FPGA Architecture

Nachiket Kapre
nachiket@uwaterloo.ca

Outline

- Introduction to FPGAs
- Recent attention to FPGA-based computing
- Introduction to FPGAs
  - Logic Architecture
  - Interconnect Architecture

What are FPGAs?

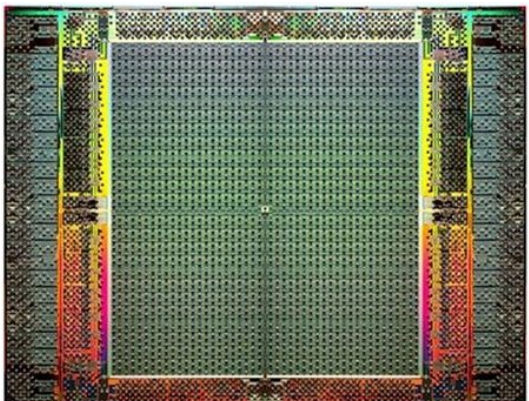


Figure: Altera Stratix IV FPGA. http://archive.eetasia.com/www.eetasia.com/ART_8800712284_480100_NT_3b7e2d3a.HTM

- **F**ield **P**rogrammable **G**ate **A**rrays
- Reconfigurable circuits programmed at boot-time
- Customized hardware tailored for your needs
- Clever use of wires to support *spatial communication*.

What are FPGAs?

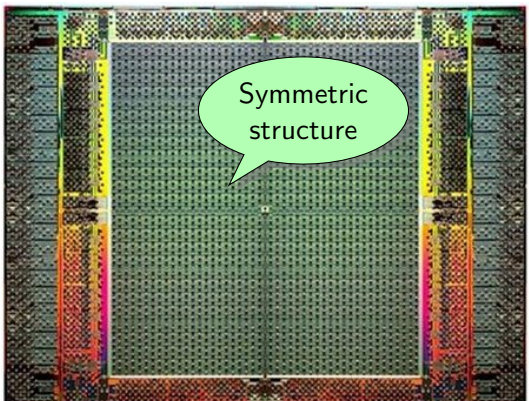
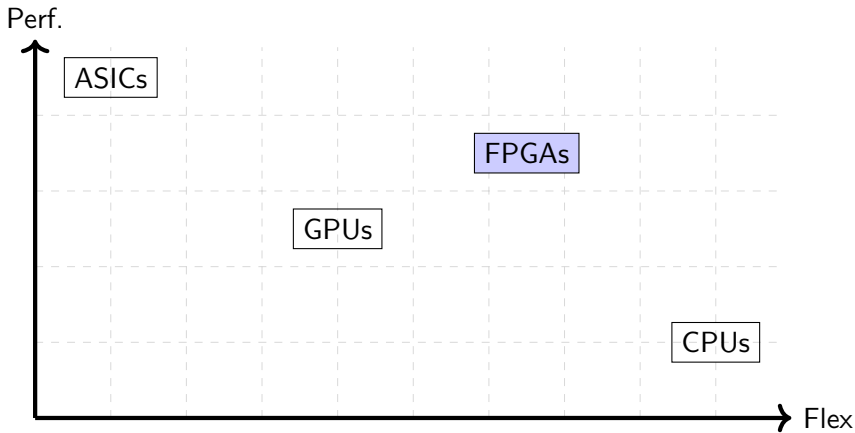


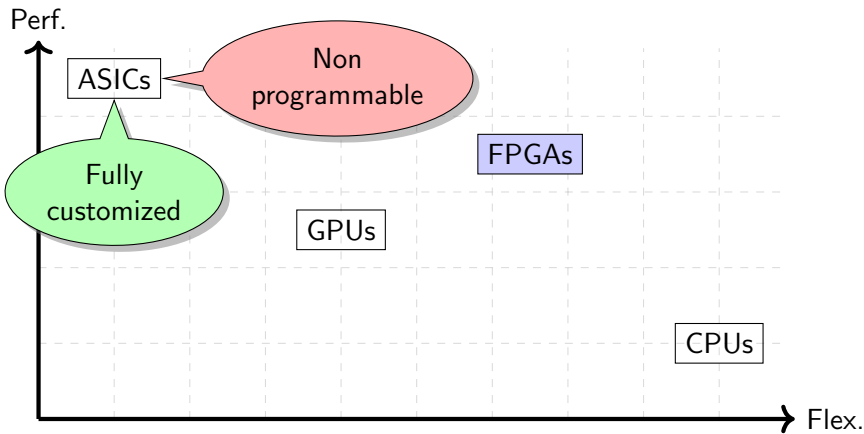
Figure: Altera Stratix IV FPGA. http://archive.eetasia.com/www.eetasia.com/ART_8800712284_480100_NT_3b7e2d3a.HTM

- **F**ield **P**rogrammable **G**ate **A**rrays
- Reconfigurable circuits programmed at boot-time
- Customized hardware tailored for your needs
- Clever use of wires to support *spatial communication*.

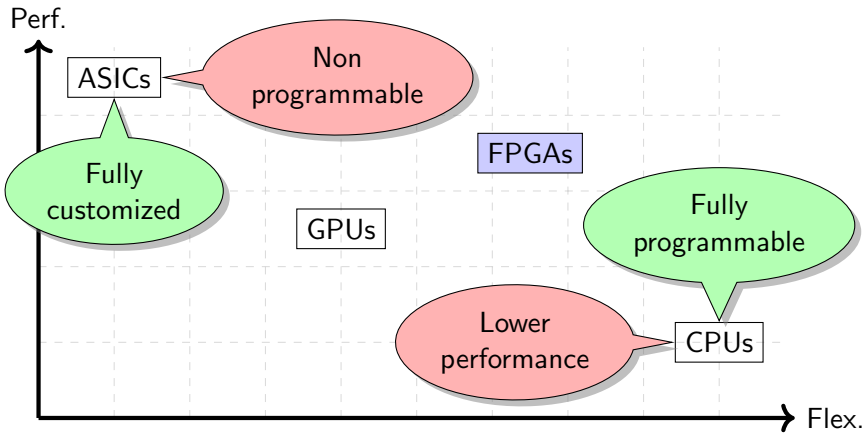
FPGAs vs. Rest of the World (Performance)



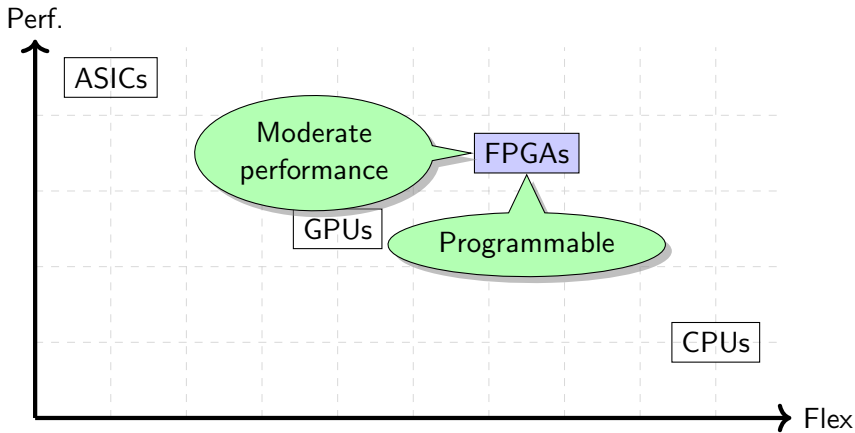
FPGAs vs. Rest of the World (Performance)



