Chapter 10: Design Options of Digital Systems

Prof. Soo-Ik Chae
Objectives

After completing this chapter, you will be able to:

- Describe the features of system-level design options
- Describe basic structures and features of cell-based ASICs
- Describe basic structures and features of gate array ASICs
- Describe basic structures and features of programmable logic devices (PLDs)
- Describe basic structures and features of field-programmable gate arrays (FPGAs)
- Understand how to model PLAs (including PLDs)
- Understand the issues of voltage tolerance and voltage compliance
Chapter 10: Design Options of Digital Systems

Implementation Options of Digital Systems

Design options of digital systems

- ASICs
- μP/DSP
- Field-programmable devices

- Full-custom
- Cell-based
- Gate array (GA)

- PLD
- FPGA
- CPLD

- Cell library
- Compiled macros (RAM/ROM/PLA)
- Platform IP

- Pure FPGA
- DSP FPGA
- Platform FPGA
System-Level Design Options

- **PCB (printed-circuit board)-based platforms**
  - They consist of discrete standard components such as μp/μC, peripherals, and memory devices.
  - NRE cost is low but final product may cost too much to be accepted.
  - It might need additional CPLD/FPGA to customize the required logic.

- **SoPC (System on a programmable chip)-based platforms**
  - They consist of hard or soft IP being configured into a system required for a given application.
  - It might need additional customized logic modules.
  - Examples are Xilinx VirtexII pro, Spartan 3 (MicroBlaze), and Altera Cyclone Series (NIOS).

- **Soft IP+ cell_library-based platforms**
  - They consist of soft IP being configured into an optimized system hardware.
  - An example is Tensilica’s Xtensa.
Design Alternatives of Digital Systems

- **PLDs**
- **FPGAs, GAs**
- **Cell-based ASICs**
- **Full-custom ASICs**

**Time to market, Cost**

**Design flexibility, Process complexity, Density, Speed, Complexity, NRE cost**
Comparison of Design Options

(a) Cost versus product volume

(b) Comparison of various digital system design options

- Full-custom ASICs
- Cell-based ASICs
- GAs
- FPGAs/CPLDs
- PLDs
Basic Structure of Cell-Based ASICs
## Basic Cell Types

<table>
<thead>
<tr>
<th>Standard cell types</th>
<th>Variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter/buffer/tristate buffer</td>
<td>1X, 2X, 4X, 8X, 16X</td>
</tr>
<tr>
<td>NAND/AND gate</td>
<td>2 ~ 8 inputs</td>
</tr>
<tr>
<td>NOR/OR gate</td>
<td>2 ~ 8 inputs</td>
</tr>
<tr>
<td>XOR/XNOR gate</td>
<td>2 ~ 8 inputs</td>
</tr>
<tr>
<td>MUX/DeMUX</td>
<td>2 ~ 8 inputs (inverted/noninverted output)</td>
</tr>
<tr>
<td>Encoder/Decoder</td>
<td>4 ~ 16 inputs (inverted/noninverted output)</td>
</tr>
<tr>
<td>Schmitt trigger circuit</td>
<td>Inverted/noninverted output</td>
</tr>
<tr>
<td>Latch/register/counter</td>
<td>D/JK(sync./async. clear/reset)</td>
</tr>
<tr>
<td>I/O pad circuits</td>
<td>Input/Output(tristate/bidirectional)</td>
</tr>
</tbody>
</table>
A Cell-Based D-Type Flip Flop

Abutting: fixed height
Basic Structures of Gate Arrays

- pMOS
- nMOS
- Gate
- $V_{DD}$
- $V_{SS}$
Basic Structures of Gate-Array ASICs

Sea of gates? No routing channels
Basic Structures of Gate Arrays

(a) NOT gate

(b) NAND gate

V_{DD} V_{SS} V_{DD} V_{SS}

Input
Output
Input
Output

pMOS nMOS

Input
Gate
Output

Input
Gate
Output

V_{DD} V_{SS} V_{DD} V_{SS}
Basic Structures of Programmable Logic Devices

(a) AND gate

(b) OR gate

Fuse broken

Shorthand notations for describing the structures of PLDs
Basic Structures of Programmable Logic Devices

