EDA322
Digital Design

Lecture 11:
Technology – Reconfigurable Hardware

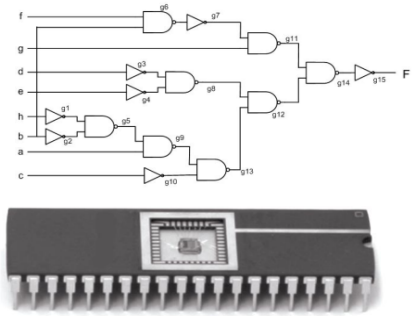
Ioannis Sourdis

Outline of Lecture 11:

- Why Reconfigurable HW is interesting?
- Reconfigurable devices:
  - Logic
  - Interconnects
  - Specialized modules and extended logic
  - Reconfiguration
- Coarse-grain Reconfigurable devices
- Summary

Computing alternatives

Hardware (Application Specific Integrated Circuits)



Advantages:

- very high performance and efficient

Disadvantages:

- not flexible (can't be altered after fabrication)
- High NRE Cost

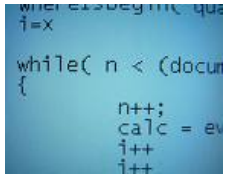
Reconfigurable computing

Advantages:

- much higher performance than software / lower performance than ASIC
- higher level of flexibility than hardware / more difficult to program than SW

•fills the gap between hardware and software

Software-programmed processors



Advantages:

- software is very flexible to change

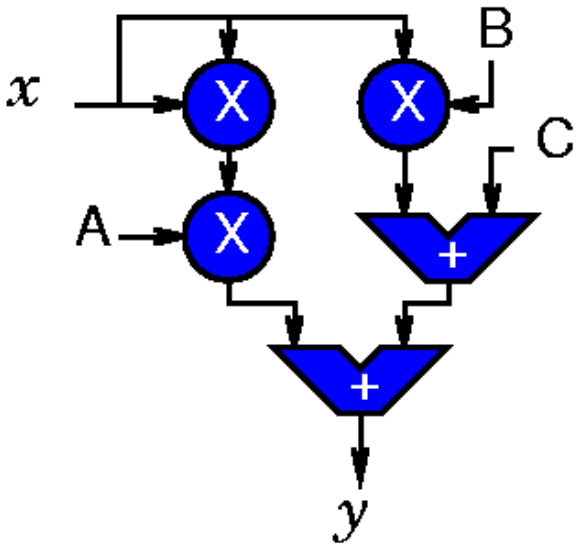
Disadvantages:

- performance can suffer if clock is not fast
- fixed instruction set by hardware

Spatial vs. Temporal computing

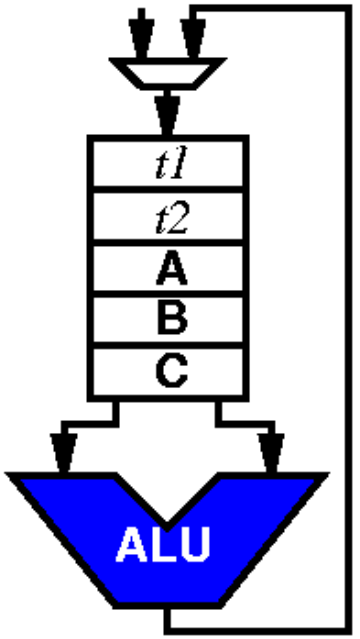
$$F = Ax^2 + Bx + C$$

Spatial-based execution (hardware)



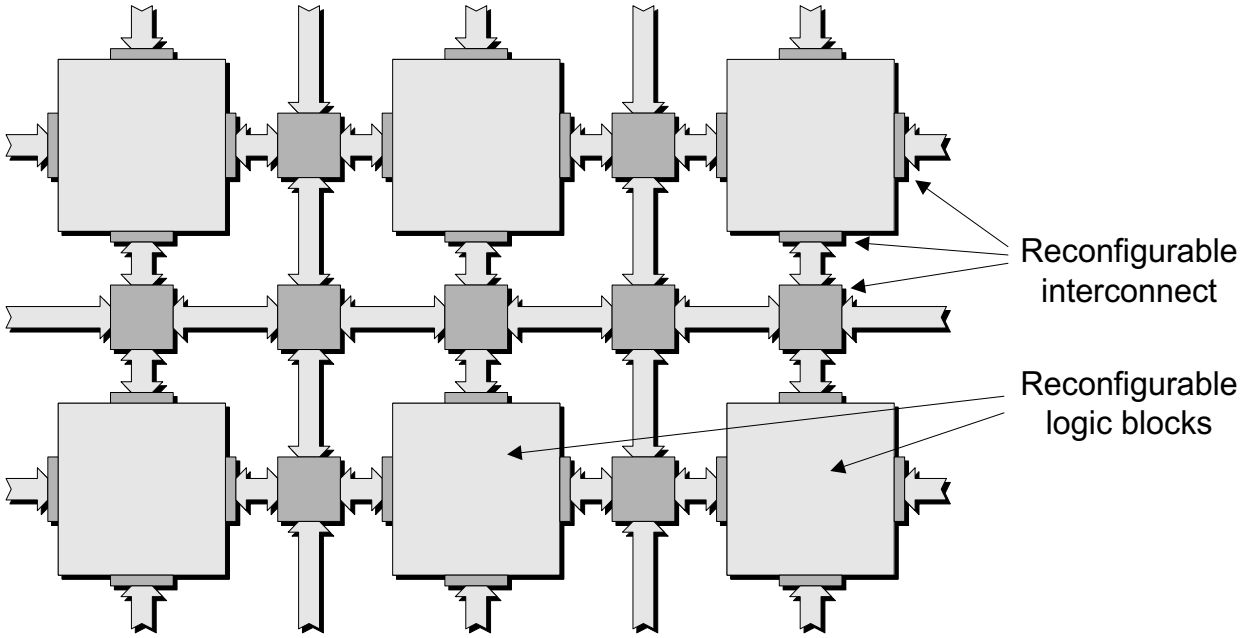
Temporal-based execution (software)

$t1 \leftarrow x$
$t2 \leftarrow \mathbf{A} \times t1$
$t2 \leftarrow t2 + \mathbf{B}$
$t2 \leftarrow \mathbf{t2} \times t1$
$y \leftarrow \mathbf{t2} + \mathbf{C}$



- Extract parallelism (or concurrency) from algorithm descriptions is one of the keys to acceleration using reconfigurable computing

Reconfigurable devices

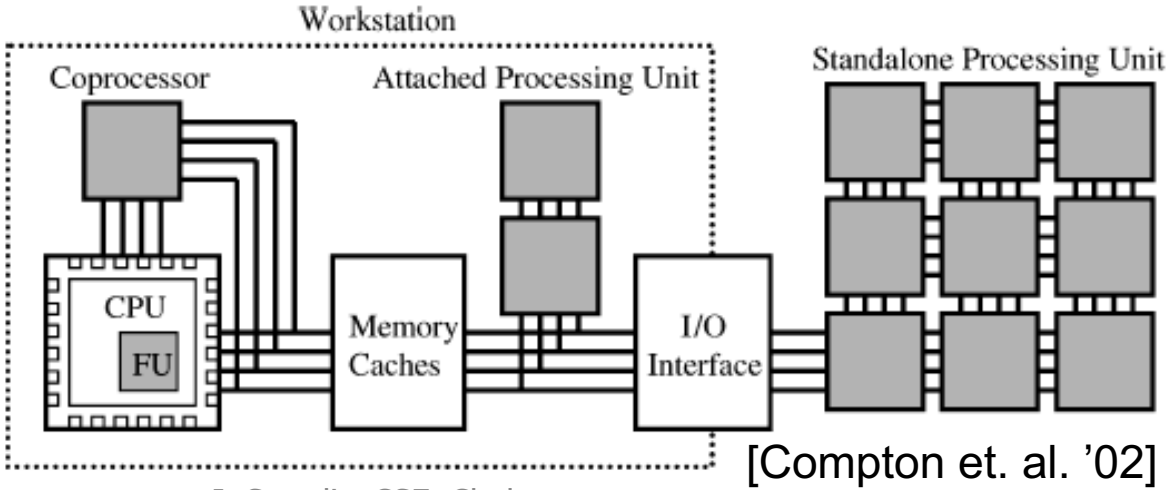


- ***Field-Programmable Gate Arrays*** *(FPGAs)* are one example of reconfigurable devices
- An FPGA consists of an array of ***configurable logic blocks*** whose functionality is determined by configuration bits
- The logic blocks are connected by a set of ***routing resources*** *(WIRES)* that are also programmable

➢ **Arbitrary custom logic circuits/functions** can be *mapped* to the reconfigurable fabric

Uses of reconfigurable devices

1. Low/med volume IC production
2. Early prototyping and logic emulation
3. Accelerating algorithms in reconfigurable computing environments
   - Reconfigurable functional units within a host processor (custom instructions)
   - Reconfigurable units used as coprocessors
   - Reconfigurable units that are accessed through external I/O or a network



[Compton et. al. '02]

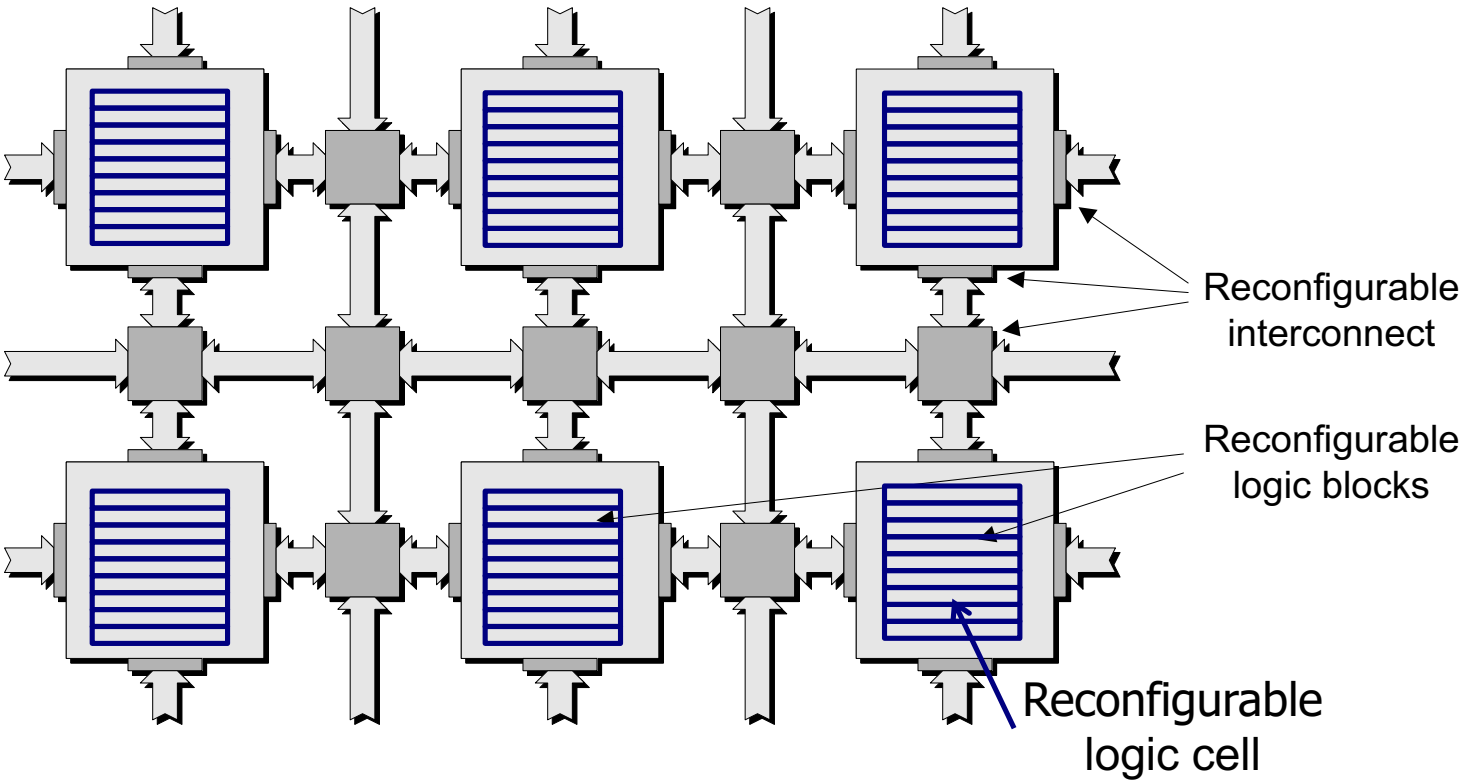
Why reconfigurable?

- **Performance, Performance/Power:** Many applications run more efficiently in Reconfigurable Hardware :
  - Streaming applications
  - Parallelism,
  - When GPPs and ASICs don't match Applications requirements (datapath, ISA, etc.)
    - **Customization required**
  - Apps with changing requirements (**adaptation**)
- **FPGAs provide:**
  - the spatial computational resources to implement massively-parallel computations directly in hardware
  - Customization
  - Adaptation

  More:
  - Reduced Time to market vs. ASIC
  - No NRE (Non recurring Engineering) cost vs. ASIC
    - **NRE costs: one-time cost to research, develop , design and test a new product**

Reconfigurable devices

FPGA devices



Reconfigurable Logic

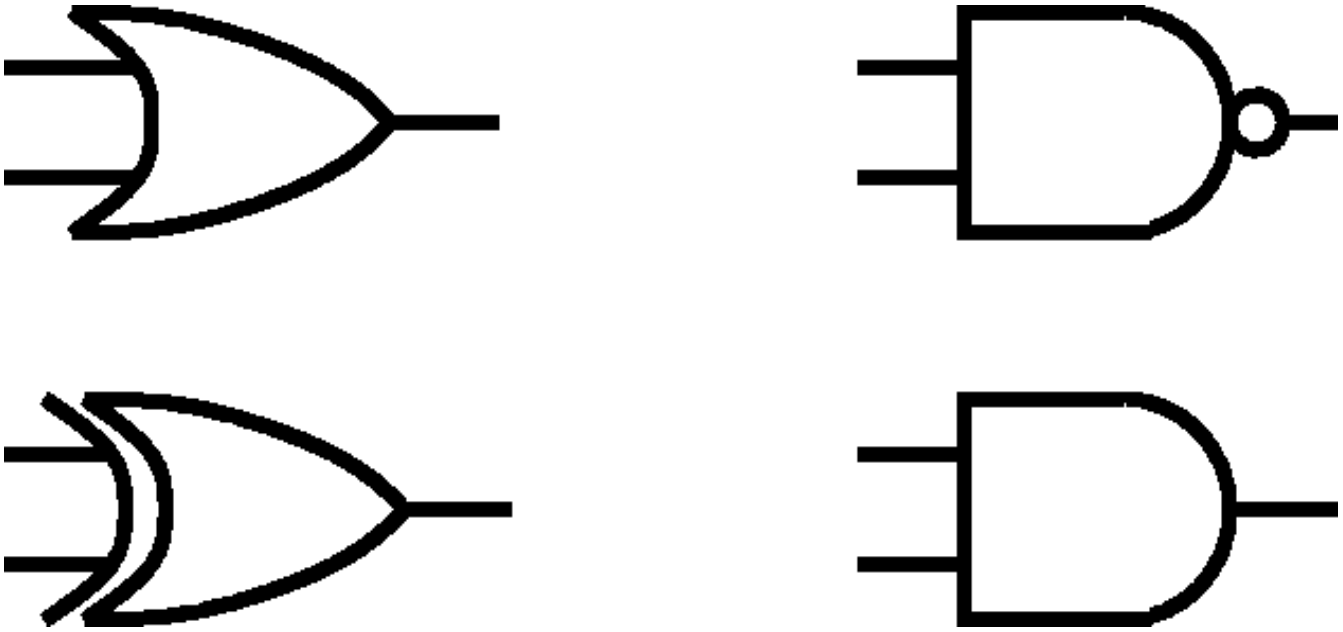
FPGAs

Field Programmable Gate Arrays

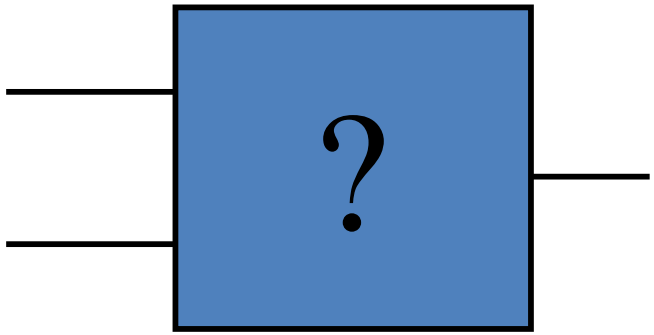
Collection of programmable “gates” embedded in a flexible interconnection network.

