Lecture 12

Digital Circuit Implementation Issues
PLAs, PALs, ROM’s, FPGA’s
- Packaging Issues
- Look Up Table method
- Multiplexer Method
- RAM & ROM method
- Xilinx and Actel Examples of FPGA’s
- I/O for FPGA’s
- Comparison of Various FPGAs
Back to Design Cycle

- Defining system requirements
- Making architectural decisions
- Decisions on Hierarchy, Regularity, Locality
- Comprehensive design of units
- Planning for implementation
- Planning for verification
- Estimating System Performance
- Implementation Technology Decision
- Measuring system performance
- Implementation
Names associated with this field:

PLD... PAL, PLA, FPLA, SPLD, CPLD
GA, MPGA, ASIC, Full Custom, Semi Custom,
ROM, PROM, EPROM, EEPROM
FPGA, LCA, VLSI, ULSI, GSI, MCM, SOC, NoC

**NEW** FPOA**

Field Programmable Object Array (FPOA) product from Mathstar.

They offer FPGA-like functionality but replaced the CLBs with ALU blocks instead. They also run at 1GHz and have large memory blocks.

Ideal associated characteristics

Field Programmability
Availability of CAD tools
CAD tool friendliness
Performance
Prototyping Costs, Production Time, Yield
Automatic transformation of HDL code into a gate level netlist is called “SYNTHESIS”
Every vendor has its own tools for synthesis, however they all use the flow shown below

1. **Specification**
2. **HDL description**
3. **Verify Design**
4. **Target Technology**
5. **Map design to PLD**
6. **Download to PLD**
Any Sum of Product (SOP) can be represented by AND-OR.

ROM, PAL, PLA are different optimized implementation of a given circuit using the AND-OR planes.

ROM: AND Fixed, OR Programmable
PAL: AND Programmable, OR fixed
PLA: AND Programmable, OR Programmable
FPGA: Programmable Logic Blocks, Programmable Interconnect
Logic Gates and Programmable switches

Programmable Logic Device as a black box

Inputs (logic variables) → Logic Gates and Programmable switches → Outputs (logic functions)
Input buffers
And inverters

Any combinational logic can be implemented with **Sum of Product** which is AND-OR implementation.

General Structure of PLD – Programmable Logic Device
## Functionality Table

<table>
<thead>
<tr>
<th>AND</th>
<th>OR</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>Fixed</td>
<td>Not Programmable</td>
</tr>
<tr>
<td>Fixed</td>
<td>Programmable</td>
<td>PROM</td>
</tr>
<tr>
<td>Programmable</td>
<td>Fixed</td>
<td>PAL</td>
</tr>
<tr>
<td>Programmable</td>
<td>Programmable</td>
<td>PLA</td>
</tr>
</tbody>
</table>
PLA: Gate Level

Programmable Fuses
Connections

OR plane

AND plane

P1

P2

P3

P4

SUM

f1

f2
PLA: Customary Schematic

AND plane

OR plane

x1 x2 x3

P1

P2

P3

P4

f1 f2
Advantages of PLA

- **Efficient** in terms of area needed for implementation on an IC chip
- Often included as part of larger chips such as microprocessors
- **Programmable** AND and OR gates
OR plane (Fixed)

AND plane (Programmable)
PAL - Programmable Array Logic

- PLA have higher programmability than PAL, however they have lower speed than PAL

  Solution → PAL for higher speed.

- Programmable AND, Fixed OR

- PAL - Simpler to manufacture, cheaper than PLA and have better performance
Flip-flops store the value produced by the OR gate output at a particular point and can hold it indefinitely.

Flip-flop output is controlled by the clock signal. On 0-1 transition of clock, flip-flop stores the value at its D input and latches the value at Q output.

2-to-1 multiplexer selects an output from the OR gate output or the flip-flop output. Tri-state buffers are placed between multiplexer and the PAL output.

Multiplexer’s output is fed back to the AND plane in PAL, which allows the multiplexer signal to be used internally in the PAL. This facilitates the implementation of circuits that have multiple stages (levels or logic gates).
For additional flexibility, extra circuitry is added at the output of each OR gate. This is also referred to macrocell.
Example: FSM Implementation

\[ S_2 = P' \ Q \ y_1, \quad R_2 = y_2, \]
\[ S_1 = P' \ Q', \quad R_1 = Q + P \]
\[ Z = y_2 \ y_1' \ P \ Q' \]

P & Q – are inputs
y_2 & y_1 are the states
Z is the output
User circuits are implemented in the programmable devices by configuring or programming these devices. Due to the large number of programmable switches in commercial chips; it is not feasible to specify manually the desired programming state for each switch. CAD systems are used to solve this problem.

Computer system that runs the CAD tools is connected to a programming unit. After design of a circuit has been completed, CAD tool generates a file (programming file or fuse map) that specifies the state of each switch in PLD. PLD is then placed into the programming unit and the programming file is transferred from the computer system to the unit. Programming unit then programs each switch individually.
PAL (or PLA) as part of a logic circuit resides with other chips on a Printed Circuit Board (PCB). PLD has to be removed from PCB for programming purposes. By placing a socket on PCB makes the removal possible. Plastic leaded chip carrier (PLCC) is the most commonly used package.

Instead of using a programming unit, it would be easier if a chip could be programmed on the PCB itself. This type of programming is called in-system programming (ISP).

So all you need: Personal computer, PLD CAD tool, the programming device and its software and the kind of PLD that the device accepts.

