Digital Circuits II

VHDL for Digital System Design: Architecture Styles
EX-OR and EX-NOR Applications

References:
2) VHDL for Programmable Logic, 1996, by Kevin Skahill, published by Addison Wesley

Spring 2015
Paul I-Hai Lin, Professor of ECET
Dept. of Computer, Electrical and Information Technology
Indiana University-Purdue University Fort Wayne

Topics of Discussion

- Design View and Corresponding Levels of Abstraction (Characteristics)
- Architecture Styles of VHDL
- VHDL Examples: EXOR and EX-NOR applications
- More Features of Altera Quartus II
Design View and Corresponding Levels of Abstraction of Digital Systems

- Behavioral
  - Algorithms
  - Register transfers
  - Boolean Expressions
  - Transfer functions

- Structural
  - Processors
  - Registers
  - Gates
  - Transistors

- Physical
  - Boards
  - Chips
  - Modules
  - Cells

VHDL for Simulation and Synthesis of Digital Systems

- The First Two Purposes of the VHDL in the design of digital systems:
  - Simulation - A virtual prototype method in making and evaluating design trade-offs prior to finalizing the design
  - Synthesis of digital circuits

- Use VHDL for Describing Digital systems
  - Describe all of the interface and signals that pass through the system
  - Describe the behavior of the system
  - Describe components and their interconnection (Structural description)
### VHDL Architecture Description Styles: Different Level of Abstraction [3]

<table>
<thead>
<tr>
<th>Behavioral</th>
<th>Test Benches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Flow</td>
<td>Sequential Descriptions</td>
</tr>
<tr>
<td>Structural Architecture</td>
<td>State Machines</td>
</tr>
<tr>
<td></td>
<td>Register Transfer</td>
</tr>
<tr>
<td></td>
<td>Selected Assignments</td>
</tr>
<tr>
<td></td>
<td>Arithmetic Operations</td>
</tr>
<tr>
<td></td>
<td>Boolean Equations</td>
</tr>
<tr>
<td></td>
<td>Hierarchy</td>
</tr>
<tr>
<td></td>
<td>Physical Information</td>
</tr>
</tbody>
</table>

### VHDL Architecture Bodies [2] Different Level of Abstraction

- Architecture Description Styles (at different level of abstraction)
  - Behavioral Architecture
  - Dataflow Architecture
  - **Structural Architecture** (can be designed with or without package)
  - Or a combination of the above type
VHDL Architecture Bodies [2]
Different Level of Abstraction

- Dataflow Architecture
  - Specify how data will be transferred from signal to signal and from input to output without the use of sequential statement
  - Use concurrent signal assignment statements
    1) Boolean equation – concurrent signal assignment statements
    2) Selective signal assignment (with, select, when) statements
    3) Conditional signal assignment statements (when, else)
  - Keywords used:
    - WHEN, ELSE
    - WITH, SELECT, WHEN

VHDL Architecture Bodies
Different Level of Abstraction

- Architecture Description Styles (at different level of abstraction)
  - Behavioral
    - Adopt a algorithmic approach that models its function
    - Use a set of sequential statements that resembles high-level language, to be executed in sequence
    - Don’t need to focus on the gate level implementation of the design
    - Keywords: PROCESS
  - Data Flow
  - Structure
  - Or a combination of the above type
An Example using Dataflow Architecture Style:

ex5_17.vhd

-- Files
LIBRARY ieee;    -- Using VHDL to Simplify Equations
USE ieee.std_logic_1164.ALL;

ENTITY ex5_17 IS
  PORT(
    a, b, c : IN std_logic;
    x, y    : OUT std_logic);
END ex5_17;

ARCHITECTURE arc OF ex5_17 IS
BEGIN
  x <= NOT((a AND b) OR (NOT b OR c));
  y <= (a NAND b) OR (b NOR c);
END arc;

---

Chemical Tank Monitoring

USE ieee.std_logic_1164.ALL;
ENTITY ex5_23 IS
  PORT(
    tank : IN std_logic_vector(2 DOWNTO 0);
    alarm : OUT std_logic);
END ex5_23;
ARCHITECTURE arc OF ex5_23 IS
BEGIN
  WITH tank SELECT
  alarm <= '0' WHEN "000", '0' WHEN "001",
           '0' WHEN "010", '1' WHEN "011",
           '0' WHEN "100", '1' WHEN "101",
           '1' WHEN "110", '1' WHEN "111",
           '0' WHEN others;
END arc;
EX-OR Gate: Application Examples

EX-OR Gate: Boolean Equation

\[ X = AB' + B'A \]

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Application Example of EX-OR
- Parity Generator/Checker
  - Check sum
- One bit half adder:
  - \( \text{SUM} = A \text{ EXOR } B \)
  - \( \text{CARRY} = A \text{ AND } B \)
- Bitwise XOR operation for reset memory
  - unsigned 8, 16, 32, 64-bit
  - \( A = 01010101 \)
  - \( A = A \text{ EXOR } A = 00000000 \) (Clear, set to zero of a memory location)
EX-OR Gate: Application Examples

- Parallel Binary Comparator
- Pseudo-random number generator
- Correlation and sequence detection
- Ethernet network busy/Idle detection

Dataflow Architecture (form 1)

```vhdl
-- xor_3_boolean.vhd
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY xor_1 IS
    PORT(
        a, b : IN std_logic;
        x : OUT std_logic);
END xor_1;

ARCHITECTURE arc_boolean OF xor_1 IS
    -- The single Concurrent statement (Boolean equation):
    -- x = a'b + ab' (in the SUM of Product form)
    BEGIN
        x <= ((NOT a) AND b) OR (a AND (NOT b));
    END arc_boolean;
```
Dataflow Architecture (form 2)

