ECE 459/559
Secure & Trustworthy Computer Hardware Design

VHDL Overview

Garrett S. Rose
Spring 2016
Recap

- Public Key Encryption (PKE)
- RSA (Rivest, Shamir and Adelman) Encryption
- Advanced Encryption Standard (AES)
- AES vs. DES
- SIMON
Summary

• Brief overview of VHDL
  – Behavioral VHDL
  – Structural VHDL
• Simple examples with VHDL
• Some VHDL for SIMON encryption
Some History of VHDL

VHSIC Hardware Description Language

- U.S. government (Dept. of Defense) initiative
  - Wanted better, concise documentation of behavior
- VHDL syntax borrows heavily from Ada (DoD requirement)
- Initial VHDL version: IEEE 1076-1987
- Standard types included in library IEEE 1164
- IEEE 1076 updated in 1993
- A few minor updates (mostly modeling) in 2000 and 2002

VHSIC = Very High Speed Integrated Circuits
**VHDL is an Entity-based Language**

- **Entity**: a unit in digital system design
- An entity describes **name** of unit, its **ports**, and **types** and **directions** of those ports
  - Port: an input or output of the design entity
  - Communication with other entities via ports
- **Example:**

```vhdl
entity my_circuit is
  port(
    a : in std_logic;
    b : in std_logic;
    c : out std_logic);
end my_circuit;
```
Signal Values

- IEEE 1164 standard defines nine values for digital signal:
  1: logic value 1
  0: logic value 0
  U: uninitialized
  Z: high impedance
  X: forcing unknown
  W: weak unknown
  L: weak 0
  H: weak 1
  “-”: don't care
Port Mode

- Identifies direction of data flow through the port
- All ports must have an identified mode:
Port Example

```vhdl
entity dumb_circuit is
    port( in1, in2, in3 : in std_logic;
         out1, out2 : out std_logic);
end dumb_circuit;

... out1 <= in1 and in2;
    out2 <= out1 or in3;
```
Port Example

```vhdl
entity dumb_circuit is
  port( in1, in2, in3 : in std_logic;
       out1, out2 : out std_logic);
end dumb_circuit;
...
out1 <= in1 and in2;
out2 <= out1 or in3;
```

- This is where mode “buffer” is useful
  - If `out1` were declared in entity as “buffer” instead of “out,” there would be no error
  - May cause other problems → try to avoid use of buffer mode
IEEE Array/Vector Type: std_logic_vector

- std_logic_vector is simply an array where each element is of type std_logic (one bit)

```vhdl
entity mux2x8 is
  port( bus_a : in std_logic_vector(7 downto 0);
        bus_b : in std_logic_vector(7 downto 0);
        sel : in std_logic;
        bus_out : out std_logic_vector(7 downto 0));
end mux2x8;
```

- In the architecture, individual bits can be referred to:

  ```vhdl
  bus_out(7) <= bus_a(7) when sel = '0' else bus_b(7);
  ```

- The whole bus can be assigned with a bitstring:

  ```vhdl
  bus_out <= "11100000";
  ```
Architecture

• The entity declares the device and describes the I/O
• The architecture describes what the device does and is

architecture my_architecture_name of my_circuit is
  -- declarative section
begin
  -- activity statements or architecture body
end my_architecture_name;

Refers to a previously declared entity
Entity & Architecture Together

- Each entity can have multiple architectures

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity my_circuit is
  port( a, b : in std_logic;
       c, d : out std_logic);
end my_circuit;

architecture behav of my_circuit is
  signal andab : std_logic;
begin
  -- these are CONCURRENT statements
  c <= andab;
  d <= not andab;
  andab <= a and b;
end behav;
```
Entity & Architecture Together

- Each entity can have multiple architectures

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity my_circuit is
  port( a, b : in std_logic;
       c, d : out std_logic);
end my_circuit;

architecture behav of my_circuit is
  signal andab : std_logic;
begin
  -- these are CONCURRENT statements
  c <= andab;
  d <= not andab;
  andab <= a and b;
end behav;
```

Library declarations required to use IEEE standard library
library IEEE;
use IEEE.std_logic_1164.all;

entity my_circuit is
  port( a, b : in std_logic;
        c, d : out std_logic);
end my_circuit;

architecture behav of my_circuit is
  signal andab : std_logic;
begin
  -- these are CONCURRENT statements
  c <= andab;
  d <= not andab;
  andab <= a and b;
end behav;

