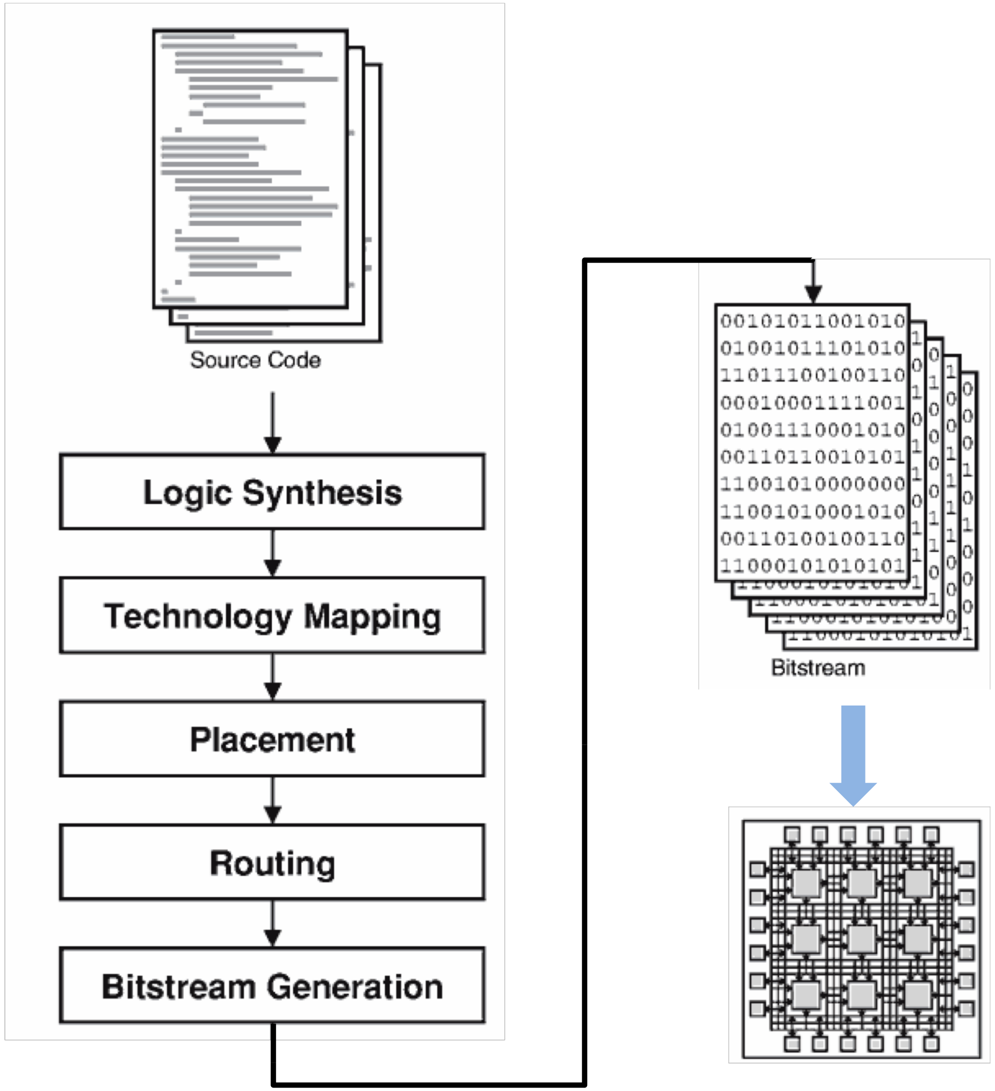


4T SRAM

+ Configuration circuitry
+ Error detection/correction
+ Security features

https://en.wikipedia.org/wiki/Static_random-

FPGA Design Flow



FPGA CAD Flow

- Input:
  - A circuit (netlist)
- Output:
  - FPGA configuration bitstream

- Main (Algorithmic) Stages:
  - Logic synthesis/optimization
  - Technology mapping
  - Packing/placement
  - Routing
  - Bitstream generation

Computing Technologies

HW, SW, and FPGA

- Traditional approaches to computation: HW & SW
- HW (ASICs)
  - Fixed on a particular application
  - Efficient: performance, silicon area, power
  - Higher cost/per application
- SW (microprocessors)
  - Used in many applications
  - Less efficient: performance, silicon area, power
  - Lower cost/per application