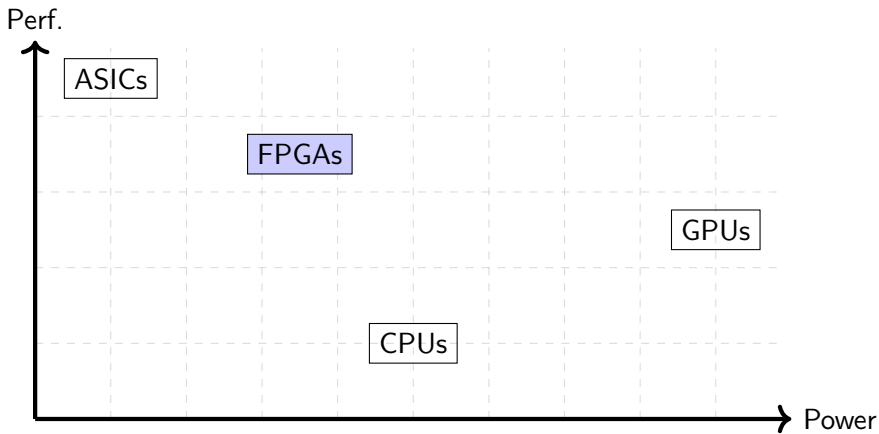
FPGAs vs. Rest of the World (Performance)



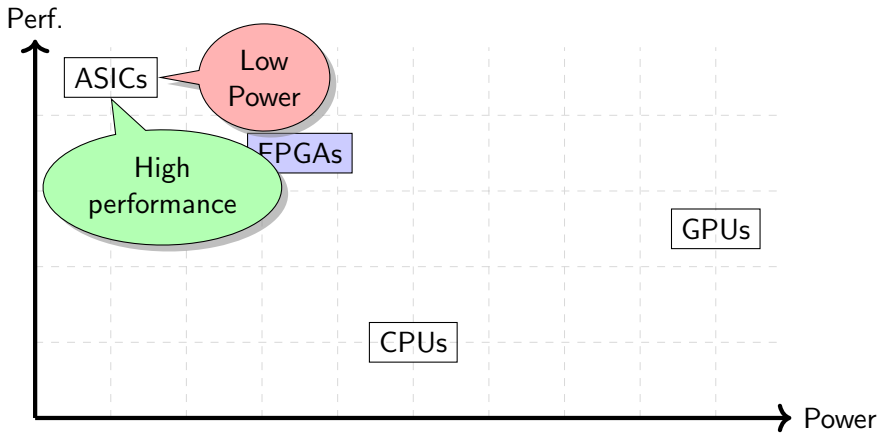
FPGAs vs. Rest of the World (Performance)



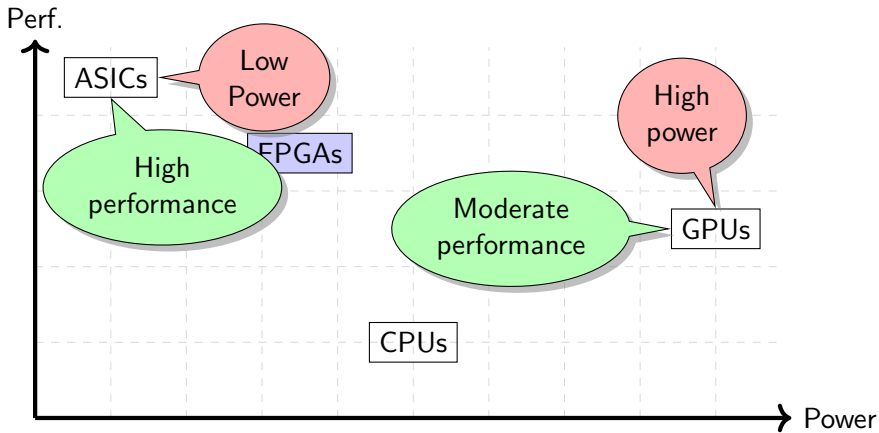
FPGAs vs. Rest of the World (Power)



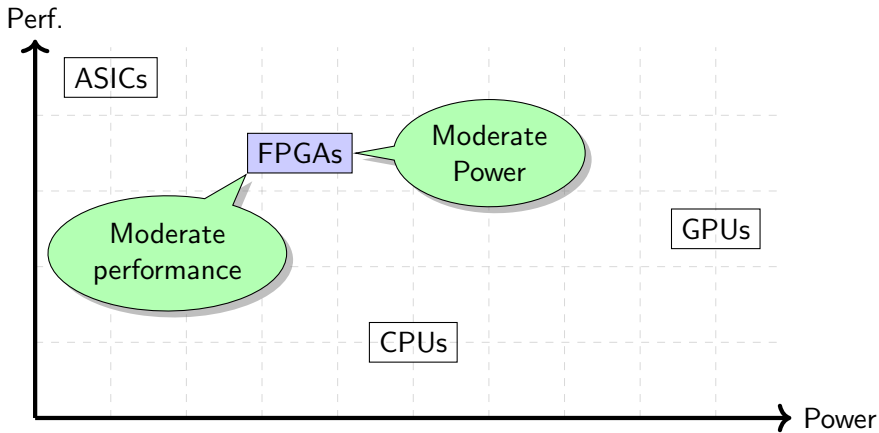
FPGAs vs. Rest of the World (Power)



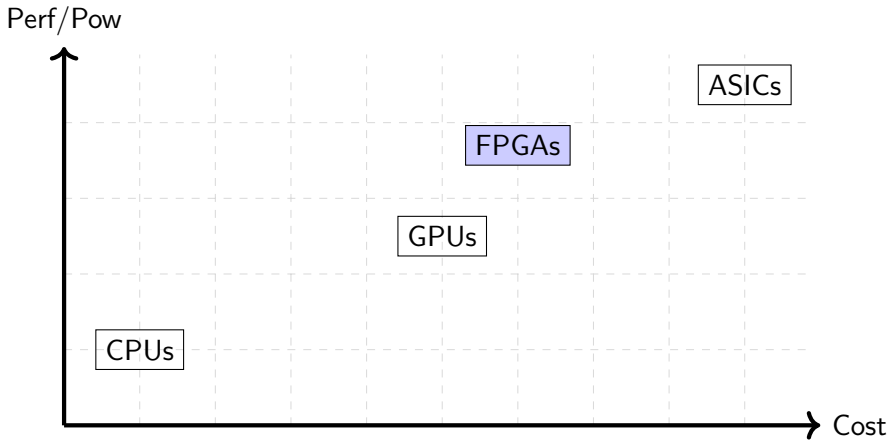
FPGAs vs. Rest of the World (Power)