…a “user programmable” alternative to gate arrays.

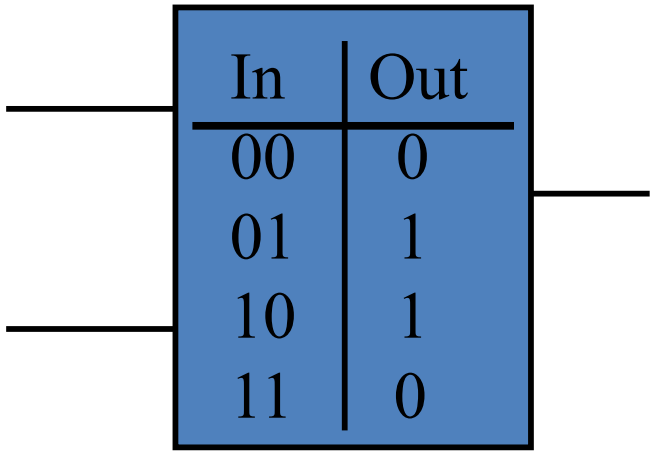
Gates



Reconfigurable Gate



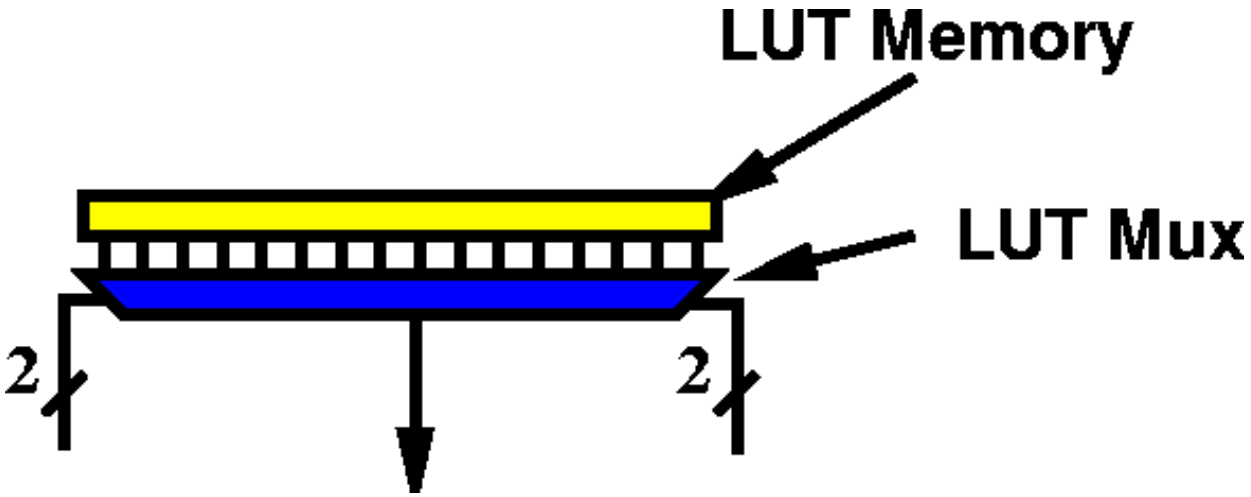
Look-Up Table



2-LUT

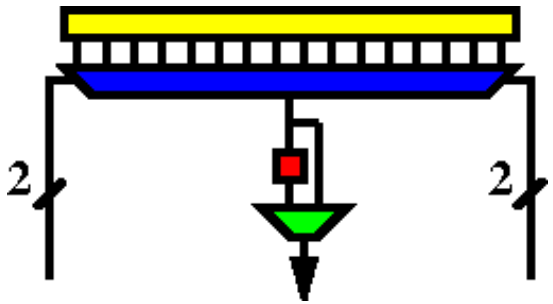
LUTs

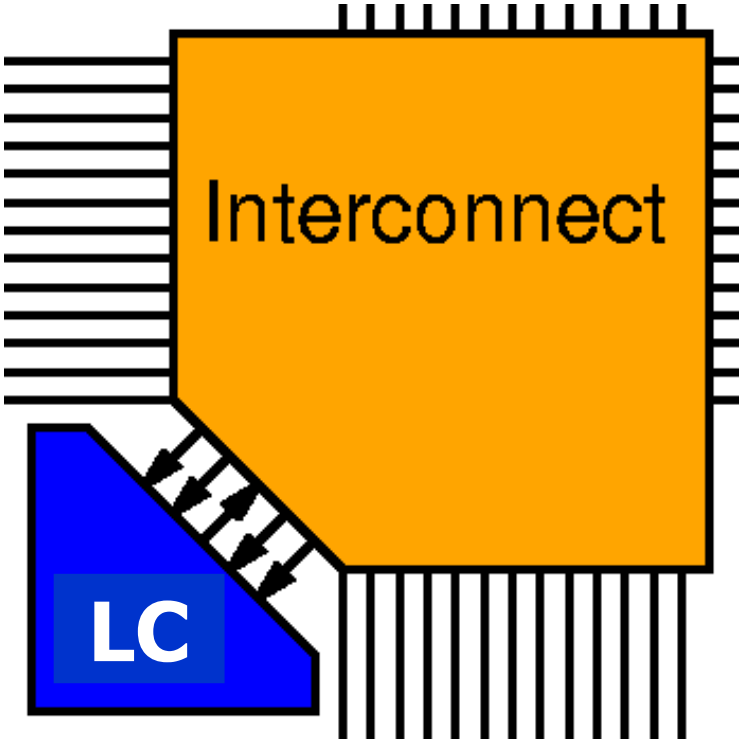
- K-LUT -- K input lookup table
- Any function of K inputs by programming table



Logic Cells:
Conventional FPGA Tile

K-LUT (typical k=4)
w/ optional
output Flip-Flop



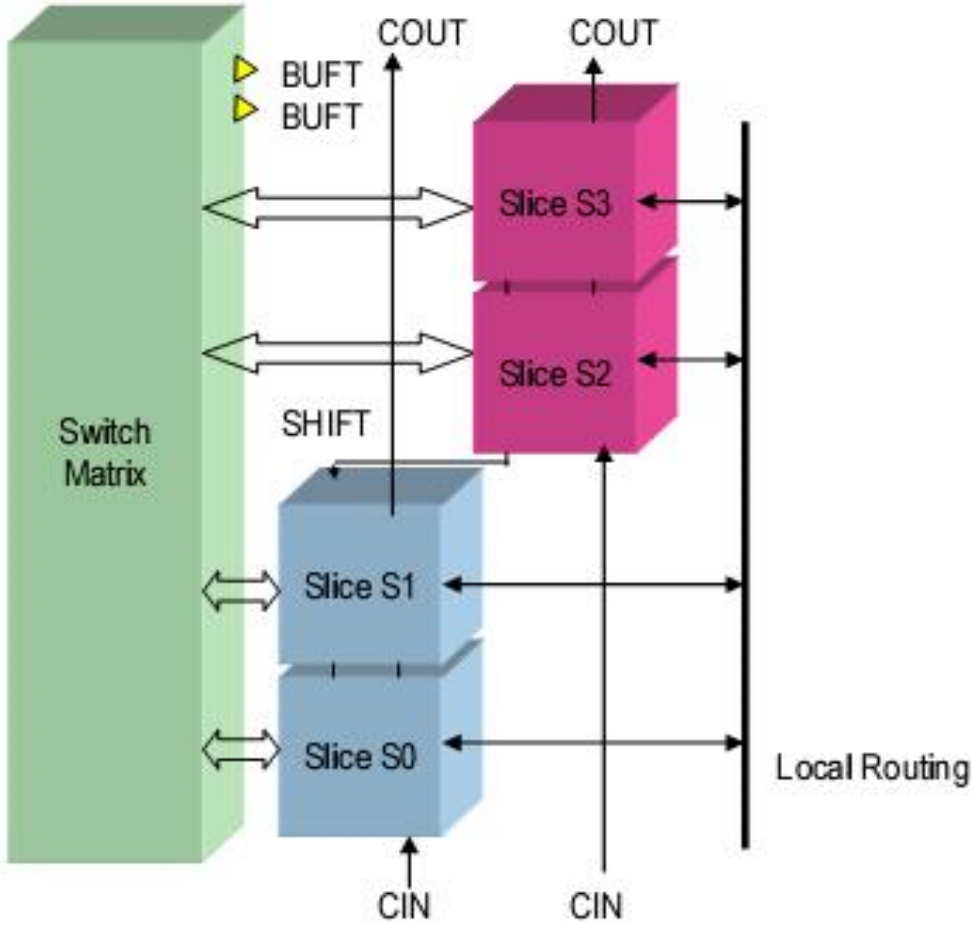


Clusters of Logic Cells (Slices, CLBs, Logic Blocks, etc.)

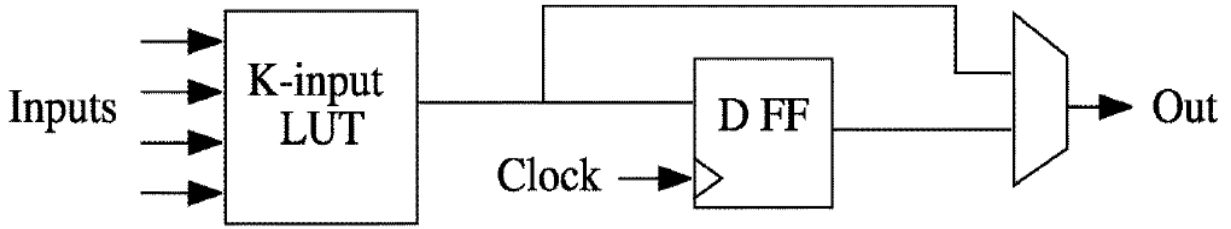
• Assume K-input LUT in each Logic Cell (LC) and assume N LCs per Logic Cluster

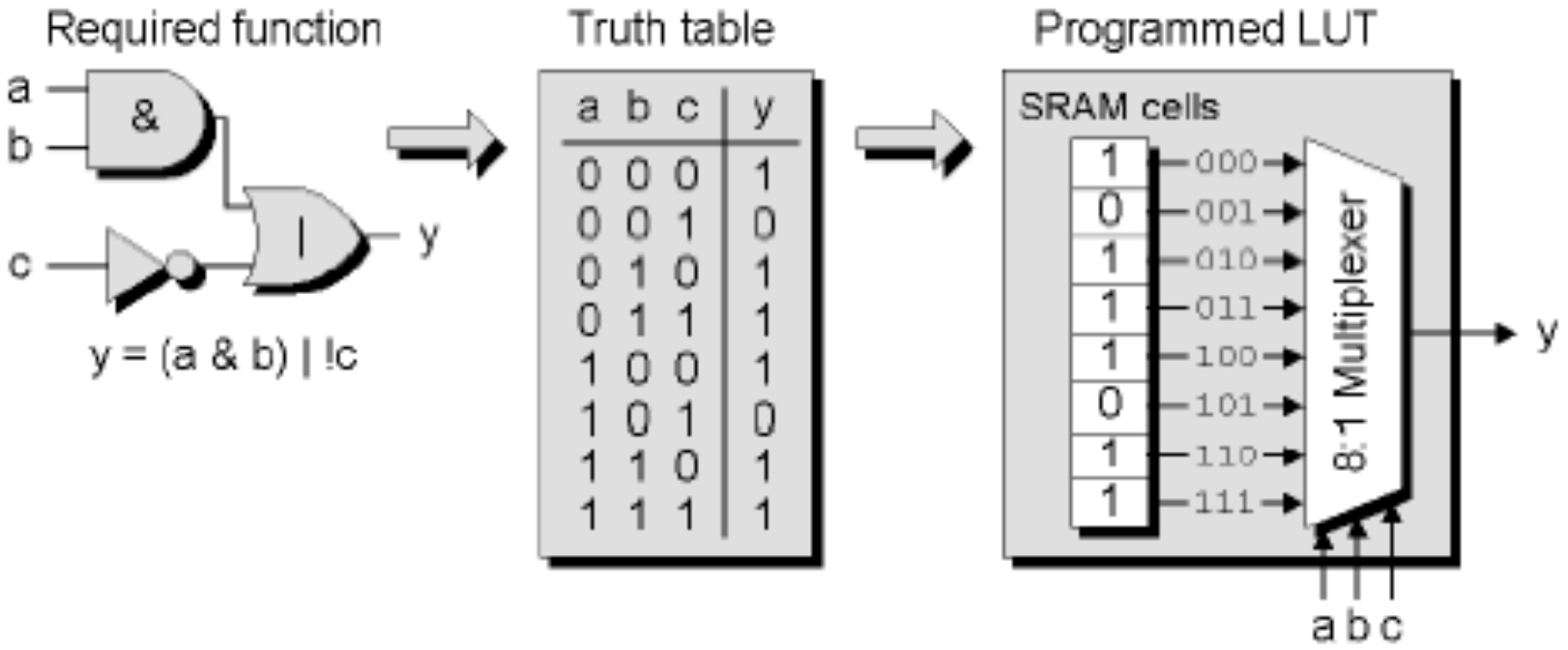
• The LCs in each logic clusters are *fully connected* or "nearly-fully" connected with each other

