FPGA Logic Cells

CSET 4650
Field Programmable Logic Devices

Dan Solarek
Field-Programmable Gate Arrays

- Xilinx FPGAs are based on Configurable Logic Blocks (CLBs)
- More generally called logic cells
- Programmable

I/O blocks not shown
Programmable Logic Cells

- All FPGAs contain a basic programmable logic cell replicated in a regular array across the chip
  - configurable logic block, logic element, logic module, logic unit, logic array block, ...
  - many other names

- There are three different types of basic logic cells:
  - multiplexer based
  - look-up table based
  - programmable array logic (PAL-like)

- We will focus on the first two types
Logic Cell Considerations

How are functions implemented?
- fixed functions (manipulate inputs only)
- programmable functionality (interconnect components)

Coarse-grained logic cells:
- support complex functions, need fewer blocks, but they are bigger so less of them on chip

Fine-grained logic cells:
- support simple functions, need more blocks, but they are smaller so more of them on chip
Fine-Grained versus Coarse-Grained

- Fine-grained FPGAs are optimized to implement glue logic and irregular structures such as state machines.
- Data paths are usually a single bit:
  - can be considered bit-level FPGAs.
- Fine-grained architectures are not suitable for wider data paths:
  - they require lots of overhead.
Fine-Grained versus Coarse-Grained

- Reconfigurable computing stresses coarse-grained devices with data path widths much higher than one bit
  - essentially word-level FPGAs
- Coarse-grained reconfigurable FPGAs are especially designed for reconfigurable computing
- Such architectures provide operator level function units (CLBs) and word-level datapaths
  - Typically, at least four-bits wide
Logic Cell Considerations

When designing (or selecting) the type of logic cell for an FPGA, some basic questions are important:

- How many inputs?
- How many functions?
  - all functions of $n$ inputs or eliminate some combinations?
  - what inputs go to what parts of the logic cell?
- Any specialized logic?
  - adder, etc.
- What register features?
Programmable Logic Cells: Keys

- What is programmable?
  - input connections
  - internal functioning of cell
  - both

- Coarser-grained than logic gates
  - typically at least 4 inputs

- Generally includes a register to latch output
  - for sequential logic use

- May provide specialized logic
  - e.g., an adder carry chain
Logic Cells as Universal Logic

- Logic cells must be flexible, able to implement a variety of logic functions.
- This requirement leads us to consider a variety of “universal logic components” as basic building blocks.
- Multiplexers (MUXs) are one of the most attractive:
  - Not too small a building block (i.e., not too fine grained)
  - Flexible
  - Easy to understand
Universal Logic Gate: Multiplexer

4-to-1 Multiplexer

\[ Y = A \overline{S_0} \overline{S_1} + B \overline{S_0} S_1 + C S_0 \overline{S_1} + D S_0 S_1 \]

**NOT**

**OR**

**AND**
Universal Logic Gate: Multiplexer

- An example logic function using the Actel Logic Module (LM)
- Connect logic signals to some or all of the LM inputs, the remaining inputs to VDD (“1”) or GND (“0”)
- This example shows the implementation of the four-input combinational logic function:
  \[ F = (A \cdot B) + (B' \cdot C) + D \]
  \[ F = B \cdot (A + D) + B' \cdot (C + D) \]
  \[ F = B \cdot F2 + B' \cdot F1 \]
Universal Logic Gate: Multiplexer

For those of you a bit rusty wrt Boolean algebra:

\[ F = (A \cdot B) + (B' \cdot C) + D \]
\[ F = (A \cdot B) + (B' \cdot C) + D \cdot 1 \]
\[ F = (A \cdot B) + (B' \cdot C) + D \cdot (B + B') \]
\[ F = (A \cdot B) + (B' \cdot C) + D \cdot B + D \cdot B' \]
\[ F = A \cdot B + B' \cdot C + B \cdot D + B' \cdot D \]

\[ F = B \cdot (A + D) + B' \cdot (C + D) \]
\[ F = B \cdot F2 + B' \cdot F1 \]