Tutorial for PAL can be found at http://courses.cs.washington.edu/courses/cse370/06au/tutorials/Tutorial_PAL.htm
Simple PLDs,
Single AND.OR plane
It is configured by programming the AND and OR plane, or may be the Flip Flop
inclusion and feedback selection,
Usually has less than 32 I/O
They are available in DIP (Dual in line package), PLCC (Plastic Lead Chip Carrier up to
100 pins. Usually less than 100 equivalent gates.

Complex PLDs
Multiple AND-OR planes
Extend the concept of the simple PLDs further by incorporating architectures that
contain several multiple logic block PAL models. Most CPLD use programmable
interconnect.
Can accommodate from 1000 to 10,000 equivalent gates.
Are available in PLCC and QFP (Quad Flap Pack) up to 200 pins
Chips containing PLDs are limited to modest sizes, typically supporting number of input and output more than 32. To accommodate circuits that require more input and outputs, either multiple PLAs or PALs can be used or a more sophisticated type of chip, called a complex programmable logic device (CLPD). CLPD is made up of multiple circuit blocks on a single chip, with internal wiring to connect the circuit blocks.

The structure of CLPD is shown on the next slide. It includes four PAL-like blocks connected by interconnection wires. Each block in turn is connected to a sub-circuit I/O block, which is attached to a number of input and output pins.
PAL-like Block

PAL-like Block
CLPD uses *quad flat pack (QFP) type* of package. QFP package has pins on all four sides and the pins extend outward from the package with a downward-curving shape. Moreover, QFP pins are much thinner and hence, they support a larger number of pins when compared to the PLCC packing.

Most CPLDs contain the same type of switch as in PLDs. Here, a separate programming unit is not used due to two main reasons. Firstly, CLPDs contain 200 + pins on the package, and these pins are often *fragile and easily bent*. Secondly, a socket would be required to hold the chip. *Sockets* are usually *quite expensive* and hence, add to the overall cost incurred.
CLPD usually support the **ISP technique**. A small connector is included on the PCB and is connected to a computer system. CLPD is programmed by transferring the programming information from the **CAD tool** to into the CLPD.

The circuitry on the CLPD that allows this type of programming is called **JTAG**, Joint Test Action Group port, and is standardized by the IEEE.

JTAG is a **non-volatile** type of programming i.e programmed state is retained permanently (for example, in case of power failure, CLPD retains the program).
The distinction between the two is blurred

Although PLDs started as small devices, today’s PLDs are anything but simple.

FPGAs fill the gap between PLDs and complex ASICs

In both cases, you can program the devices yourself, using design entry and simulation.

All FPGAs have regular array of basic cells that are configured by the programmer using special software that program the chips by programming the interconnection.

Each vendor has tool supplier that provides custom tools for their products.

The programming methodology is usually non permanent, allowing re-programmability
Advantage:
FPGAs have lower prototyping costs
FPGAs have shorter production times

Disadvantage:
FPGAs have lower speed of operation in comparison to MPGAs
Say by a factor 3 to 5
FPGAs have a lower logic density in comparison to MPGAs
Say by a factor of 8 to 12
FPGAs

Consists of uncommitted logic arrays and user programmable interconnection.

The interconnect programming is done through programmable switches.

The Logic circuits are implemented by partitioning the logic into blocks and then interconnecting the blocks with the programmable switches.

The architecture of an FPGA varies from device to device, vendor to vendor; it can be based on CPLDs, EPROMS, EEPROMS, LUT, Buses, PALS.

The interconnect is also varied from EPROM, static RAM, antifuse, EEPROM.
FPGAs Classifications

FPGA types

- Implementation Architecture
  - Symmetrical Array
  - Row based Array
  - Hierarchial PLD
  - Sea of Gates

- Logic Implementation
  - Look Up table
  - Multiplexer based
  - PLD Block
  - NAND Gates

- Interconnect Technology
  - Static Ram
  - Antifuse
  - E/EPROM
Consists of an array of uncommitted elements that can be interconnected in a general way. Like a PAL the interconnection between the elements are user programmable. The interconnect compromises segments of wires, where segments may be of various lengths. Present in the interconnect are programmable switches that serve to connect the logic blocks to the wire segments or one wire segment to another. Logic circuits are implemented in the FPGA by partitioning the logic into logic blocks and then interconnecting the blocks as required via switches. To facilitate the implementation of a wide variety of circuits, it is important that an FPGA be as versatile as possible. There are many ways to design an FPGA, involving trade offs in the complexity and flexibility of both the logic blocks and the interconnection resources.
Logic Block and Interconnection:

The *architecture of logic blocks* vary from simple combinational logic to complex EPROMs, LUT, Buses etc.. The *routing architecture* can also be variable including pass-transistors controlled by static RAM cells, anti fuses, EPROM transistors. Each company provides a

*variety of architecture* of the logic blocks and *routing architecture*.
Classes of common commercial FPGA

Symmetrical Array

Row-based

Logic Block

Sea-of-Gates

Interconnect overlayed on Logic Blocks

PLD Block

Hierarchical PLD

Various Block Architecture & Routing Architecture
Table 2. HardCopy IV E Devices Overview

<table>
<thead>
<tr>
<th>Device</th>
<th>ASIC Gates</th>
<th>Memory Bits</th>
<th>I/O Pins</th>
<th>PLLs</th>
<th>FPGA Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>HC4E2YZ</td>
<td>3.9M</td>
<td>8.1</td>
<td>296 - 480</td>
<td>4</td>
<td>EP4SE110</td>
</tr>
<tr>
<td>HC4E3YZ</td>
<td>9.2M</td>
<td>10.7</td>
<td>296 - 480</td>
<td>4</td>
<td>EP4SE230</td>
</tr>
<tr>
<td>HC4E5YZ</td>
<td>9.5M</td>
<td>16.8</td>
<td>480 - 864</td>
<td>4/8/12</td>
<td>EP4SE360</td>
</tr>
<tr>
<td>HC4E6YZ</td>
<td>11.5M</td>
<td>16.8</td>
<td>736 - 880</td>
<td>8/12</td>
<td>EP4SE530</td>
</tr>
<tr>
<td>HC4E7YZ</td>
<td>13.3M</td>
<td>16.8</td>
<td>736 - 880</td>
<td>8/12</td>
<td>EP4SE680</td>
</tr>
</tbody>
</table>