HW, SW, and FPGA

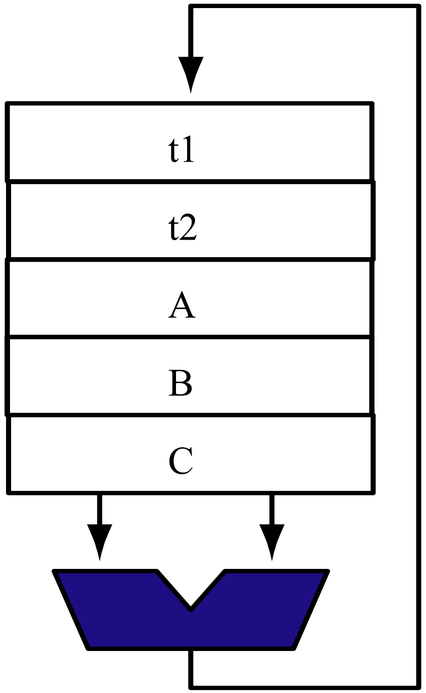
- Field Programmable Gate Arrays (FPGAs)
  - Spatial computing: similar to HW
  - Reprogrammable: similar to SW
  - Faster than SW and more flexible than HW
  - Harder to program than SW
  - Less efficient than HW: performance, power consumption & silicon area

Temporal vs Spatial Computing (SW vs. HW)

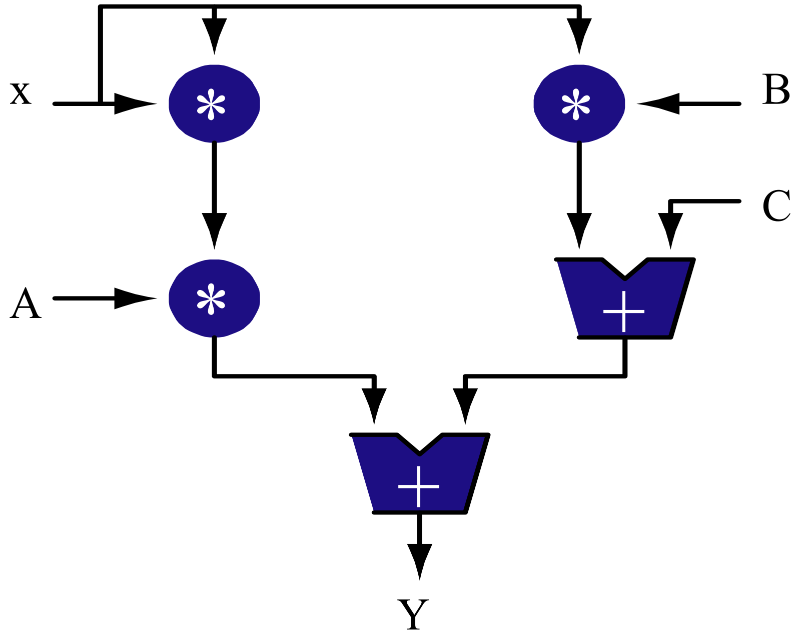
$$y = Ax^2 + Bx + C$$

Temporal Computation

t1 = x
t2 = t1 * A
t2 = t2 + B
t2 = t2 * t1
y = t2 + C



Spatial Computation



Why SW is Slower?

- Generality:
  - Instruction set may not provide the operations your program needs
  - Processors provide hardware that may not be useful in every program or in every cycle of a given program: Multipliers, Dividers
- Instruction Memory
  - Program instructions and intermediate results stored in memory.
  - Accessing memory is very slow.
- Bit Width Mismatches
  - General purpose processors have a fixed bit width, and all computations are performed on that many bits

SW or FPGA?

