VHDL Overview

Arturo Díaz Pérez
Centro de Investigación y de Estudios Avanzados del IPN
Laboratorio de Tecnologías de Información
Departamento de Computación
adiaz@cinvestav.mx
Outline

♦ History
♦ Existing Languages
♦ VHDL Requirements
♦ VHDL Language
♦ VHDL Based Design Process
♦ Levels of Abstraction
VHDL Milestones

- 1981 DoD, Woods Hole MA. Workshop on HDLs.
- 1983 Dod. Requirements were established. Contract awarded to IBM, TI, Intermetrics.
- 1984 IBM, TI, Intermetrics. VHDL 2.0 was defined.
- December 1984. VHDL 6.0 was released. Software development started.
- 1985. VHDL 7.2 was released to IEEE.
- May 1985. Standard VHDL 1076/A.
- 1993. VHDL 1076-1993 was approved.
VHDL Requirements Hierarchy

- Use various levels of abstraction for defining a system.
- Upper level systems are partitioned into lower.

Diagram:

```
Stack   ALU   MUX   Counter
   Bit n  Bit n-1  Bit 0
Adder   MUX   ALU std Logic
   AND   OR   MUX    NOT
```
VHDL Language

♦ A concurrent language for hardware description
♦ Allows sequential bodies

- Entity
- Architecture
- Package
- Configurations
- Strong Timing Support
VHDL Environment

- Text Editor
- Analyzer
- Library System
- Design Libraries
- VHDL Simulator
- Layout Synthesizer
- Netlist Synthesizer
- Other Tools

Library Management

Library Environment
Existing HDLs

♦ AHPL: A Hardware Programming Language
♦ CDL: Computer Design Language
♦ CONLAN: CONsensus LANguage
♦ IDL: Interactive Design Language
♦ ISPS: Instruction Set Processor Specification
♦ TEGAS: TEst Generation And Specification
♦ TI-HDL: Texas Instruments Hardware Description Language
♦ ZEUS: An HDL by GE corporation
VHDL Requirements

- **General Features**: documentation, high level design, simulation, synthesis, test, automatic hardware.
- **Design Hierarchy**: Multilevel description, partitioning.
- **Library Support**: Standard packages, cell based design.
- **Sequential Statements**: Behavioral software-like constructs.
- **Generic Design**: Binding to specific libraries.
- **Type Declaration**: strongly typed language.
- **Subprograms**.
- **Timing**: delay and concurrency.
- **Structural specification**: wiring components.
VHDL Design Process

IF 110 found on x_in THEN
  z_out := NOT y_in
ELSE
  z_out := 0

x_in

y_in

enable

z_out
VHDL Behavioral Description

0/(z_out := NOT y_in)
ENTITY moore_110_detector IS
  PORT( x, clk: IN std_logic;
       z: OUT std_logic );
END moore_1110_detector;

ARCHITECTURE behavioral OF moore_110_detector IS
  TYPE state IS (reset, goto1, goto11, goto110);
  SIGNAL current: state := reset;
BEGIN
  PROCESS(clk)
  ...

BEGIN

PROCESS( clk )
BEGIN
  IF clk’event AND clk = ‘1’ THEN
    CASE current IS
      WHEN reset =>
        IF x = ‘1’ THEN current <= goto1;
        ELSE current <= reset; END IF;
      WHEN goto1 =>
        IF x = ‘1’ THEN current <= goto11;
        ELSE current <= reset; END IF;
      WHEN goto11 =>
        IF x = ‘1’ THEN current <= goto11;
        ELSE current <= reset; END IF;
      WHEN goto110 =>
        IF x = ‘1’ THEN current <= goto1;
        ELSE current <= reset; END IF;
    END CASE;
  END IF;
END PROCESS;

z <= ‘1’ WHEN current = goto110 ELSE ‘0’;
END behavioral;
Hardware Modeling Requirements

