ECE 545
Lecture 8

FPGA Devices
& FPGA Design Flow
Required Reading

Xilinx, Inc.

Spartan-3 FPGA Family

Spartan-3 FPGA Family Data Sheet

Module 1:

• Introduction
• Features
• Architectural Overview
• Package Marking

Module 2:

• CLB Overview
Required Reading

Xilinx, Inc.

Spartan-3 FPGA Family
Spartan-3 Generation FPGA User Guide

Chapter 5 Using Configurable Logic Blocks (CLBs)
Chapter 6 Using Look-Up Tables as Distributed RAM
Chapter 7: Using Look-Up Tables as Shift Registers (SRL16)
Chapter 9: Using Carry and Arithmetic Logic
Two competing implementation approaches

**ASIC**
*Application Specific Integrated Circuit*
- designed all the way from behavioral description to physical layout
- designs must be sent for expensive and time consuming fabrication in semiconductor foundry

**FPGA**
*Field Programmable Gate Array*
- no physical layout design; design ends with a bitstream used to configure a device
- bought off the shelf and reconfigured by designers themselves
What is an FPGA?

- Configurable Logic Blocks
- I/O Blocks
- Block RAMs
Which Way to Go?

**ASICs**
- High performance
- Low power
- Low cost in high volumes

**FPGAs**
- Off-the-shelf
- Low development cost
- Short time to market
- Reconfigurability
Other FPGA Advantages

• Manufacturing cycle for ASIC is very costly, lengthy and engages lots of manpower
  • Mistakes not detected at design time have large impact on development time and cost
  • FPGAs are perfect for rapid prototyping of digital circuits

• Easy upgrades like in case of software

• Unique applications
  • reconfigurable computing
Major FPGA Vendors

SRAM-based FPGAs

• Xilinx, Inc.
• Altera Corp.

{ Share about 85% of the market
• Atmel
• Lattice Semiconductor

Flash & antifuse FPGAs

• Microsemi SoC Products Group (formerly Actel Corp.)
• Quick Logic Corp.
Xilinx

- Primary products: FPGAs and the associated CAD software
- Main headquarters in San Jose, CA
- Fabless* Semiconductor and Software Company
  - UMC (Taiwan) {*Xilinx acquired an equity stake in UMC in 1996}
  - Seiko Epson (Japan)
  - TSMC (Taiwan)
  - Samsung (Korea)
Xilinx FPGA Families

- **Old families**
  - XC3000, XC4000, XC5200
  - Old 0.5\(\mu\)m, 0.35\(\mu\)m and 0.25\(\mu\)m technology. Not recommended for modern designs.

- **High-performance families**
  - Virtex (220 nm)
  - Virtex-E, Virtex-EM (180 nm)
  - Virtex-II (130 nm)
  - Virtex-II PRO (130 nm)
  - Virtex-4 (90 nm)
  - Virtex-5 (65 nm)
  - Virtex-6 (40 nm)

- **Low Cost Family**
  - Spartan/XL – derived from XC4000
  - Spartan-II – derived from Virtex
  - Spartan-IIIE – derived from Virtex-E
  - Spartan-3 (90 nm)
  - Spartan-3E (90 nm) – logic optimized
  - Spartan-3A (90 nm) – I/O optimized
  - Spartan-3AN (90 nm) – non-volatile,
  - Spartan-3A DSP (90 nm) – DSP optimized
  - Spartan-6 (45 nm)
CLB Structure
General structure of an FPGA

Programmable interconnect

Programmable logic blocks
Xilinx Spartan 3 CLB

Configurable logic block (CLB)

- Slice
  - Logic cell
  - Logic cell

- Slice
  - Logic cell
  - Logic cell
CLB Slice = 2 Logic Cells
Xilinx Multipurpose LUT (MLUT)
Spartan 3 CLB Structure

Left-Hand SLICEM
(Logic or Distributed RAM
or Shift Register)