-- xor_2_dataflow.vhd
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY xor_1 IS
  PORT(
    a, b : IN std_logic;
    x : OUT std_logic);
END xor_1;
ARCHITECTURE arc_dataflow OF xor_1 IS
-- Concurrent statement(s) alone are used to specify how data is
-- transferred to
-- the output by the input signals.
BEGIN
  x <= '1' WHEN (a /= b) ELSE '0';
-- "NOT equal to" operator "/="
END arc_dataflow;

Prof. Paul Lin 15

Dataflow Architecture (form 2)

Prof. Paul Lin 16
Behavioral Architecture

-- xor_1_behavioral.vhd
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY xor_1 IS
    PORT(
        a, b : IN std_logic;
        x   : OUT std_logic);
END xor_1;

ARCHITECTURE arc_behavioral OF xor_1 IS
    -- comp is the name of the PROCESS. signal: a, b, called sensitivity list
    -- When a signal in sensitivity list changes the sequential statements are
    -- executed in the order they appear.
    -- The PROCESS comp only executed when there is an event on one of
    -- signals.
    BEGIN
        comp: PROCESS (a, b)
        BEGIN
            -- architectural description
            IF a /= b THEN  -- "NOT equal to" operator "/=
                x <= '1';  -- single quote for bit value
            ELSE
                x <= '0';
            END IF;
        END PROCESS comp;
    END arc_behavioral;
Behavioral Architecture

Parity Generator and Checker [1]

- Transmission of binary information from one digital device to another
  - Bit errors (0 -> 1, or 1 -> 0) may be caused by external electrical noise or other distribution
  - Examples:
    - BCD 5 (0101) => BCD 4 (0100)
    - Decimal 0: ASCII 30 (0011 0000) Hex => Decimal 1: ASCII 31 Hex (0011 0001) Hex
  - Extra bit added: Odd parity, Even Parity
    - 4-bit become 5-bit, 7-bit becomes 8-bit
    - Odd parity: Sum of all bits odd
    - Even parity: Sum of all bits even
- Parity Check at the Receiving side
  - Error indicator: Odd parity generated => Even parity received
  - Can only detect one-bit error
Parity Generator and Checker [1]

- Figure 6.8 Odd-parity generator/check system

![Odd-parity generator/check system diagram](image)

Parity Generator and Checker [1]

- Figure 6.9 Even and odd-parity generators

![Even and odd-parity generators diagram](image)
Parity Generator and Checker [1]

- Figure 6.11 5-bit even-parity checker

Parity Generator and Checker [1]

- Figure 6.10 8-bit even parity generator
Example 6-8. The 74280 Parity Generator and Input Bus Configuration

- Start Altera Quartus II software
- Step 1. Create a new project
- Step 2. Insert a Block Diagram File (.bdf), enter the symbol name 74280b (groups the input as a bus – vector configuration) at the Name field

- Right click the “Node line” between the INPUT and D[8..0] of 74280b and change it to “Bus Line”.
Example 6-8. The 74280 Parity Generator and Input Bus Configuration

- Processing > Start Compilation
- Tools > Technology Map Viewer (Post Mapping)

File > New > Vector Waveform File

D > Value > Count Value > Radix > Binary > Timing >
Count Every 1 us > OK
Example 6-8. The 74280 Parity Generator and Input Bus Configuration

Example 6-9. Parallel Binary Comparator
- File > New > Design Files > Block Diagram/Schematic File
Example 6-9. Parallel Binary Comparator

- Double click Input names and change to A[3..0] and B[3..0]
- Add two Bus lines, connect EXNOR gates inputs to the two bus
- Right click each Node line > Properties the add A[0]...A[3] and B[0]...B[3]

Example 6-9. Parallel Binary Comparator

- Vector Waveform File
- End time = 16 us
- Grid = 1 us
- A, B – Hexadecimal radix, from 0 to F

- B: Value > Arbitrary Value
- Changes for the values
  - 3 => 5, 6 => 2, B => 4, E => 9
Example 6-9. Parallel Binary Comparator

- Assignments > Settings > Remove ex6_9.bdf
- File > New > VHDL file
  -- ex6_9.vhd
  -- Parallel Binary Comparator
  LIBRARY ieee;
  USE ieee.std_logic_1164.ALL;
  ENTITY ex6_9 IS
    PORT(
      A : IN std_logic_vector (3 DOWNTO 0);
      B : IN std_logic_vector (3 DOWNTO 0);
      W : OUT std_logic);
  END ex6_9;
  ARCHITECTURE arc OF ex6_9 IS
  BEGIN
    W <= (A(0) XNOR B(0)) AND (A(1) XNOR B(1)) AND
      (A(2) XNOR B(2)) AND (A(3) XNOR B(3));
  END arc;
Example 6.9. Parallel Binary Comparator

![Diagram of Parallel Binary Comparator]
Example 6-10. FPGA Controlled Inverter

-- ex6_10.vhd
-- Controlled Inverter using a FOR LOOP within the PROCESS
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ex6_10 IS
PORT ( c : IN std_logic;
        d : IN std_logic_vector (3 DOWNTO 0);
        x : OUT std_logic_vector (3 DOWNTO 0));
END ex6_10;
ARCHITECTURE arc OF ex6_10 IS
BEGIN
    PROCESS (c,d)
    BEGIN
        FOR i IN 3 DOWNTO 0 LOOP
            x(i) <= d(i) XOR c;
        END LOOP;
    END PROCESS;
END arc;
Summary & Conclusion