Example use of Shannon’s expansion theorem
### Universal Logic Gate: Multiplexer

#### 2-to-1 Multiplexer

<table>
<thead>
<tr>
<th>Function, F</th>
<th>F=</th>
<th>Canonical form</th>
<th>Min-terms</th>
<th>Min-term code</th>
<th>Function number</th>
<th>M1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 '0'</td>
<td>'0'</td>
<td>'0'</td>
<td>none</td>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2 NOR1-1(A, B)</td>
<td>(A+B')</td>
<td>A·B</td>
<td>1</td>
<td>0010</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3 NOT(A)</td>
<td>A'</td>
<td>A·B' + A·B</td>
<td>0, 1</td>
<td>0011</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4 AND1-1(A, B)</td>
<td>A·B'</td>
<td>A·B'</td>
<td>2</td>
<td>0100</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>5 NOT(B)</td>
<td>B'</td>
<td>A·B' + A·B'</td>
<td>0, 2</td>
<td>0101</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6 BUF(B)</td>
<td>B</td>
<td>A·B' + A·B</td>
<td>1, 3</td>
<td>1010</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7 AND(A, B)</td>
<td>A·B</td>
<td>A·B</td>
<td>3</td>
<td>1000</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>8 BUF(A)</td>
<td>A</td>
<td>A·B' + A·B</td>
<td>2, 3</td>
<td>1100</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>9 OR(A, B)</td>
<td>A+B</td>
<td>A·B' + A·B + A·B' + A·B</td>
<td>1, 2, 3</td>
<td>1110</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>10 '1'</td>
<td>'1'</td>
<td>A·B' + A·B + A·B' + A·B</td>
<td>0, 1, 2, 3</td>
<td>1111</td>
<td>15</td>
<td>1</td>
</tr>
</tbody>
</table>
2-to-1 MUX WHEEL

- A 2:1 MUX viewed as a function wheel
- Any of the gates shown in the WHEEL can be generated by appropriate connections of A0, A1, SA, O and 1
- Any 2-input logic function can be generated
- Invert and buffer can be generated
Anti-fuse FPGA Examples

- Families of FPGAs differ in:
  - physical means of implementing user programmability,
  - arrangement of interconnection wires, and
  - the basic functionality of the logic blocks

- Most significant difference is in the method for providing flexible blocks and connection

Anti-fuse based (ex: Actel)

- Non-volatile, relatively small
- fixed (non-reprogrammable)
Actel ACT FPGAs

- Uses antifuse technology
- Based on channeled gate array architecture
- Each logic element (labelled ‘L’) is a combination of multiplexers which can be configured as a multi-input gate
- Fine-grain architecture
ACT 1 Simple Logic Module

- The ACT 1 Logic Module (LM, the Actel basic logic cell)
  - three 2-to-1 MUX
  - 2-input OR gate
- The ACT 1 family uses just one type of LM
- ACT 2 and ACT 3 FPGA families both use two different types of LM
ACT 1 Simple Logic Module

- An example Actel LM implementation using pass transistors (without any buffering)
The ACT 1 Logic Module is two function wheels, an OR gate, and a 2:1 MUX

- \( \text{WHEEL}(A, B) = \text{MUX}(A_0, A_1, SA) \)
- \( \text{MUX}(A_0, A_1, SA) = A_0 \cdot SA' + A_1 \cdot SA \)

Each of the inputs (\( A_0, A_1, \) and \( SA \)) may be A, B, '0', or '1'
ACT 1 Simple Logic Module

- Multiplexer-based logic module.
- Logic functions implemented by interconnecting signals from the routing tracks to the data inputs and select lines of the multiplexers.
- Inputs can also be tied to a logical 1 or 0, since these signals are always available in the routing channel.
ACT 1 Simple Logic Module