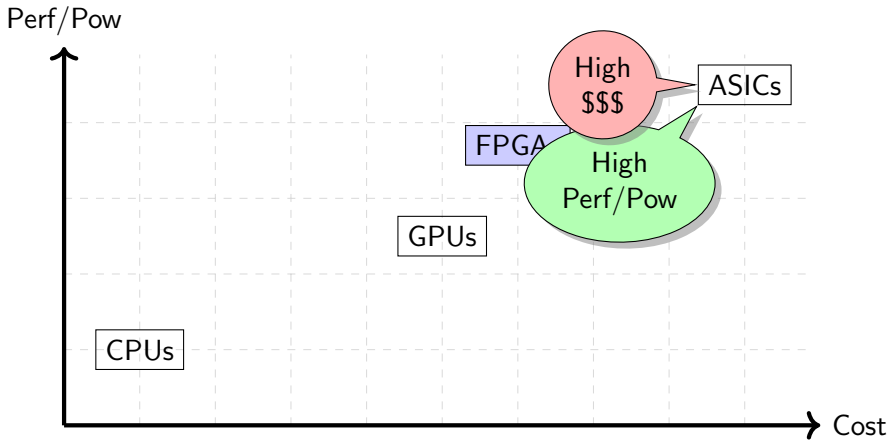
FPGAs vs. Rest of the World (Power)



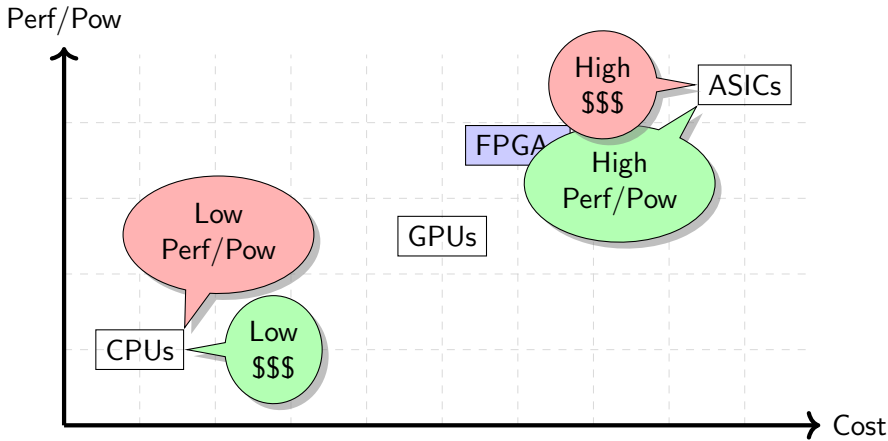
FPGAs vs. Rest of the World ($ Cost)



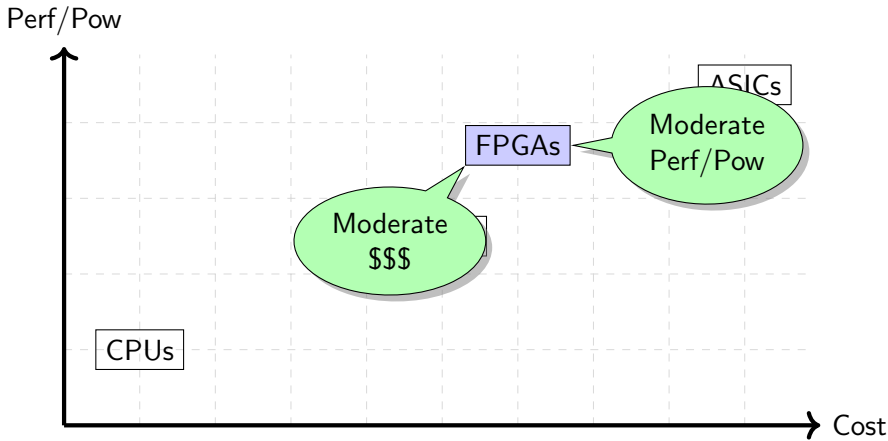
FPGAs vs. Rest of the World ($ Cost)



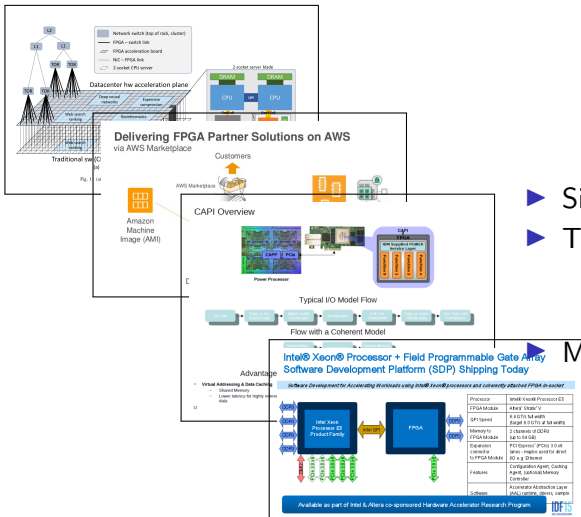
FPGAs vs. Rest of the World ($ Cost)



FPGAs vs. Rest of the World ($ Cost)



Why do FPGAs matter now?



- Significant industrial interest!
- Two factors:
  - **Cloud Computing** (energy)
  - **Machine Learning** (new function)
- Microsoft, Amazon, IBM, Intel

Microsoft Cloud-Scale Architecture (MICRO 2016)

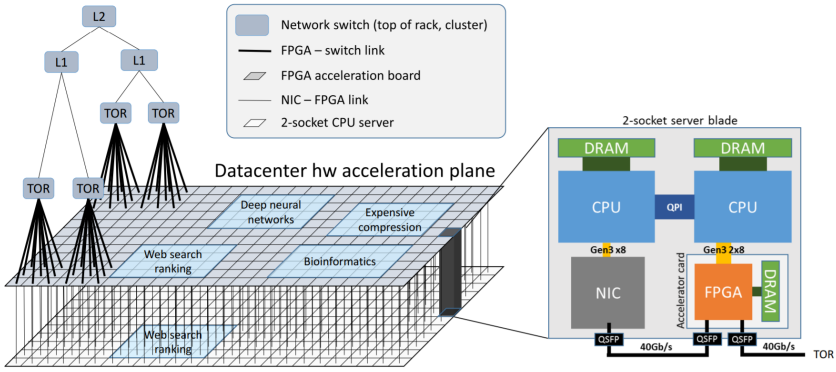


Fig. 1. (a) Decoupled Programmable Hardware Plane, (b) Server + FPGA schematic.

Figure: Catapult FPGA server `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/10/Cloud-Scale-Acceleration-Architecture.pdf`

Microsoft Cloud-Scale Architecture (MICRO 2016)

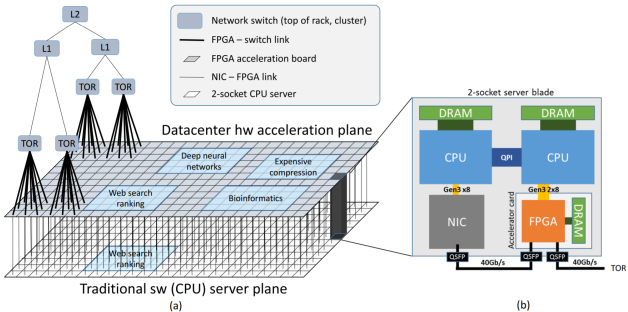


Fig. 1. (a) Decoupled Programmable Hardware Plane, (b) Server + FPGA schematic.