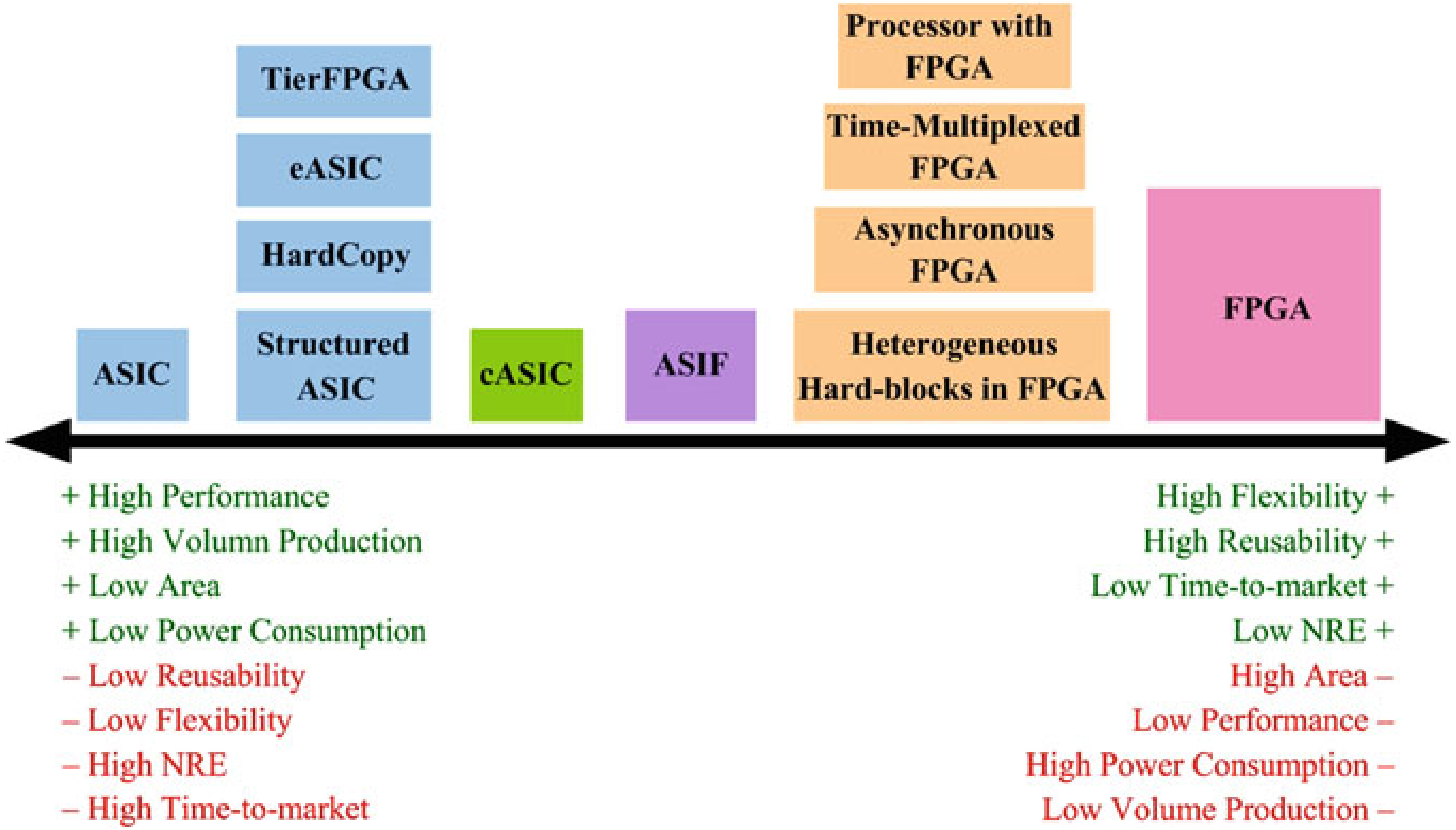
- CPUs – cheaper, faster, sequential, fix data format
  - Sequential, control-oriented applications
- FPGA – costlier, slower, parallel, custom data op.
  - Applications with data parallelism
- FPGA wins if

$$(\text{programming} + \text{exec time})_{\text{FPGA}} <= (\text{compilation} + \text{exec time})_{\text{CPU}}$$

How about ASIC HW?

- Dedicated -> not programmable.
- Takes long time and high cost to design and develop (typical processor takes a handful of years to design, with design teams of a few hundred engineers)
  - High non-recurring cost (NRE) -> very expensive!
- Justification for high cost: high volume applications, or high-performance is more desired

ASIC vs FPGA


TierFPGA
eASIC
HardCopy
Structured ASIC
ASIC
cASIC
ASIF
Processor with FPGA
Time-Multiplexed FPGA
Asynchronous FPGA
Heterogeneous Hard-blocks in FPGA
FPGA
+ High Performance
+ High Volumn Production
+ Low Area
+ Low Power Consumption
– Low Reusability
– Low Flexibility
– High NRE
– High Time-to-market
High Flexibility +
High Reusability +
Low Time-to-market +
Low NRE +
High Area –
Low Performance –
High Power Consumption –
Low Volume Production –

ASIC vs FPGA

- Time-to-Market
  - FPGA 6-12 month shorter
- Cost
  - FPGA much less expensive in low-volume applications
- Development time
  - FPGA shorter as no need to fabricate
- Power consumption
  - ASIC is better – no need to run SRAMs
- Debug and Verification
  - FPGA easier – direct test in-device

Instance–Specific Design

- ASIC targets a particular application
- ASIC more efficient than FPGA in application
- FPGA can be more efficient if it is customized to particular instances of an application
  - Encryption design for specific password
  - reduce area/power, higher performance
- Customizations
  - Data width
  - Constant folding
  - Function adaptation

Applications

- **Low-cost customizable digital circuitry**
  - Can be used to make any type of digital circuit.
  - Rapid with product development with design software. Upgradable.
- **High-performance computing**
  - Complex algorithms are off-loaded to an FPGA co-processor.
  - Application-specific hardware
  - FPGAs are inherently parallel and can have very efficient hardware algorithms: typical speed increase is x10 - x100.
- **Evolvable hardware**
  - Hardware can change its own circuitry.
  - Neural Networks.
- **Digital Signal Processing**