Right-Hand SLICEL
(Logic Only)

Switch Matrix

Interconnect to Neighbors

DS099-2_05_082104
CLB Slice Structure

- Each slice contains two sets of the following:
  - Four-input LUT
    - Any 4-input logic function (16x1 ROM),
    - or 16-bit x 1 sync RAM (SLICEM only)
    - or 16-bit shift register (SLICEM only)
  - Carry & 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
Multipurpose Look-Up Table (MLUT)
MLUT as 16x1 ROM
LUT (Look-Up Table) Functionality

- Look-Up tables are primary elements for logic implementation
- Each LUT can implement any function of 4 inputs
5-Input Functions implemented using two LUTs

- One CLB Slice can implement any function of 5 inputs
- Logic function is partitioned between two LUTs
- F5 multiplexer selects LUT
5-Input Functions implemented using two LUTs
MLUT as 16x1 RAM
**Distributed RAM**

- CLB LUT configurable as Distributed RAM
  - A single LUT equals 16x1 RAM
  - Two LUTs Implement Single and Dual-Port RAMs
  - Cascade LUTs to increase RAM size
- Synchronous write
- Synchronous/Asynchronous read
  - Accompanying flip-flops used for synchronous read
MLUT as 16-bit Shift Register (SRL16)
Shift Register

- Each LUT can be configured as shift register
  - Serial in, serial out
- Dynamically addressable delay up to 16 cycles
- For programmable pipeline
- Cascade for greater cycle delays
- Use CLB flip-flops to add depth
Using Multipurpose Look-Up Tables in the Shift Register Mode (SRL16)

Inferred from behavioral description in VHDL for shift-registers with
- one serial input, one serial output
- no reset, no set
Cascading LUT Shift Registers into Shift Registers Longer than 16 bits
Shift Register

- Register-rich FPGA
  - Allows for addition of pipeline stages to increase throughput
- Data paths must be balanced to keep desired functionality

9-Cycle imbalance
Carry & Control Logic
**Full-adder**

\[ x + y + c_{in} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ c_{out} \end{pmatrix} = \begin{pmatrix} \text{c}_{\text{out}} \\ \text{s} \end{pmatrix} \]
### Full-adder

#### Alternative implementations

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>$c_{out}$</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$c_{in}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$c_{in}$</td>
<td>$\overline{c_{in}}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$c_{in}$</td>
<td>$\overline{c_{in}}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$c_{in}$</td>
</tr>
</tbody>
</table>
Full-adder
Alternative implementations

Implementation used to generate fast carry logic in Xilinx FPGAs

\[
\begin{array}{c|c|c}
  x & y & c_{\text{out}} \\
  0 & 0 & y \\
  0 & 1 & c_{\text{in}} \\
  1 & 0 & c_{\text{in}} \\
  1 & 1 & y \\
\end{array}
\]

\[p = x \oplus y\]
\[g = y\]
\[s = p \oplus c_{\text{in}} = x \oplus y \oplus c_{\text{in}}\]
Carry & Control Logic in Spartan 3 FPGAs

LUT

Hardwired (fast) logic
Simplified View of Spartan-3 FPGA Carry and Arithmetic Logic in One Logic Cell
Simplified View of Carry Logic in One Spartan 3 Slice
\[ T_{\text{ripple-add}} = T_{FA}(x,y \rightarrow c_{out}) + (k - 2) \times T_{FA}(c_{in} \rightarrow c_{out}) + T_{FA}(c_{in} \rightarrow s) \]

**Fig. 5.5**  Critical path in a \( k \)-bit ripple-carry adder.
Critical Path for an Adder Implemented Using Xilinx Spartan 3/Spartan 3E FPGAs
## Number and Length of Carry Chains for Spartan 3 FPGAs