- 8 Input combinational function
- 702 possible combinational functions
- 2-to-1 Multiplexer
  \[ Y = A \cdot \overline{S} + B \cdot S \]
ACT 1 Simple Logic Module

- Implementation of a three-input AND gate
ACT 1 Simple Logic Module

- Implementation of S-R Latch
ACT 2 and ACT 3 Logic Modules

- The C-Module for combinational logic
- Actel introduced S-Modules (sequential) which basically add a flip-flop to the MUX based C-Module
  - ACT 2 S-Module
  - ACT 3 S-Module
ACT 2 Logic Module: C-Mod

- 8-input combinational function
- 766 possible combinational functions
ACT 2 Logic Module: C-Mod

- Example of a Logic Function Implemented with the Combinatorial Logic Module
ACT 3 Logic Module: S-Mod

- Sequential Logic Module
- Up to 7-input function plus D-type flip-flop with clear
- The storage element can be either a register or a latch.
  - It can also be bypassed so the logic module can be used as a Combinatorial Logic Module
ACT 2 and ACT 3 Logic Modules

The equivalent circuit (without buffering) of the SE (sequential element)
ACT 2 and ACT 3 Logic Modules

- The SE configured as a positive-edge-triggered D flip-flop
Actel Logic Module Analysis

- Actel uses a fine-grain architecture which allows you to use almost all of the FPGA
- Synthesis can map logic efficiently to a fine-grain architecture
- Physical symmetry simplifies place-and-route (swapping equivalent pins on opposite sides of the LM to ease routing)
- Matched to small antifuse programming technology
- LMs balance efficiency of implementation and efficiency of utilization
- A simple LM reduces performance, but allows fast and robust place-and-route
Altera FLEX 10K

- **Altera FLEX 10K Block Diagram**

- The EAB is a block of RAM with registers on the input and output ports, and is used to implement common gate array functions.

- The EAB is suitable for multipliers, vector scalars, and error correction circuits.
Embedded Array Block (EAB)

- Memory block, can be configured:
  - 256 x 8,  512 x 4,  1024 x 2,  2048 x 1

Figure 3. Examples of Combining EABs
Altera FLEX 10K

- Embedded Array Block
- Logic functions are implemented by programming the EAB with a read only pattern during configuration, creating a large LUT.
Altera FLEX 10K

- Logic Array Block
  - Each LAB consists of eight LEs, their associated carry and cascade chains, LAB control signals, and the LAB local interconnect.
  - The LAB provides the coarse-grained structure to the Altera architecture
Altera FLEX 10K

- Logic Element (LE)
- The LE is the smallest unit of logic in the FLEX 10K architecture
- contains a four-input LUT
- contains a programmable flip-flop with a synchronous enable, a carry chain, and a cascade chain.
- drives both the local and the FastTrack Interconnect.
### Table 1. FLEX 10K Device Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>EPF10K10</th>
<th>EPF10K20</th>
<th>EPF10K30</th>
<th>EPF10K40</th>
<th>EPF10K50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical gates (logic and RAM), Note (1)</td>
<td>10,000</td>
<td>20,000</td>
<td>30,000</td>
<td>40,000</td>
<td>50,000</td>
</tr>
<tr>
<td>Usable gates</td>
<td>7,000 to 31,000</td>
<td>15,000 to 63,000</td>
<td>22,000 to 69,000</td>
<td>29,000 to 93,000</td>
<td>36,000 to 116,000</td>
</tr>
<tr>
<td>Logic elements (LEs)</td>
<td>576</td>
<td>1,152</td>
<td>1,728</td>
<td>2,304</td>
<td>2,880</td>
</tr>
<tr>
<td>Logic array blocks (LABs)</td>
<td>72</td>
<td>144</td>
<td>216</td>
<td>288</td>
<td>360</td>
</tr>
<tr>
<td>Embedded array blocks (EABs)</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Total RAM bits</td>
<td>6,144</td>
<td>12,288</td>
<td>12,288</td>
<td>16,384</td>
<td>20,480</td>
</tr>
<tr>
<td>Maximum user I/O pins</td>
<td>134</td>
<td>189</td>
<td>246</td>
<td>189</td>
<td>310</td>
</tr>
</tbody>
</table>
# Altera FLEX 10K Family