Reading

- Paper at http://www.cse.usf.edu/~haozheng/teaching/cda4253/

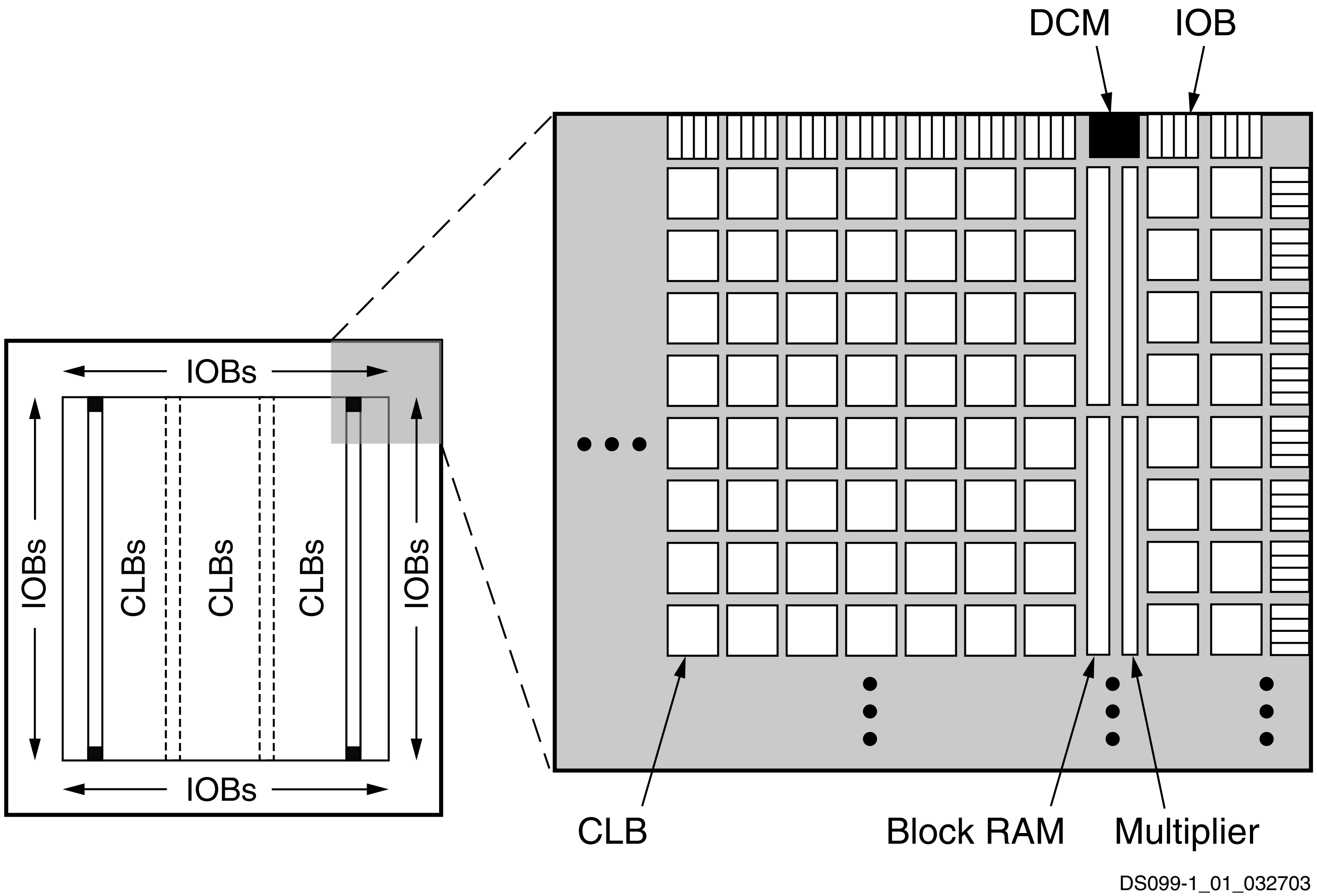
  FPGA Architectures: An Overview

  Section 2.1, 2.2, 2.3, 2.4 (skip 2.4.1.1, 2.4.2.2, 2.4.2.3),

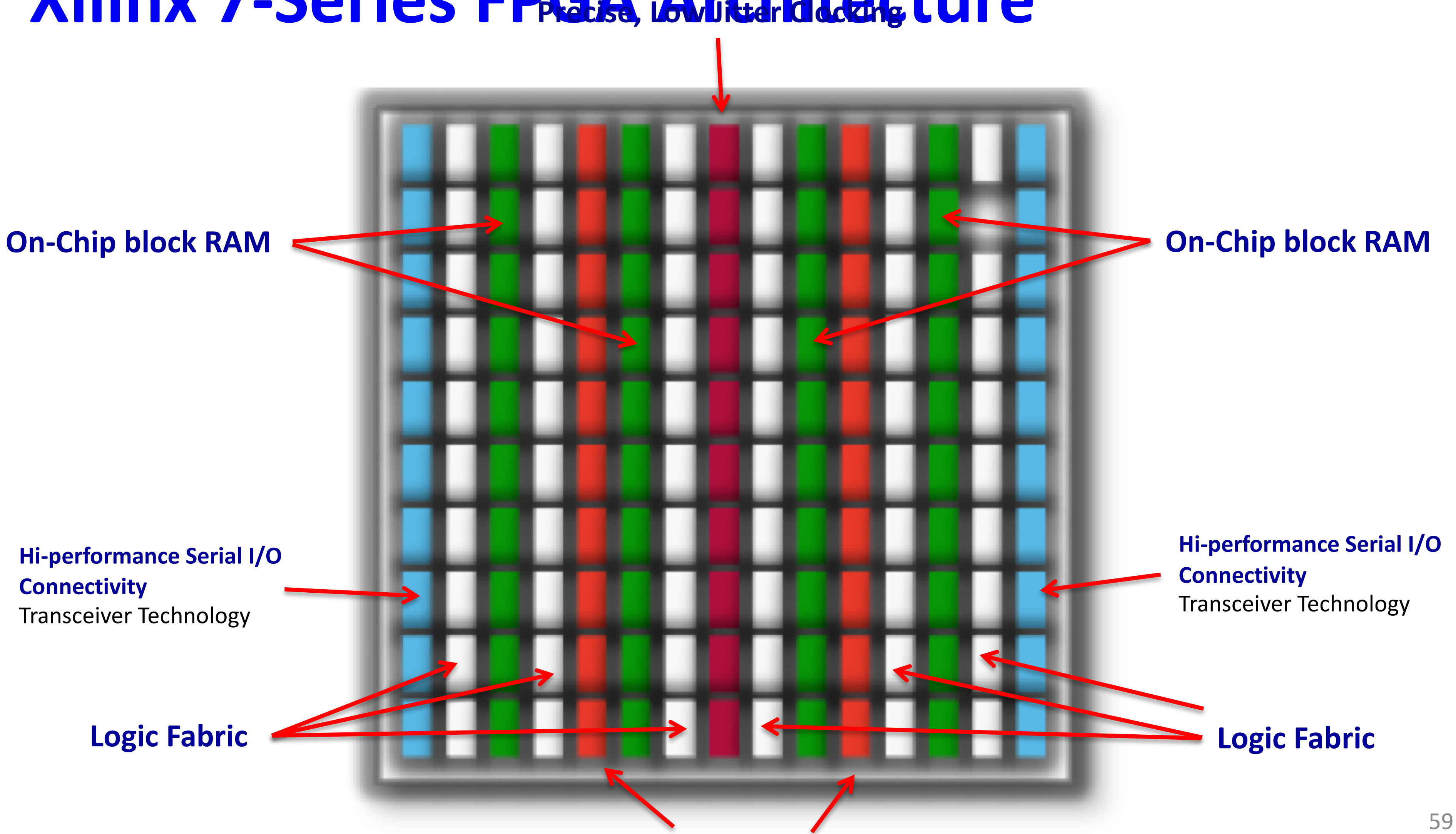
  Skim 2.6

Xilinx 7-Series Devices

Xilinx FPGA Architecture



Xilinx 7-Series FPGA Architecture


Precise, Low Jitter Clocking
On-Chip block RAM
On-Chip block RAM
Hi-performance Serial I/O Connectivity
Transceiver Technology
Hi-performance Serial I/O Connectivity
Transceiver Technology
Logic Fabric
Logic Fabric