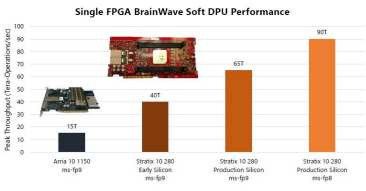
Figure: Catapult FPGA server `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/10/Cloud-Scale-Acceleration-Architecture.pdf`

- FPGAs as accel. + network pkt processors
- Stratix V D5 FPGA@172K ALMs (40% Shell)

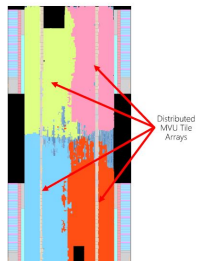
- Enable large FPGAs clusters to work together
- Web workloads
  - Several independent queries
  - Fast latency of response required
- Packet processing
  - Stream of network data (encryption)
  - Throughput processing required
- 250K machines $<20\mu$s
- Latency-optimized

Microsoft BrainWave AI Accelerator (ISCA 2018)

BrainWave Soft DPU Performance



Arria 10 1150 (20nm)	Stratix 10 280 Early Silicon (14nm)
ms-fp9	ms-fp9
316K ALMs (74%)	858K ALMs (92%)
1442 DSPs (95%)	5,760 DSPs (100%)
2,564 M20Ks (95%)	8,151 M20Ks (70%)
160 GOPS/W	320 GOPS/W → 720 GOPS/W (production)



BrainWave Soft DPU
Floorplan on Stratix 10 280

Figure: Microsoft BrainWave FPGA Accelerator https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/

- Hundreds, Thousdands of FPGAs in Azure can outperform Google TPU
- Arria10/Stratix 10 cutting-edge Intel FPGAs

- Build simple matrix-vector multiplication units
- Keep neural network weights inside the chip
  - Power and performance penalty for off-chip access
- Scale design to hundreds/thousands of chips
  - Adapt to changing requirements of AI workloads

Xilinx ARM CPU+FPGA chip

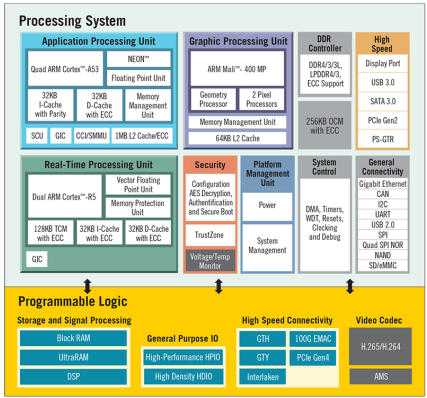


Figure: Hybrid ARM+FPGA chip `https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html`

- Heterogenous chip with ARM CPU, real-time CPUs, GPUs + FPGAs
  - Host code can run on ARM Cortex A53 quad-core CPU
  - Graphics code runs on ARM Mali 400 MP (OpenCL)
  - Real-time ARM Cortex R5 (floating-point)
- Rich on-chip connectivity solutions (AXI)
  - AXI-HP for tput.
  - ACP for coherence

How should you prepare?

- Understand when to use an FPGA. Not useful in all cases.
- Know how the FPGA is internally organized. Good to know when constructing designs from the bottom-up.
- Direct the CAD tools cleverly. They're slow, finicky, and annoying to use. Master their use.
- Learn VHDL, Verilog (or some HDL). Learn OpenCL.
- Have a scaling plan. To larger/smaller chips.

What is inside an FPGA?

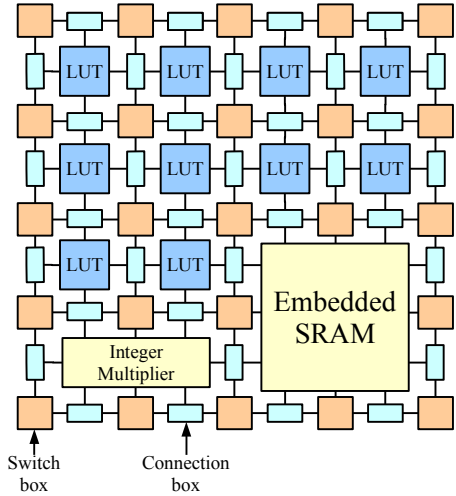


Figure: High-Level View of an FPGA chip. (*thesis.library.caltech.edu/6159/*)

Logic Elements

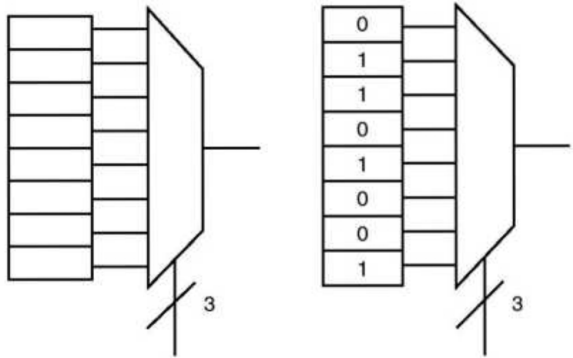
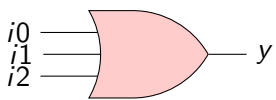


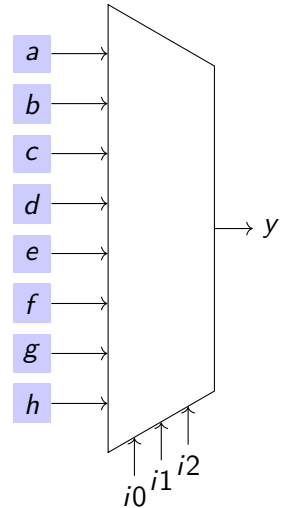
FIGURE 1.1 ¤ A 3-LUT schematic (a) and the corresponding 3-LUT symbol and truth table (b) for a logical XOR.

Figure: A Lookup-Table in an FPGA.