Notes:
1. Y = I/O count, Z = package type (see the product catalog for more information)
2. ASIC gates calculated as 12 gates per logic element (LE), 5,000 gates per 18 x 18 multiplier (SRAMs, PLLs, test circuitry, I/O registers not included in gate count)
3. Not including MLABs
Design Flow
Process Diagram
A designer implementing a circuit on an FPGA must have access to CAD tools for that type of FPGA. The following steps summarize the process:

1) **Logic Entry**: Either simulate capture or entering VHDL description or specifying Boolean expansions.

2) **Translate** to Boolean & optimize

3) Transform into a circuit of **FPGA logic blocks** through a technology mapping program (minimizing # of blocks).

4) **Decides** what to place in each block in FPGA array (minimizing total length of interconnect)

5) **Assigns** the FPGA’s wire segments and chooses programmable switches to establish required interconnection.
6) The output of the CAD system is fed to the programming unit that configures the final FPGA chip.

Depending upon correct VHDL or design entry, the entire process of implementing a circuit in an FPGA can take from a few minutes to about an hour.
Any logic function can expanded in form of a Boolean variable:

\[ F = A \cdot F + \overline{A} \cdot F \]

For example assume \( F = A \cdot B + A \cdot B \cdot C + A \cdot B \cdot C \cdot A \)

Then in the expansion

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

\[ = A \cdot (B \cdot C) + A \cdot (B + C) \]

Then this can be implemented with a MUX
F1 = B \cdot \overline{C} \quad F2 = B + C

These functions can be broken down further into:

F1 = B (B \cdot \overline{C}) + B (B \cdot \overline{C})

= B \cdot \overline{C} + B \cdot 0

F2 = B (B + C) + B (B + C)

= B \cdot 1 + B \cdot C
Functions can also be expanded into canonical form. Then $F$ is expanded as

$$F = A \cdot B + A \cdot B \cdot C + A \cdot B \cdot C$$

$$F = A \cdot B \cdot (C + C) + A \cdot B \cdot C + A \cdot B \cdot C$$

$$= A \cdot B \cdot C + A \cdot B \cdot C + A \cdot B \cdot C + A \cdot B \cdot C$$

$$= A \cdot B \cdot C + A \cdot (B \cdot C + B \cdot C + B \cdot C)$$

$$= A \cdot F_1 + A \cdot F_2$$  \text{In turn this can be implemented in MUX:}
Therefore 2-1 multiplexer is a general block that can represent any gate:

**AND Gate**

\[
F = A \cdot B \\
F = A \cdot (A \cdot B) + \overline{A}(A \cdot B) \\
= A \cdot B + \overline{A} \cdot 0
\]

**OR Gate**

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

**Ex-OR**

\[
F = A \cdot B + A \cdot \overline{B}
\]
Functions that can be implemented using just 2:1 MUX (No inverter at the input).

<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) (A+B')</td>
<td>A'B</td>
<td>1</td>
<td>0010</td>
<td>2</td>
<td>B 0 A</td>
<td></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>2</td>
<td>0100</td>
<td>4</td>
<td>A 0 B</td>
<td></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>3</td>
<td>1000</td>
<td>8</td>
<td>A 0 B</td>
<td></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 1 A</td>
</tr>
</tbody>
</table>

10 '1' '1' A'B' + A'B' + A'B' + A'B | 0, 1, 2, 3 | 1111 | 15 | 1 1 1 |

If there are no 2 input rails available, XOR, NAND & NOR cannot be implemented directly. There is a need for more MUXs to be used as inverters.
ACT1 module has three 2:1 Muxs with AND-OR logic at the select of final MUX and implements all 2 input functions, most 3 input and many 4 input functions.

Software module generator for ACT1 takes care of all this.

Apart from variety of combinational logic functions, the ACT1 module can implement sequential logic cells in a flexible and efficient manner. For example an ACT1 module can be used for a transparent Latch or two modules for a flip flop.
General Architecture of Actel FPGAs

ACT-1 Logic Module
The basic Architecture of Actel FPGA is similar to that found in MPGAs, consisting of rows of programming block with horizontal routing channels between the rows. Each routing switch in these FPGAs is implemented by the PLICE Anti fuse.
**ACTEL Logic Module**

\[ F = A.B + B.C + D \]

\[ = B [A.B + B.C + D] + B[A.B + B.C + D] \]

\[ = A.B + B.D + B.C + B.D \]

\[ = B.(A+D) + B.(C+D) \]
**S-Module (ACT 2)**

- Inputs: D00, D01, D10, D11, A1, B1, A0, CLR
- Outputs: SE, Q
- Logic: Y-OR, Master Latch, Slave Latch

**S-Module (ACT 3)**

- Inputs: D00, D01, D10, D11, A1, B1, A0, CLR, CLK
- Outputs: SE, Q
- Logic: Y-OR, Master Latch, Slave Latch, Combinational Logic for Clear and Clock

**ACTEL ACT C-Module**

- Inputs: M1, D00, D01, D10, D11, A1, B1, A0, B0
- Outputs: OUT, S0, S1

**SE (Sequential Element)**

- Inputs: D, C2, C1, CLR
- Outputs: Q
- Logic: 1Z Latch, Master Latch, Slave Latch, Combinational Logic for Clear and Clock
ACT1 module is simple logical block. It does not have built in function to generate a Flip Flop. Although it can generate a FF if required. ACT2 and ACT3 that has separate FF module is used for Sequential Circuits.