Xilinx 7-Series Family

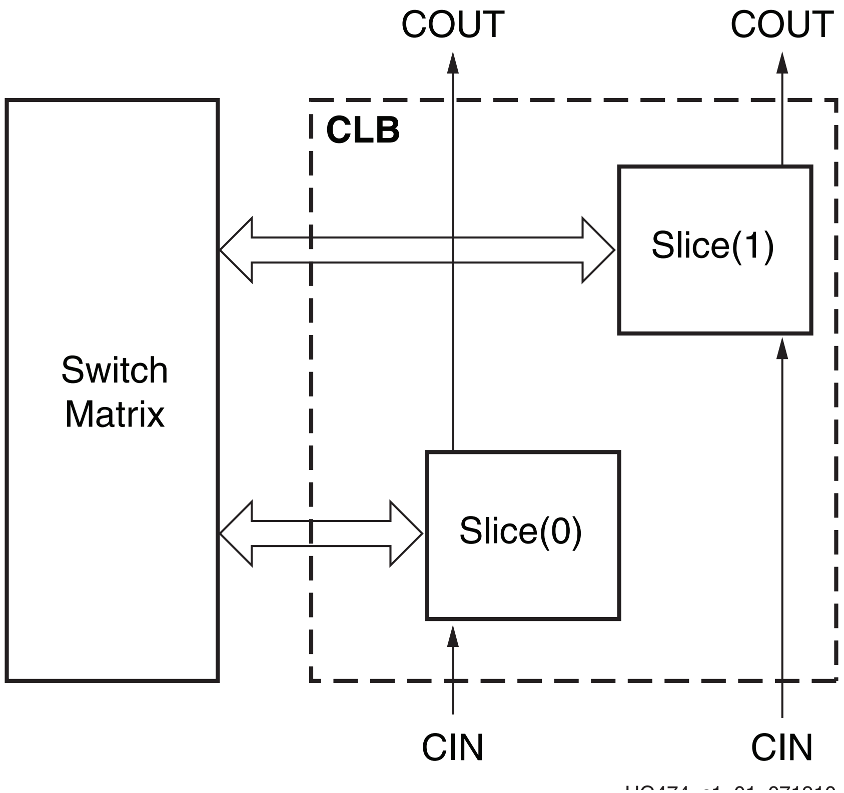
Maximum Capability	ARTIX 7 Lowest Power and Cost	KINTEX 7 Industry's Best Price/Performance	VIRTEX 7 Industry's Highest System Performance
Logic Cells	20K – 355K	70K – 480K	285K – 2,000K
Block RAM	12 Mb	34 Mb	65 Mb
DSP Slices	40 – 700	240 – 1,920	700 – 3,960
Peak DSP Perf.	504 GMACS	2,450 GMACs	5,053 GMACS
Transceivers	4	32	88
Transceiver Performance	3.75Gbps	6.6Gbps and 12.5Gbps	12.5Gbps, 13.1Gbps and 28Gbps
Memory Performance	1066Mbps	1866Mbps	1866Mbps
I/O Pins	450	500	1,200
I/O Voltages	3.3V and below	3.3V and below 1.8V and below	3.3V and below 1.8V and below

Xilinx Artix-7

- Low end 7-series FPGA manufactured using 28nm
- Based on 6-input LUT
  - Configurable as distributed memory
- Support DDR3 memory interfaces
- High-speed serial interfaces supporting multi-gigabit communications
- On-chip DSPs, multipliers, and block RAMs
- Clock management tiles to provide high precise and low jitter clock signals

Xilinx Artix-7 - CLBs

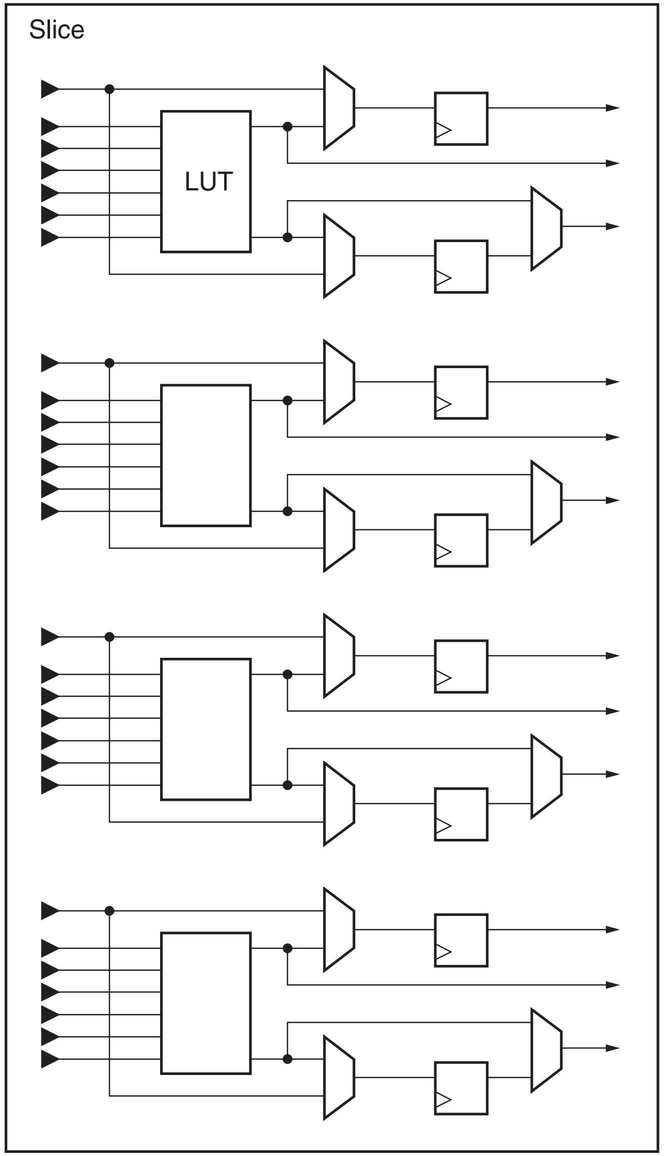
- 8 **6-LUTs**
- 16 **FFs**
- 2 carry chains
- 256b **distributed RAM**
- 128b **shift register**



- The abundant FFs can be used to improve design performance with pipelining.