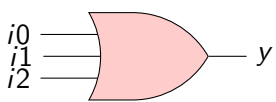
Packing Logic into LUTs



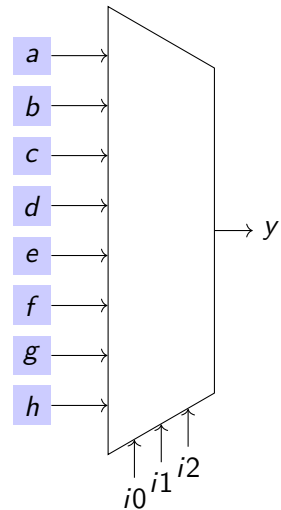
i0	i1	i2	y
0	0	0	a
0	0	1	b
0	1	0	c
0	1	1	d
1	0	0	e
1	0	1	f
1	1	0	g
1	1	1	h



Packing Logic into LUTs

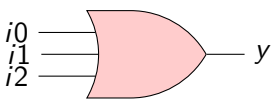


i0	i1	i2	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

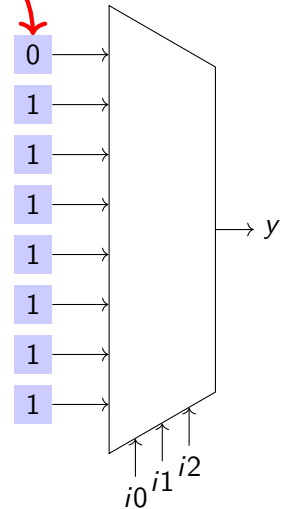


Packing Logic into LUTs

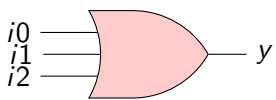
Program y column into SRAM at bootup



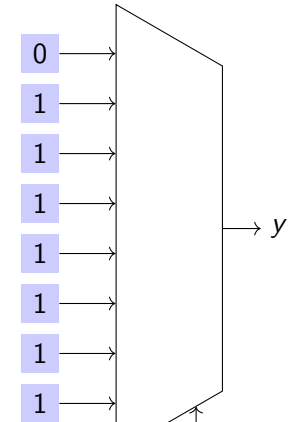
i0	i1	i2	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Packing Logic into LUTs



i0	i1	i2	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Logic Elements

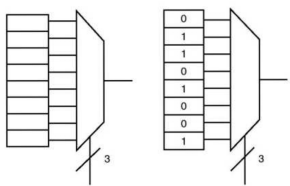
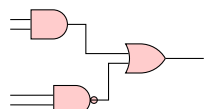


FIGURE 1.1 ¤ A 3-LUT schematic (a) and the corresponding 3-LUT symbol and truth table (b) for a logical XOR.

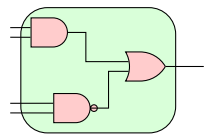
Figure: A Lookup-Table in an FPGA.

- Boolean table mapped to SRAMs.
- k-input table has 2^k cells.
- Old FPGAs had 4-LUTs – now have 5/6-input LUTs
- LUTs typically packed into clusters (CLBs/ALMs)
  - Richer local connectivity within cluster
- LUT contents programmed at boot-time

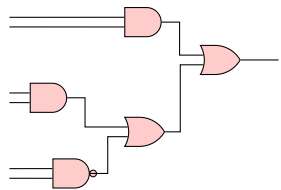
Example 1 - How to pack into 4-LUTs?



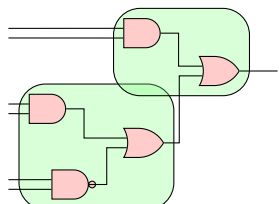
Example 1 - How to pack into 4-LUTs?



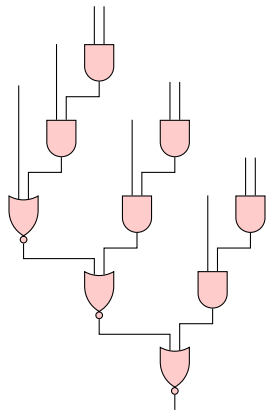
Example 2 - How many LUTs here?



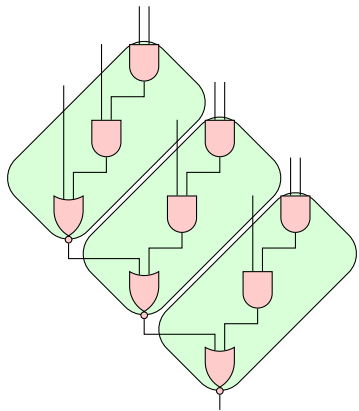
Example 2 - How many LUTs here?



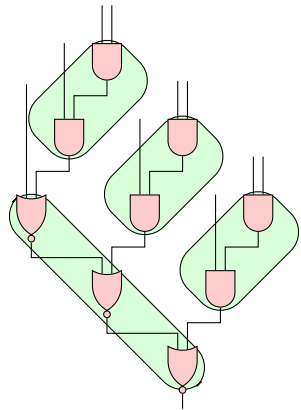
Area or Delay



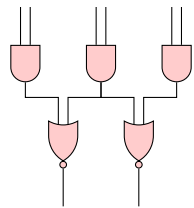
Area or Delay



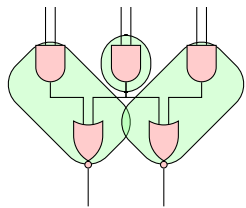
Area or Delay



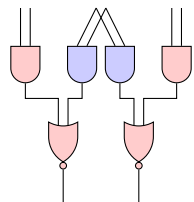
Fanout



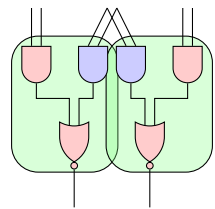
Fanout



Fanout



Fanout



Rough Sketch of the LUT packing algorithm

- Assign a LUT for each output of the circuit
- Perform breadth-first search to pack gates into k-LUT
- Stop when you reach the primary inputs of the circuit, or the number of inputs to the breadth-first wavefront exceed k
  - You have to traverse all the way to the primary inputs to confirm no k-LUT packing possible
- When breadth-first search fails to find an input count $\leq$k, start a depth-first search
- Depth-first search can be conducted on all inputs of the gates. Thus, multiple correct solutions are possible.
- If you encounter a register, stop the expansion.
- If you encounter a fanout (same gate output used in multiple places), you **may** have to stop the expansion
  - One option is to consider replication. This is only useful if it reduces the final LUT count
  - Another option is to continue the expansion (covered under the breadth-first search to primary inputs)
  - If all else fails, you are force to place LUT at the fanout position

Interconnect Fabric

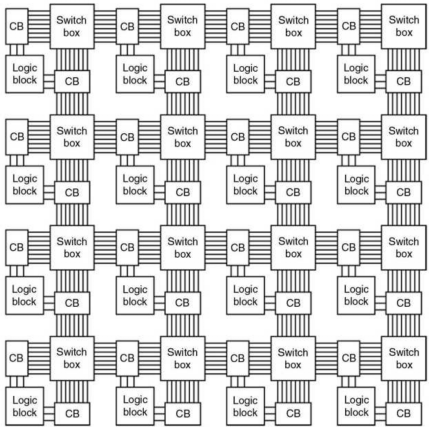


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Figure: Routing Fabric on an FPGA.

Interconnect Fabric

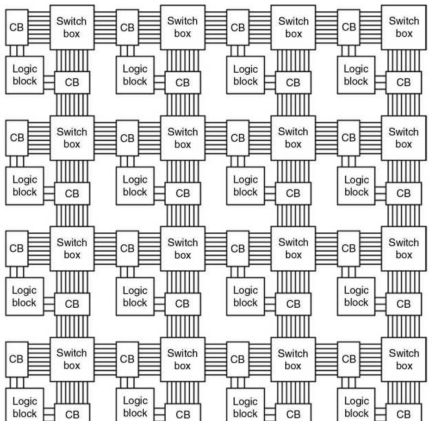


FIGURE 1.5 ◘ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Figure: Routing Fabric on an FPGA.