## FLEX 10K Devices (continued)

### Table 2. FLEX 10K Device Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>EPF10K70</th>
<th>EPF10K100</th>
<th>EPF10K100A</th>
<th>EPF10K130V</th>
<th>EPF10K250A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical gates (logic and RAM), Note (1)</td>
<td>70,000</td>
<td>100,000</td>
<td>130,000</td>
<td>250,000</td>
<td></td>
</tr>
<tr>
<td>Usable gates</td>
<td>46,000 to 118,000</td>
<td>62,000 to 158,000</td>
<td>82,000 to 211,000</td>
<td>149,000 to 310,000</td>
<td></td>
</tr>
<tr>
<td>LEs</td>
<td>3,744</td>
<td>4,992</td>
<td>6,656</td>
<td>12,160</td>
<td></td>
</tr>
<tr>
<td>LABs</td>
<td>468</td>
<td>624</td>
<td>832</td>
<td>1,520</td>
<td></td>
</tr>
<tr>
<td>EABs</td>
<td>9</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Total RAM bits</td>
<td>18,432</td>
<td>24,576</td>
<td>32,768</td>
<td>40,960</td>
<td></td>
</tr>
<tr>
<td>Maximum user I/O pins</td>
<td>358</td>
<td>406</td>
<td>470</td>
<td>470</td>
<td></td>
</tr>
</tbody>
</table>

Note to tables:
(1) For designs that require JTAG boundary-scan testing, the built-in JTAG circuitry contributes up to 31,250 additional
Altera FPGA Family Summary

- **Altera Flex10K/10KE**
  - LEs (Logic elements) have 4-input LUTS (look-up tables) +1 Flip-Flop
  - Fast Carry Chain between LE’s, cascade Chain for logic operations
  - Large blocks of SRAM available as well

- **Altera Max7000/Max7000A**
  - EEPROM based, very fast (Tpd = 7.5 ns)
  - Basically a PLD architecture with programmable interconnect.
  - Max 7000A family is 3.3 v
Xilinx LCA

- Xilinx LCA (a trademark, denoting logic cell array) basic logic cells, configurable logic blocks or CLBs, are bigger and more complex than the Actel or QuickLogic cells.
- The Xilinx CLBs contain both combinational logic and flip-flops.
- Coarse-grain architecture
- Xilinx Mature Products: XC3000, XC4000, XC5200
Latches are used to:
- make or break cross-point connections in the interconnect
- define the function of the logic blocks
- set user options:
  - within the logic blocks
  - in the input/output blocks
  - global reset/clock

“Configuration bit stream” can be loaded under user control

All latches are strung together in a shift chain

Latch-based (Xilinx, Altera, …)
Logic Lookup Table

- LUT is used instead of basic gates or MUXs
- Specify logic functions to be implemented as a simple truth table
- n-input LUT can handle function of $2^n$ inputs
- A LUT is actually a small (1-bit) RAM
  - FPGA LUTs can be used as RAM
A Two-Input Lookup Table

- LUTs can be implemented using MUXs
- We do not normally care about the implementation, just the functioning

(a) Circuit for a two-input LUT
(b) \( f_1 = \bar{x}_1\bar{x}_2 + x_1x_2 \)
(c) Storage cell contents in the LUT
A Three-Input LUT

- A simple extension of the two-input LUT leads to the figure at right.
- Again, at this point we are interested in function and not form.
Inclusion of a Flip-Flop with a LUT

- A Flip-Flop can be selected for inclusion or not
- Latches the LUT output

---

The diagram shows a LUT with inputs $\text{In}_1$, $\text{In}_2$, and $\text{In}_3$. The output of the LUT is connected to a flip-flop, which can be selected for inclusion or bypassed. The flip-flop has a clock input and outputs $D$ and $Q$. The select signal determines whether the flip-flop is used or bypassed.
Xilinx XC3000 CLB