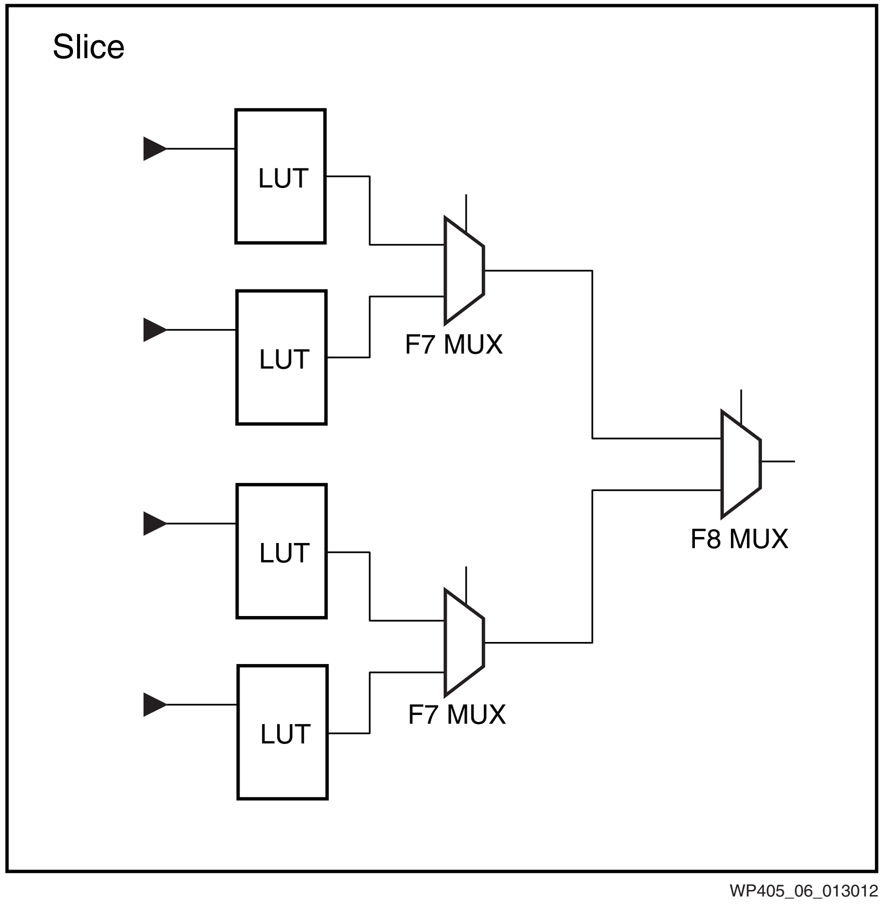
Xilinx Artix-7 – CLBs
Slice Architecture

- 4 6-LUTs
- 8 FFs
- Carry logic for fast addition
- Other local wires w/o global routing