- Wiring organized into horizontal and vertical tracks
- Circuit connections mapped to this array structure
- Switches along a path programmed at boot-time
- Multiple parallel (non-overlapping) tracks for different circuit wires

Programming Routes

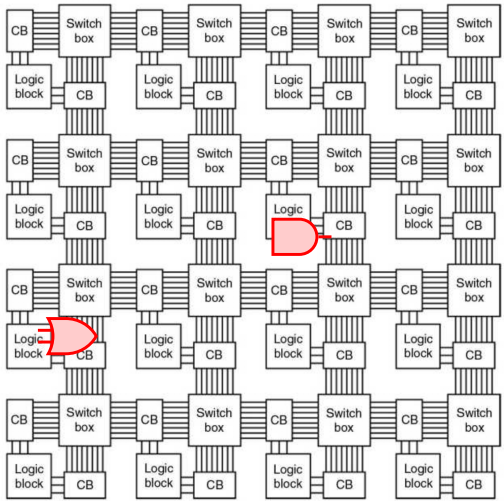


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Programming Routes

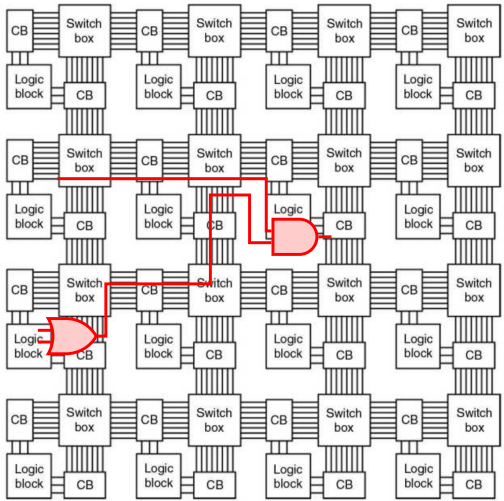


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Programming Routes

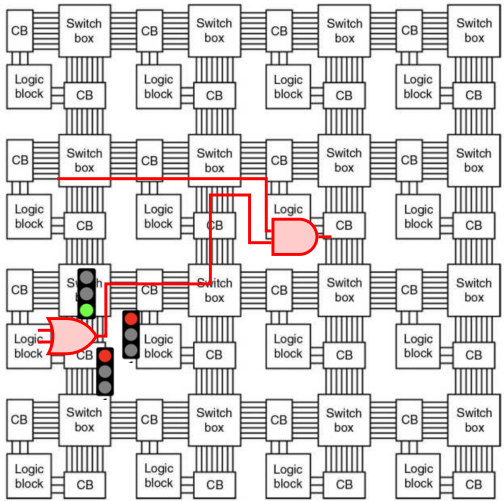


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Programming Routes

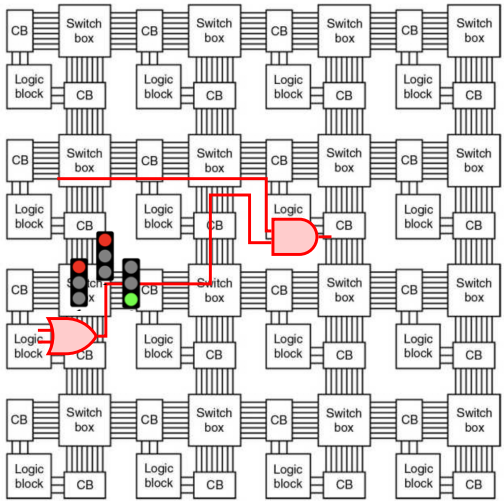


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Programming Routes

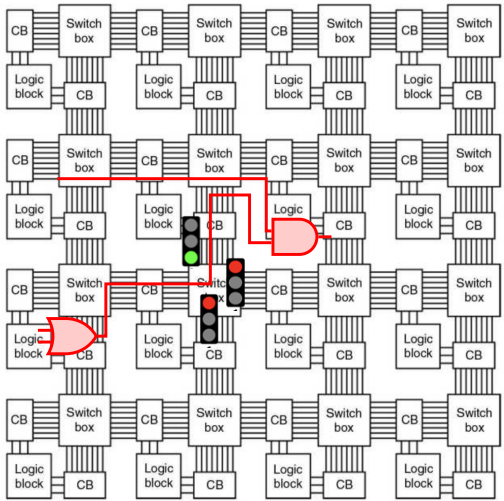


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Programming Routes

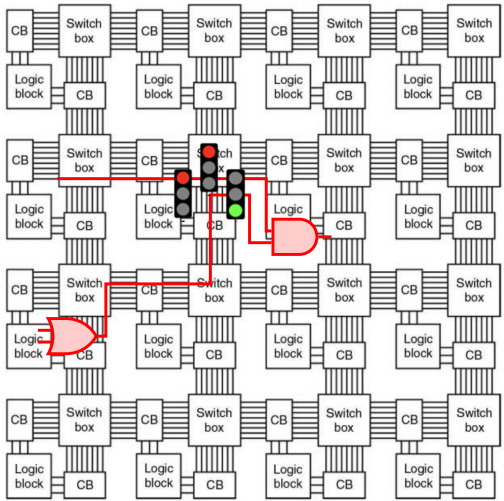


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Programming Routes

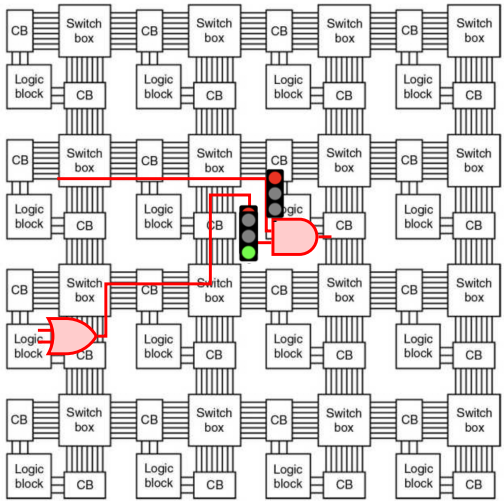
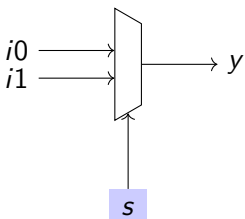


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

Switching element

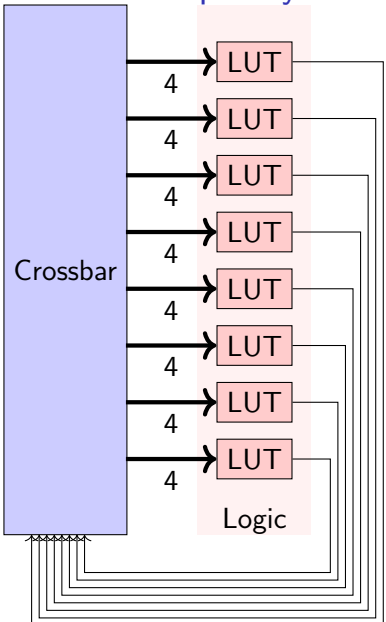


- Steer data from source to destination through a series of configured switches
- Switch configuration is *programmed* into SRAM cells at bootup time.
- Configuration does not change at runtime, connection persists during entire lifetime of circuit operation
- Physical implementations can sometimes use a single transistor + SRAM cell configurations.

