Lecture 5: FPGAs
FPGAs are in widespread use

Far more different designs are implemented in FPGAs than in custom chips.
And in the Data-Center
EECS151 FPGA Lab Board
FPGA: Xilinx Virtex-5 XC5VLX110T

Virtex-5 enhanced die photo

A die is an unpackaged part
FPGA Overview

- Basic structure: two-dimensional array of logic blocks and flip-flops with a means for the user to configure (program):
  1. the interconnection between the logic blocks,
  2. the function of each block.

Simplified version of FPGA internal architecture
Why are FPGAs Interesting?

- Technical viewpoint:
  - For hardware/system-designers, like ASICs - only better: “Tape-out” new design every few minutes/hours.
  - “reconfigurability” or “reprogrammability” may offer other advantages over fixed logic?
    - In-field reprogramming? Dynamic reconfiguration? Self-modifying hardware, evolvable hardware?

Of course, the higher flexibility comes at the expense of larger die area, slower circuits, and more energy per operation.
Why are FPGAs Interesting?

- Staggering logic capacity growth (10000x):

<table>
<thead>
<tr>
<th>Year Introduced</th>
<th>Device</th>
<th>Logic Cells</th>
<th>“logic gate equivalents”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>XC2064</td>
<td>128</td>
<td>1024</td>
</tr>
<tr>
<td>2011</td>
<td>XC7V2000T</td>
<td>1,954,560</td>
<td>15,636,480</td>
</tr>
</tbody>
</table>

- FPGAs have tracked Moore’s Law better than any other programmable device.
Why are FPGAs Interesting?

- Logic capacity now only part of the story: on-chip RAM, high-speed I/Os, “hard” function blocks, ...
- Modern FPGAs are “reconfigurable systems on a chip”

Xilinx Virtex-5 LX110T

- 64 ALUs
- 148 36Kb SRAM Blocks
- 10GBps Serdes
- Ethernet MACs
- PCI express Phy

Xilinx ZYNQ - embedded ARM cores
Energy Efficiency of CPU versus ASIC versus FPGA


\[
\therefore \text{FPGA} : \text{CPU} = 70x
\]

Similar story for performance efficiency
FPGA Internals
Background for upcoming technical details

Review: **mux** or multiplexor is a combinational logic circuit that chooses between $2^N$ inputs under the control of $N$ control signals.

A **latch** is a 1-bit memory (similar to a but “level sensitive” not edge-triggered, closer to an SRAM storage cell).
**FPGA Programmability**

- FPGA programmability allows users to:
  1. define function of configurable logic blocks (CLBs),
  2. establish interconnection paths between CLBs
  3. set other options, such as clock, reset connections, and I/O.
- Most FPGAs have **“SRAM based”** programmability.

**Programmable Cross-points**

- Latch-based (Xilinx, Intel/Altera, ...)

![Programmable Cross-points Diagram]

- MOSFET used as a “switch”

+ reconfigurable
- volatile
- relatively large.
User Programmability

- Latches store the configuration.
- Configuration *bitstream* is loaded under user control.
- “partial reconfiguration”: a selective part of the array can be reprogrammed without disturbing the other parts.
- Dynamic / runtime reconfiguration: reprogramming during a computation.
- Most commonly the entire device is programmed when the system is booted.

Diagram: Diagram of memory elements and interconnections, labeled with FF (flip-flops) and logic gates.
Look up table (LUT)
- implements any combinational logic function

Register (Flip-flop)
- optionally stores output of LUT
4-LUT Implementation

- LUTs size named by number of inputs
- n-bit LUT is implemented as a $2^n \times 1$ memory:
  - inputs choose one of $2^n$ memory locations.
  - memory locations (latches) are loaded with values from user’s configuration bit stream.
  - Inputs to mux control are the LUT inputs.
- Result is a general purpose “logic gate”.
  - n-LUT can implement any function of n inputs!

Latches programmed as part of configuration bit-stream
LUT as general logic gate

- An n-LUT is a direct implementation of a function truth-table.
- Each latch location holds the value of the function corresponding to one input combination.

Example: 2-input functions

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A 2-lut implements any function of 2 inputs.

How many of these are there?

How many functions of n inputs?