**Timing Models & Critical Path**

Exact timing (delays) on any FPGA chip cannot be estimated until place and routing step has been performed. This is due to the delay of the interconnect. A critical path of SE in is shown on the next slide.
Actel ACT3 timing model

Model with numerical values

Taking S-module as one sequential cct.
### TABLE 5.2 ACT 3 timing parameters* [1]

<table>
<thead>
<tr>
<th>Family</th>
<th>Delay*</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT 3-3 (data book)</td>
<td>$t_{PD}$</td>
<td>2.9</td>
<td>3.2</td>
<td>3.4</td>
<td>3.7</td>
<td>4.8</td>
</tr>
<tr>
<td>ACT3-2 (calculated)</td>
<td>$t_{PD}/0.85$</td>
<td>3.41</td>
<td>3.76</td>
<td>4.00</td>
<td>4.35</td>
<td>5.65</td>
</tr>
<tr>
<td>ACT3-1 (calculated)</td>
<td>$t_{PD}/0.75$</td>
<td>3.87</td>
<td>4.27</td>
<td>4.53</td>
<td>4.93</td>
<td>6.40</td>
</tr>
<tr>
<td>ACT3-Std (calculated)</td>
<td>$t_{PD}/0.65$</td>
<td>4.46</td>
<td>4.92</td>
<td>5.23</td>
<td>5.69</td>
<td>7.38</td>
</tr>
</tbody>
</table>

* $V_{DD} = 4.75$ V, $T_J$ (junction) = 70 °C. **Logic module + routing delay.** All propagation delays in nanoseconds.

* The Actel '1' speed grade is **15 %** faster than 'Std'; '2' is **25 %** faster than 'Std'; '3' is **35 %** faster than 'Std'.

---
### ACT timing parameters...

**TABLE 5.3 ACT 3 Derating factors* [1]**

<table>
<thead>
<tr>
<th>$V_{DD}$ / V</th>
<th>−55</th>
<th>−40</th>
<th>0</th>
<th>25</th>
<th>70</th>
<th>85</th>
<th>125</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>0.72</td>
<td>0.76</td>
<td>0.85</td>
<td>0.90</td>
<td>1.04</td>
<td>1.07</td>
<td>1.17</td>
</tr>
<tr>
<td>4.75</td>
<td>0.70</td>
<td>0.73</td>
<td>0.82</td>
<td>0.87</td>
<td>1.00</td>
<td>1.03</td>
<td>1.12</td>
</tr>
<tr>
<td>5.00</td>
<td>0.68</td>
<td>0.71</td>
<td>0.79</td>
<td>0.84</td>
<td>0.97</td>
<td>1.00</td>
<td>1.09</td>
</tr>
<tr>
<td>5.25</td>
<td>0.66</td>
<td>0.69</td>
<td>0.77</td>
<td>0.82</td>
<td>0.94</td>
<td>0.97</td>
<td>1.06</td>
</tr>
<tr>
<td>5.5</td>
<td>0.63</td>
<td>0.66</td>
<td>0.74</td>
<td>0.79</td>
<td>0.90</td>
<td>0.93</td>
<td>1.01</td>
</tr>
</tbody>
</table>

- **Worst-case (Commercial):** $V_{DD} = 4.75$ V, $T_A$ (ambient) = +70 °C. Commercial: $V_{DD} = 5$ V ± 5 %,
- **T_A** (ambient) = 0 to +70 °C. Industrial: $V_{DD} = 5$ V ± 10 %, $T_A$ (ambient) = −40 to +85 °C.
- **Military** $V_{DD} = 5$ V ± 10 %, $T_C$ (case) = −55 to +125 °C.
Look Up Table (LUT)

A k input LUT can implement any Boolean function of k variables. The inputs are used as addresses that can retrieve the $2^k$ by 1-bit memory that stores the truth table of the Boolean function.

Since the size of the memory increases with the number of inputs, k, in order to optimize this mapping and reduce the size of the memory there are a variety of algorithms that map a Boolean network, from a given equation, into a circuit of k-input LUT. These algorithms minimize either the total number of LUTs or the number of levels of LUTs in the final circuit. Minimizing the total number of LUTs reduces the CLB requirements while minimizing the levels of LUTs improves the delay.
\[ f_1 = (abc + def) (g + h + i) (jk + lm) \]

This can be implemented by Four 5 input LUT
LookUp Tables: LUT...

Function to be implemented:

\[ f_1 = x_1 \cdot x_2 + x_1 \cdot x_2 \]

Two input LUT
Before programming

Storage Cell contents in the LUT
After programming

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Storage Cell contents in the LUT
After programming

\[ f_1 = x_2 \cdot x_1 + x_2 \cdot x_1 \]
Xilinx uses the configuration cell, ie a static ram shown to store a ‘1’ or ‘0’ to drive the gates of other transistors on the chip to on or off to make connections or to break the connections.

The cell is constructed from two cross-coupled Inverters and uses standard CMOS process.

This method has the advantage or immediate re-programmability. By changing the configuration cells new designs can be implemented almost immediately. New designs encoded in a bit patterns can be sent directly by any sort of mail if needed.