Connecting millions of LUTs

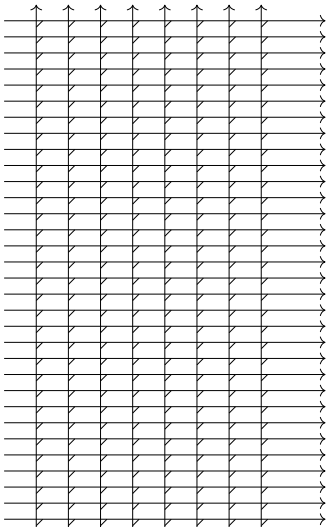
- How do we connect N hardware components in a programmable fashion?
- For a configurable fabric, anyone can connect to anyone
  - Not known until you write RTL for your hardware
  - Connection pattern changes with each design
  - Connections must stay alive during entire lifetime of operation
- For small N, use a crossbar
- For large N, need a scalable multi-stage fabric

Hardware complexity of crossbar



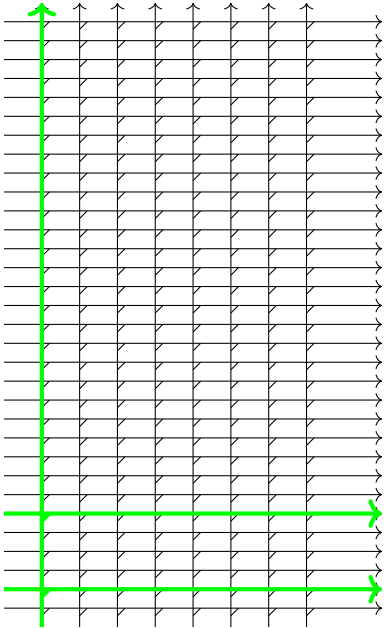
- For 8 4-LUTs, we have N=8, k=4
- Total number of outputs from logic = $N = 8$
- Total number of inputs to logic = $k \times N = 32$
- For logic, we have 8 outputs and 32 inputs
- We must design a crossbar with 8 inputs and 32 outputs

Switch Crossbar

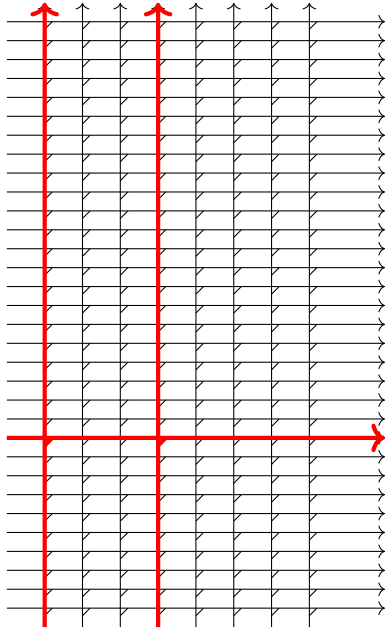


- The cluster interconnect for small sizes can be a crossbar
- A crossbar allows **any** input to connect to **any** output
  - Can visualize each output wire as an 8-input mux
  - For 8-input, 32-output switch, we need 32 8:1 muxes
  - Can implement a mux efficiently as one row of crossbar
- Each crosspoint in a crossbar is one transistor + one SRAM cell
- For N inputs and M outputs, we need $N \times M$ SRAM cells

Allowed



Disallowed



Clustering

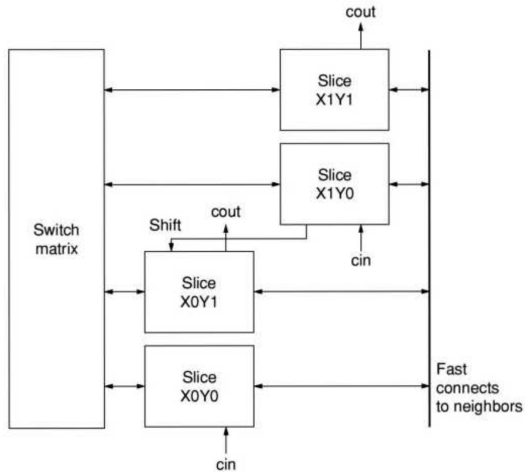


FIGURE 1.16 ¤ Xilinx Virtex-II Pro configurable CLB. (*Source:* Adapted from [8], Figure 32, p. 35.)

Figure: A Clustered Logic Block in an FPGA.

Clustering

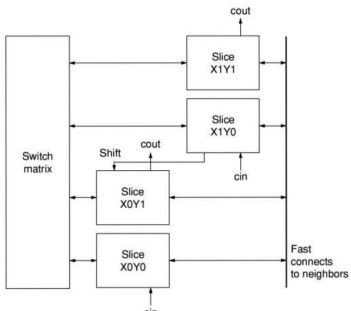
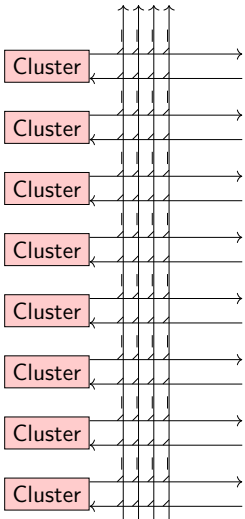


FIGURE 1.16 ¤ Xilinx Virtex-II Pro configurable CLB. (*Source:* Adapted from [8], Figure 32, p. 35.)

Figure: A Clustered Logic Block in an FPGA.

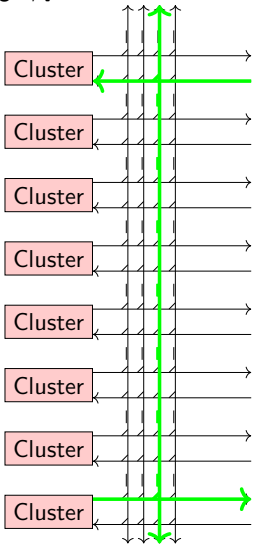
- LUTs clustered into a CLB (configurable logic block)
- Often circuits need lot of local connectivity
- Richer internal switch, reduce wiring pressure on fabric

Understanding chip-scale connections

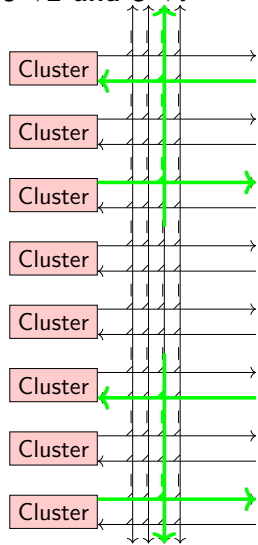


- To connect a large number of hardware components on a chip, a crossbar is too expensive
- Idea: allow only certain subset of possible connections
- Segmented interconnect to allow reuse same wire for different routes
  - Segment control requires one transistor + SRAM cell for 1D layout
  - For 2D, this becomes the 6 transistor + 6 SRAM cells sbox
- Each cluster input and output can hop on or off the shared wires
  - Each crosspoint here requires one transistor + one SRAM cell
  - This is a cbox

0→7


Cluster
Cluster
Cluster
Cluster
Cluster
Cluster
Cluster
Cluster

0→2 and 5→7


Cluster
Cluster
Cluster
Cluster
Cluster
Cluster
Cluster
Cluster

Connection Box

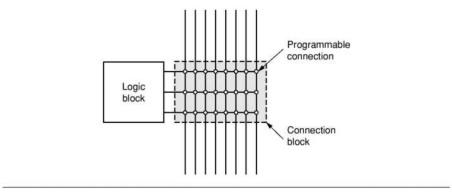


FIGURE 1.6 ◘ Detail of a connection block.