Example: 4-lut

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>F(0,0,0,0)</th>
<th>F(0,0,0,1)</th>
<th>F(0,0,1,0)</th>
<th>F(0,0,1,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td></td>
<td>store in 1st latch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td>store in 2nd latch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FPGA Generic Design Flow

- **Design Entry:**
  - HDL (hardware description languages: Verilog, VHDL)

- **Design Implementation:**
  - Logic synthesis (in case of using HDL entry) followed by,
  - Partition, place, and route to create configuration bit-stream file

- **Design verification:**
  - Optionally use simulator to check function
  - Load design onto FPGA device (cable connects PC to development board), optional “logic scope” on FPGA
  - check operation at full speed in real environment.

https://digitalsystemdesign.in/fpga-implementation-step-by-step/
Example Partition, Placement, and Route

- Simplified FPGA structure:

Example Circuit:
- collection of gates and flip-flops

Circuit combinational logic must be “covered” by 4-input 1-output LUTs.

Flip-flops from circuit must map to FPGA flip-flops.
(Best to preserve “closeness” to CL to minimize wiring.)

Best placement in general attempts to minimize wiring.

Vdd, GND, clock, and global resets are all “prewired”.

19
Example Partition, Placement, and Route

Example Circuit:
- collection of gates and flip-flops

Two partitions. Each has single output, no more than 4 inputs, and no more than 1 flip-flop. In this case, inverter goes in both partitions.

Note: (with 4-LUTs) the partition can be arbitrarily large as long as it has not more than 4 inputs and 1 output, and no more than 1 flip-flop.
Configurable Interconnect

- Design Challenges (topology):
  - traversing long wires incurs delay and energy
  - switches (transistors) add significant delay
  - Mapping time

switch matrix could be more richly populated

“connection block”
Xilinx FPGAs (interconnect detail)
Embedded Hard Blocks

- Many important functions are not efficient when implemented in the reconfigurable fabric:
  - multiplication, large memory, ...
- Dedicated blocks take relatively little area and therefore could go unused.
Colors represent different types of resources:

Logic
Block RAM
DSP (ALUs)
Clocking
I/O
Serial I/O + PCI

A routing fabric runs throughout the chip to wire everything together.
# State-of-the-Art - Xilinx FPGAs

<table>
<thead>
<tr>
<th>Device Name</th>
<th>45nm</th>
<th>28nm</th>
<th>20nm</th>
<th>16nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARTAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtex Ultra-scale</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kintex</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artix</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spartan</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Device Specifications

<table>
<thead>
<tr>
<th></th>
<th>VU3P</th>
<th>VU5P</th>
<th>VU7P</th>
<th>VU9P</th>
<th>VU11P</th>
<th>VU13P</th>
<th>VU27P</th>
<th>VU29P</th>
<th>VU31P</th>
<th>VU33P</th>
<th>VU35P</th>
<th>VU37P</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Logic Cells (K)</td>
<td>862</td>
<td>1,314</td>
<td>1,724</td>
<td>2,586</td>
<td>2,835</td>
<td>3,780</td>
<td>2,835</td>
<td>3,780</td>
<td>962</td>
<td>962</td>
<td>1,907</td>
<td>2,852</td>
</tr>
<tr>
<td>CLB Flip-Flops (K)</td>
<td>788</td>
<td>1,201</td>
<td>1,576</td>
<td>2,364</td>
<td>2,592</td>
<td>3,456</td>
<td>2,592</td>
<td>3,456</td>
<td>879</td>
<td>879</td>
<td>1,743</td>
<td>2,607</td>
</tr>
<tr>
<td>CLB LUTs (K)</td>
<td>394</td>
<td>601</td>
<td>788</td>
<td>1,182</td>
<td>1,296</td>
<td>1,728</td>
<td>1,296</td>
<td>1,728</td>
<td>440</td>
<td>440</td>
<td>872</td>
<td>1,304</td>
</tr>
<tr>
<td>Max. Dist. RAM (Mb)</td>
<td>12.0</td>
<td>18.3</td>
<td>24.1</td>
<td>36.1</td>
<td>36.2</td>
<td>48.3</td>
<td>36.2</td>
<td>48.3</td>
<td>12.5</td>
<td>12.5</td>
<td>24.6</td>
<td>36.7</td>
</tr>
<tr>
<td>Total Block RAM (Mb)</td>
<td>25.3</td>
<td>36.0</td>
<td>50.6</td>
<td>75.9</td>
<td>70.9</td>
<td>94.5</td>
<td>70.9</td>
<td>94.5</td>
<td>23.6</td>
<td>23.6</td>
<td>47.3</td>
<td>70.9</td>
</tr>
<tr>
<td>UltraRAM (Mb)</td>
<td>90.0</td>
<td>132.2</td>
<td>180.0</td>
<td>270.0</td>
<td>270.0</td>
<td>360.0</td>
<td>270.0</td>
<td>360.0</td>
<td>90.0</td>
<td>90.0</td>
<td>180.0</td>
<td>270.0</td>
</tr>
<tr>
<td>HBM DRAM (GB)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>HBM AXI Interfaces</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Clock Mgmt Tiles (CMTs)</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>DSP Slices</td>
<td>2,280</td>
<td>3,474</td>
<td>4,560</td>
<td>6,840</td>
<td>9,216</td>
<td>12,288</td>
<td>9,216</td>
<td>12,288</td>
<td>2,880</td>
<td>2,880</td>
<td>5,952</td>
<td>9,024</td>
</tr>
<tr>
<td>Peak INT8 DSP (TOP/s)</td>
<td>7.1</td>
<td>10.8</td>
<td>14.2</td>
<td>21.3</td>
<td>28.7</td>
<td>38.3</td>
<td>28.7</td>
<td>38.3</td>
<td>8.9</td>
<td>8.9</td>
<td>18.6</td>
<td>28.1</td>
</tr>
<tr>
<td>PCIe® Gen3 x16</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>PCIe Gen3x16/Gen4x8/CCIX(1)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>100G Interlaken</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>100G Ethernet w/ K4 RS-FEC</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>11</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Max. Single-Ended HP I/Os</td>
<td>520</td>
<td>832</td>
<td>832</td>
<td>832</td>
<td>624</td>
<td>832</td>
<td>520</td>
<td>676</td>
<td>208</td>
<td>208</td>
<td>416</td>
<td>624</td>
</tr>
<tr>
<td>GTY 32.75Gb/s Transceivers</td>
<td>40</td>
<td>80</td>
<td>80</td>
<td>120</td>
<td>96</td>
<td>128</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>64</td>
<td>96</td>
</tr>
<tr>
<td>GTM 58Gb/s PAM4 Transceivers</td>
<td>32</td>
<td>48</td>
<td></td>
<td></td>
<td>16/32</td>
<td>24/48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100G / 50G K4P4 FEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
1. CCIX: Channelized Compute Interface eXtensions
2. Extended: Extensive support for various interfaces
Atoms: 5-input Look Up Tables (LUTs)