The disadvantage of using SRAM technology is it is a volatile technology. If power is turned off then, the information is lost. Alternatively, configuration data can be loaded from a permanently programmed memory (PROM) so that every time the system is turned on, the information regarding cells are down loaded automatically.

The SRAM based FPGAs have a larger area overhead (5Transistors/cell) than the fused or anti fused devices. Advantages are fast programmability and uses standard CMOS Process
An anti fuse is normally an open circuit until a programming current is forced though it (about 5mA @ 18 Volts).

The two prominent methods are Poly to Diffusion (Actel) and Metal to Metal (Via Link). In a Poly-diffusion anti fuse the high current density causes a large power dissipation in a small area. Once fused the contact is permanent.
Anti fuse (Actel)....

This will melt a thin insulating dielectric between polysilicon and diffusion and form a thin (about 20nm in diameter) permanent, and resistive silicon link. The programming process also drives dopand atoms from the poly and diffusion electrodes. The fabrication process and Programming current controls the average resistance of blown anti fuses.

<table>
<thead>
<tr>
<th>Actel Device</th>
<th># of Anti fuses</th>
<th>% Blown Anti fuses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1010</td>
<td>112,000</td>
<td></td>
</tr>
<tr>
<td>A1225</td>
<td>250,000</td>
<td></td>
</tr>
<tr>
<td>A1280</td>
<td>750,000</td>
<td></td>
</tr>
</tbody>
</table>

To design and program an Actel FPGA, designers iterate between design entry and simulation when design is verified both by functional tests. **Once a designer has completed place-and-route using Actel's Designer software and verified its timing, the program generates an AFM (Actel fuse map) programming file.** The Chip is plugged into a socket on a special programming box that generates the programming voltage.
Anti fuse (Actel)....

Metal-Metal Anti fuse (Via Link)

Same principle as previous slide but different process with 2 main advantages

1) Direct metal to metal eliminating connection between poly and metal or diffusion to metal thus reducing parasitic capacitance and interconnect space requirement.

2) Lower resistance.

Routing wires

Routing wires

% Blown Anti fuses

Anti fuse Resistance Ω

Thin amorphous Si

M3

M2

M2

M3

4λ

2λ

4λ
Alterna MAX 5K and Xilinx ELPDs both use UV-erasable “electrically programmable read-only` memory” (EPROM) cells as their programming technology. The EPROM cell is almost as small as an anti fuse.

---

EEPROM is a special process that the transistor has double gate one for selection and a floating gate for programming./re-programming. Disadvantage is slow re-configuration time, high ON_Resistance due to the floating gate, high static power consumption. Advantages being non-volatile.
Altera MAX 5K and Xilinx ELPDs both use UV-erasable “electrically programmable read-only memory” (EPROM) cells as their programming technology. The EPROM cell is almost as small as an anti fuse.

An EPROM looks like a normal transistor except it has a second floating gate.

(a) Applying a programming voltage $V_{pp} (>12)$ to the drain of the n-channel, programs the cell. A high electric field causes electrons flowing towards the drain to move so fast they “jump” across the insulating gate oxide where they are trapped on the bottom of the floating gate.

(b) Electrons trapped on the floating gate raise the threshold voltage. Once programmed an n-channel EPROM remains off even with $V_{dd}$ applied to the gate. An unprogrammed n-channel device will turn on as normal with a top-gate voltage $V_{dd}$.

(c) Exposure to an ultra-violet (UV) light will erase the EPROM cell. An absorbed light quantum gives an electron enough energy to jump for the floating gate.
EPLD package can be bought in a windowed package for development, erase it and use it again. Programming EEPROM transistors is similar to programming an UV-erasable EPROM transistor, but the erase mechanism is different. In an EEPROM transistor and electric field is also used to remove electrons from the floating gate of a programmed transistor. This is faster than the UV-procedure and the chip doesn’t have to removed from the system.
### EPROM and EEPROM....

<table>
<thead>
<tr>
<th>Programming Technology</th>
<th>Volatile</th>
<th>Re-Program.</th>
<th>Chip Area</th>
<th>R(ohms)</th>
<th>C(FF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static RAM Cells</td>
<td>yes</td>
<td>In circuit</td>
<td>Large</td>
<td>1-2K</td>
<td>10-20 ff</td>
</tr>
<tr>
<td>PLICE Anti-fuse</td>
<td>no</td>
<td>no</td>
<td>Small anti-Fuse. Large Prog. Trans.</td>
<td>300-500</td>
<td>3-5ff</td>
</tr>
<tr>
<td>Via Link Anti-fuse</td>
<td>no</td>
<td>no</td>
<td>Small anti-Fuse. Large Prog. Trans.</td>
<td>50-80</td>
<td>1.3ff</td>
</tr>
<tr>
<td>EPROM</td>
<td>no</td>
<td>Out of Circuit</td>
<td>Small</td>
<td>2-4K</td>
<td>10-20ff</td>
</tr>
<tr>
<td>EEPROM</td>
<td>no</td>
<td>In Circuit</td>
<td>2x EPROM</td>
<td>2-4K</td>
<td>10-20ff</td>
</tr>
</tbody>
</table>

Table 2.1 Characteristics of Programming Technologies
- Can be static RAM cells, Anti fuse, EPROM transistor and EEPROM transistors.

- The programming elements are used to implement the programmable connections among the FPGA’s logic blocks, and a typical FPGA may contain some 5000,000 programming elements.