Figure: Left: Connection Box between the Logic Block and Wiring. Right: Highway onramp.

Connection Box

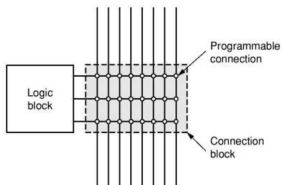


FIGURE 1.6 ¤ Detail of a connection block.

Figure: Connection Box between the Logic Block and Wiring.

- Allow CLB (LUT) inputs and outputs to interface with chip wiring
- Geometrically simpler and compact connectivity
  - Each crossing needs one transistor + SRAM cell
  - Only need to get onto the highway or not
- Connection statically determined by SRAM cell per intersection
- Can be *depopulated* to reduce number of switches

Switch Box

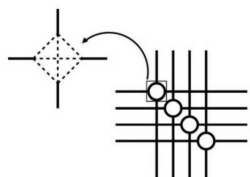


FIGURE 1.7 ¤ An example of a common switch block architecture.

Figure: Left: Switch Box between the Horizontal/Vertical wiring tracks. Right: Highway cloverleaf intersection

Switch Box

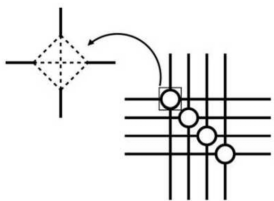


FIGURE 1.7 ¤ An example of a common switch block architecture.

Figure: Switch Box between the Horizontal/Vertical wiring tracks.

- Allows wires to cross from horizontal to/from vertical segments
- Crosspoint can support diamond pattern
  - Size possible paths – each controlled with transistor + SRAM cell
  - More complex than cbox as directionality of signal now known + turns permitted
- Various connection patterns can be supported
  - Can cross horizontal channels for better routability

How do you program an FPGA?

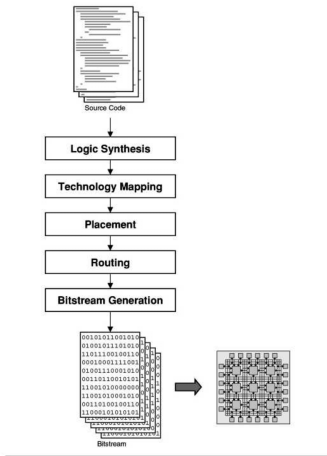


FIGURE 1.2 ¤ A typical FPGA mapping flow.

Figure: FPGA programming flow.

- Hardware described in Register-Transfer Level (RTL)
  - Gates, Registers, Wires
  - Use simulators to verify function
- **Synthesis**: RTL text converted into a netlist of gates
- **Tech Mapping**: gates $\rightarrow$ LUTs/CLBs
- **Placement**: LUTs/CLBs assigned physical X,Y positions
- **Routing**: Set switches to impl. wires between LUTs/CLBs
- **Bitstream**: Create executable for boot-time loading on FPGA

Placement

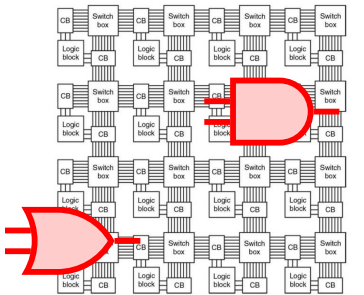


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

- First LUT can be placed in any of N possible locations.
- Second LUT can be placed in any of $N-1$ possible locations.
- We must pack M LUTs into N possible chip locations: $N \times (N-1) \times (N-2) \times \ldots (N-M+1)$
- **Solution**: Allow LUTs to place wherever they best want. Identify overused locations, and redistribute until each LUT has unique location. Objective is to minimize wirelength.

Routing

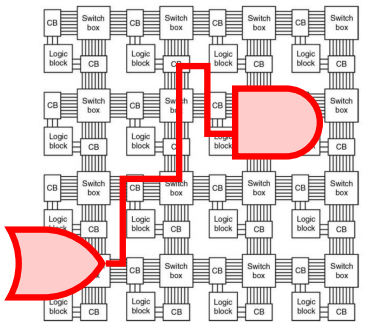


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

- Once all LUTs are placed, we must route connections
- Space of possible connections is large, we must route M connections (possibly with fanout)
- **Solution**: Allow routes to take the resource they most want. Identify conflicts. Add cost to in-demand resources. Reroute until each route has unique path. Objective is to minimize timing delays.

Routing

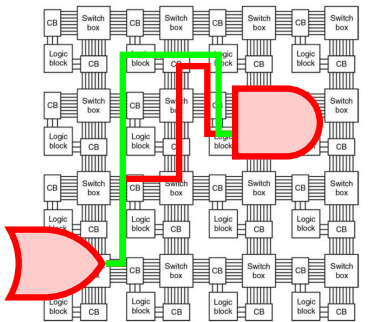


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

- Once all LUTs are placed, we must route connections
- Space of possible connections is large, we must route M connections (possibly with fanout)
- **Solution**: Allow routes to take the resource they most want. Identify conflicts. Add cost to in-demand resources. Reroute until each route has unique path. Objective is to minimize timing delays.

Routing

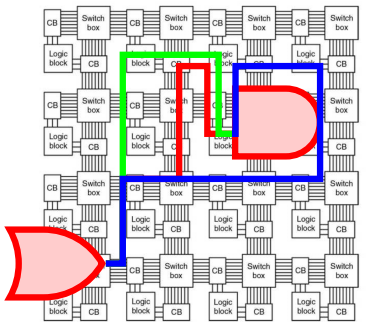


FIGURE 1.5 ¤ An island-style architecture with connect blocks and switch boxes to support more complex routing structures. (The difference in relative sizes of the blocks is for visual differentiation.)

- Once all LUTs are placed, we must route connections
- Space of possible connections is large, we must route M connections (possibly with fanout)
- **Solution**: Allow routes to take the resource they most want. Identify conflicts. Add cost to in-demand resources. Reroute until each route has unique path. Objective is to minimize timing delays.

Wrapup

- Introduced the need and tradeoffs for FPGA technology $\rightarrow$ need for configurable silicon resources in datacenters
- Peeked inside an FPGA to understand its building blocks
  - Lookup Tables $\rightarrow$ Logic
  - Connection and Switch boxes $\rightarrow$ Interconnect
- Evaluated the overall FPGA design flow
  - Placement and Routing phases
  - Emerging HLS flows