Computes any 5-input logic function.

Timing is independent of function.

Latches set during configuration.
Virtex 6-LUTs: Composition of 5-LUTs

May be used as one 6-input LUT ($D_6$ out) ...

... or as two 5-input LUTS ($D_6$ and $D_5$)

Combinational logic (post configuration)
**Configurable Logic Blocks (CLBs)**

_Slices define regular connections to the switching fabric, and to slices in CLBs above and below it on the die._
The simplest view of a slice

- Four 6-LUTs
- Four Flip-Flops
- Switching fabric may see combinational and registered outputs.

An actual Virtex slice adds many small features to this simplified diagram. We show them one by one ...
Two 7-LUTs per slice ...

Extra multiplexers (F7AMUX, F7BMUX)
Extra inputs (AX and CX)
Or one 8-LUTs per slice ...

Third multiplexer (F8MUX)

Third input (BX)
Extra muxes to chose LUT option ...

From eight 5-LUTs ... to one 8-LUT.

Combinational or registered outs.

Flip-flops unused by LUTs can be used standalone.
We can map ripple-carry addition onto carry-chain block.

The carry-chain block also useful for speeding up other adder structures and counters.
Putting it all together ... a SLICEL.

The previous slides explain all SLICEL features.

About 50% of the are SLICELs.

The other slices are SLICEMs, and have extra features.
Recall: 5-LUT architecture ...

- **32 Latches.**
- **Configured to 1 or 0.**

Some parts of a logic design need many state elements.

SLICEMs replace normal 5-LUTs with circuits that can act like 5-LUTs, but can alternatively use the 32 latches as RAM, ROM, shift registers.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>1</td>
</tr>
<tr>
<td>000001</td>
<td>0</td>
</tr>
<tr>
<td>000010</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11101</td>
<td>0</td>
</tr>
<tr>
<td>11110</td>
<td>0</td>
</tr>
<tr>
<td>11111</td>
<td>1</td>
</tr>
</tbody>
</table>
Virtex DSP48E Slice

Efficient implementation of multiply, add, bit-wise logical.
Throughout the semester, we will look at different FPGA features in-depth.

- Switch fabric
- Block RAM
- DSP48 (ALUs)
- Clocking
- I/O
- Serial I/O + PCI