  • The programming element should consume as little chip area as possible.
  • The programming element should have a low “ON” resistance and very high “OFF” resistance.
  • The programming element contributes low parasitic capacitance to the wiring.
  • It should be possible to reliably fabricate a large number of programming elements on a single chip
  • Re-programmability is derived features for these elements.
FPGAs

Implementation Architecture:
- Symmetrical Array
- Row based
- Hierarchical PLD
- Sea of Gates

Logic Implementation:
- Look Up Table
- Multiplexer based
- PLD Block
- NAND gates

Technology of Interconnection:
- Static RAM
- Anti fuse
- EPROM
- EEPROM
Modern hardware development process is based on HDL designs and IP core.
Standard cell ASIC design cost is in millions of dollars, effectively making the FPGA the best alternative.
Example the XILINX Vertix 4 FF series has the following COREs:
PowerPC® processors (with a new APU interface), tri-mode Ethernet MACs, 622 Mb/s to 6.5 Gb/s serial transceivers, dedicated DSP slices, high-speed clock management circuitry, and source-synchronous interface blocks, 18 x 18, two’s complement, signed Multiplier with Optional pipeline stages
And Built-in Accumulator (48-bit) Adder/Subtractor
Embedded Units

In more complex FPGAs there are many specialized circuitry, particularly for DSP. These include a variety of Adders, Multipliers, Processors, Memory, Digital to Analog converters and so on.

For example:

- Memory units of RAM 16 K to 10M RAM with different organizations
- Multipliers 25 * 18 bits or 18 * 18 bits multipliers from Xilinx, and Altera
- Adders A variety of adders example 48 bit adders from Xilinx
- Processors MicroBlaze, IBM Power PC, Pico Blaze, from Xilinx
- – ARM 9, Nios, MIPS from Altera
FPGA Growth

- FPGA market is expected to reach USD 7.23 Billion by 2022
- A key influencing factor for the growth of FPGAs is their fast time-to-market (TTM). FPGAs offer fastest TTM, compared to their counterparts ASICs and ASSPs?
- ASIC: Application Specific Integrated Circuits.
- ASSP : An application specific standard product.
- ASSP is an integrated circuit that implements a specific function that is used in a wide market.
- ASICs combine a collection of functions and are designed by or for one customer.
the global FPGA market is categorized as:

Telecommunication, Military & Aerospace, Consumer Electronics, Industrial, Automotive, Medical, Computing, Others
The FPGA Cores (IP modules):
- Prevents others from looking inside the core to see how they work.
- Many FPGAs have an array of IP (Pre made modules) can perform many complicated tasks.
- Example: IP modules that implement a soft CPU that can be used as a general computer.

FPGA Advantages
- The design can be written, tested and simulated on the computer.
- The Verified designs can be portable to other FPGA devices, for repeatable and rapid deployment
- Multiple people can work on the same HDL files and increase the speed of circuit development.
- FPGA can be rewritten as many times as needed. The flash memory, which stores the program to configure the FPGA on power up, will be the limiting factor, with a re-write limit of about 100,000.
- There are Many FPGA coming to the market every day What I gave you is a basic principal Always keep uptodate and choose the right FPGA that fits your requirements
- Example: Current Xilinx product portfolio based on 28nm and 20nm planar and 16Fin FET+ technologies and keeps changing
# FPGA Comparison Table

<table>
<thead>
<tr>
<th>Features</th>
<th>Artix-7</th>
<th>Kintex-7</th>
<th>Virtex-7</th>
<th>Spartan-6</th>
<th>Virtex-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cells</td>
<td>352,000</td>
<td>480,000</td>
<td>2,000,000</td>
<td>150,000</td>
<td>760,000</td>
</tr>
<tr>
<td>BlockRAM</td>
<td>19Mb</td>
<td>34Mb</td>
<td>68Mb</td>
<td>4.8Mb</td>
<td>38Mb</td>
</tr>
<tr>
<td>DSP Slices</td>
<td>1,040</td>
<td>1,920</td>
<td>3,600</td>
<td>180</td>
<td>2,016</td>
</tr>
<tr>
<td>DSP Performance (symmetric FIR)</td>
<td>1,248GMACS</td>
<td>2,845GMACS</td>
<td>5,335GMACS</td>
<td>140GMACS</td>
<td>2,419GMACS</td>
</tr>
<tr>
<td>Transceiver Count</td>
<td>16</td>
<td>32</td>
<td>96</td>
<td>8</td>
<td>72</td>
</tr>
<tr>
<td>Transceiver Speed</td>
<td>6.6Gb/s</td>
<td>12.5Gb/s</td>
<td>28.05Gb/s</td>
<td>3.2Gb/s</td>
<td>11.18Gb/s</td>
</tr>
<tr>
<td>Total Transceiver Bandwidth (full duplex)</td>
<td>211Gb/s</td>
<td>800Gb/s</td>
<td>2,784Gb/s</td>
<td>50Gb/s</td>
<td>536Gb/s</td>
</tr>
<tr>
<td>Memory Interface (DDR3)</td>
<td>1,066Mb/s</td>
<td>1,866Mb/s</td>
<td>1,866Mb/s</td>
<td>800Mb/s</td>
<td>1,066Mb/s</td>
</tr>
<tr>
<td>PCI Express® Interface</td>
<td>Gen2x4</td>
<td>Gen2x8</td>
<td>Gen3x8</td>
<td>Gen1x1</td>
<td>Gen2x8</td>
</tr>
<tr>
<td>Agile Mixed Signal (AMS)/XADC</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Configuration AES</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>I/O Pins</td>
<td>600</td>
<td>500</td>
<td>1,200</td>
<td>576</td>
<td>1,200</td>
</tr>
<tr>
<td>I/O Voltage</td>
<td>1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V</td>
<td>1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V</td>
<td>1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V</td>
<td>1.2V, 1.5V, 1.8V, 2.5V, 3.3V</td>
<td>1.2V, 1.5V, 1.8V, 2.5V</td>
</tr>
<tr>
<td>EasyPath Cost Reduction Solution</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## FPGAs…[1]