<table>
<thead>
<tr>
<th>Device</th>
<th>Number of Carry Chains</th>
<th>Bits per Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3S50</td>
<td>24</td>
<td>64</td>
</tr>
<tr>
<td>XC3S200</td>
<td>40</td>
<td>96</td>
</tr>
<tr>
<td>XC3S400</td>
<td>56</td>
<td>128</td>
</tr>
<tr>
<td>XC3S1000</td>
<td>80</td>
<td>192</td>
</tr>
<tr>
<td>XC3S1500</td>
<td>104</td>
<td>256</td>
</tr>
<tr>
<td>XC3S2000</td>
<td>128</td>
<td>320</td>
</tr>
<tr>
<td>XC3S4000</td>
<td>144</td>
<td>384</td>
</tr>
<tr>
<td>XC3S5000</td>
<td>160</td>
<td>416</td>
</tr>
</tbody>
</table>
Bottom Operand Input to Carry Out Delay

$T_{OPCYF}$

0.9 ns for Spartan 3
Carry Propagation Delay $t_{BYP}$

0.2 ns for Spartan 3
Carry Input to Top Sum Combinational Output Delay $T_{CINY}$

1.2 ns for Spartan 3
Critical Path Delays and Maximum Clock Frequencies (into account surrounding registers)

- 8 bits: 3.0 ns or 333 MHz
- 16 bits: 3.8 ns or 263 MHz
- 32 bits: 5.4 ns or 185 MHz
- 64 bits: 8.6 ns or 116 MHz
Fast Carry Logic

- Each CLB contains separate logic and routing for the fast generation of sum & carry signals
  - Increases efficiency and performance of adders, subtractors, accumulators, comparators, and counters
- Carry logic is independent of normal logic and routing resources
Accessing Carry Logic

- All major synthesis tools can infer carry logic for arithmetic functions
  - Addition (SUM <= A + B)
  - Subtraction (DIFF <= A - B)
  - Comparators (if A < B then…)
  - Counters (count <= count +1)
Logic Cell = \( \frac{1}{2} \) of a CLB Slice

<table>
<thead>
<tr>
<th>Part of Logic Cell</th>
<th>Mode of Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multi-purpose Look-Up Table</strong> (MLUT)</td>
<td>ROM (4-bit LUT)</td>
<td>A piece of combinational logic with 4 inputs and 1 output.</td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>16 x 1 single-port RAM.</td>
</tr>
<tr>
<td></td>
<td>SR (shift register)</td>
<td>Up to 16-bit shift-register with serial input, serial output, and no reset.</td>
</tr>
<tr>
<td><strong>Carry &amp; Control</strong></td>
<td></td>
<td>1 stage of an adder.</td>
</tr>
<tr>
<td><strong>Storage Element</strong></td>
<td>FF (Flip-flop)</td>
<td>D Flip-flop = 1-bit register, with optional set, reset, and enable.</td>
</tr>
<tr>
<td></td>
<td>Latch</td>
<td>D Latch = 1-bit latch register, with optional set, reset, and enable.</td>
</tr>
</tbody>
</table>
# CLB Slice = 2 Logic Cells

<table>
<thead>
<tr>
<th>Part of Logic Cell</th>
<th>Mode of Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-purpose Look-Up Table (MLUT)</td>
<td>ROM</td>
<td>Two pieces of combinational logic with 4 inputs and 1 output or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>One piece of combinational logic with 5 inputs and 1 output.</td>
</tr>
<tr>
<td></td>
<td>RAM</td>
<td>16 x 2 single-port RAM or 32 x 1 single-port RAM or 16 x 1 dual-port RAM.</td>
</tr>
<tr>
<td></td>
<td>SR (shift register)</td>
<td>Up to 32-bit shift-register with serial input, serial output, and no reset.</td>
</tr>
<tr>
<td>Carry &amp; Control (C&amp;C)</td>
<td></td>
<td>2 stages of an adder.</td>
</tr>
<tr>
<td>Storage Element (SE)</td>
<td>FF (Flip-flop)</td>
<td>2-bit register, with optional set, reset, and enable.</td>
</tr>
<tr>
<td></td>
<td>Latch</td>
<td>2-bit latch register, with optional set, reset, and enable.</td>
</tr>
</tbody>
</table>
Examples:

Determine the amount of Spartan 3 resources needed to implement a given circuit
Circuit 1:
Top level
Circuit 1:
F – function

2-to-4 Decoder

Full Adder
Circuit 2:
Top level
Circuit 2:
F – function

Priority Encoder

Half Adder
Circuit 3: Top level
Circuit 4: Top level
Other Components of Spartan 3 FPGAs
RAM Blocks and Multipliers in Xilinx FPGAs
Combinational and Registered Multiplier

MULT18X18

\( A[17:0] \)
\( B[17:0] \)
\( P[35:0] \)

MULT18X18S

\( A[17:0] \)
\( B[17:0] \)
\( C \)
\( CE \)
\( R \)
\( P[35:0] \)
Dedicated Multiplier Block
Block RAM

- Most efficient memory implementation
  - Dedicated blocks of memory
- Ideal for most memory requirements
  - 4 to 36 memory blocks in Spartan 3
    - 18 kbits = 18,432 bits per block (16 k without parity bits)
  - Use multiple blocks for larger memories
- Builds both single and true dual-port RAMs
- Synchronous write and read (different from distributed RAM)
Block RAM can have various configurations (port aspect ratios)

- 16k x 1
- 8k x 2
- 4k x 4
- 2k x (8+1)
- 1024 x (16+2)
# Block RAM Port Aspect Ratios

<table>
<thead>
<tr>
<th>Total Data Path Width (w bits)</th>
<th>DI/DO Data Bus Width (w-p bits) (^1)</th>
<th>DIP/DOP Parity Bus Width (r bits) (^2)</th>
<th>ADDR Bus Width (w-p-1:0)</th>
<th>DI/DO [w-p-1:0]</th>
<th>DIP/DOP [p-1:0]</th>
<th>ADDR [r-1:0]</th>
<th>No. of Addressable Locations (n) (^3)</th>
<th>Block RAM Capacity (w*n bits) (^4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>[0:0]</td>
<td>-</td>
<td>[13:0]</td>
<td>16,384</td>
<td>16,384</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>13</td>
<td>[1:0]</td>
<td>-</td>
<td>[12:0]</td>
<td>8,192</td>
<td>16,384</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>12</td>
<td>[3:0]</td>
<td>-</td>
<td>[11:0]</td>
<td>4,096</td>
<td>16,384</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>1</td>
<td>11</td>
<td>[7:0]</td>
<td>[0:0]</td>
<td>[10:0]</td>
<td>2,048</td>
<td>18,432</td>
</tr>
<tr>
<td>18</td>
<td>16</td>
<td>2</td>
<td>10</td>
<td>[15:0]</td>
<td>[1:0]</td>
<td>[9:0]</td>
<td>1,024</td>
<td>18,432</td>
</tr>
<tr>
<td>36</td>
<td>32</td>
<td>4</td>
<td>9</td>
<td>[31:0]</td>
<td>[3:0]</td>
<td>[8:0]</td>
<td>512</td>
<td>18,432</td>
</tr>
</tbody>
</table>
Single-Port Block RAM

(b) Single-Port
Dual-Port Block RAM
Input/Output Blocks (IOBs)
Basic I/O Block Structure

- Three-State
- FF Enable
- Clock
- Set/Reset
- Output
- FF Enable
- Direct Input
- FF Enable
- Registered Input
IOB Functionality

• IOB provides interface between the package pins and CLBs
• Each IOB can work as uni- or bi-directional I/O
• Outputs can be forced into High Impedance
• Inputs and outputs can be registered
  • advised for high-performance I/O
• Inputs can be delayed
Spartan-3 Family Attributes
# Spartan-3 FPGA Family Members