• Xilinx Virtex4:
  - 4-input LUTs
  - 2 LCs = 1 Slice
  - 4 Slices= 1 CLB



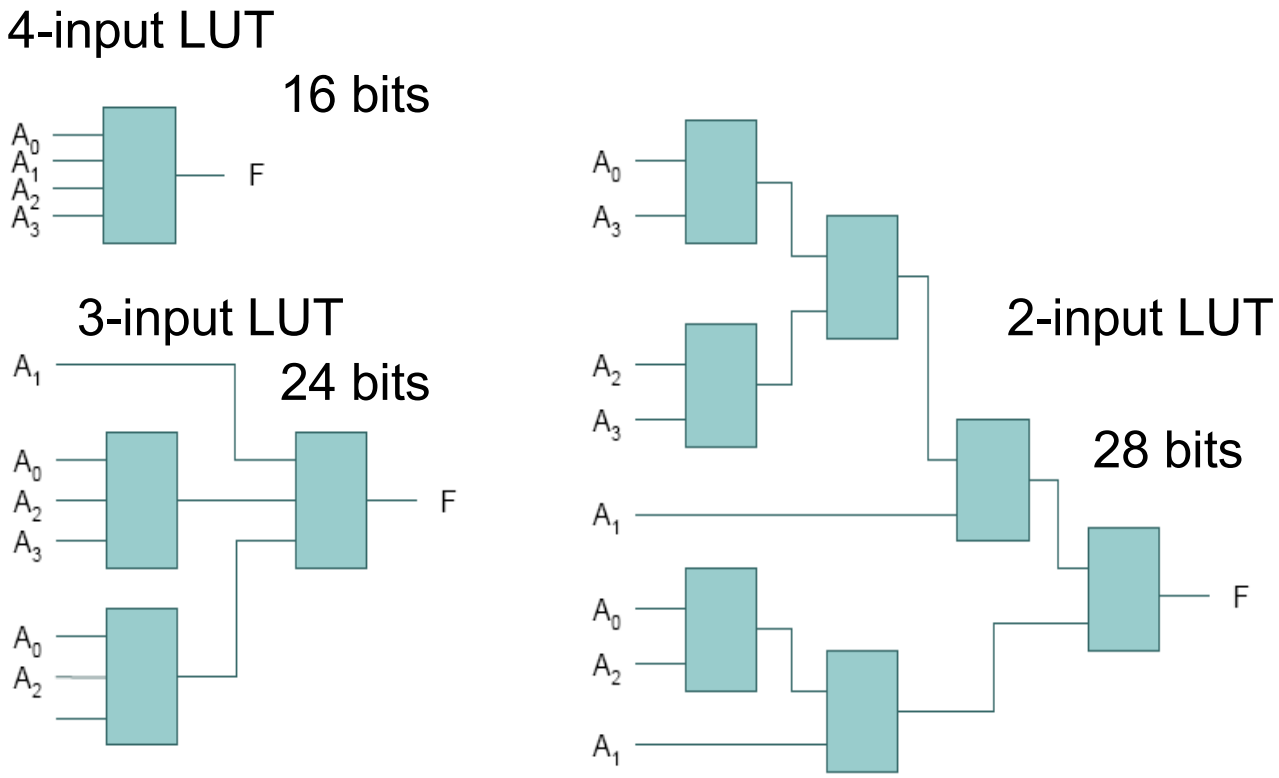
Mapping logic to a Logic Cell





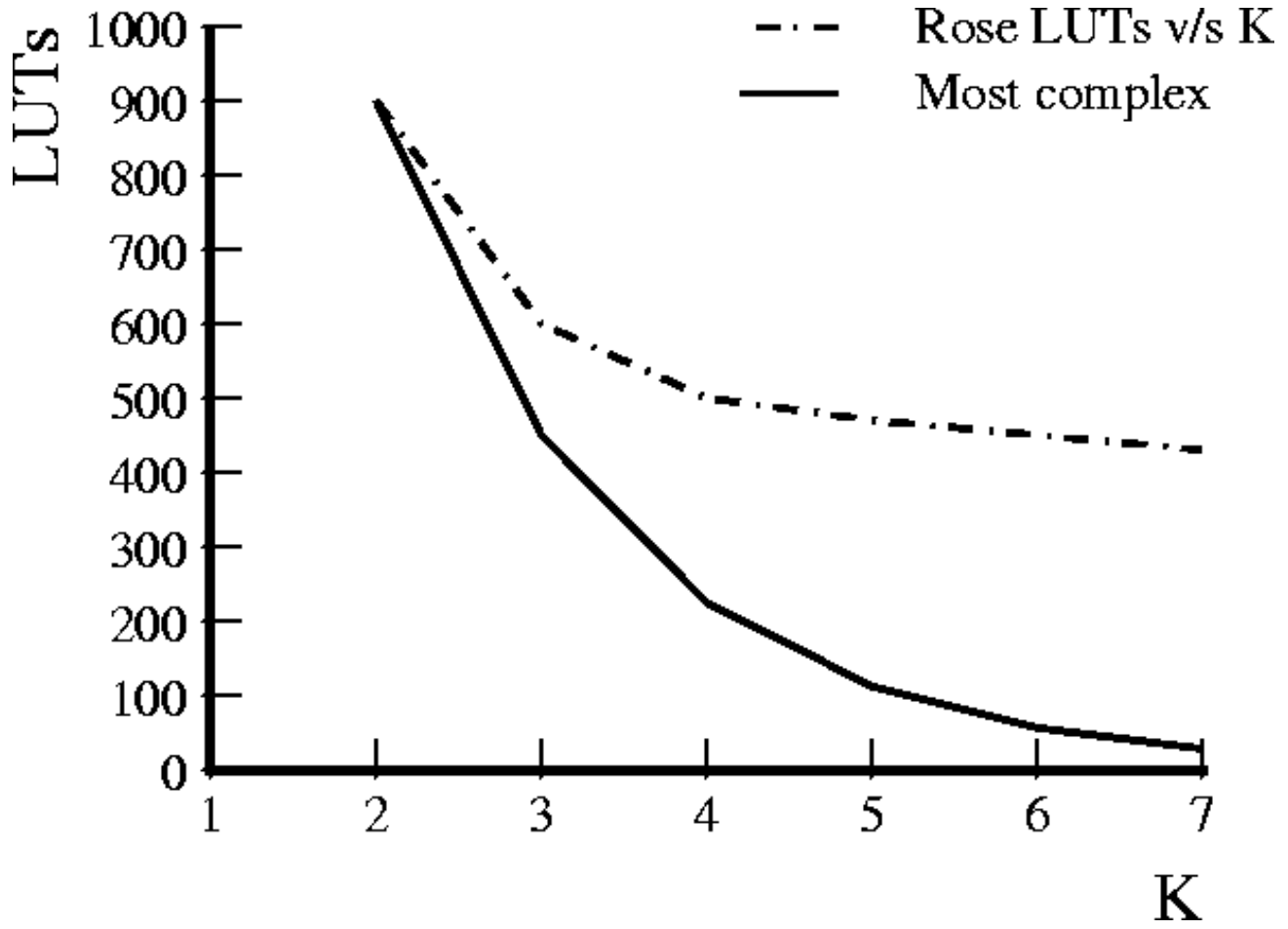
Example

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



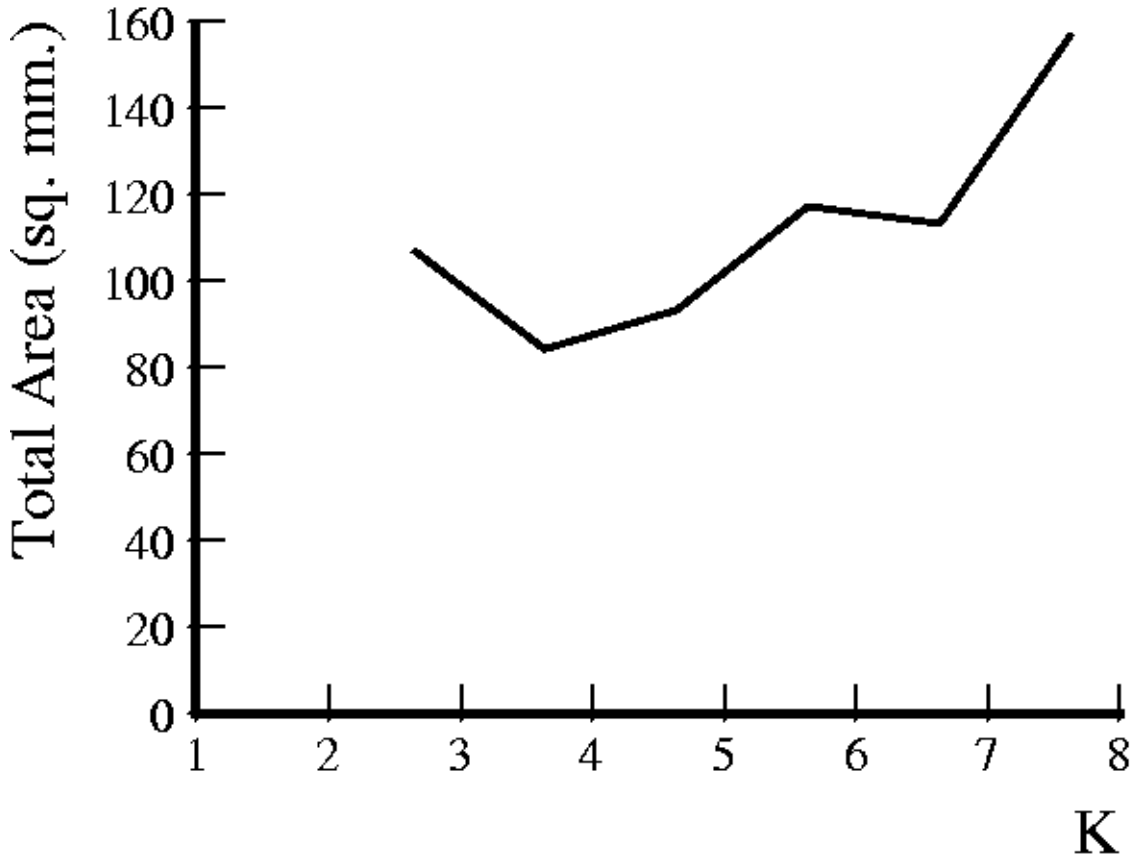
- **What is the number of bits in a K-input table?**
- **How many different Boolean functions can a K-input LUT implement?**
- **What is the best LUT size?**

LUT Count vs. base LUT size



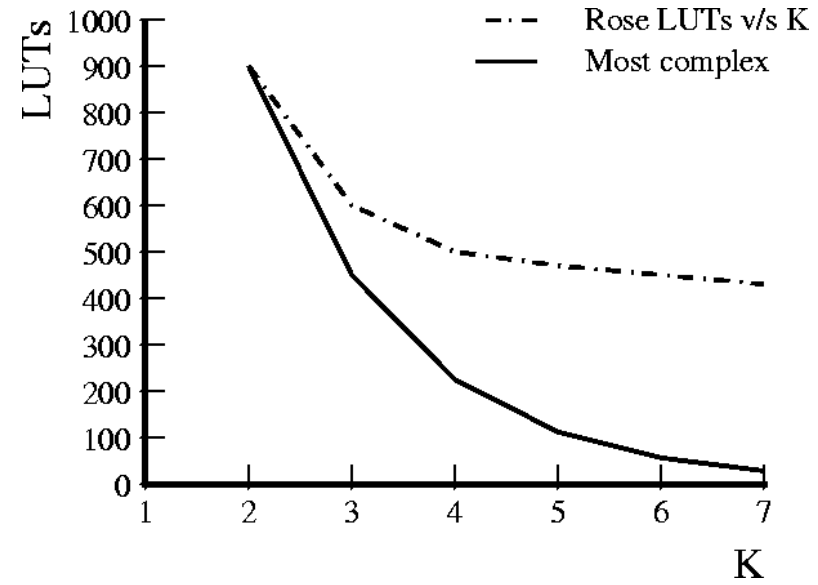
Base LUT size: Rose et. al. empirical study

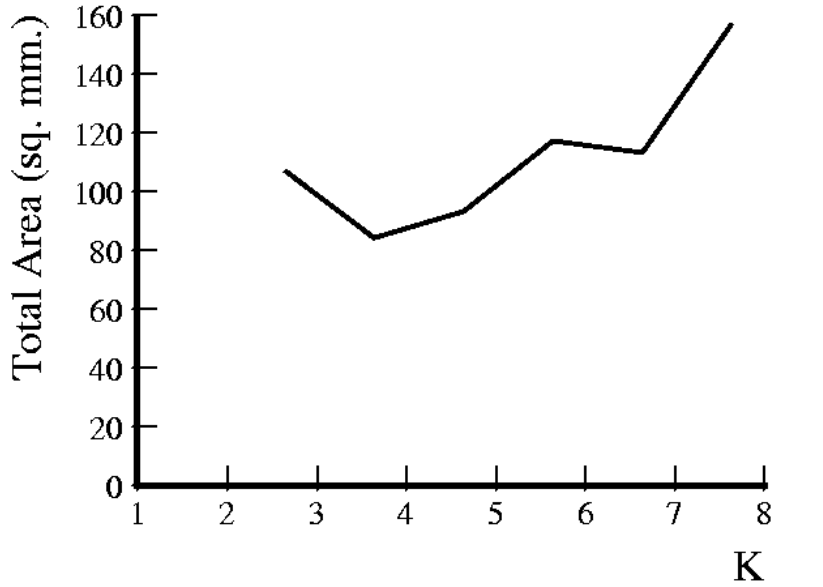
- Minimum LUT **Area**
  - at K=4
  - robust for different switch sizes
    - (wire widths)



Area: U.Toronto Experiments (Rose etal.)

- Minimum LUT Area K=4
- Bigger LUTs
  - handle complicated functions efficiently
  - less interconnect overhead
- Smaller LUTs
  - handle regular functions efficiently
  - interconnect allows exploitation of compute structure
- Depends on the typical complexity/structure of the logic/design?

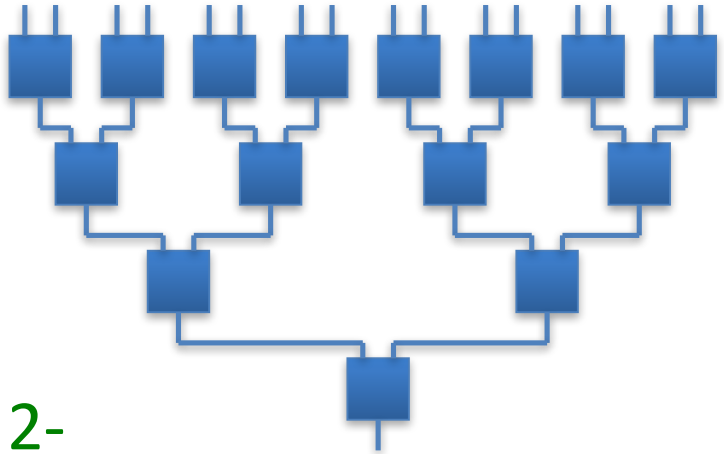




Delay?

- Circuit Depth in LUTs?