(a) ROM structure

(b) PLA structure

(c) PAL structure
Basic Structures of CPLDs

(a) CPLD basic structure

(b) pLSI basic structure
Basic Structures of FPGAs

(a) Matrix type

(b) Row type
Basic Structures of Programmable Interconnections

(a) SRAM device

(b) Flash device

(c) Antifuse device
### Comparison of programmable interconnections

<table>
<thead>
<tr>
<th>Process technology</th>
<th>SRAM</th>
<th>Flash</th>
<th>Antifuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming approach</td>
<td>CMOS</td>
<td>Standard 2-level polysilicon</td>
<td>New type polysilicon</td>
</tr>
<tr>
<td></td>
<td>Shift register</td>
<td>FAMOS</td>
<td>Avalanche</td>
</tr>
<tr>
<td>Area</td>
<td>Very large</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Capacitance</td>
<td>≈ 2 kΩ</td>
<td>≈ 2 kΩ</td>
<td>≈ 500 Ω</td>
</tr>
<tr>
<td>Resistance</td>
<td>≈ 50 fF</td>
<td>≈ 15 fF</td>
<td>≈ 5 fF</td>
</tr>
</tbody>
</table>
Basic Structures of ROMs

- AND array
- OR array
- Chip enable (CE)
- Output enable (OE)
- n-input lines
- m-output lines
- n×2^n decoder

Diagram showing the connections and components of a ROM (Read-Only Memory) with n-input lines, m-output lines, and a decoder.
Basic Applications of ROMs

(a) Truth table

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁ A₀</td>
<td>O₁ O₀</td>
</tr>
<tr>
<td>0  0</td>
<td>0  1</td>
</tr>
<tr>
<td>0  1</td>
<td>1  0</td>
</tr>
<tr>
<td>1  0</td>
<td>0  0</td>
</tr>
<tr>
<td>1  1</td>
<td>1  1</td>
</tr>
</tbody>
</table>

(b) Logic circuit

Fuse broken

Fuse intact

2 x 4 decoder

O₁ O₀
### Basic Structures of PLAs

Programmable AND gate array

- $n$ input lines
- $m$ output lines
- $k$ product terms

Programmable OR gate array

- $n$ input lines
- $m$ output lines
- $k$ product terms
Basic Applications of PLAs

(a) PLA programming table

<table>
<thead>
<tr>
<th>Product term</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>xy</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x'y</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>wx'y'</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>w'xz</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>wxz</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Logic circuit

Digital System Designs and Practices Using Verilog HDL and FPGAs @ 2008, John Wiley
## Basic Structures of PALs

<table>
<thead>
<tr>
<th>PAL devices</th>
<th>Package pins</th>
<th>AND-gate inputs</th>
<th>Primary inputs</th>
<th>Bidirectional I/Os</th>
<th>Registered outputs</th>
<th>Combinational outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL16L8</td>
<td>20</td>
<td>16</td>
<td>10</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>PAL16R4</td>
<td>20</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>PAL16R6</td>
<td>20</td>
<td>16</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>PAL16R8</td>
<td>20</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>PAL20L8</td>
<td>24</td>
<td>20</td>
<td>14</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>PAL20R4</td>
<td>24</td>
<td>20</td>
<td>12</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>PAL20R6</td>
<td>24</td>
<td>20</td>
<td>12</td>
<td>2</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>PAL20R8</td>
<td>24</td>
<td>20</td>
<td>12</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>
A PAL Example --- 16R8  Chapter 10: Design Options of Digital Systems
PLA Modeling

- PLA device structures can be one of the following types:
  - **SOP (sum of product)** are composed of an AND plane and an OR plane.
    - AND plus OR array.
    - NAND plus NAND array.
  - **POS (product of sum)** are composed of an OR plane and an AND plane.
    - OR plus AND array.
    - NOR plus NOR array.
- An AND plane or an OR plane is also called a **personality** array (or memory).
PLA Modeling (p.303 in LRM)

- A PLA device
  - is modeled by a group of system tasks.
  - is modeled by an appropriate combination of two system tasks.