<table>
<thead>
<tr>
<th>Device</th>
<th>System Gates</th>
<th>Equivalent Logic Cells$^1$</th>
<th>CLB Array (One CLB = Four Slices)</th>
<th>Distributed RAM Bits (K=1024)</th>
<th>Block RAM Bits (K=1024)</th>
<th>Dedicated Multipliers</th>
<th>DCMs</th>
<th>Maximum User I/O</th>
<th>Maximum Differential I/O Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3S50$^2$</td>
<td>50K</td>
<td>1,728</td>
<td>16/12/192</td>
<td>12K</td>
<td>72K</td>
<td>4</td>
<td>2</td>
<td>124</td>
<td>56</td>
</tr>
<tr>
<td>XC3S200$^2$</td>
<td>200K</td>
<td>4,320</td>
<td>24/20/480</td>
<td>30K</td>
<td>216K</td>
<td>12</td>
<td>4</td>
<td>173</td>
<td>76</td>
</tr>
<tr>
<td>XC3S400$^2$</td>
<td>400K</td>
<td>8,064</td>
<td>32/28/896</td>
<td>56K</td>
<td>288K</td>
<td>16</td>
<td>4</td>
<td>264</td>
<td>116</td>
</tr>
<tr>
<td>XC3S1000$^2,3$</td>
<td>1M</td>
<td>17,280</td>
<td>48/40/1,920</td>
<td>120K</td>
<td>432K</td>
<td>24</td>
<td>4</td>
<td>391</td>
<td>175</td>
</tr>
<tr>
<td>XC3S1500$^3$</td>
<td>1.5M</td>
<td>29,952</td>
<td>64/52/3,328</td>
<td>208K</td>
<td>576K</td>
<td>32</td>
<td>4</td>
<td>467</td>
<td>221</td>
</tr>
<tr>
<td>XC3S2000</td>
<td>2M</td>
<td>46,080</td>
<td>80/64/5,120</td>
<td>320K</td>
<td>720K</td>
<td>40</td>
<td>4</td>
<td>565</td>
<td>270</td>
</tr>
<tr>
<td>XC3S4000$^3$</td>
<td>4M</td>
<td>62,208</td>
<td>96/72/6,912</td>
<td>432K</td>
<td>1,728K</td>
<td>96</td>
<td>4</td>
<td>712</td>
<td>312</td>
</tr>
<tr>
<td>XC3S5000</td>
<td>5M</td>
<td>74,880</td>
<td>104/80/8,320</td>
<td>520K</td>
<td>1,872K</td>
<td>104</td>
<td>4</td>
<td>784</td>
<td>344</td>
</tr>
</tbody>
</table>

**Notes:**
1. Logic Cell = 4-input Look-Up Table (LUT) plus a ‘D’ flip-flop. "Equivalent Logic Cells" equals "Total CLBs" x 8 Logic Cells/CLB x 1.125 effectiveness.
FPGA Nomenclature

**Pb-Free Packaging**

For additional information on Pb-free packaging, see [XAPP427](#): "Implementation and Solder Reflow Guidelines for Pb-Free Packages".