- "Simple Function" --> M-input AND
  - 1 table lookup in M-LUT
  - $\log_k(M)$ in K-LUT

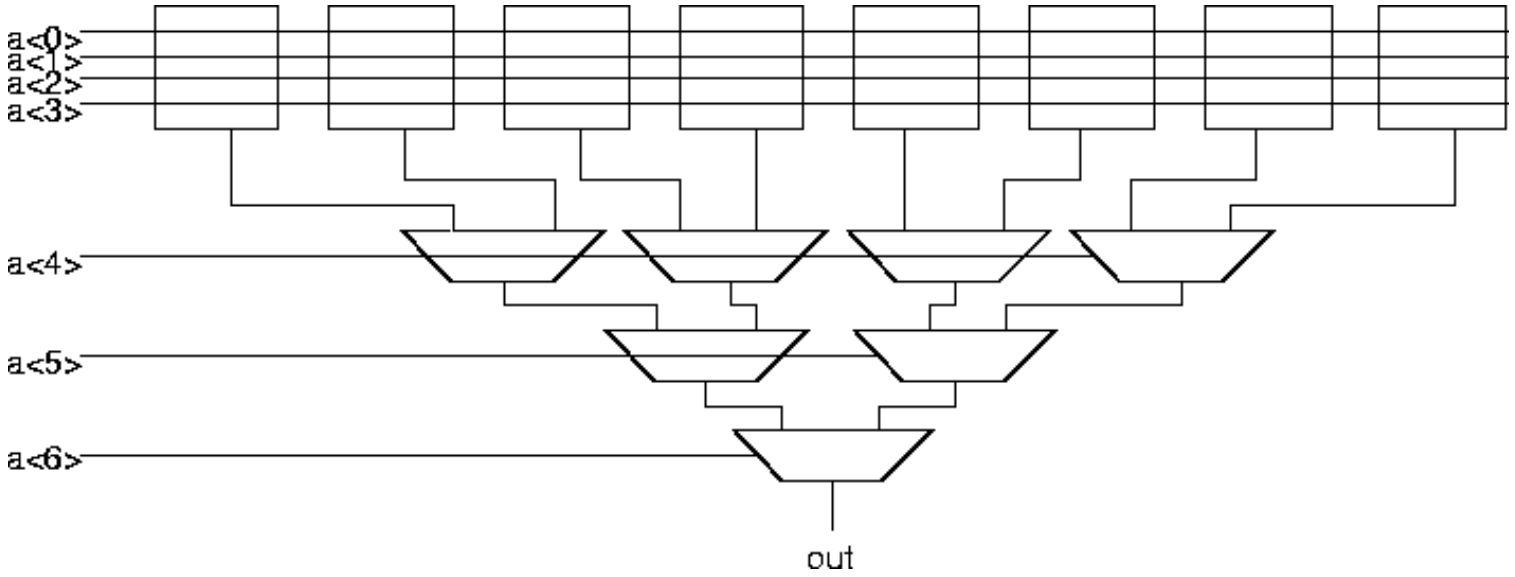


Example:

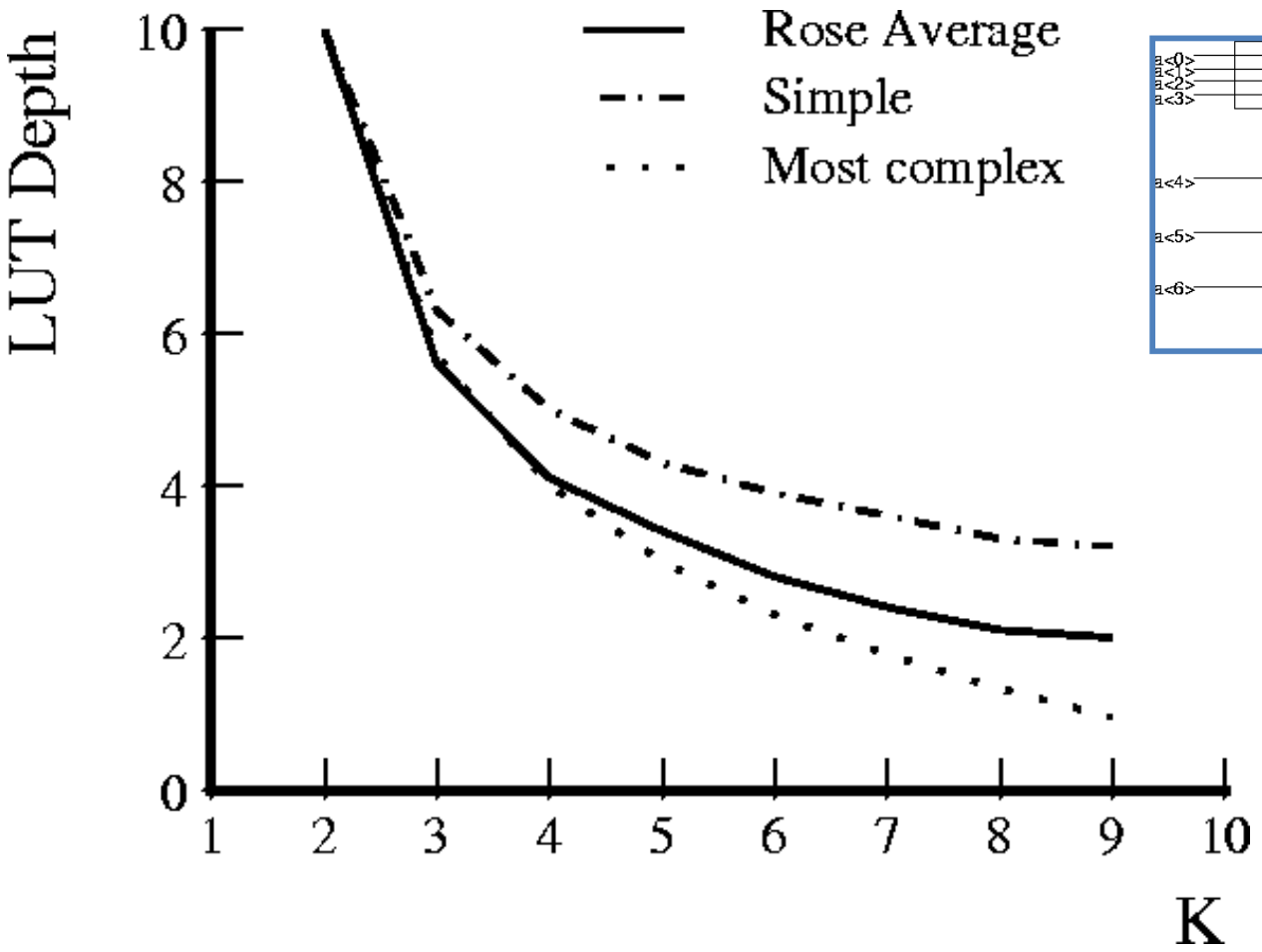
- to implement a 16-input AND with 2-input LUTs, we need **$\log_2(16)=4$** levels of LUTs
  - think of a binary tree, each node been an LUT implementing a 2-input AND

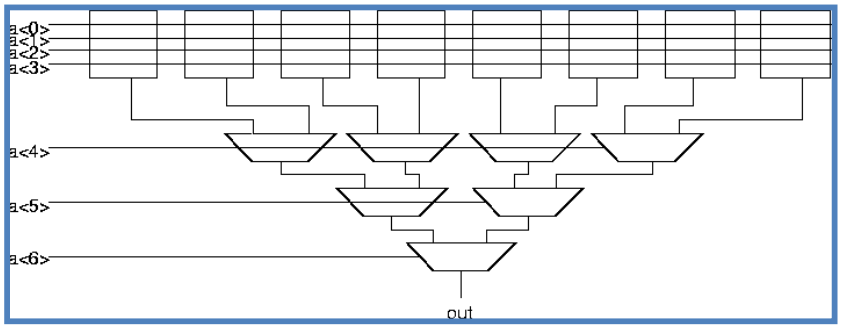
Delay?

- M-input “Complex” function
  - 1 table lookup for K-LUT
  - Depth ≤ (M-K) +1

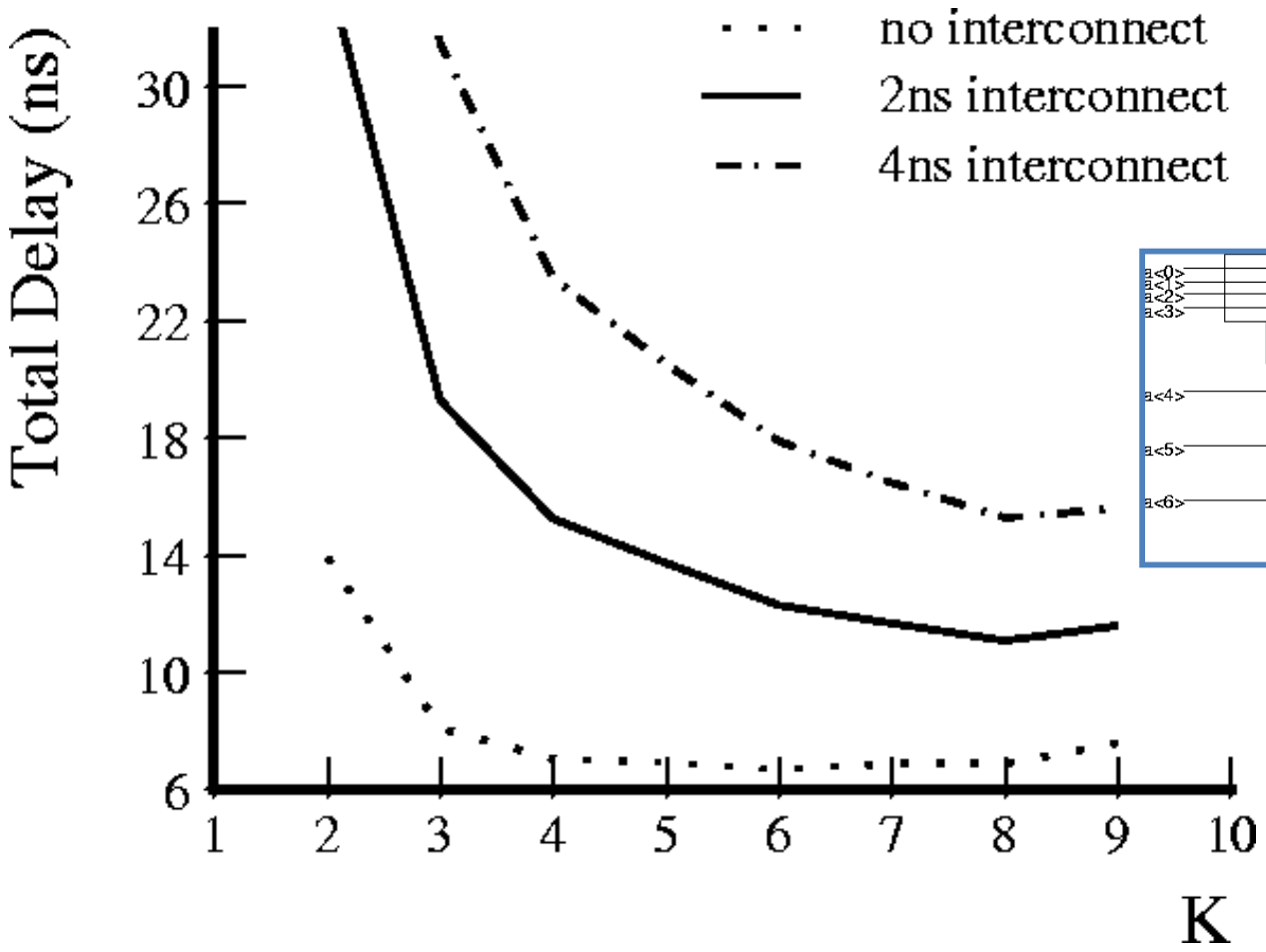


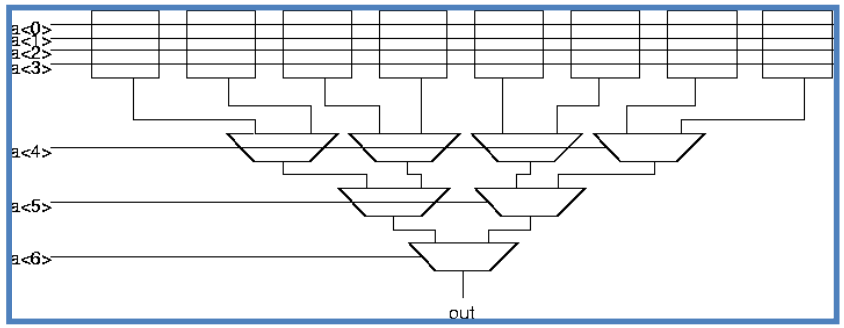
Circuit Depth vs. K





Delay vs. K





Delay = Depth $\times$ (t_{LUT}+ $t_{Interconnect}$)

Observations

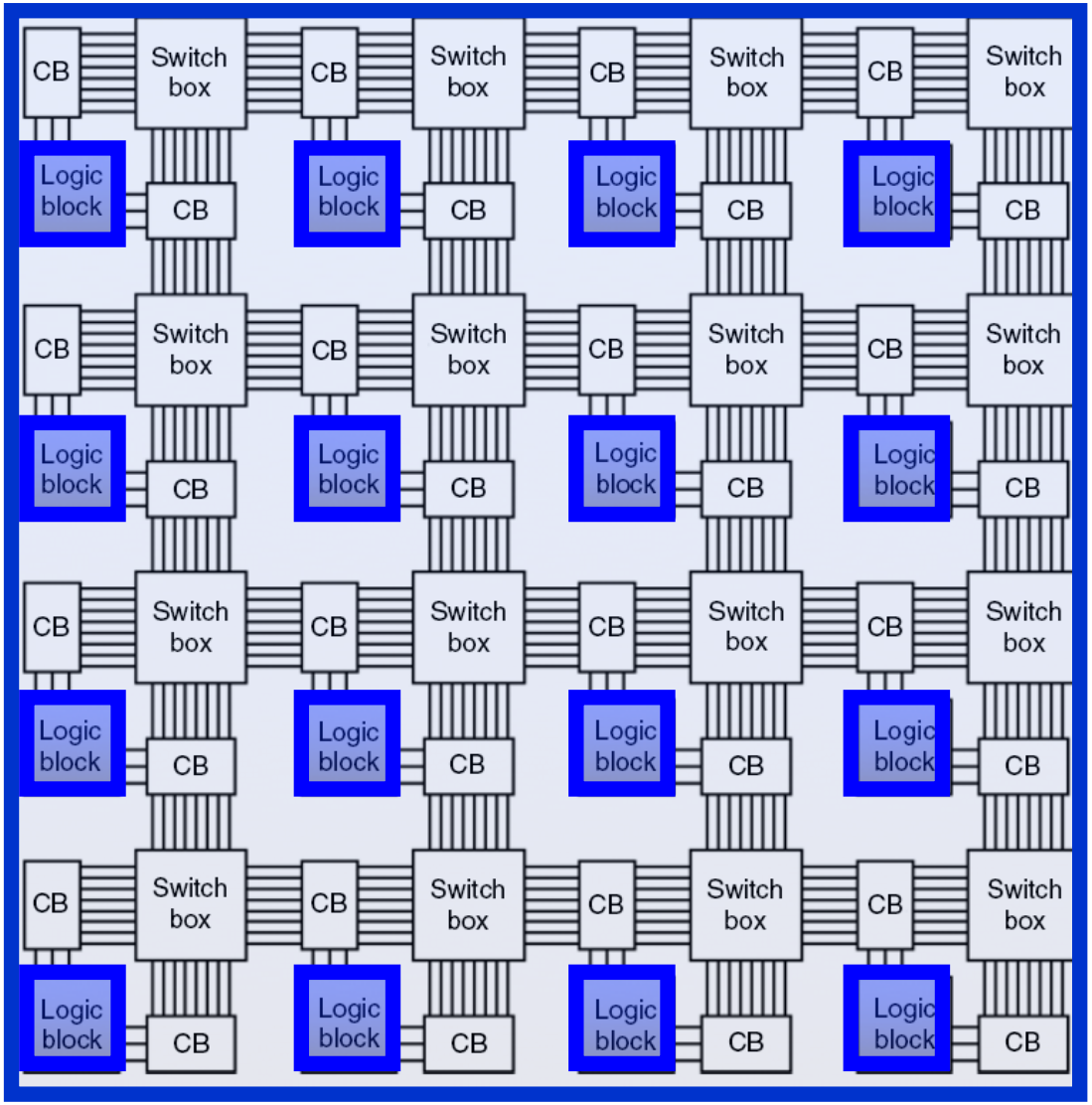
- “Larger” logic blocks :
  - less interconnect crossing
    - Interconnects are expensive
  - lower interconnect delay
  - logic blocks get slower
  - less area efficient
    - don’t match structure in computation
    - Large logic blocks are often underutilized