- A PLA device can be modeled as:
  - **asynchronous**: the evaluations are executed whenever any input is changed.
  - **synchronous**: the evaluation time can be controlled in a predictable way.

- The output terms are updated without any delay.
Chapter 10: Design Options of Digital Systems

PLA Modeling

- PLA modeling system tasks
  - input_terms: nets or variables
  - output_terms: variables
  - logic: and | or | nand | nor

$async$logic$format(mem_id, input_terms, output_terms);
$sync$logic$format(mem_id, input_terms, output_terms);
Array or Plane formats (17.5.4 in LRM)

- Array format: only 1 or 0 is used to denote whether inputs is taken or not
- Plane format: complies with Espresso
  - 1 or 0: true or complement value is taken
  - x: denote that the worst-case input is taken
  - z, ?: denote don’t care

<table>
<thead>
<tr>
<th>Array format</th>
<th>Plane format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>Synchronous</td>
</tr>
<tr>
<td>$\text{async}&amp;\text{array}$</td>
<td>$\text{sync}&amp;\text{array}$</td>
</tr>
<tr>
<td>$\text{async}&amp;\text{nand}\text{array}$</td>
<td>$\text{sync}&amp;\text{nand}\text{array}$</td>
</tr>
<tr>
<td>$\text{async}&amp;\text{or}\text{array}$</td>
<td>$\text{sync}&amp;\text{or}\text{array}$</td>
</tr>
<tr>
<td>$\text{async}&amp;\text{nor}\text{array}$</td>
<td>$\text{sync}&amp;\text{nor}\text{array}$</td>
</tr>
</tbody>
</table>
PLA Modeling

wire a1, a2, a3, a4, a5, a6, a7;
reg b1, b2, b3;
wire [1:7] awire;
reg [1:3] breg;
reg [1:7] mema [1:3]; // define personality memory
   // asynchronous AND plane
   $async$and$array(mema, {a1, a2, a3, a4, a5, a6, a7}, {b1, b2, b3});
   // or using the following statement
   $async$and$array(mema, awire, breg);
   // synchronous AND plane
forever @(posedge clock)
   $sync$and$array(mema, {a1, a2, a3, a4, a5, a6, a7}, {b1, b2, b3});
PLA Modeling

- Logic array personality is declared as an array of regs.
  - The width is equal to the number of input terms.
  - The depth is equal to the number of output terms.
- Logic array personality is loaded into memory using
  - system tasks $readmemb or $readmemh.
  - the procedural assignment statements.
- Logic array personality can be changed dynamically during simulation.
PLA Modeling

- The array format
  - uses only 1 or 0 to denote the input value is taken or not.
- The plane format (complies with Espresso)
  - 1 or 0: the true or complement input value is taken.
  - x: the worst case input value is taken.
  - ? or z: don’t care.
PLA Modeling

- Define PLA personality from file.
  - In this example only one and plane is considered.

```verilog
// an example of the usage of the array format.
module async_array(a1, a2, a3, a4, a5, a6, a7, b1, b2, b3);
input   a1, a2, a3, a4, a5, a6, a7 ;
output b1, b2, b3;
reg      [1:7] mem[1:3];  // memory declaration for array personality
reg      b1, b2, b3;
initial begin
// setup the personality from the file array.dat
$readmemb ("array.dat", mem);
// setup an asynchronous logic array with the input
// and output terms expressed as concatenations
$async$array (mem,{a1, a2, a3, a4, a5, a6, a7},{b1, b2, b3});
end
endmodule
```

\[
\begin{align*}
b1 &= a1 \& a2 \\
b2 &= a3 \& a4 \& a5 \\
b3 &= a5 \& a6 \& a7
\end{align*}
\]