<table>
<thead>
<tr>
<th>Device</th>
<th>Speed Grade</th>
<th>Package Type / Number of Pins</th>
<th>Temperature Range (TJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3S50</td>
<td>-4</td>
<td>VQ(G)109 100-pin Very Thin Quad Flat Pack (VQFP)</td>
<td>C: Commercial (-5°C to 85°C)</td>
</tr>
<tr>
<td>XC3S2000</td>
<td>-5</td>
<td>CP(G)132 132-pin Chip-Scale Package (CSP)</td>
<td>C: Commercial (-5°C to 85°C)</td>
</tr>
<tr>
<td>XC3S4000</td>
<td></td>
<td>TQ(G)144 144-pin Thin Quad Flat Pack (TQFP)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td>XC3S1000</td>
<td></td>
<td>PQ(G)208 208-pin Plastic Quad Flat Pack (PQFP)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td>XC3S1500</td>
<td></td>
<td>FT(G)256 256-ball Fine-Pitch Thin Ball Grid Array (FTBGA)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td>XC3S2000</td>
<td></td>
<td>FG(G)320 320-ball Fine-Pitch Ball Grid Array (FBGA)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td>XC3S4000</td>
<td></td>
<td>FG(G)456 456-ball Fine-Pitch Ball Grid Array (FBGA)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td>XC3S5000</td>
<td></td>
<td>FG(G)676 676-ball Fine-Pitch Ball Grid Array (FBGA)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FG(G)900 900-ball Fine-Pitch Ball Grid Array (FBGA)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FG(G)1156 1156-ball Fine-Pitch Ball Grid Array (FBGA)</td>
<td>I: Industrial (-40°C to 100°C)</td>
</tr>
</tbody>
</table>
FPGA Nomenclature Example

**XC3S1500-4FG320**

- **Spartan 3 family**
- 1500 k gates = 1.5 M equivalent logic gates
- Speed grade -4 = standard performance
- 320 pins
- Package type
FPGA Design Flow
Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds.....

Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
  port (clock, reset, encr_decr: in std_logic;
        data_input: in std_logic_vector(31 downto 0);
        data_output: out std_logic_vector(31 downto 0);
        out_full: in std_logic;
        key_input: in std_logic_vector(31 downto 0);
        key_read: out std_logic);
end RC5_core;

Functional simulation

Synthesis

Post-synthesis simulation
FPGA Design process (2)

- Implementation
- Configuration
- Timing simulation
- On chip testing
Tools used in FPGA Design Flow

- **Functionally verified VHDL code**
- **Xilinx XST**
- **Synplify Premier**
- **Xilinx ISE**

<table>
<thead>
<tr>
<th>Design</th>
<th>VHDL code</th>
<th>Synthesis</th>
<th>Netlist</th>
<th>Implementation</th>
<th>Bitstream</th>
</tr>
</thead>
</table>

Diagram shows the flow from Design to VHDL code, then to Synthesis, Netlist, Implementation, and finally Bitstream.
Synthesis Tools

Xilinx XST

Synplify Premier

... and others
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
    A1<=A when (NEG_A='0') else not A;
    B1<=B when (NEG_B='0') else not B;
    Y<=Y1 when (NEG_Y='0') else not Y1;
    MUX_0<=A1 and B1;
    MUX_1<=A1 or B1;
    MUX_2<=A1 xor B1;
    MUX_3<=A1 xnor B1;

    with (L1 & L0) select
    Y1<=MUX_0 when "00",
        MUX_1 when "01",
        MUX_2 when "10",
        MUX_3 when others;

end MLU_DATAFLOW;
Circuit netlist (RTL view)
Mapping

LUT0

LUT1

LUT2

LUT3

LUT4

LUT5

FF1

FF2
RTL view in Synplify Premier

General logic structures can be recognized in RTL view.
Crossprobing between RTL view and code

- Each port, net or block can be chosen by mouse click from the browser or directly from the RTL View.
- By double-clicking on the element its source code can be seen:

```
-- generate divided clock
IF enable_xmit_clk = '1' OR enable_rcv_clk = '1' THEN
  IF clk_cnt >= unsigned(div_msbs_lsb) - 1 THEN
    clk_cnt <= |others => '0'|;
  ELSE
    clk_cnt <= unsigned(clk_cnt) + 1;
  END IF;
ELSE
  clk_cnt <= (0=>'1', others=>'0');
END IF;
END PROCESS clk_div;
```

- Reverse crossprobing is also possible: if section of code is marked, appropriate element of RTL View is marked too:
Technology View in Synplify Pro

- Technology view is a mapped RTL view. It can be seen by pressing button or by double-click on “.srm” file.
- As in case of “RTL View”, buttons can be used here.
- Two additional buttons are enabled:
  - show critical path
  - open timing analyst

Pay attention: technology view is usually large and presented on number of sheets.