Reconfigurable Interconnects

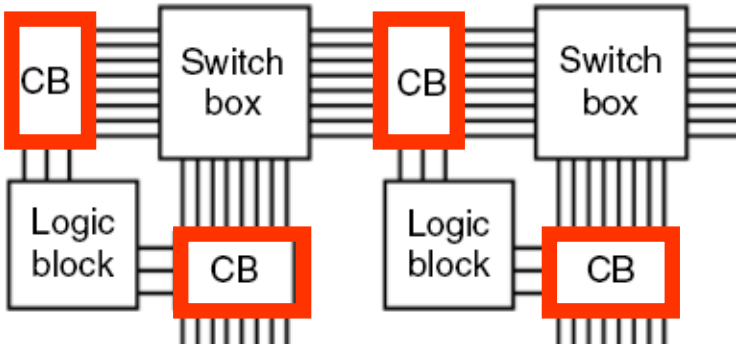
Reconfigurable Interconnects

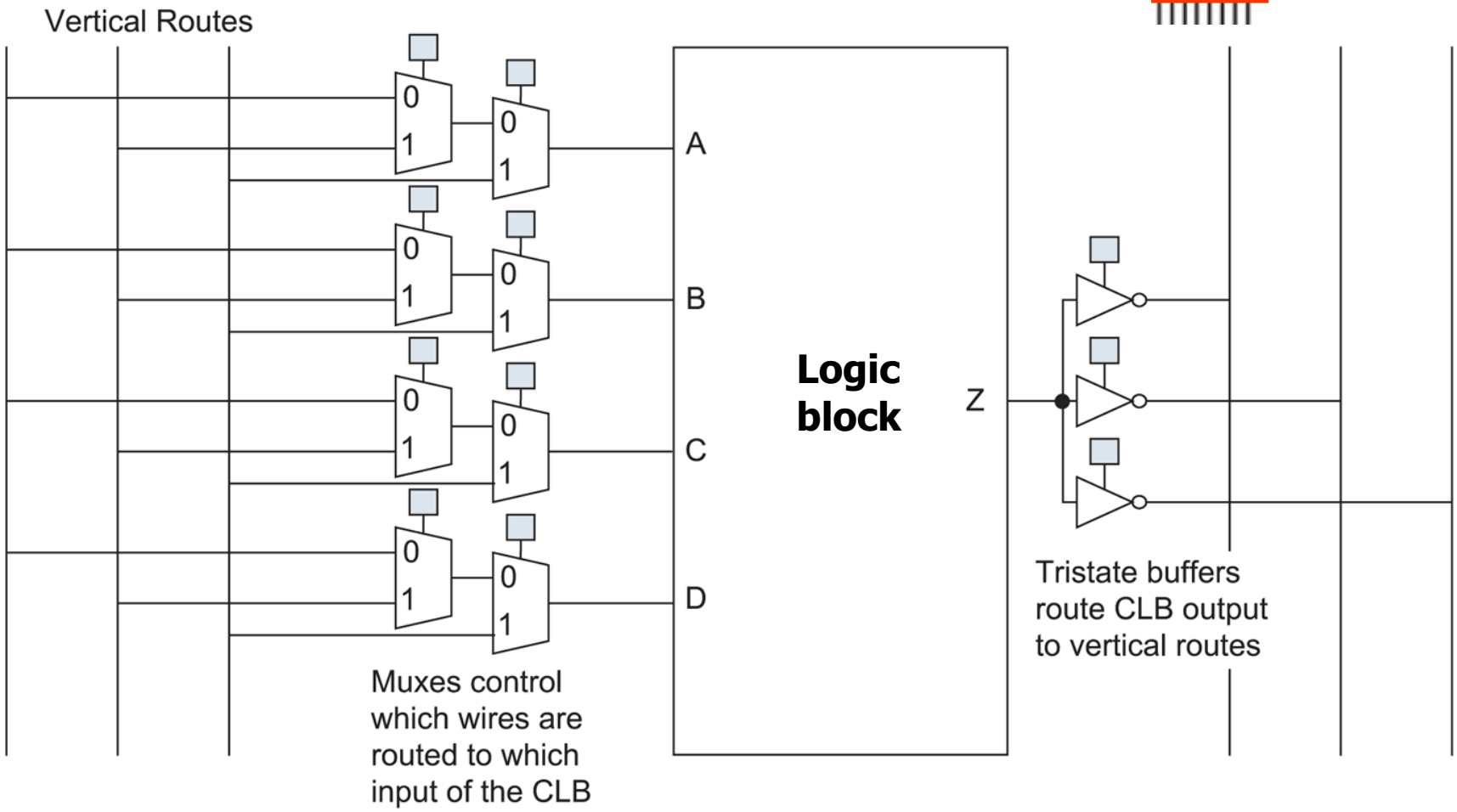
- Connect computational node output(s) to the inputs of another computational node
- **Interconnects more crucial than logic?**
  - Wire delay grows quadratically as a function of its length → avoid using long wires unless necessary
  - Technology scaling reduces device (logic) delay but increases wire delay
  - Area? Power?

Island-style FPGA

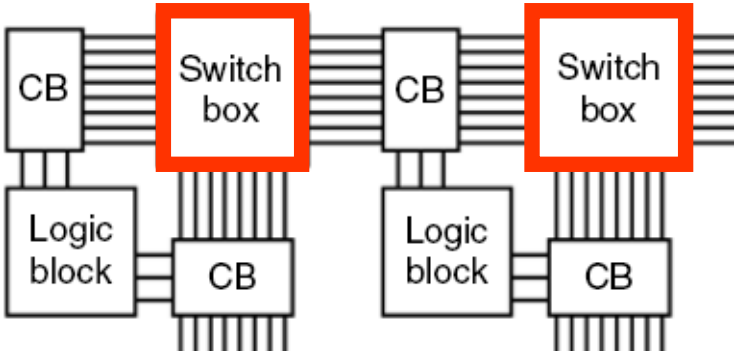


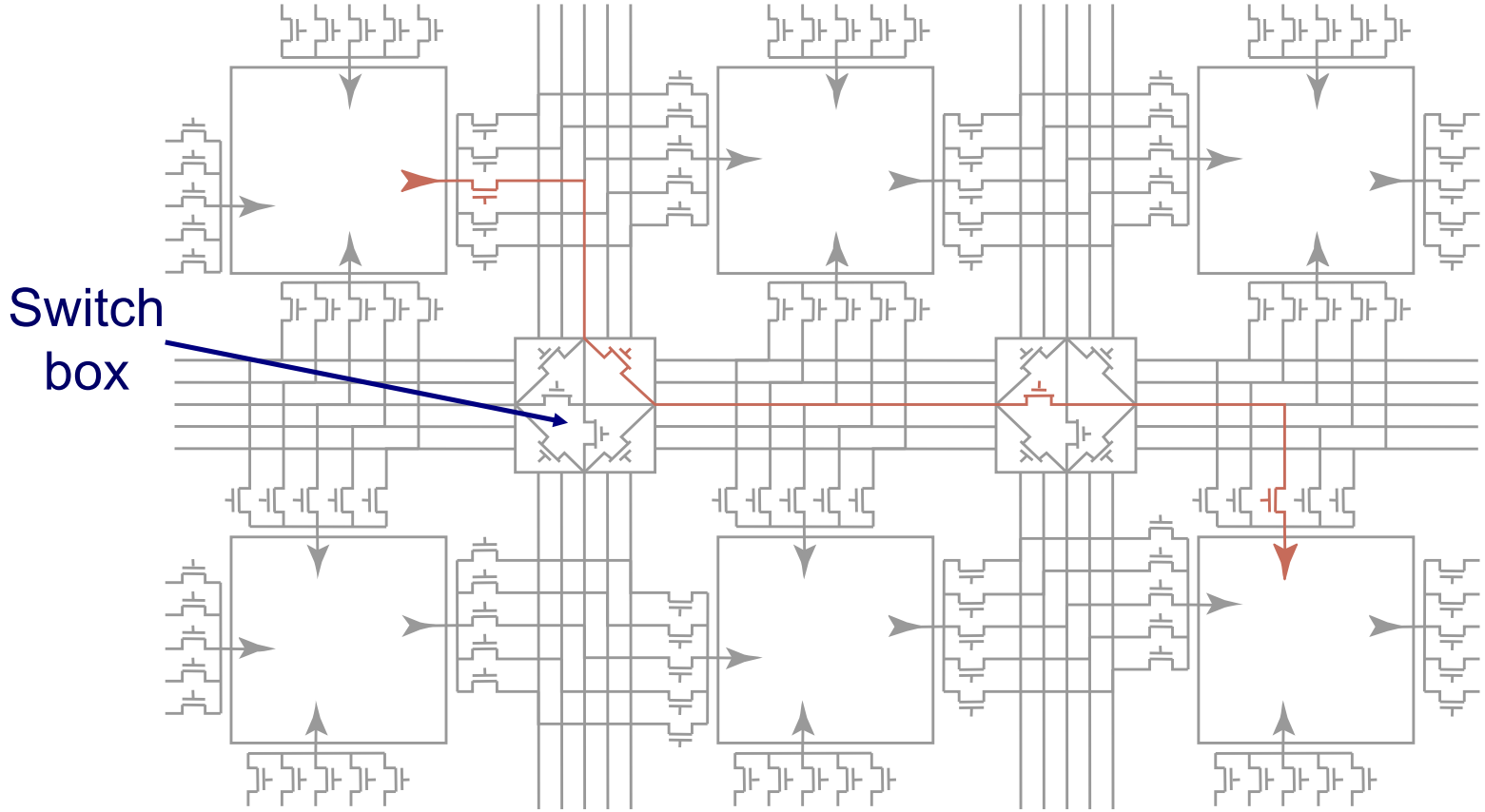
Connection


CB
Switch box
CB
Switch box
Logic block
CB
Logic block
CB

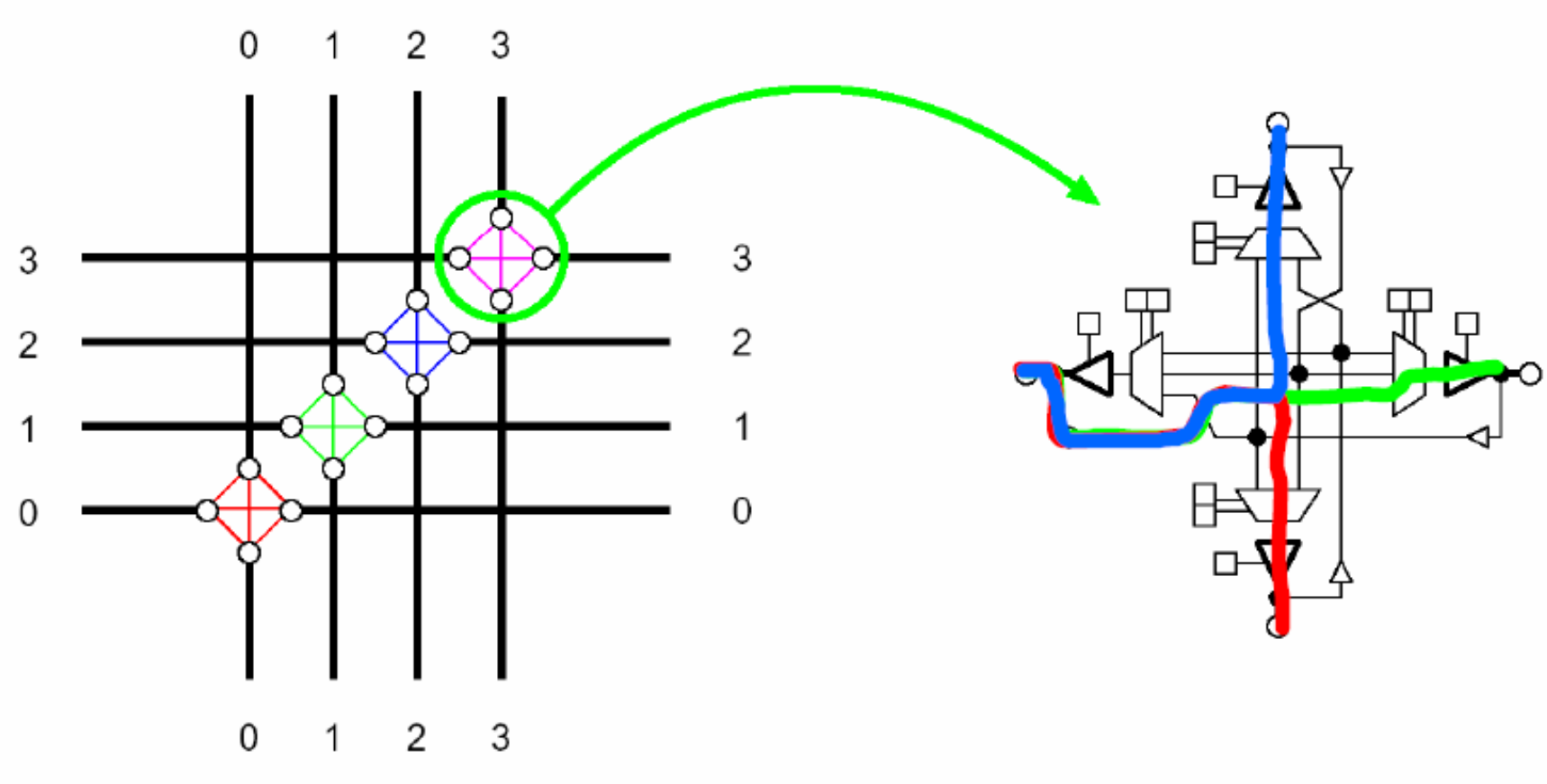

Vertical Routes
A
B
C
D
Logic block
Z
Muxes control which wires are routed to which input of the CLB
Tristate buffers route CLB output to vertical routes