1100000
0011100
0000111
PLA Modeling

module sync_array(a0, a1, a2, a3, a4, a5, a6, a7, b0, b1, b2);
input    a0, a1, a2, a3, a4, a5, a6, a7;
output  b0, b1, b2;
reg       b0, b1, b2;
reg [7:0] mem[0:2];
// an example of the usage of array format
initial begin   // using procedural assignment statements.
    $readmemb ("array.dat", mem);
    forever @(posedge clk)
        $asyncand$array(mem, {a0,a1,a2,a3,a4,a5,a6,a7}, {b0,b1,b2});
end
endmodule
Module pla(a0, a1, a2, a3, a4, a5, a6, a7, b0, b1, b2);
input a0, a1, a2, a3, a4, a5, a6, a7;
output b0, b1, b2;
reg b0, b1, b2;
reg [7:0] mem[0:2];

// an example of the usage of array format
initial begin  // using procedural assignment statements.
    mem[0] = 8'b11001100;
    mem[1] = 8'b00110011;
    mem[2] = 8'b00001111;
$async$and$array(mem, {a0,a1,a2,a3,a4,a5,a6,a7},
    {b0,b1,b2});
end
endmodule

A = {a0, a1, a2, a3, a4, a5, a6, a7}
B = {b0, b1, b2}
mem[0] = 8'b11001100;
mem[1] = 8'b00110011;
mem[2] = 8'b00001111;

A = 11001100 -> B = 100
A = 00110011 -> B = 010
A = 00001111 -> B = 001
A = 10101010 -> B = 000
A = 01010101 -> B = 000
A = 11000000 -> B = 000
A = 00111111 -> B = 011
A PLA Modeling Example --- Logic Circuit

(a) PLA programming table

<table>
<thead>
<tr>
<th>Product</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>$xy'$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$xz$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$y'z$</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) PLA symbolic diagram

\[
f_1(x, y, z) = xy' + xz + x'y'z'
\]

\[
f_2(x, y, z) = xz + y'z
\]

(c) logic circuit
module pla_example(input x, y, z, output reg f1, f2);
// An example of the usage of the array format.
reg   p0, p1, p2, p3;   // internal minterms
reg  [0:5] mem_and[0:3];
reg  [0:3] mem_or[1:2];
wire  x_n, y_n, z_n;
assign x_n = ~x, y_n = ~y, z_n = ~z;
initial begin:
  pla_model_array
    mem_and[0] = 6'b100100;   // define AND plane
    mem_and[1] = 6'b100010;
    mem_and[2] = 6'b000110;
    mem_and[3] = 6'b011001;
$async$and$array(mem_and, {x, x_n, y, y_n, z, z_n}, {p0, p1, p2, p3});
    mem_or[1] = 4'b1101;     // define OR plane
    mem_or[2] = 4'b0110;
$async$or$array(mem_or, {p0, p1, p2, p3}, {f1, f2});
end
endmodule
A PLA Modeling Example – The Plane Format

module pla_example(input x, y, z, output reg f1, f2);
reg p0, p1, p2, p3;       // internal minterms
reg [0:2] mem_and[0:3];  // and plane personality matrix
reg [0:3] mem_or[1:2];   // or plane personality matrix

// An example of the usage of the plane format
initial begin: pla_model_plane
    $async$and$plane(mem_and, {x, y, z}, {p0, p1, p2, p3});
    mem_and[0] = 3'b10?;   // define AND plane
    mem_and[1] = 3'b1?1;
    mem_and[2] = 3'b?01;   // 000110 in array format
    mem_and[3] = 3'b010;
    $async$or$plane(mem_or, {p0, p1, p2, p3}, {f1, f2});
    mem_or[1] = 4'b11?1;   // define OR plane
    mem_or[2] = 4'b?11?;   // 0110 in array format
end
endmodule
Xilinx Basic Structure of CPLD: XC9500XL
CPLD: XC9500XL --- Function Block

From switch matrix 54

Programmable AND array

Product-term allocators

Macrocell 1
Macrocell 2
Macrocell 3

... ...