Technology view is presented using device primitives.
Viewing critical path

- Critical path can be viewed by pressing on

- Delay values are written near each component of the path
Timing Analyst

- Timing analyst opened by pressing on
- Timing analyst gives a possibility to analyze different paths in the design
- Timing analyst can be opened only from Technology View
Implementation

• After synthesis the entire implementation process is performed by FPGA vendor tools
Xilinx Implementation

Translate  Map  Place&Route  Timing  Configure

InputFile = c:\Documents and Settings\Milind Parekar\My Documents\ECE_449\ALU\implement\xie0.ini
Executing C:\Xilinx\bin\nt\ngdbuild.exe -p 2S100TQ144-6  -sd "c:\Documents and Settings\Milind Parekar\My Documents\ECE_449\ALU\synthesis" -sd "c:\Documents and Settings\Milind Parekar\My Documents\ECE_449\ALU\compile" -sd "c:\Documents and Settings\Milind Parekar\My Documents\ECE_449\ALU\src" -sd "C:\Program Files\Aldec\Active-HDL 6.2\vlib\SPARTAN2\compile" -uc "ALU.ucf" "ALU.edf" "ALU.ngd"

c:\Documents and Settings\Milind Parekar\My Documents\ECE_449\ALU\implement\ver1\rev1>set XILINX=C:\Xilinx

c:\Documents and Settings\Milind Parekar\My Documents\ECE_449\ALU\implement\ver1\rev1>set PATH=C:\Xilinx\bin\nt
Translation

- Electronic Design Interchange Format
  - EDIF
- Timing Constraints
  - NCF
  - UCF
- User Constraint File
- Constraint Editor or Text Editor
- Native Constraint File
- Translation
- Native Generic Database file
Mapping
Placing
Routing

Programmable Connections

FPGA

LUT0

LUT1

LUT2

LUT3

LUT4

LUT5

FF1

FF2
Configuration

• Once a design is implemented, you must create a file that the FPGA can understand
  • This file is called a bit stream: a BIT file (.bit extension)

• The BIT file can be downloaded directly to the FPGA, or can be converted into a PROM file which stores the programming information
Two main stages of the FPGA Design Flow

Synthesis

- Code analysis
- Derivation of main logic constructions
- Technology independent optimization
- Creation of “RTL View”

Implementation

- Technology dependent

RTL Synthesis → Map

- Mapping of extracted logic structures to device primitives
- Technology dependent optimization
- Application of “synthesis constraints”
- Netlist generation
- Creation of “Technology View”

Map → Place & Route

- Placement of generated netlist onto the device
- Choosing best interconnect structure for the placed design
- Application of “physical constraints”

Place & Route → Configure

- Bitstream generation
- Burning device
Report files
Map report header

Release 8.1i Map I.24
Xilinx Mapping Report File for Design 'Lab3Demo'

Design Information
---------------------
Command Line : c:\Xilinx\bin\nt\map.exe -p 3S1500FG320-4 -o map.ncd -pr b -k 4 -cm area -c 100 Lab3Demo.ngd Lab3Demo.pcf
Target Device : xc3s1500
Target Package : fg320
Target Speed : -4
Mapper Version : spartan3 -- $Revision: 1.34 $
Mapped Date : Tue Feb 13 17:04:54 2007
Map report

Design Summary
-------------
Number of errors: 0
Number of warnings: 0
Logic Utilization:
   Number of Slice Flip Flops: 30 out of 26,624 1%
   Number of 4 input LUTs: 38 out of 26,624 1%
Logic Distribution:
   Number of occupied Slices: 33 out of 13,312 1%
      Number of Slices containing only related logic: 33 out of 33 100%
      Number of Slices containing unrelated logic: 0 out of 33 0%
         *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs: 62 out of 26,624 1%
   Number used as logic: 38
   Number used as a route-thru: 24
   Number of bonded IOBs: 10 out of 221 4%
      IOB Flip Flops: 7
   Number of GCLKs: 1 out of 8 12%
Related and Unrelated Logic

Related logic is defined as being logic that shares connectivity – e.g. two LUTs are "related" if they share common inputs. When assembling slices, Map gives priority to combine logic that is related. Doing so results in the best timing performance.