Switch blocks



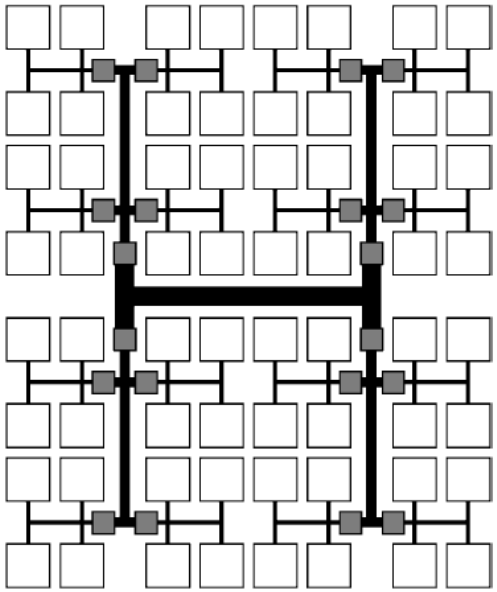


Bidirectional switch details



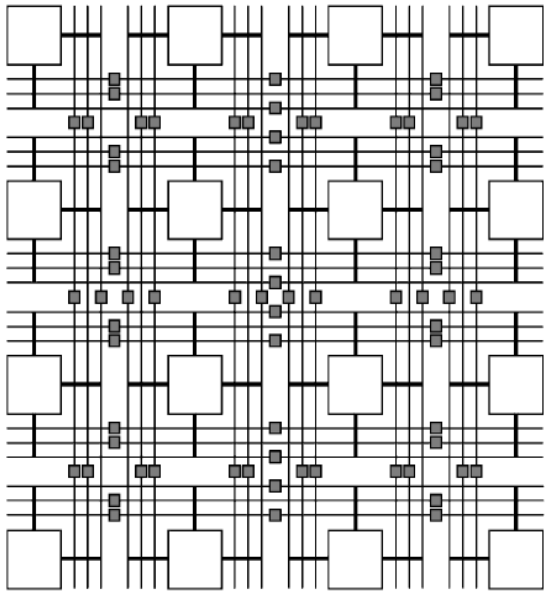
Hierarchical and Segmented Interconnects

hierarchical Interconnects



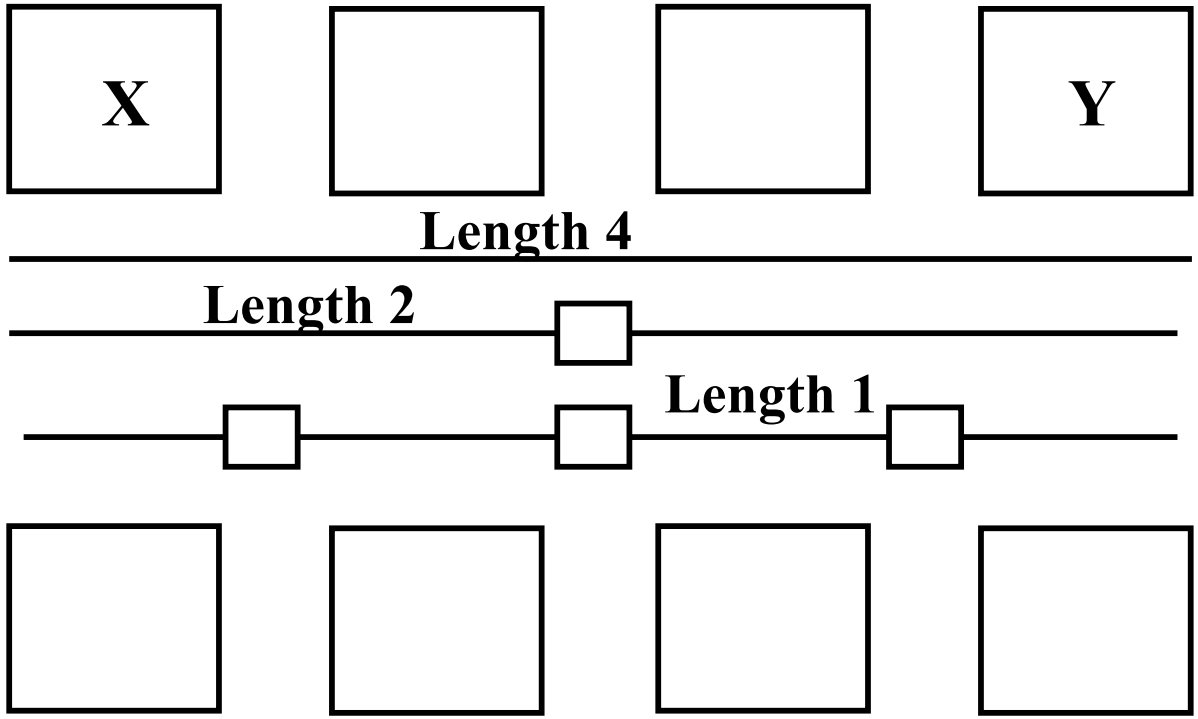
- Routing within a “group” of logic blocks occur at the local level
- Longer hierarchical wires connect different groups

segmented Interconnects



- Short wires accommodate local traffic
- Short wires can be connected together using switch boxes to emulate longer wires
- Also contain long wires to allow efficient communication without passing through switches

Segmentation



- Segmentation distribution: how many of each length?
- Longer length
  - Better performance? ☺
  - Reduced routability? ☹

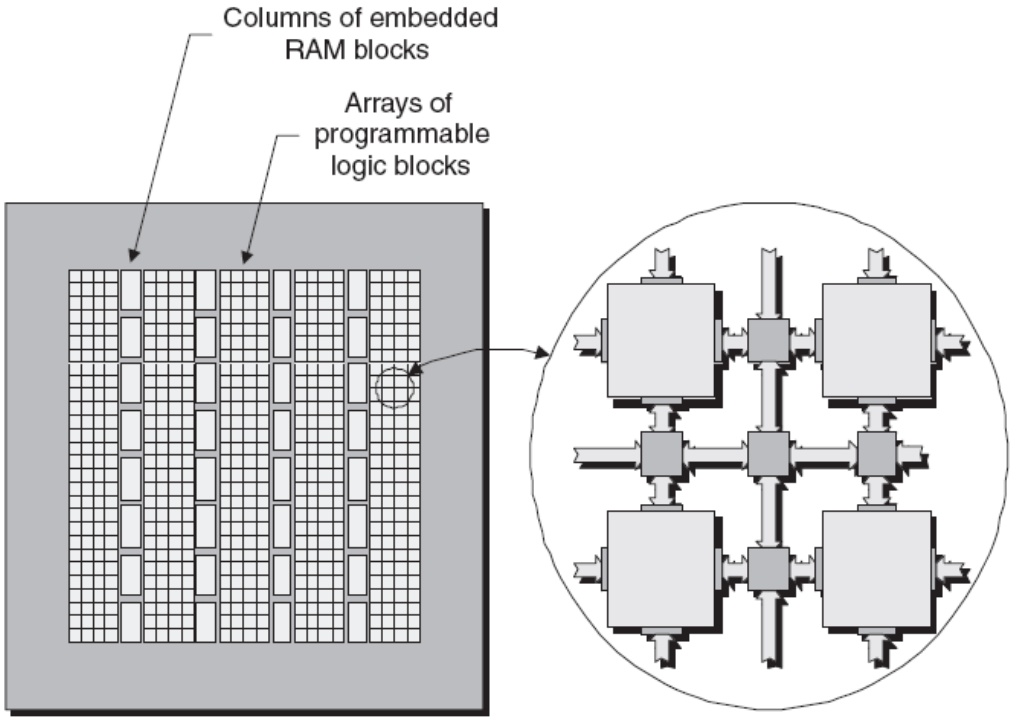
Specialized Modules

Heterogeneous reconfigurable environments

- Reconfigurable fabric might contain non-reconfigurable elements that interface to the logic blocks through the reconfigurable interconnect fabric
- Examples:
  - Embedded memory
  - Embedded multipliers, adders, MAC
  - Embedded processors (e.g. PPC)

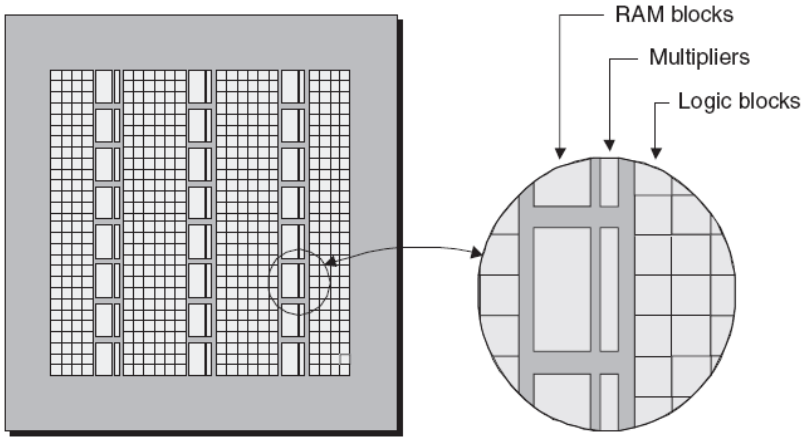
Embedded memory blocks

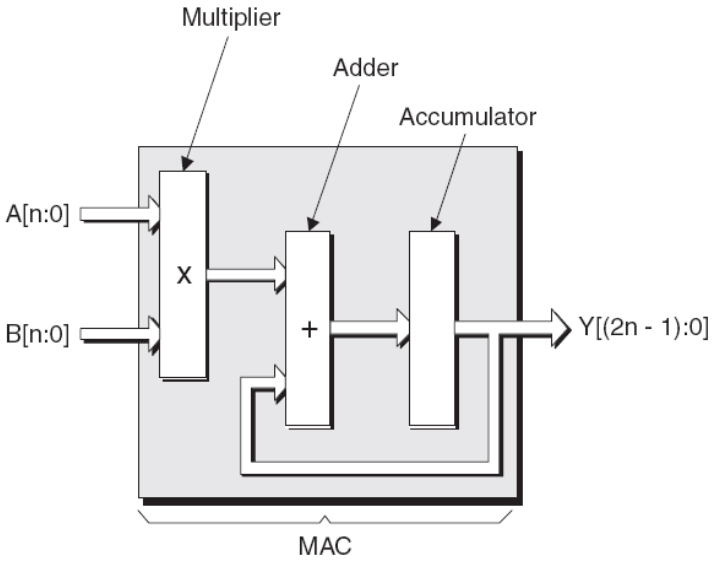
- Costly to implement memory with configurable logic blocks → add hard chunks of RAM blocks
- Position/size vary depending on the FPGA device. Size varies from few Kbits (or tens of Kbits) per block RAM
- Each block can be used independently or combined to form larger RAM blocks
- Could be single or dual-port RAMs
- Reconfigurable dimensions, e.g.:
  - 16k entries of 1 bit to
    512 entries of 32 bits



Embedded multipliers and adders

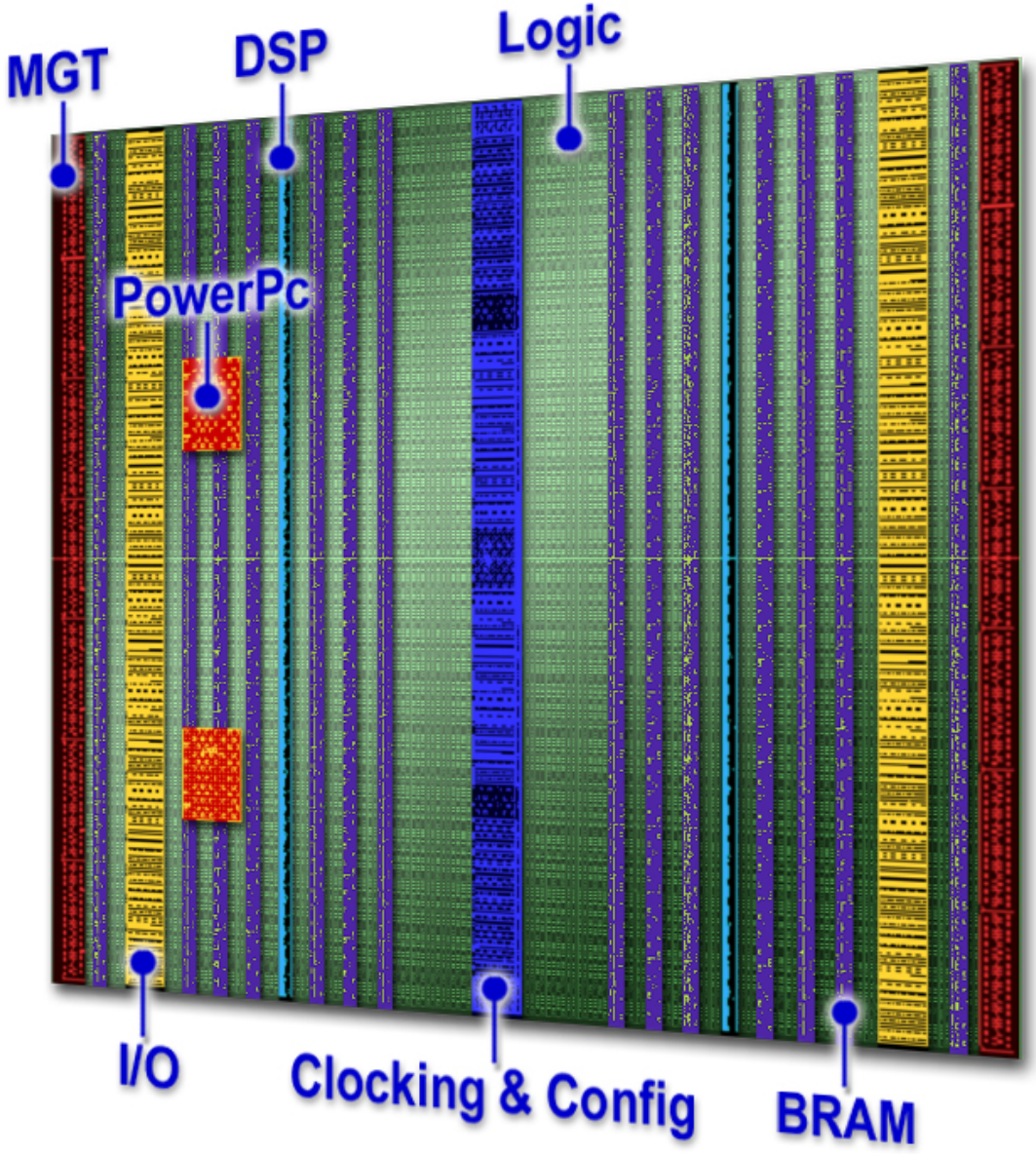
- Multipliers are inherently slow if implemented by connecting a large number of programmable logic blocks → add hard-wired multiplier blocks
- Typically located close to the embedded RAM blocks
- Some FPGA use Multiply-And-Accumulate (MAC) blocks (useful in DSP applications)





…and more

- Embbedded processors
- Rocket I/Os
- Etc.