Macrocell 16
Macrocell 17
Macrocell 18

To switch matrix 18

Output 18

PTOE 18

Global set/reset 1

Global clock 3

To I/O blocks
CPLD: XC9500XL --- Macro of Function Block

Additional product terms (From other macrocells)
Global set/reset
Global clocks

Product-term allocator

Product-term set
Product-term clock enable
Product-term clock
Product-term reset
Product-term OE

54

OUT
PTOE

To switch matrix
To I/O blocks

S
D/T
Q
EC
CK
R
CPLD: XC9500XL --- Switch Matrix
CPLD: XC9500XL --- I/O Block

- Macrocell
- Product-term OE
- PTOE
- OUT
- Slew-rate control
- Programmable ground
- Pull-up resistor
- I/O/GTS1
- I/O/GTS2
- To other I/O blocks
- To switch matrix

Diagram showing the I/O block with connections to other I/O blocks, the switch matrix, and various control signals such as V_{CCINT} and the programmable ground.
MAX7000 Family --- Basic Structure

LAB: logic array block,    PIA: programmable interconnect array
Chapter 10: Design Options of Digital Systems

MAX7000 Family --- Macrocell Structure

Logic array

Parallel logic expanders (from other macrocells)

Global clear

Global clocks

Preset

Clock/Clock enable

Clear

Clear select

Vcc

Programmable flip-flop

From I/O pin

To IOB

To PIA

36 signals from PIA

16 expander product terms

Shared logic expanders

Product-term select matrix

16 expander product terms

36 signals from PIA
MAX7000 Family --- Sharable Expander

36 signals from PIA  16 shared expanders

Use a product term as an input
MAX7000 Family --- Parallel Expander

Use product terms up to 20 from the neighbor macrocell
MAX7000 Family --- IOB
Xilinx Basic Structure of FPGA: XC4000XL
FPGA: XC4000XL --- Configurable Logic Block

Function of G1-G4

Function of F1-F4

Function of F', G', H1

Slew-rate control

Digital System Designs and Practices Using Verilog HDL and FPGAs @ 2008, John Wiley
FPGA: XC4000XL --- PIA

- Interconnection lines:
  - Single-length lines
  - Double-length lines
  - Global long-length lines
FPGA: XC4000XL --- PIA

(a) Single-length lines
(b) Double-length lines

PSM: programmable switch matrix
Chapter 10: Design Options of Digital Systems

FPGA: XC4000XL --- Input/Output Block

Diagram showing the I/O block with various components labeled including output clock, input clock, clock enable, output MUX, input buffer, output buffer, slew-rate control, and pull down/up resistor.
**FPGA: XC4000XL --- Macro Library Example**

- Macros may be categorized into two basic classes:
  - **Soft macros** only describe logic and interconnections.
  - **Hard macros** describe where the elements are placed.

<table>
<thead>
<tr>
<th>Soft macros</th>
<th>Hard macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulators, Counters</td>
<td>Accumulators, Counters</td>
</tr>
<tr>
<td>Data/shift registers</td>
<td>Data/shift registers</td>
</tr>
<tr>
<td>Flip-Flops/latches</td>
<td>RAM</td>
</tr>
<tr>
<td>FSMs</td>
<td>Adders</td>
</tr>
<tr>
<td>RAMs/ROMs</td>
<td>Comparators</td>
</tr>
<tr>
<td>Adder/Subtractors</td>
<td>Dividers</td>
</tr>
<tr>
<td>Comparators</td>
<td>Encoders/decoders</td>
</tr>
<tr>
<td>Dividers</td>
<td>Gates/buffers</td>
</tr>
<tr>
<td>Encoders/decoders</td>
<td>Logical shifters</td>
</tr>
<tr>
<td>Gates/buffers</td>
<td>Multiplexers</td>
</tr>
<tr>
<td>Logical shifters</td>
<td>Multipliers</td>
</tr>
<tr>
<td>Multiplexers</td>
<td>Parity checkers</td>
</tr>
<tr>
<td>Multipliers</td>
<td>Tristate buffers</td>
</tr>
<tr>
<td>Parity checkers</td>
<td></td>
</tr>
<tr>
<td>Tristate buffers</td>
<td></td>
</tr>
</tbody>
</table>
# Features of Spartan-XL and Spartan-II Series

