5.1 Actel ACT

5.1.1 ACT 1 Logic Module

The Actel ACT architecture

(a) Organization of the basic logic cells

(b) The ACT 1 Logic Module (LM, the Actel basic logic cell). The ACT 1 family uses just one type of LM. ACT 2 and ACT 3 FPGA families both use two different types of LM

(c) An example LM implementation using pass transistors (without any buffering)

(d) An example logic macro. Connect logic signals to some or all of the LM inputs, the remaining inputs to VDD or GND
5.1.2 Shannon’s Expansion Theorem

- We can use the Shannon expansion theorem to expand \( F = A \cdot F(A='1') + A' \cdot F(A='0') \)

Example: \( F = A' \cdot B + A \cdot B \cdot C' + A' \cdot B' \cdot C = A \cdot (B \cdot C') + A' \cdot (B + B' \cdot C) \)

- \( F(A='1') = B \cdot C' \) is the cofactor of \( F \) with respect to (wrt) \( A \) or \( F_A \)
- If we expand \( F \) wrt \( B \), \( F = A' \cdot B + A \cdot B \cdot C' + A' \cdot B' \cdot C = B \cdot (A' + A \cdot C') + B' \cdot (A' \cdot C) \)
- Eventually we reach the unique canonical form, which uses only minterms
- (A minterm is a product term that contains all the variables of \( F \)—such as \( A \cdot B' \cdot C \))

Another example: \( F = (A \cdot B) + (B' \cdot C) + D \)

- Expand \( F \) wrt \( B \): \( F = B \cdot (A + D) + B' \cdot (C + D) = B \cdot F_2 + B' \cdot F_1 \)
- \( F \) = 2:1 MUX, with \( B \) selecting between two inputs: \( F(A='1') \) and \( F(A='0') \)
- \( F \) also describes the output of the ACT 1 LM
- Now we need to split up \( F_1 \) and \( F_2 \)
- Expand \( F_2 \) wrt \( A \), and \( F_1 \) wrt \( C \): \( F_2 = A + D = (A \cdot 1) + (A' \cdot D); F_1 = C + D = (C \cdot 1) + (C' \cdot D) \)
- \( A, B, C \) connect to the select lines and ‘1’ and \( D \) are the inputs of the MUXes in the ACT 1 LM
- Connections: \( A_0 = D, A_1 = '1', B_0 = D, B_1 = '1', S_A = C, S_B = A, S_0 = '0', \) and \( S_1 = B \)
5.1.3 Multiplexer Logic as Function Generators

The 16 logic functions of 2 variables:

- 2 of the 16 functions are not very interesting (F='0', and F='1')
- There are 10 functions that we can implement using just one 2:1 MUX
- 6 functions are useful: INV, BUF, AND, OR, AND1-1, NOR1-1

### Boolean functions using a 2:1 MUX

<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>
<th>A0</th>
<th>A1</th>
<th>SA</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 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 NOR1-1(A, B)</td>
<td>(A+B')</td>
<td>A·B'</td>
<td>1</td>
<td>0010</td>
<td>2</td>
<td>0 0 A</td>
<td></td>
<td></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 1 A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 AND1-1(A, B)</td>
<td>A·B'</td>
<td>A·B'</td>
<td>2</td>
<td>0100</td>
<td>4</td>
<td>A 0 B</td>
<td></td>
<td></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 1 B</td>
<td></td>
<td></td>
<td></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 0 B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 AND(A, B)</td>
<td>A·B</td>
<td>A·B</td>
<td>3</td>
<td>1000</td>
<td>8</td>
<td>0 0 A</td>
<td></td>
<td></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 1 A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 OR(A, B)</td>
<td>A+B</td>
<td>A·B' + A·B' + A·B</td>
<td>1, 2, 3</td>
<td>1110</td>
<td>13</td>
<td>B 1 A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 '1'</td>
<td>'1'</td>
<td>A·B' + A·B + A·B' + A·B</td>
<td>0, 1, 2, 3</td>
<td>1111</td>
<td>15</td>
<td>1 1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example of using the WHEEL functions to implement \( F = \text{NAND}(A, B) = (A \cdot B)' \)

- 1. First express \( F \) as the output of a 2:1 MUX: we do this by expanding \( F \) \emph{wrt} \( A \) (or \emph{wrt} \( B \); since \( F \) is symmetric) \( F = A \cdot (B') + A' \cdot ('1') \)
- 2. Assign WHEEL1 to implement INV(B), and WHEEL2 to implement '1'
- 3. Set the select input to the MUX connecting WHEEL1 and WHEEL2, \( S_0+S_1=A \). We can do this using \( S_0=A, S_1='1' \)
5.1.4 ACT 2 and ACT 3 Logic Modules