Unrelated logic shares no connectivity. Map will only begin packing unrelated logic into a slice once 99% of the slices are occupied through related logic packing.

Note that once logic distribution reaches the 99% level through related logic packing, this does not mean the device is completely utilized. Unrelated logic packing will then begin, continuing until all usable LUTs and FFs are occupied. Depending on your timing budget, increased levels of unrelated logic packing may adversely affect the overall timing performance of your design.
Place & route report

Asterisk (*) preceding a constraint indicates it was not met. This may be due to a setup or hold violation.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Requested</th>
<th>Actual</th>
<th>Logic Levels</th>
<th>Absolute Slack</th>
<th>Number of errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>* TS_CLOCK = PERIOD TIMEGRP &quot;CLOCK&quot; 5 ns</td>
<td>5.000ns</td>
<td>5.140ns</td>
<td>4</td>
<td>-0.140ns</td>
<td>5</td>
</tr>
<tr>
<td>HIGH 50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS_gen1Hz_Clock1Hz = PERIOD TIMEGRP &quot;gen1</td>
<td>5.000ns</td>
<td>4.137ns</td>
<td>2</td>
<td>0.863ns</td>
<td>0</td>
</tr>
<tr>
<td>&quot;gen1Hz_Clock1Hz&quot; 5 ns HIGH 50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Post layout timing report

Clock to Setup on destination clock CLOCK

<table>
<thead>
<tr>
<th>Src:Rise</th>
<th>Src:Fall</th>
<th>Src:Rise</th>
<th>Src:Fall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest:Rise</td>
<td>Dest:Rise</td>
<td>Dest:Fall</td>
<td>Dest:Fall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Clock</th>
<th>CLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.140ns</td>
</tr>
</tbody>
</table>

### Timing summary:

- Timing errors: 9  Score: 543
- Constraints cover 574 paths, 0 nets, and 187 connections

### Design statistics:

- Minimum period: 5.140ns  (Maximum frequency: 194.553MHz)
## Xilinx FPGA Devices

<table>
<thead>
<tr>
<th>Technology</th>
<th>Low-cost</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>120/150 nm</td>
<td></td>
<td>Virtex 2, 2 Pro</td>
</tr>
<tr>
<td>90 nm</td>
<td>Spartan 3</td>
<td>Virtex 4</td>
</tr>
<tr>
<td>65 nm</td>
<td></td>
<td>Virtex 5</td>
</tr>
<tr>
<td>45 nm</td>
<td>Spartan 6</td>
<td></td>
</tr>
<tr>
<td>40 nm</td>
<td></td>
<td>Virtex 6</td>
</tr>
</tbody>
</table>
## Altera FPGA Devices

<table>
<thead>
<tr>
<th>Technology</th>
<th>Low-cost</th>
<th>Mid-range</th>
<th>High-performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>130 nm</td>
<td>Cyclone</td>
<td></td>
<td>Stratix</td>
</tr>
<tr>
<td>90 nm</td>
<td>Cyclone II</td>
<td></td>
<td>Stratix II</td>
</tr>
<tr>
<td>65 nm</td>
<td>Cyclone III</td>
<td>Arria I</td>
<td>Stratix III</td>
</tr>
<tr>
<td>40 nm</td>
<td>Cyclone IV</td>
<td>Arria II</td>
<td>Stratix IV</td>
</tr>
</tbody>
</table>