- The block diagram for an XC3000 family CLB illustrates all of these features.
- To simplify the diagram, programmable MUX select lines are not shown.
- Combinational function is a LUT.
Xilinx XC3000 CLB

- A 32-bit look-up table (LUT)
- CLB propagation delay is fixed (the LUT access time) and independent of the logic function
- 7 inputs to the XC3000 CLB:
  - 5 CLB inputs (A–E), and
  - 2 flip-flop outputs (QX and QY)
- 2 outputs from the LUT (F and G).
- Since a 32-bit LUT requires only five variables to form a unique address \((32 = 2^5)\), there are multiple ways to use the LUT
Xilinx XC3000 CLB

- Use 5 of the 7 possible inputs (A–E, QX, QY) with the entire 32-bit LUT
  - the CLB outputs (F and G) are then identical

- Split the 32-bit LUT in half to implement 2 functions of 4 variables each
  - choose 4 input variables from the 7 inputs (A–E, QX, QY).
  - you have to choose 2 of the inputs from the 5 CLB inputs (A–E); then one function output connects to F and the other output connects to G

- You can split the 32-bit LUT in half, using one of the 7 input variables as a select input to a 2:1 MUX that switches between F and G
  - to implement some functions of 6 and 7 variables
Xilinx XC4000 Family CLB

- The block diagram for the XC4000 family CLB is similar to that of the CLB of the XC3000.
- Carry logic connections shown.
XC4000 Logic Block

- Two four-input LUTs that feed a three-input LUT
- Special fast carry logic hard-wired between CLBs
- MUX control logic maps four control inputs C1-C4 into the four inputs:
  - LUT input (H1)
  - direct in (DIN)
  - enable clock (EC)
  - set/reset control for flip-flops (S/R)
- Control inputs C1-C4 can also be used to control the use of the F’ and G’ LUTs as 32 bits of SRAM
Two 4-Input Functions

Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)
5-Input Function

Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)
CLB Used as RAM

Figure 4: 16x2 (or 16x1) Edge-Triggered Single-Port RAM
Xilinx XC5200 Family

- Xilinx XC5200 family **Logic Cell (LC)** and configurable logic block (CLB).
Xilinx XC5200 Family

- Basic Cell is called a *Logic Cell* (LC) and is similar to, but simpler than, CLBs in other Xilinx families.
- Term CLB is used here to mean a group of 4 LCs (LC0-LC3).
Xilinx Spartan Family

- Memory Resources
- I/O Connectivity
- System Clock Management
  - Digital Delay Lock Loops (DLLs)
- Logic & Routing
Xilinx Spartan CLB

- Spartan-IIE CLB Slice
- Two identical slices in each CLB
- Each slice has 2 LUT-FF pairs with associated carry
- Two 3-state buffers (BUFT) associated with each CLB, accessible by all CLB outputs
Xilinx Spartan CLB

- Each slice contains two sets of the following:
  - Four-input LUT
    - Any 4-input logic function
    - Or 16-bit by 1 sync RAM
    - Or 16-bit shift register
  - Carry and Control
    - Fast arithmetic logic
    - Multiplier logic
    - Multiplexer logic
  - Storage element
    - Latch or flip-flop
    - Set and reset
    - True or inverted inputs
    - Sync. or async. control

The University of Toledo
Xilinx Virtex-E CLB

- Two CLB “logic slices”
Xilinx LUTs: Pros and Cons

Using LUTs to implement combinational logic has both advantages and disadvantages.

Disadvantages:
- An inverter is as slow as a five input NAND
- Routing between cells is more complex than Actel because of coarse-grain architecture

Advantages:
- Simplifies timing
- Same delay for any function of five variables
- Can be used directly as SRAM
Quicklogic FPGA

- Quicklogic (Cypress)
- Logic Cell