- ACT 1 requires 2 LMs per flip-flop: with unknown interconnect capacitance
- ACT 2 and ACT 3 use two types of LMs, one includes a D flip-flop
- ACT 2 C-Module is similar to the ACT 1 LM but can implement five-input logic functions
  - **combinatorial** module implements **combinational** logic (blame MMI for the misuse of terms)
- ACT 2 S-Module (**sequential module**) contains a C-Module and a **sequential element**
5.1.5 Timing Model and Critical Path

Example of timing calculations (a rather complex examination of internal module timing):

- The setup and hold times, measured inside (not outside) the S-Module, are $t'_{\text{SUD}}$ and $t'_{\text{H}}$ (a prime denotes parameters that are measured inside the S-Module)
- The clock–Q propagation delay is $t'_{\text{CO}}$
- The parameters $t'_{\text{SUD}}$, $t'_{\text{H}}$, and $t'_{\text{CO}}$ are measured using the internal clock signal $\text{CLK}_i$
- The propagation delay of the combinational logic inside the S-Module is $t'_{\text{PD}}$
- The delay of the combinational logic that drives the flip-flop clock signal is $t'_{\text{CLKD}}$
- From outside the S-Module, with reference to the outside clock signal $\text{CLK}_1$:

$$t_{\text{SUD}} = t'_{\text{SUD}} + (t'_{\text{PD}} - t'_{\text{CLKD}}), \quad t_{\text{H}} = t'_{\text{H}} + (t'_{\text{PD}} - t'_{\text{CLKD}}), \quad t_{\text{CO}} = t'_{\text{CO}} + t'_{\text{CLKD}}$$

- We do not know the internal parameters $t'_{\text{SUD}}$, $t'_{\text{H}}$, and $t'_{\text{CO}}$, but assume reasonable values:

$$t'_{\text{SUD}} = 0.4\text{ns}, \quad t'_{\text{H}} = 0.1\text{ns}, \quad t'_{\text{CO}} = 0.4\text{ns}.$$  

- $t'_{\text{PD}}$ (combinational logic inside the S-Module) is equal to the C-Module delay, so $t'_{\text{PD}} = 3\text{ns}$ for the ACT 3
- We do not know $t'_{\text{CLKD}}$; assume a value of $t'_{\text{CLKD}} = 2.6\text{ns}$ (the exact value does not matter)
- Thus the external S-Module parameters are: $t_{\text{SUD}} = 0.8\text{ns}, \ t_{\text{H}} = 0.5\text{ns}, \ t_{\text{CO}} = 3.0\text{ns}$
- These are the same as the ACT 3 S-Module parameters (I chose $t'_{\text{CLKD}}$ so they would be)
- Of the 3.0ns combinational logic delay: 0.4ns increases the setup time and 2.6ns increases the clock–output delay, $t_{\text{CO}}$
- Actel says that the combinational logic delay is buried in the flip-flop setup time. But this is borrowed money—you have to pay it back.

5.1.6 Speed Grading

- Speed grading (or speed binning) uses a binning circuit
- Measure $t_{\text{PD}} = (t_{\text{PLH}} + t_{\text{PHL}})/2$ — and use the fact that properties match across a chip
- Actel speed grades are based on 'Std' speed grade
Actel ACT 2 and ACT 3 Logic Modules

(a) The C-Module for combinational logic
(b) The ACT 2 S-Module
(c) The ACT 3 S-Module
(d) The equivalent circuit (without buffering) of the SE (sequential element)
(e) The SE configured as a positive-edge–triggered D flip-flop

- '1' speed grade is approximately 15 percent faster than 'Std'
- '2' speed grade is approximately 25 percent faster than 'Std'
- '3' speed grade is approximately 35 percent faster than 'Std'.
5.1 Actel ACT

Timing views from inside and outside the Actel ACT S-module

(a) Timing parameters for a 'Std' speed grade ACT 3

(b) Flip-flop timing

(c) An example of flip-flop timing based on ACT 3 parameters
5.1.7 Worst-Case Timing

**Keywords and concepts:** Using synchronous design you worry about how slow your circuit may be—not how fast • **ambient temperature**, \( T_A \) • package **case temperature**, \( T_C \) (military) • temperature of the chip, the **junction temperature**, \( T_J \) • nominal operating conditions: \( V_{DD}=5.0\,\text{V} \) and \( T_J=25^\circ\text{C} \) • worst-case commercial conditions: \( V_{DD}=4.75\,\text{V} \) and \( T_J=+70^\circ\text{C} \)

- always design using worst-case timing • derating factors • critical path delay between registers • **process corner** (slow–slow • fast–fast • slow–fast • fast–slow) • Commercial. \( V_{DD}=5\,\text{V} \pm 5\% \), \( T_A \) (ambient)=0 to +70°C • Industrial. \( V_{DD}=5\,\text{V} \pm 10\% \), \( T_A \) (ambient)=−40 to +85°C • Military: \( V_{DD}=5\,\text{V} \pm 10\% \), \( T_C \) (case)=−55 to +125°C • Military: Standard MIL-STD-883C Class B • Military extended: unmanned spacecraft