- Proper modeling requires simultaneous processing
- Waveform shows node values

VHDL
Objects and Classes

- Signals for hardware carriers
- Variables are temporary carriers
- Constants for fixed parameters
- A signal is an object whose class is signal
Signals and Variables

- Signal assignments have a time component
- \texttt{x\_sig <= value AFTER 6 NS;}
Concurrent & sequential assignments

**sequential body**
```
  x_sig <= val1;
  y_sig <= a_sig;
  z_sig <= val2
WAIT FOR 5 NS;
  z_sig <= val3;
```

**concurrent body**
```
  x_sig <= val1;
  y_sig <= a_sig;
  z_sig <= val2, val3 AFTER 5 NS;
```

- `val2` and `val3` are sequentially placed on `z_sig`
- Assignment of `a_sig` to `y_sig` is done:
  - in the concurrent body, when event occurs on `a_sig`
  - in the sequential body, when program flow reaches it
Conventions & Syntax

ARCHITECTURE demo OF example IS
  SIGNAL
    a, b, c: std_logic := '0';
  BEGIN
    a <= '1';
    b <= NOT a;
    c <= a;
  END demo;
**VHDL Dataflow description**

- Clock level timing details are specified
- Can generate corresponding VHDL

<table>
<thead>
<tr>
<th>STATE</th>
<th>V0</th>
<th>V1</th>
<th>0</th>
<th>x_in</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td></td>
<td>00,0</td>
<td>01,0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td></td>
<td>00,0</td>
<td>10,0</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>0</td>
<td></td>
<td>00,1</td>
<td>10,0</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>1</td>
<td></td>
<td>--,--</td>
<td>--,--</td>
</tr>
</tbody>
</table>
Moore_110_Detector

Logical Part
- d logic 1
- d logic 2
- z logic

Memory Part
- dff0
- dff1
Moore_110_Detector: Overall Structure

VHDL

D 0 Q
CLK

D 1 Q
CLK

memory part

clk

logical part

z logic

z

D logic 0

D logic 1

x

VHDL

Vhdl-Overview- 20
Moore_110_detector: Logical Functions

ENTITY dff IS
  PORT( d, clk: IN std_logic; q: OUT std_logic );
END dff;
--
ARCHITECTURE dataflow OF dff IS
BEGIN
  b: BLOCK (clk = '1' AND NOT clk'STABLE)
  BEGIN
    q <= d;
  END BLOCK;
END dataflow;

ENTITY logicfunction_f IS
  PORT( i1, i2, i3: IN std_logic; o1: OUT std_logic );
END logicfunction_f;
--
ARCHITECTURE dataflow OF logicfunction_f IS
BEGIN
  o1 <= ((NOT i1) AND i2) OR ((NOT i2) AND i1 AND i3 );
END dataflow;
Moore_110_detector: Logical Functions

ENTITY logicfunction_g IS
  PORT( i1, i2, i3: IN std_logic; o1: OUT std_logic );
END logicfunction_g;

ARCHITECTURE dataflow OF logicfunction_g IS
BEGIN
  o1 <= (i2 AND (NOT i1) AND (NOT i3)) OR (i2 AND i1 AND i3) OR ((NOT i2) AND (NOT i1) AND i3);
END dataflow;

ENTITY logicfunction_z IS
  PORT( i1, i2, i3: IN std_logic; o1: OUT std_logic );
END logicfunction_z;

ARCHITECTURE dataflow OF logicfunction_z IS
BEGIN
  o1 <= (i2 AND (NOT i1) AND (NOT i3));
END dataflow;
ENTITY logical_part IS
  PORT( x, q0_in, q1_in: IN std_logic; d0_out, d1_out, z_out: OUT std_logic );
END logical_part;