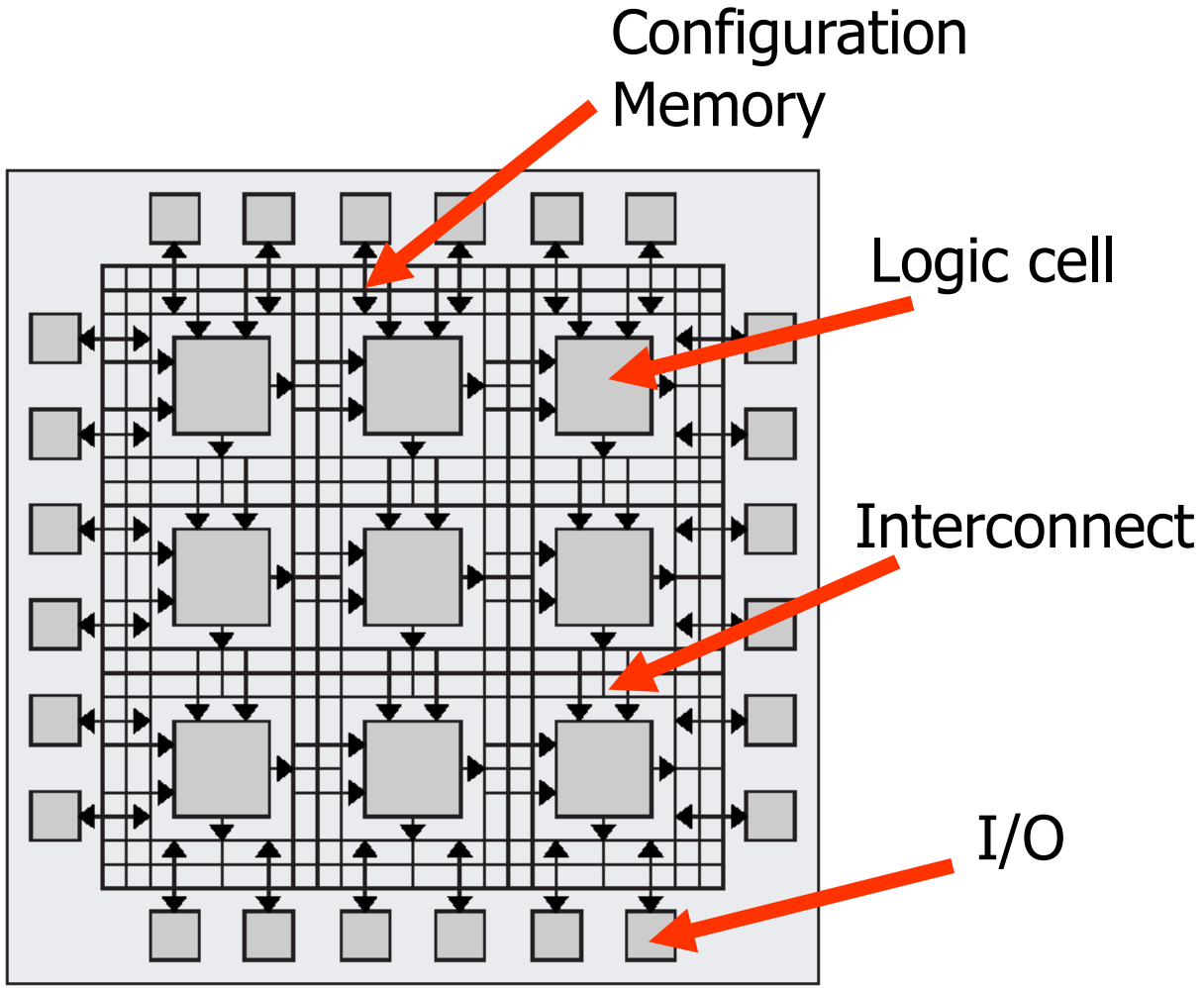


Virtex-7

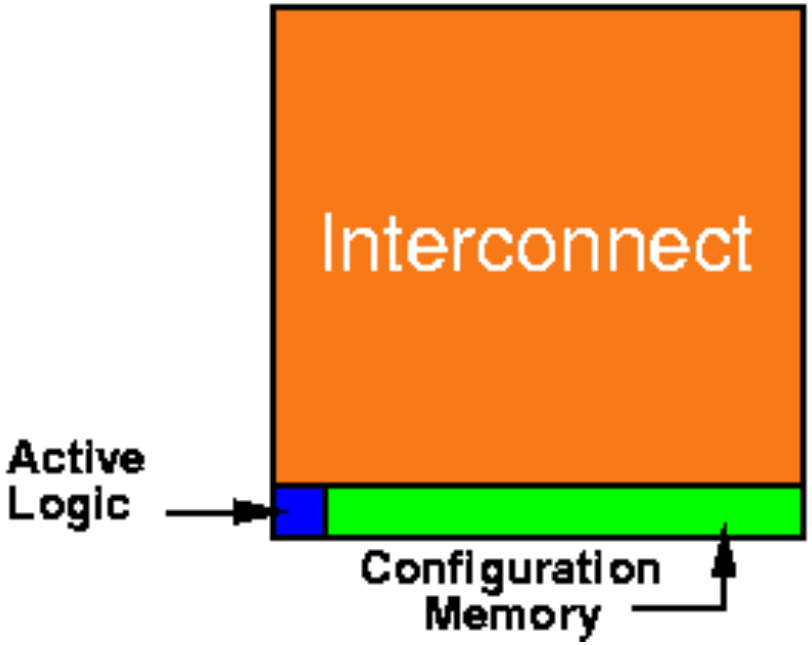
- **28-nm technology!**
- 6-input LUT configured also as 2x5-input
- 4 LC= 1 Slice
- 2 Slices = 1 CLB (Conf. Logic Block)
- Up to 68 Mbit (8,5 MByte) on-chip RAM

Maximum Capability	Artix-7 Family	Kintex-7 Family	Virtex-7 Family
Logic Cells	215K	478K	1,955K
Block RAM(1)	13 Mb	34 Mb	68 Mb
DSP Slices	740	1,920	3,600
Peak DSP Performance(2)	929 GMAC/s	2,845 GMAC/s	5,335 GMAC/s
Transceivers	16	32	96
Peak Transceiver Speed	6.6 Gb/s	12.5 Gb/s	28.05 Gb/s
Peak Serial Bandwidth (Full Duplex)	211 Gb/s	800 Gb/s	2,784 Gb/s
PCIe Interface	x4 Gen2	x8 Gen2	x8 Gen3
Memory Interface	1,066 Mb/s	1,866 Mb/s	1,866 Mb/s
I/O Pins	500	500	1,200
I/O Voltage	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V
Package Options	Low-Cost, Wire-Bond, Lidless Flip-Chip	Low-Cost, Lidless Flip-Chip and High-Performance Flip-Chip	Highest Performance Flip-Chip

What about Area, Delay, Power?

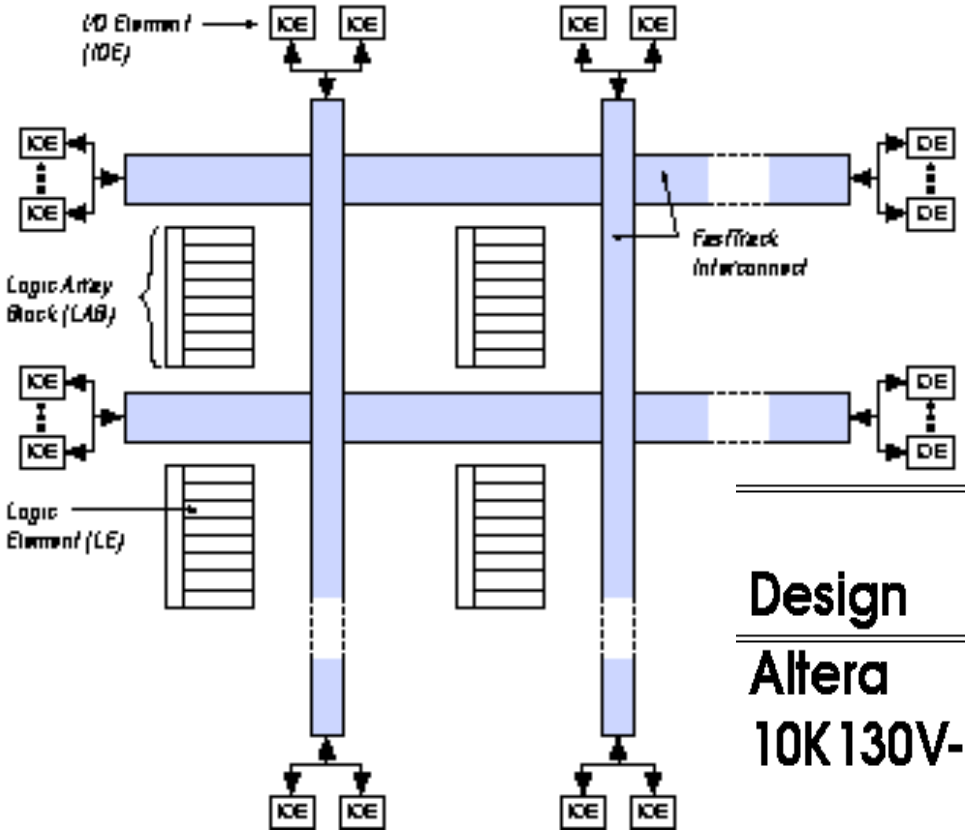


Area



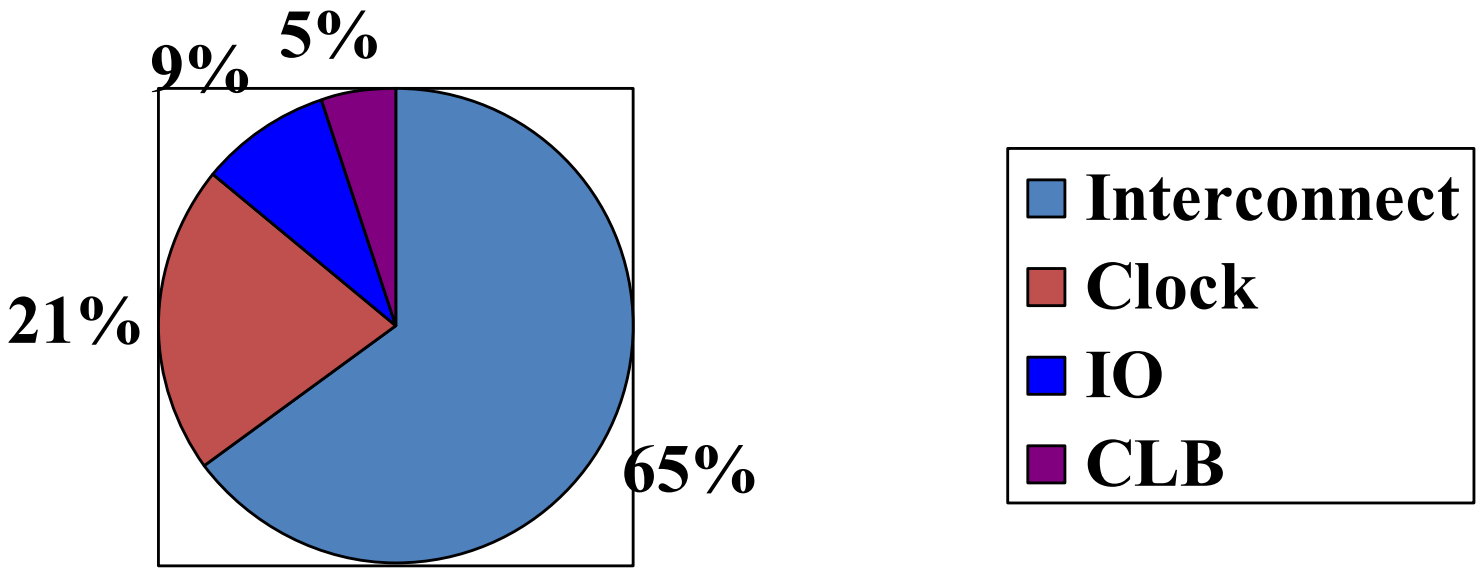
Function	Area (λ^2)	
LUT MUX + ff	20K	(generous, closer to 10K)
Programming Memory	80K	(240K typical unencoded)
Interconnect	700K	(for $N_p = 2048$)

Delay



Design	Path	Total Delay	LUT Delay	Inter. %
Altera 10K130V-2	LUT-local-LUT	2.5 ns	2.1 ns	16%
	LUT-row-local-LUT	6.6 ns	2.1 ns	68%
	LUT-column-local-LUT	11.1 ns	2.1 ns	81%
	LUT-row-column-local-LUT	15.6 ns	2.1 ns	87%
	LUT-row-fanout-local-LUT (fanout)	28 ns	2.1 ns	90%

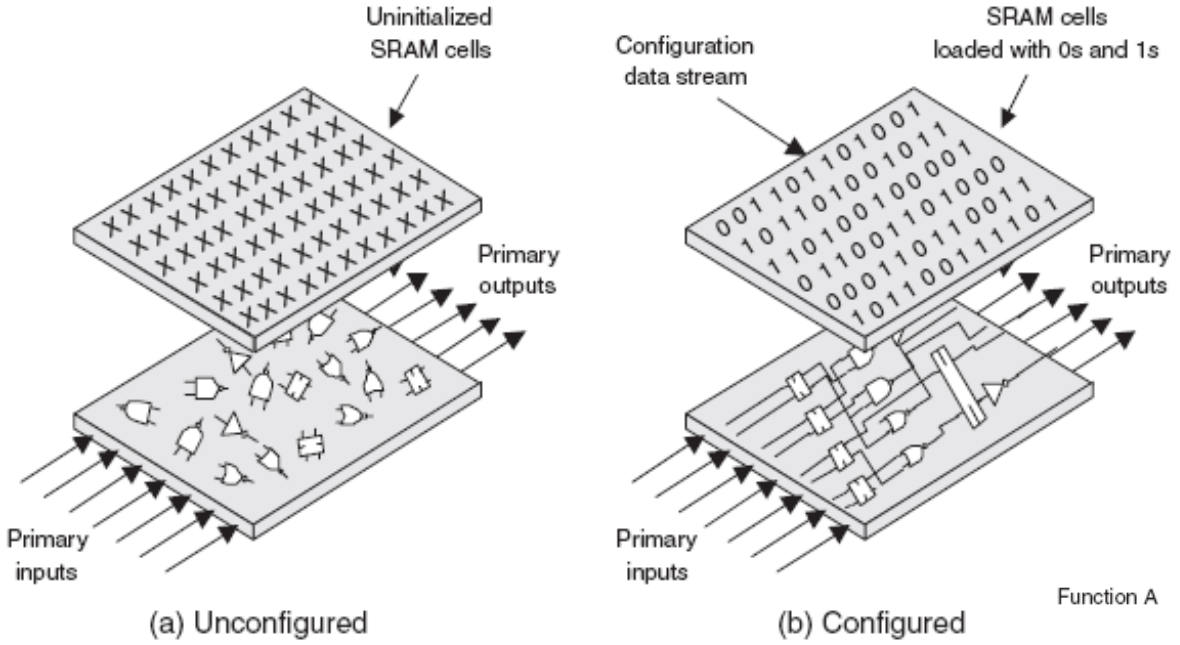
Power



XC4003A data from Eric Kusse (UCB MS 1997)