### ACT 3 timing parameters

<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>

### ACT 3 derating factors

<table>
<thead>
<tr>
<th>( V_{DD}/\text{V} )</th>
<th>Temperature ( T_J ) (junction)/( ^\circ\text{C} )</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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
</tbody>
</table>

5.1.8 Actel Logic Module Analysis

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

5.2 Xilinx LCA

Keywords and concepts: Xilinx LCA (a trademark, logic cell array) • configurable logic block • coarse-grain architecture

5.2.1 XC3000 CLB
• A 32-bit look-up table (LUT)
• CLB propagation delay is fixed (the LUT access time) and independent of the logic function
• 7 inputs to the XC3000 CLB: 5 CLB inputs (A–E), and 2 flip-flop outputs (QX and QY)
• 2 outputs from the LUT (F and G). Since a 32-bit LUT requires only five variables to form a unique address (32=2^5), there are several ways to use the LUT:
  • Use 5 of the 7 possible inputs (A–E, QX, QY) with the entire 32-bit LUT (the CLB outputs (F and G) are then identical)
  • Split the 32-bit LUT in half to implement 2 functions of 4 variables each; choose 4 input variables from the 7 inputs (A–E, QX, QY). You have to choose 2 of the inputs from the 5 CLB inputs (A–E); then one function output connects to F and the other output connects to G.
  • You can split the 32-bit LUT in half, using one of the 7 input variables as a select input to a 2:1 MUX that switches between F and G (to implement some functions of 6 and 7 variables).

5.2.2 XC4000 Logic Block
The Xilinx XC3000 CLB (configurable logic block)
(Source: Xilinx.)
The Xilinx XC4000 family CLB (configurable logic block). (Source: Xilinx.)
5.2.3 XC5200 Logic Block

The Xilinx XC5200 family **Logic Cell (LC)** and configurable logic block (CLB). *(Source: Xilinx.)*

5.2.4 Xilinx CLB Analysis

The use of a LUT has advantages and disadvantages:

- An inverter is as slow as a five-input NAND
- A LUT simplifies timing of synchronous logic
- Matched to large SRAM programming technology

Xilinx uses two speed-grade systems:

- Maximum guaranteed toggle rate of a CLB flip-flop (in MHz) as a suffix—higher is faster
- Example: Xilinx XC3020-125 has a toggle frequency of 125MHz
- Delay time of the combinational logic in a CLB in ns—lower is faster
- Example: XC4010-6 has \( t_{ilo} = 6.0\text{ns} \)
- Correspondence between grade and \( t_{ilo} \) is fairly accurate for the XC2000, XC4000, and XC5200 but not for the XC3000
Xilinx LCA timing model (XC5210-6)  
(Source: Xilinx.)
5.3 Altera FLEX

The Altera FLEX architecture

(a) Chip floorplan
(b) Logic Array Block (LAB)
(c) Details of the Logic Element (LE)

(Source: Altera (adapted with permission).)
A registered PAL with \( i \) inputs, \( j \) product terms, and \( k \) macrocells. (Source: Altera (adapted with permission).)

Features and keywords:
- product-term line
- programmable array logic
- bit line
- word line
- programmable-AND array (or product-term array)
- pull-up resistor
- wired-logic
- wired-AND
- macrocell
- 22V10 PLD
5.4.1 Logic Expanders

The Altera MAX architecture (the macrocell details vary between the MAX families—the functions shown here are closest to those of the MAX 9000 family macrocells) (Source: Altera (adapted with permission).) (a) Organization of logic and interconnect (b) LAB (Logic Array Block) (c) Macrocell

Features:
- Logic expanders and expander terms (helper terms) increase term efficiency
- Shared logic expander (shared expander, intranet) and parallel expander (internet)
- Deterministic architecture allows deterministic timing before logic assignment
- Any use of two-pass logic breaks deterministic timing
- Programmable inversion increases term efficiency
5.4.2 Timing Model

(a) A direct path through the logic array and a register

(b) Timing for the direct path

(c) Using a parallel expander

(d) Parallel expander timing

(e) Making two passes through the logic array to use a shared expander

(f) Timing for the shared expander (there is no register in this path)

Altera MAX timing model (ns for the MAX 9000 series, ‘15’ speed grade) (Source: Altera.)
5.4.3 Power Dissipation in Complex PLDs

Key points: static power • Turbo Bit

5.5 Summary

Key points: The use of multiplexers, look-up tables, and programmable logic arrays • The difference between fine-grain and coarse-grain FPGA architectures • Worst-case timing design • Flip-flop timing • Timing models • Components of power dissipation in programmable ASICs • Deterministic and nondeterministic FPGA architectures

5.6 Problems