ARCHITECTURE structural OF logical_part IS
  COMPONENT c1 PORT( i1, i2, i3: IN std_logic; o1: OUT std_logic );
  END COMPONENT;
  FOR d_logic0: c1 USE ENTITY WORK.logicfunction_g( dataflow );
  COMPONENT c2 PORT( i1, i2, i3: IN std_logic; o1: OUT std_logic );
  END COMPONENT;
  FOR d_logic1: c1 USE ENTITY WORK.logicfunction_f( dataflow );
  COMPONENT c2 PORT( i1, i2, i3: IN std_logic; o1: OUT std_logic );
  END COMPONENT;
  FOR z_logic: c1 USE ENTITY WORK.logicfunction_z( dataflow );
BEGIN
  d_logic0: c1 PORT MAP( q0_in, q1_in, x, d0_out );
  d_logic1: c1 PORT MAP( q0_in, q1_in, x, d1_out );
  d_logic1: c1 PORT MAP( q0_in, q1_in, x, z_out );
END structural;
ENTITY memory_part IS
    PORT( d0_in, d1_in, clk: IN std_logic; q0_out, q1_out: OUT std_logic );
END memory_part;
--
ARCHITECTURE structural OF memory_part IS
    COMPONENT m
        PORT( d, clk: IN std_logic; q: OUT std_logic );
    END COMPONENT;
    FOR dff0, dff1: m USE ENTITY WORK.dff( dataflow );
BEGIN
    dff0: m PORT MAP( d0_in, clk, q0_out );
    dff1: m PORT MAP( d1_in, clk, q1_out );
END structural;
ARCHITECTURE structural OF moore_110_detector IS
    COMPONENT l
        PORT(x, q0_in, q1_in: IN std_logic; d0_out, d1_out, z_out: OUT std_logic );
    END COMPONENT;
    FOR lpart: l USE ENTITY WORK.logical_part( structural );
    COMPONENT m
        PORT(d0_in, d1_in, clk: IN std_logic; q0_out, q1_out: OUT std_logic );
    END COMPONENT;
    FOR mpart: m USE ENTITY WORK.memory_part( structural );
    SIGNAL conn0, conn1, conn2, conn3: std_logic;
BEGIN
    lpart: l PORT MAP(x, conn0, conn1, conn2, conn3, z);
    mpart: m PORT MAP( conn2, conn3, clk, conn0, conn1);
END structural;
ENTITY Moore_test IS END Moore_test;

ARCHITECTURE input_output OF moore_test IS

    COMPONENT comp1 PORT(x, clk: IN std_logic; z: OUT std_logic );
    END COMPONENT;

    FOR c1: comp1 USE ENTITY WORK. moore_110_detector(behavioral);
    FOR c2: comp1 USE ENTITY WORK. moore_110_detector(structural);

    SIGNAL x_in, clock, z_beh, z_struct, compare_out: std_logic;

BEGIN

    x_in <= '0', '1' AFTER 500 NS,
              '0' AFTER 900 NS, '1' AFTER 1100 NS,
              '0' AFTER 1300 NS, '1' AFTER 1500 NS,
              '0' AFTER 1900 NS, '1' AFTER 2100 NS,
              '0' AFTER 2300 NS, '1' AFTER 2500 NS,

    clock <= NOT clock AFTER 100 NS WHEN NOW < 3000 NS ELSE clock;

    c1: comp1 PORT MAP( x_in, clock, z_beh );
    c2: comp1 PORT MAP( x_in, clock, z_struct );

    compare_out <= z_beh XOR z_struct;

END input_output;
Testbench

♦ Test may be done for various purposes
  ■ Verify the design
  ■ Check the delays
  ■ Find maximum clock speed
  ■ Compare behavioral & dataflow
  ■ A testbench can instantiate two versions of a component
  ■ XOR gates can be used to flag discrepancies
Summary

- VHDL is a standard language for describing hardware at different levels: behavioral or structural
- Usually, VHDL source code is translated to an intermediate code
- Library support must be provided with any VHDL tool
- An entity and architectural module must be constructed for each block
- Testbench can be realized by constructing test modules
- Proper modeling requires simultaneous processing
- Signals and variables are used for carriers
- Signal assignments have a time component
- Assignment to a signal causes an EVENT on the signal if the signal changes value