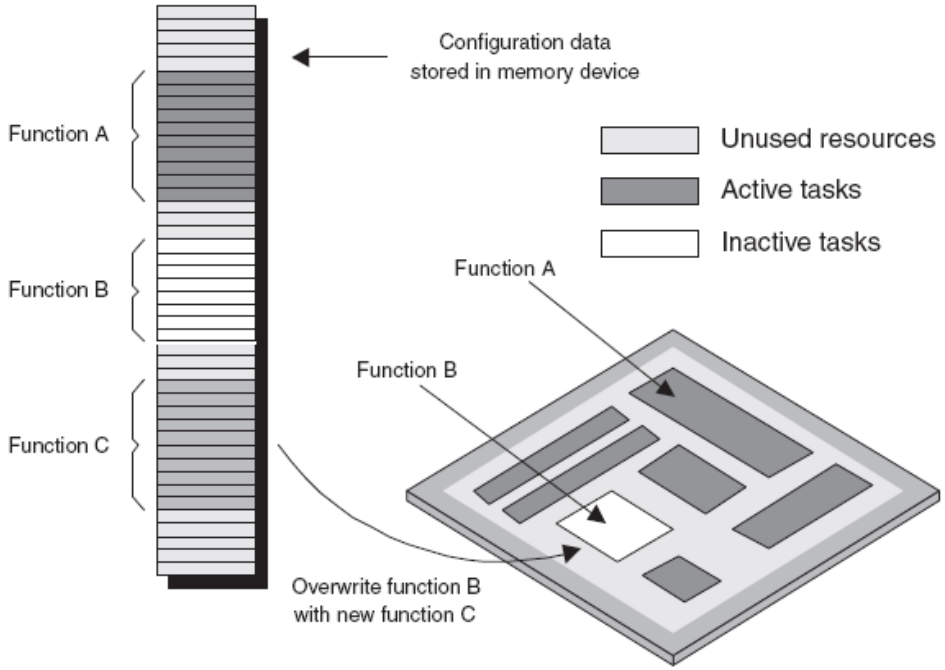
Reconfiguration

Reconfiguring FPGAs

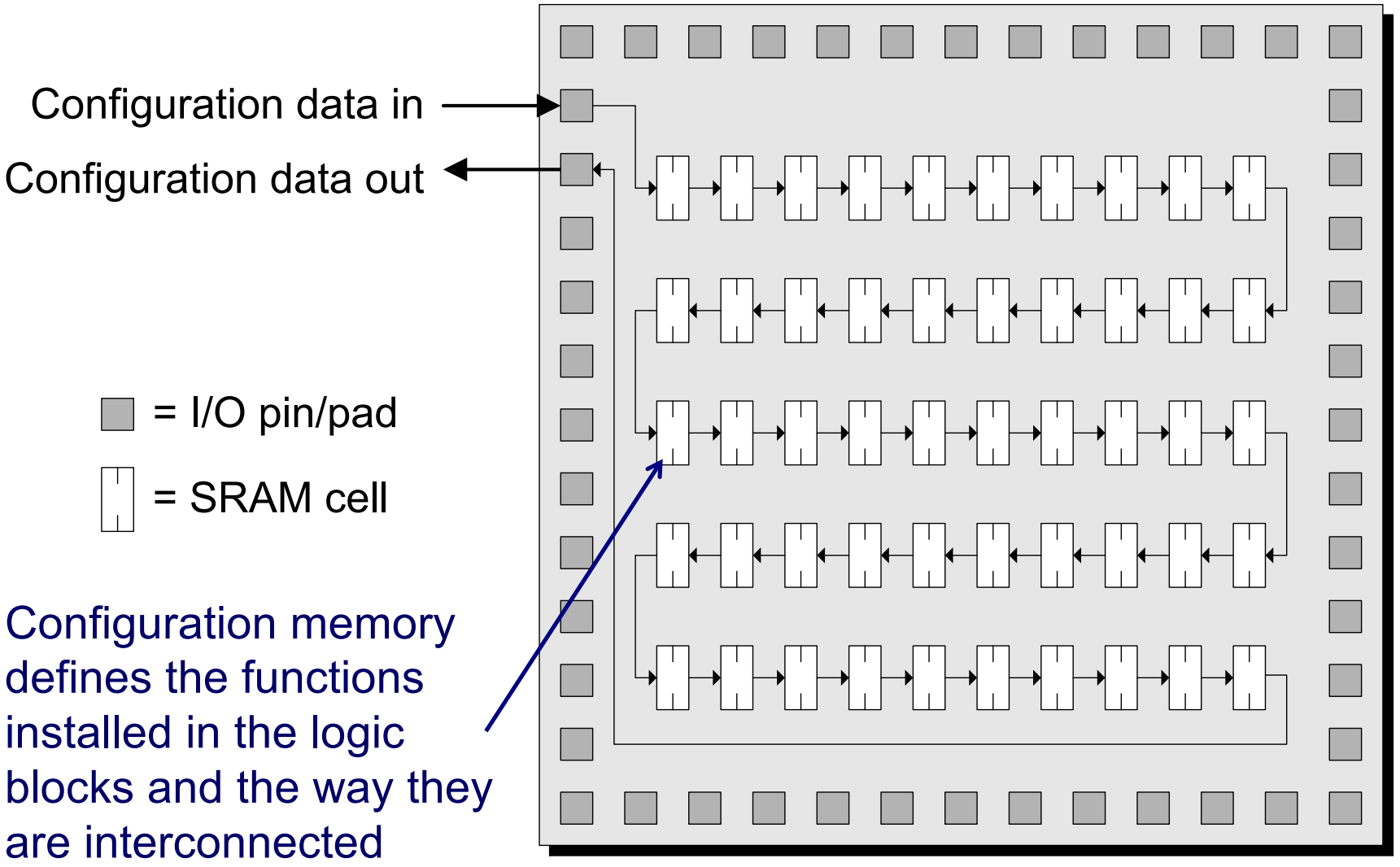


FPGAs can be dynamically reconfigured before runtime or during runtime

- full
- partial



Reconfiguring an FPGA



Reconfiguration Memory

Anti-fuse

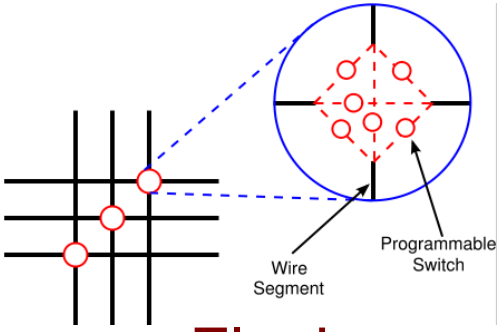
by default is OFF; when programmed it is ON (creating a short-circuit between the endpoints).

Advantages:

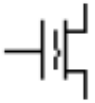
- negligible delay
- small area overhead
- no soft-errors and bit flips

Disadvantages:

- not really reconfigurable; one time programmable



Flash



Advantages:

- programming not lost when device is turned off.
- Fewer transistors than SRAM
- Lower power than SRAM

Disadvantages:

- Limited writes (~milions)
- Slower writes than SRAM
- Higher voltage than circuits

SRAM

SRAM bit cell stores the programmability of the device

Advantages:

- can be reconfigured quickly and as repeatedly as required
- no special fabrication steps

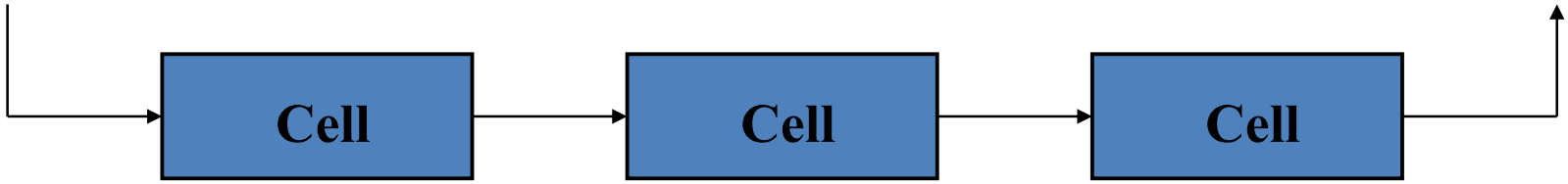
Disadvantages:

- takes more area, and power
- loses charge when turned off

Coarse-grain Reconfigurable Devices

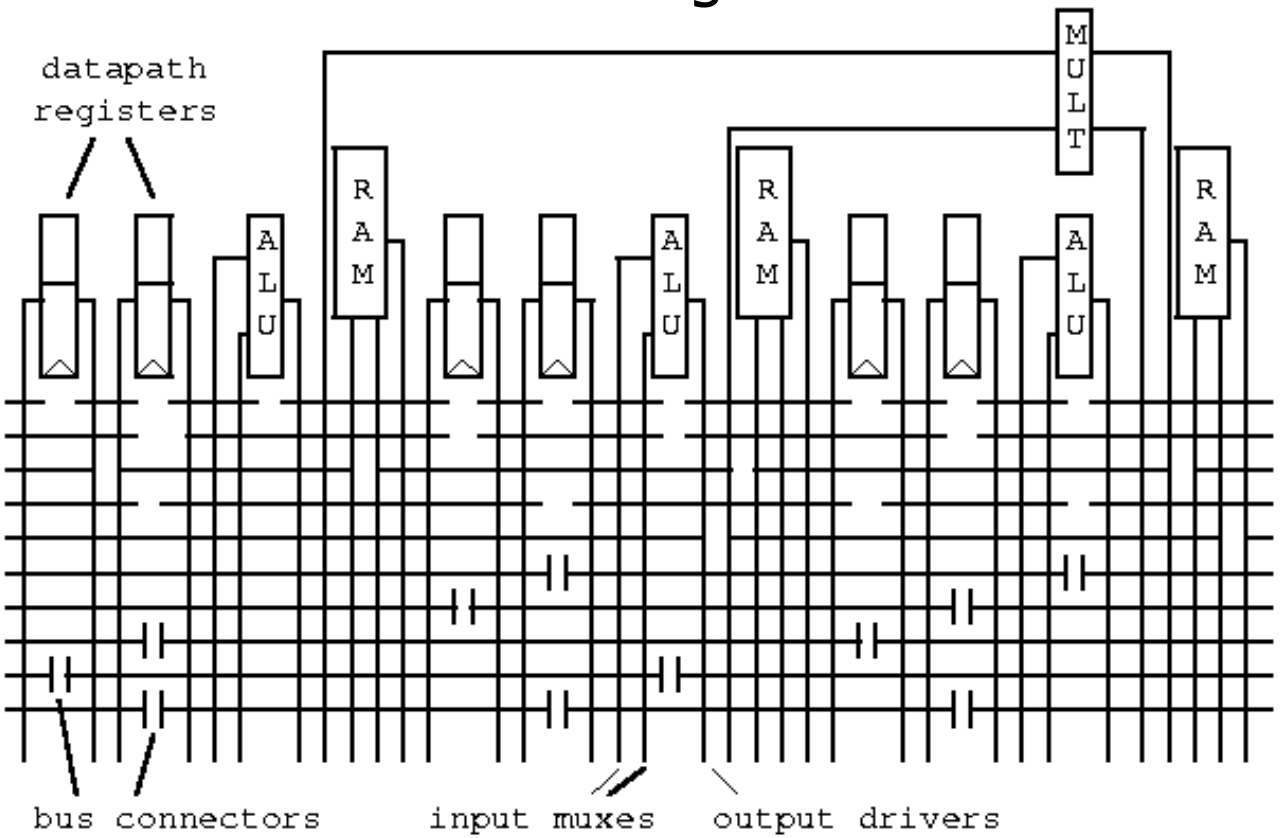
Rapid

- Reconfigurable Pipeline Datapath
- Ebeling – University of Washington
- Uses hard-coded functional units (ALU, Memory, multiply)
- Good for signal processing
- Linear array of processing elements.

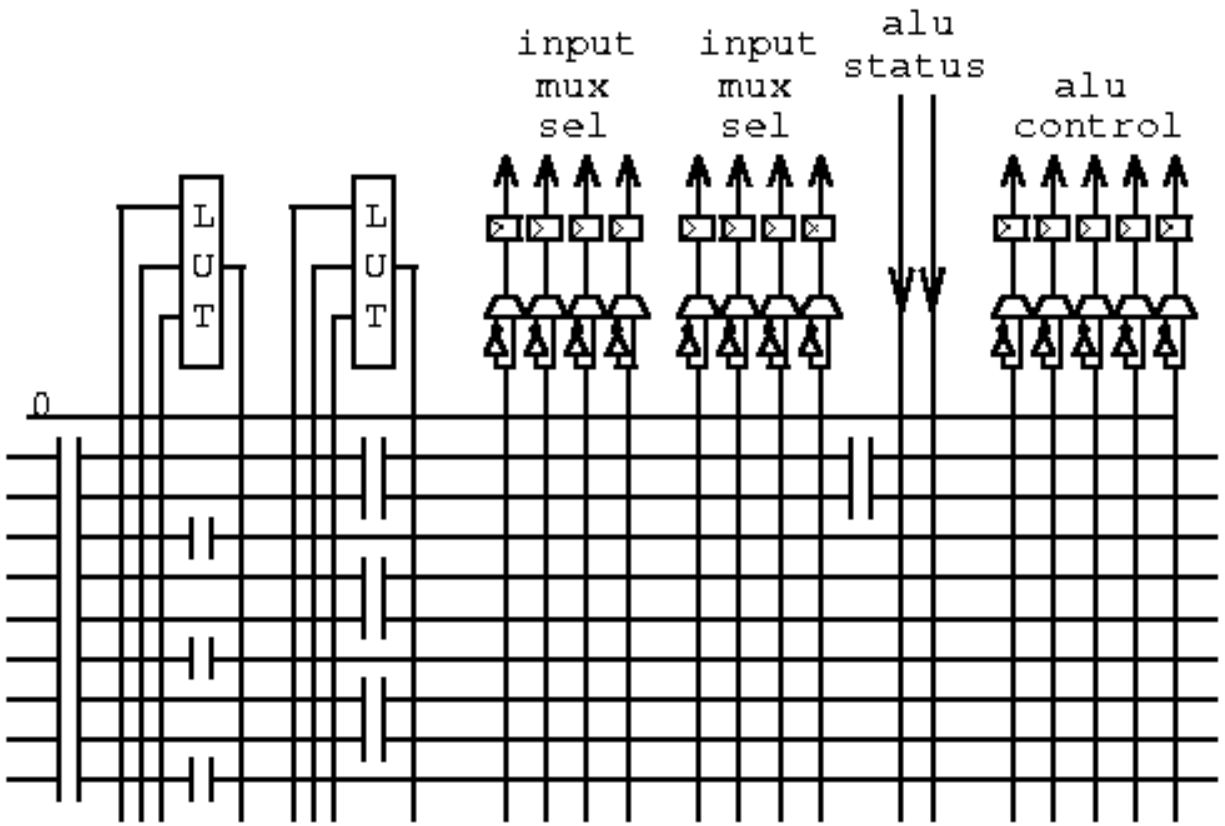


Rapid Datapath

- Segmented linear architecture
- All RAMs and ALUs are pipelined
- Bus connectors also contain registers



Rapid Control Path



- In addition to static control, control pipeline allows dynamic control.
- LUTs provide simple programmability.
- Cells can be chained together to form continuous pipe.

Fine-grain vs. Coarse-grain Reconfigurable devices

- **Fine-grain:** accomodate arbitrary functionality at what Cost?
  - Circuit area
  - Power
  - Op. Frequency
  - Complex configuration
- **Coarse-grain** RC to reduce the above costs
  - Less flexible, but...
  - More area efficient,
  - Less power hungry
  - Faster

- Tradeoff: **Cost/flexibility**
  - Cost= Resources, Speed, Power, Complexity of building a system, etc.

What is Reconfigurable?

- Is an ALU reconfigurable or Re-Programmable?
- Are the Coarse-Grain Reconfigurable Devices actually Reconfigurable?

Quiz 13-1

http://m.socrative.com/student/#joinRoom

room number: 713113

- Q1: How many different gates can a 4-input LUT accommodate?
  - 1, 4, 8, 16
- Q2: Is an ALU is reconfigurable?
- Q3: Most of the area in an FPGA chip is occupied due to:
  - (a) logic cells, (b) Interconnects, (c) configuration memory
- Q4: Put the FPGA parts below in order starting from the most power-hungry towards the less power-hungry:
  - (a) logic cells, (b) Interconnects, (c) clock, (d) IO
- Q5: An FPGA with configuration memory based on __(a) keeps its configuration even after power off. An FPGA with configuration memory based on ____(b) can be configured only once.
  - Anti-fuse, flash, SRAM

Reconfigurable vs. Re-Programmable

- Is an ALU reconfigurable or Re-Programmable?
- Are the Coarse-Grain Reconfigurable Devices actually Reconfigurable?

Reconfigurable Computing is Computing via post-fabrication, spatially programmed connection of processing elements.

– Andre DeHon

The difference between reconfigurable and reprogrammable is that the first can implement an arbitrary number of functions directly in hardware, while the second supports only a predefined –during fabrication- finite number of functions.

– Stamatis Vassiliadis

Summary

- Why Reconfigurable?
- Reconfigurable Devices:
  - Fine-grain Reconfigurable
    - Logic
    - Interconnects
    - Specialized Units
    - Reconfiguration Memory
  - Coarse-grain
    - Better speed, Less Area/power cost, …less Flexible
- What is Reconfigurable?

- This lecture is not included in the book at all
- You can check Xilinx Spartan6 Datasheet
  - will be used in Labs 6-7
  - http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

- Next Lecture 12:
  - Arithmetic Units