<table>
<thead>
<tr>
<th>Company</th>
<th>General Architecture</th>
<th>Logic Block Type</th>
<th>Programming Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xilinx</td>
<td>Symmetrical Array</td>
<td>Look-up Table</td>
<td>Static RAM</td>
</tr>
<tr>
<td>Actel</td>
<td>Row-based</td>
<td>Multiplexer-Based</td>
<td>Anti-fuse</td>
</tr>
<tr>
<td>Altera</td>
<td>Hierarchical-PLD</td>
<td>PLD Block</td>
<td>EPROM</td>
</tr>
<tr>
<td>Plessey</td>
<td>Sea-of-Gates</td>
<td>NAND-gate</td>
<td>Static RAM</td>
</tr>
<tr>
<td>PLUS</td>
<td>Hierarchical-PLD</td>
<td>PLD Block</td>
<td>EPROM</td>
</tr>
<tr>
<td>AMD</td>
<td>Hierarchical-PLD</td>
<td>PLD Block</td>
<td>EEPROM</td>
</tr>
<tr>
<td>QuickLogic</td>
<td>Symmetrical Array</td>
<td>Multiplexer-Based</td>
<td>Anti-fuse</td>
</tr>
<tr>
<td>Algotronix</td>
<td>Sea-of-gates</td>
<td>Multiplexers &amp; Basic Gate</td>
<td>Static RAM</td>
</tr>
<tr>
<td>Concurrent</td>
<td>Sea-of-gates</td>
<td>Multiplexers &amp; Basic Gate</td>
<td>Static RAM</td>
</tr>
<tr>
<td>Crosspoint</td>
<td>Row-based</td>
<td>Transistors Pairs &amp; Multiplexers</td>
<td>Anti-fuse</td>
</tr>
</tbody>
</table>

Table 2.2 Summary of Commercially Available FPGAs
<table>
<thead>
<tr>
<th>Lead Type</th>
<th>40-Lead DIP</th>
<th>44-Lead PLCC</th>
<th>132-Lead PQFP</th>
<th>40 lead</th>
<th>132 lead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resistance, mΩ</td>
<td>125</td>
<td>98</td>
<td>102</td>
<td>3.6</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>98</td>
<td>101</td>
<td>2.2</td>
<td>8.2</td>
</tr>
<tr>
<td>Inductance, nH</td>
<td>22</td>
<td>4.6</td>
<td>10</td>
<td>2.1</td>
<td>6.7</td>
</tr>
<tr>
<td></td>
<td>3.9</td>
<td>3.3</td>
<td>7.2</td>
<td>.8</td>
<td>5.1</td>
</tr>
<tr>
<td>Capacitance, pF</td>
<td>.68</td>
<td>.12</td>
<td>.12</td>
<td>.21</td>
<td>.04</td>
</tr>
<tr>
<td></td>
<td>.12</td>
<td>.16</td>
<td>.15</td>
<td>.02</td>
<td>.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.08</td>
</tr>
</tbody>
</table>

**DIP** (Dual In-line Package)  
**PLCC** (Plastic Leaded Chip Carrier)  
**PQFP** (Plastic Quad Flat Package)  
**TAB** (Taped Automated Bonding)
Classic Package Hierarchy
[Intel Corp.]
Area Array Packages

Cross Section of Flip-Chip Ball Grid Array (FC-BGA)
Which Package should we select?

- Industry trend is going for **Area Array Packages**
  - Bond wires contribute parasitic inductance
  - According to some policies, industry is urged to use pb-Free products
  - The number of needed pins growing up
- Packaging Innovations
  - System In Package (SiP)
  - Wafer Level Package (WLP)
http://electronics.stackexchange.com/questions/128120/reason-of-multiple-gnd-and-vcc-on-an-ic (90nm Technology)
Reasons for having multiple supply lines.

- Current has to be distributed, it is impractical that any pad can take the total current. The resistance drop is prohibiting.

- Power coming in from any one pin will probably have to snake it’s away around a lot of stuff to get to every part of the device. Multiple power lines gives the device multiple avenues to pull power from, which keeps the voltage from dipping as much during high current events.

- Need for a clean supply voltage at certain areas.

- Analog devices require special attention and probably different voltage supply.

- Heat distribution, and removal.
The figure represents all of the power and ground pins on a Virtex 4 FPGA in a BGA package with 1513 pins. The FPGA can draw up to 30 or 40 amps at 1.2 volts. Every I/O pin is adjacent to at least one power or ground pin, minimizing the inductance and therefore the generated crosstalk.

Today’s FPGAs structure

Today’s generation of FPGAs consist of various mixes of configurable embedded IPs (large blocks) such as: SRAM, transceivers, I/Os, logic blocks, arithmetic units such as adders and multipliers, microprocessors and routing. Most FPGAs contain programmable logic components called logic elements (LEs) and a hierarchy of reconfigurable interconnects. You can configure LEs to perform complex combinational functions, or merely simple logic gates. Most FPGAs include memory elements, which may be simple flipflops or complete blocks of memory.
Altera’s Stratix’ advertisement

Highest bandwidth, highest integration 28-nm FPGAs with ultimate flexibility
New class of application-targeted devices with integrated 28-Gbps and backplane-capable 12.5-Gbps transceivers, integrated hard intellectual property (IP) blocks including Embedded HardCopy® Blocks, and user-friendly partial reconfiguration
30% lower total power compared to Stratix® IV FPGAs
Low-risk, low-cost path to HardCopy ASICs for higher volume production
28-nm FPGAs providing industry’s lowest system cost and power
Six variants offer mix of logic, 3.125-Gbps or 5-Gbps transceivers, and single- or dual-core ARM Cortex-A9 hard processor system
Delivers up to 40 percent lower total power and up to 30 percent lower static power vs. the previous generation
High level of integration with abundant hard IP blocks
Altera’s Cyclone II FPGA Starter Development Board (around $200.)
As it matures, the cost of 20 nm technology may never cross over the cost of 28 nm technology.