<table>
<thead>
<tr>
<th>Spartan/XL FPGAs</th>
<th>XCS05/XL</th>
<th>XCS10/XL</th>
<th>XCS20/XL</th>
<th>XCS30/XL</th>
<th>XCS40/XL</th>
</tr>
</thead>
<tbody>
<tr>
<td>System gates</td>
<td>5k</td>
<td>10k</td>
<td>20k</td>
<td>30k</td>
<td>40k</td>
</tr>
<tr>
<td>Logic cells</td>
<td>238</td>
<td>466</td>
<td>950</td>
<td>1,368</td>
<td>1,862</td>
</tr>
<tr>
<td>Max. no. of logic gates</td>
<td>3,000</td>
<td>5,000</td>
<td>10,000</td>
<td>13,000</td>
<td>20,000</td>
</tr>
<tr>
<td>Flip-flops</td>
<td>360</td>
<td>616</td>
<td>1,120</td>
<td>1,536</td>
<td>2,016</td>
</tr>
<tr>
<td>Max. RAM bits</td>
<td>3,200</td>
<td>6,272</td>
<td>12,800</td>
<td>18,432</td>
<td>25,088</td>
</tr>
<tr>
<td>Max. available I/O</td>
<td>77</td>
<td>112</td>
<td>160</td>
<td>192</td>
<td>224</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spartan-II FPGAs</th>
<th>XC2S15</th>
<th>XC2S30</th>
<th>XC2S50</th>
<th>XC2S100</th>
<th>XC2S150</th>
<th>XC2S200</th>
</tr>
</thead>
<tbody>
<tr>
<td>System gates</td>
<td>15k</td>
<td>30k</td>
<td>50k</td>
<td>100k</td>
<td>150k</td>
<td>200k</td>
</tr>
<tr>
<td>Logic cells</td>
<td>432</td>
<td>972</td>
<td>1,728</td>
<td>2,700</td>
<td>3,888</td>
<td>5,292</td>
</tr>
<tr>
<td>Blocks RAM (kb)</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
</tr>
<tr>
<td>DLL</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Max. Dist. RAM (kb)</td>
<td>4</td>
<td>13.5</td>
<td>24</td>
<td>37.5</td>
<td>54</td>
<td>73.5</td>
</tr>
<tr>
<td>Max. available I/O</td>
<td>86</td>
<td>132</td>
<td>176</td>
<td>196</td>
<td>260</td>
<td>284</td>
</tr>
</tbody>
</table>

## Spartan/XL FPGAs
- **XCS05/XL**
  - System gates: 5k
  - Logic cells: 238
  - Max. no. of logic gates: 3,000
  - Flip-flops: 360
  - Max. RAM bits: 3,200
  - Max. available I/O: 77

## Spartan-II FPGAs
- **XC2S15**
  - System gates: 15k
  - Logic cells: 432
  - Blocks RAM (kb): 16
  - DLL: 4
  - Max. Dist. RAM (kb): 4
  - Max. available I/O: 86
# Features of Spartan-3 Series

<table>
<thead>
<tr>
<th>Spartan-III FPGAs</th>
<th>XC3 S50</th>
<th>XC3 S200</th>
<th>XC3 S400</th>
<th>XC3 S1000</th>
<th>XC3 S1500</th>
<th>XC3 S2000</th>
<th>XC3 S4000</th>
<th>XC3 S5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>System gates</td>
<td>50k</td>
<td>200k</td>
<td>400k</td>
<td>1000k</td>
<td>1500k</td>
<td>2000k</td>
<td>4000k</td>
<td>5000k</td>
</tr>
<tr>
<td>Logic cells</td>
<td>1,728</td>
<td>4,320</td>
<td>8,064</td>
<td>17,280</td>
<td>29,952</td>
<td>46,080</td>
<td>62,208</td>
<td>74,880</td>
</tr>
<tr>
<td>18 * 18 multipliers</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>96</td>
<td>104</td>
</tr>
<tr>
<td>Blocks RAM (kb)</td>
<td>72</td>
<td>216</td>
<td>288</td>
<td>432</td>
<td>576</td>
<td>720</td>
<td>1,728</td>
<td>1,872</td>
</tr>
<tr>
<td>Max. Dist. RAM (kb)</td>
<td>12</td>
<td>30</td>
<td>56</td>
<td>120</td>
<td>208</td>
<td>320</td>
<td>432</td>
<td>520</td>
</tr>
<tr>
<td>DCMs</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>I/O standards</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Max. Diff. I/O pairs</td>
<td>56</td>
<td>76</td>
<td>116</td>
<td>175</td>
<td>221</td>
<td>270</td>
<td>312</td>
<td>344</td>
</tr>
<tr>
<td>Max. Single ended I/O</td>
<td>124</td>
<td>173</td>
<td>264</td>
<td>391</td>
<td>487</td>
<td>565</td>
<td>712</td>
<td>784</td>
</tr>
</tbody>
</table>
Chapter 10: Design Options of Digital Systems