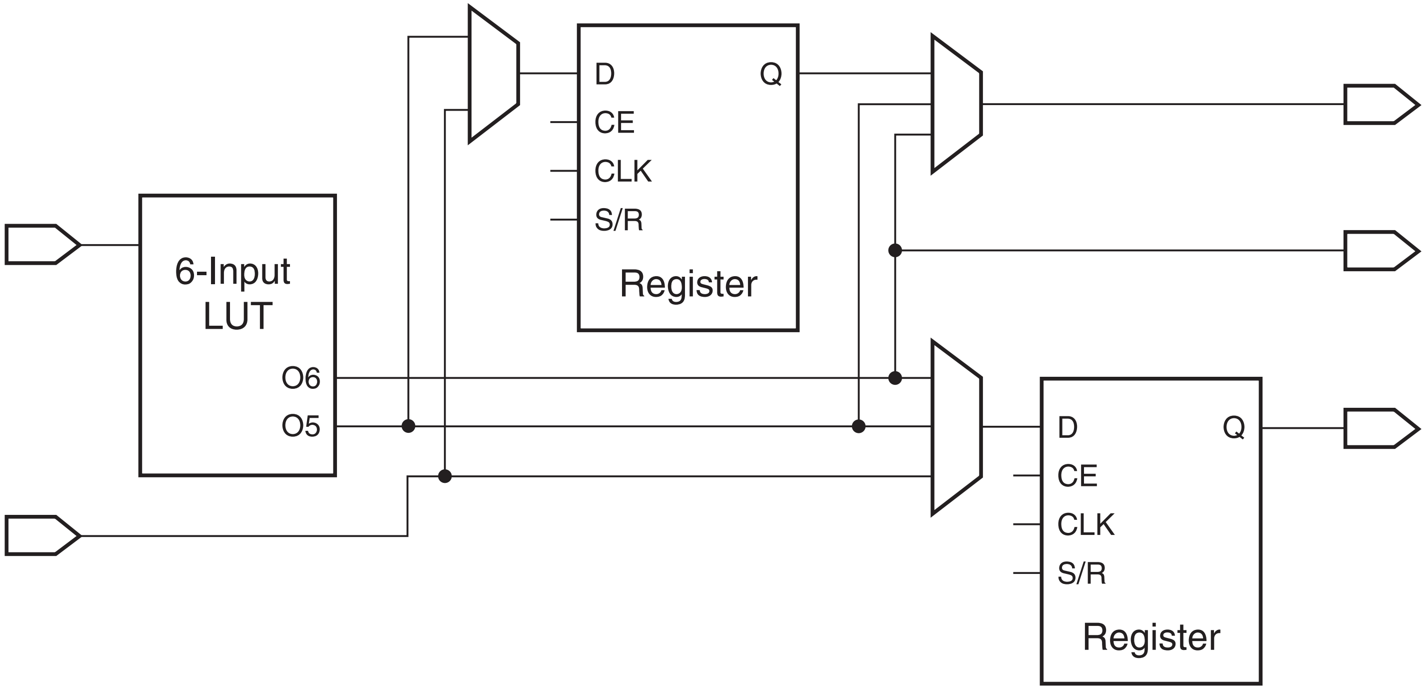


Xilinx Artix-7 CLBs Slice Architecture

Wide-function MUXs to implement functions with 8 inputs.



Xilinx Artix-7 – CLBs



- Each 6-LUT implements any 6-input functions, or
- Two 5-input functions with shared inputs.

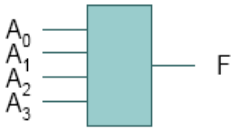
Distributed RAMs

- Slices in CLBs of type SLICEM can be configured as synchronous RAMs
  - 256x1b single port
  - 128x1b dual/single port
- Can also be configured as ROM with up to 256b.
- Can be instantiated by using special VHDL components.

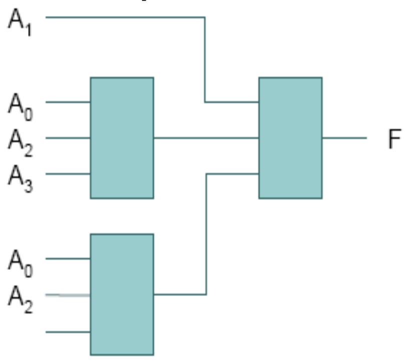
Backup

$$F = A_0A_1A_3 + A_1A_2\bar{A}_3 + \bar{A}_0\bar{A}_1\bar{A}_2$$

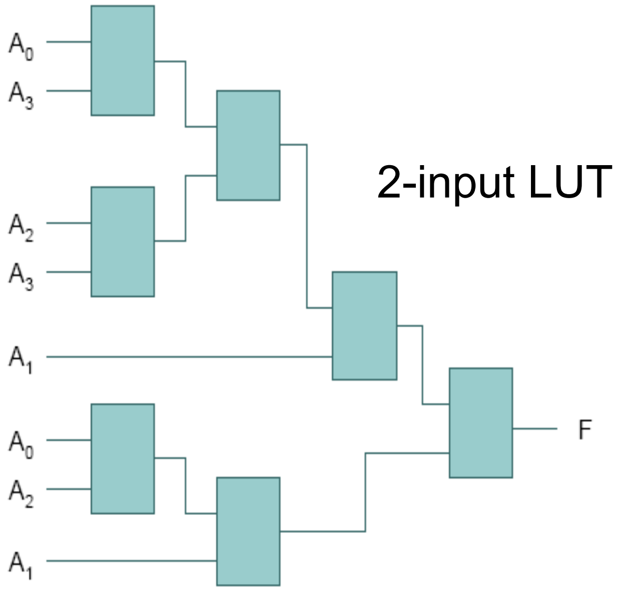
4-input LUT



3-input LUT



2-input LUT



Xilinx Artix-7

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices[2]	Block RAM Blocks[3]		
		Slices[1]	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)
XC7A15T	16,640	2,600	200	45	50	25	900
XC7A35T	33,280	5,200	400	90	100	50	1,800
XC7A50T	52,160	8,150	600	120	150	75	2,700
XC7A75T	75,520	11,800	892	180	210	105	3,780
XC7A100T	101,440	15,850	1,188	240	270	135	4,860
XC7A200T	215,360	33,650	2,888	740	730	365	13,140

Xilinx Artix-7

Device	CMTs[4]	PCIe[5]	GTPs	XADC Blocks	Total I/O Banks[6]	Max User I/O[7]
XC7A15T	5	1	4	1	5	250
XC7A35T	5	1	4	1	5	250
XC7A50T	5	1	4	1	5	250
XC7A75T	6	1	8	1	6	300
XC7A100T	6	1	8	1	6	300
XC7A200T	10	1	16	1	10	500