- Transistor cost = F (yield(t), scaling factor, wafer cost)
- Crossover not quite on the 2-year (8Q) cadence
- 20 or 14 nm cost barely goes below the previous one, no saving
The method of update allows customers to migrate their 20nm designs and benefit from the performance per watt advantages of FinFET technology.
2D vs. 2.5D vs. 3D ICs 101

By: Clive Maxfield  4/8/2012 12:08 PM EDT

Birds-eye view of circuit board with individually packaged chips

Birds-eye view of circuit board with a System-on-Chip (SoC) device

Birds-eye view of circuit board with a System-in-Package (SiP) device
3D Structures

A simple form of 3D IC/SiP

A more complex “True 3D IC/SiP”

Connecting dice using wires running down the sides 3D stack
Project Team work
Many views of the same Object
FINAL WORD

• Thank you for being good students.
• I hope you have learned something in this class, that it will be useful in your future endeavor.
• Always go to the root of any problem that you are solving, whether engineering or social.
• Be a Good engineer, Never forget your Engineering ethics.
• Always keep your mind open to new ideas and development, and have vision as were the world is heading and try to be there before others.
• Do NOT forget the “environment”.
• Be a team player.
• Always be a dignified Engineer, respect yourself and other people’s dignity.
• Be just to yourself and give justice to others.
• Always Have good intentions with your thinking,
• actions and speaking.

THANK YOU
References

General Architecture of Xilinx FPGAs

I/O Block

Vertical Routing Channel

Horizontal Routing Channel

Configurable Logic Block

General Architecture of Xilinx FPGAs
Basic logic cells CLBs (Configurable Logic Blocks) are bigger and more complex than the Actel or Quick Logic cells. The Xilinx LCA basic cell is an example of a coarse grain architecture that has both combinational logic and Flip Flop (FF).

The XC3000 has five logic inputs, as common clock, FF, MUXs,……Using programmable MUXs connected to the SRAM programming cells, outputs of two CLBs X and Y can be independently connected to the outputs of FF Qx and Qy or to the outputs of the Combinational Logic F & G.

A 32-bit Look Up Table (LUT) stored in 32 bits of SRA, provides the ability to implement combinational logic. If 5-input AND is being implemented for e.g. $F = ABCDE$. The content of LUT cell number 31 in the 32-bit SRAM is then set to ‘1’ and all other SRAM cells are set to ‘0’. When the input variables are applied it will act as a 5-input AND. This means that the CLB propagation delay is fixed equal to the SRAM Access time.
Xilinx Design Flow

1. Design Specification
2. VHDL Code
3. Simulation Using Modelsim VHDL system simulator
4. Synthesizing Using XST
5. Implementaion Using Xilinx ISE
6. Generate Circuit.bit
7. Target Technology Xilinx xc2s100-5q144 FPGA

- Test bench output files
- Report Files
- Report files

Report Files

96
There are seven inputs in XC3000 CLB, the 5 inputs A→E and the FF outputs. LUT can be broken into two halves and two functions of four variables each can be implemented. Instead. Two of the inputs can be chosen from 5 CLB inputs (A-E) and then one function output connects to F and the other output connects to G.

There are other methods of splitting the LUT...
Extra Circuitry in FPGA logic block
The LUT can generate any function of up to four variables or any two functions of three variables. Outputs can be also registered.
XC2000 Interconnect

Long Lines

Connection to CLB not shown for clarity

Direct Interconnect

General Purpose Interconnect
PLA Example

\[ P_1 = x_1 x_2 \]
\[ P_2 = x_1 \overline{x}_3 \]
\[ P_3 = \overline{x}_1 x_2 x_3 \]
\[ P_4 = x_1 x_3 \]

\[ f_1 = x_1 x_2 + x_1 \overline{x}_3 + \overline{x}_1 \overline{x}_2 x_3 \]

\[ f_2 = x_1 x_2 + x_1 \overline{x}_3 + \overline{x}_1 \overline{x}_2 x_3 + x_1 x_3 \]
Design a PLA, PAL and ROM at a gate level to realize the following sum of product functions:

\[
X(A,B,C) = A \cdot B + A \cdot B \cdot C + A \cdot B \cdot C
\]

\[
Y(A,B,C) = A \cdot B + A \cdot B \cdot C
\]

\[
Z(A,B,C) = A + B
\]
ROM Implementation

- $X = \sum m_6, m_7$
- $Y = \sum m_6, m_7$
- $Z = \sum m_7, m_6, m_5, m_4, m_3, m_2$

- Fixed ✧ programmed
PAL Implementation

Product terms
ABC, AB, A, B

• Fixed  ♦ programmed
Example....

PLA Implementation

Product terms
ABC, AB, A, B

• Fixed  ♦ programmed
4 way to arrange single 1’s

4 way to arrange two 1’s

6 ways to arrange two 1’s

All 1’s

All 0’s
F = a' (b' c + b d) + a (e' f + e g)
Three-input LUT

read/write

D
Data

Q

Q

x₁

x₂

x₃

0/1

0/1

0/1

0/1

0/1

0/1

0/1

f