Features of Virtex-II Series

<table>
<thead>
<tr>
<th>Device</th>
<th>System Gates</th>
<th>Array Row x Col.</th>
<th>Slices</th>
<th>Maximum Distributed RAM Kbits</th>
<th>Multiplier Blocks</th>
<th>SelectRAM Blocks</th>
<th>DCMs</th>
<th>Max I/O Pads(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2V40</td>
<td>40K</td>
<td>8 x 8</td>
<td>256</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>88</td>
</tr>
<tr>
<td>XC2V80</td>
<td>80K</td>
<td>16 x 8</td>
<td>512</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>XC2V250</td>
<td>250K</td>
<td>24 x 16</td>
<td>1,536</td>
<td>48</td>
<td>24</td>
<td>24</td>
<td>8</td>
<td>200</td>
</tr>
<tr>
<td>XC2V500</td>
<td>500K</td>
<td>32 x 24</td>
<td>3,072</td>
<td>96</td>
<td>32</td>
<td>32</td>
<td>8</td>
<td>264</td>
</tr>
<tr>
<td>XC2V1000</td>
<td>1M</td>
<td>40 x 32</td>
<td>5,120</td>
<td>160</td>
<td>40</td>
<td>40</td>
<td>8</td>
<td>432</td>
</tr>
<tr>
<td>XC2V1500</td>
<td>1.5M</td>
<td>48 x 40</td>
<td>7,680</td>
<td>240</td>
<td>48</td>
<td>48</td>
<td>8</td>
<td>528</td>
</tr>
<tr>
<td>XC2V2000</td>
<td>2M</td>
<td>56 x 48</td>
<td>10,752</td>
<td>336</td>
<td>56</td>
<td>56</td>
<td>8</td>
<td>624</td>
</tr>
<tr>
<td>XC2V3000</td>
<td>3M</td>
<td>64 x 56</td>
<td>14,336</td>
<td>448</td>
<td>96</td>
<td>96</td>
<td>12</td>
<td>720</td>
</tr>
<tr>
<td>XC2V4000</td>
<td>4M</td>
<td>80 x 72</td>
<td>23,040</td>
<td>720</td>
<td>120</td>
<td>120</td>
<td>12</td>
<td>912</td>
</tr>
<tr>
<td>XC2V6000</td>
<td>6M</td>
<td>96 x 88</td>
<td>33,792</td>
<td>1,056</td>
<td>144</td>
<td>144</td>
<td>12</td>
<td>1,104</td>
</tr>
<tr>
<td>XC2V8000</td>
<td>8M</td>
<td>112 x 104</td>
<td>46,592</td>
<td>1,456</td>
<td>168</td>
<td>168</td>
<td>12</td>
<td>1,108</td>
</tr>
</tbody>
</table>

(1) Max I/O Pads refers to the maximum number of input/output pins available on the device.
Altera APEX 20K Family --- Basic Structure

Row interconnect

Column interconnect

IOE

MegaLAB

Row interconnect
ESB (embedded system block) can be configured as a block of macrocells or implement various types of memory blocks.
APEX 20K Family --- Logic Element Structure
APEX 20K Family --- Structure of ESB

Dedicated clocks
Global signals
4 4
65
Macrocell array

Local interconnect

From adjacent LAB

Macrocell inputs(1-16)
clk[1:0]
enable[1:0]
CLRN[1:0]

To row and column interconnect

Macrocell array

MegaLAB interconnect
Altera Stratix Family --- Basic Structure