Entity declaration defines the outer “black box” view of the design
Entity & Architecture Together

- Each entity can have multiple architectures

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity my_circuit is
  port( a, b : in std_logic;
       c, d : out std_logic);
end my_circuit;

architecture behav of my_circuit is
  signal andab : std_logic;
begin
  -- these are CONCURRENT statements
  c <= andab;
  d <= not andab;
  andab <= a and b;
end behav;
```

Architecture is the main body of the design, describing the operation

Statements are concurrent
Describing Architecture Practically

- How should the architecture behave?
- Of what pieces is it composed and how are they connected?
- The architecture is described by defining any needed subelements, how they behave and putting it all together.

*a 4-bit shift register*
Describing Architecture Practically
4-bit Shift Register

• Define the design:
  – Inputs: DIN, CLK (need clock)
  – Outputs: 4 bits (std_logic) or 4-bit word (std_logic_vector)

• Behavior in English:
  – On rising edge of clock, shift DIN into LSB, LSB into 2\textsuperscript{nd} position, 2\textsuperscript{nd} position to 3\textsuperscript{rd}, and 3\textsuperscript{rd} position to MSB
  – “positions” must instantiate memory elements (e.g. flip-flops)
Describing Architecture Practically

4-bit Shift Register

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity shr_4b is
  port( DIN, CLK       : in  std_logic;
        Q0, Q1, Q2, Q3 : out std_logic);
end shr_4b;
```
Describing Architecture Practically
4-bit Shift Register - Behavioral

```vhdl
architecture behavioral of shr_4b is
begin
storage: process is
variable str_q0, str_q1, str_q2, str_q3 : std_logic;
begins
if CLK = '1' then
str_q0 := DIN;
str_q1 := str_q0;
str_q2 := str_q1;
str_q3 := str_q2;
end if;
Q0 <= str_q0 after 2 ns;
Q1 <= str_q1 after 2 ns;
Q2 <= str_q2 after 2 ns;
Q3 <= str_q3 after 2 ns;
wait on DIN, CLK;
end process storage;
end behavioral;
```
Behavioral Architecture

- Abstract or algorithmic description – “high level”
- Structure is not defined specifically
- Usually contains:
  - **process** statements – can contain sequential statements
    --may look more like software
  - **variable** declarations – for use in process
  - signal assignment statements
  - **wait** statements – temporarily suspends process, can model delay and timing
Describing Architecture Practically
4-bit Shift Register - Structural

• Need some additional, lower-level, entities

entity d_latch is
  port( D, CLK : in  std_logic;
      Q : out std_logic);
end entity d_latch;

architecture basic of d_latch is
begin
  dl_behav: process is
  begin
    if CLK = '1' then
      Q <= D after 2 ns;
    end if;
    wait on DIN, CLK;
  end process dl_behav;
end basic;
library IEEE;
use IEEE.std_logic_1164.all;

entity shr_4b is
    port( DIN, CLK : in std_logic;
          Q0, Q1, Q2, Q3 : out std_logic);
end shr_4b;
Describing Architecture Practically
4-bit Shift Register - Structural

```vhdl
architecture structural of shr_4b is
  signal tmp_q0, tmp_q1, tmp_q2, tmp_q3 : std_logic;
begin
  bit0: entity work.d_latch(basic)
    port map(DIN, CLK, tmp_q0);
  bit1: entity work.d_latch(basic)
    port map(tmp_q0, CLK, tmp_q1);
  bit2: entity work.d_latch(basic)
    port map(tmp_q1, CLK, tmp_q2);
  bit3: entity work.d_latch(basic)
    port map(tmp_q2, CLK, tmp_q3);

  Q0 <= tmp_q0;
  Q1 <= tmp_q1;
  Q2 <= tmp_q2;
  Q3 <= tmp_q3;
end structural;
```
Structural Architecture

• Implements module as composition of subsystems
• Contains:
  - **signal** declarations for internal interconnections (wires)
    --entity ports also treated as signals
  - Component instances of previously declared entity and architecture pairs for subsystems
  - **port maps** in component instances
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity encrypt is
  port ( clk          : in std_logic;
        reset        : in std_logic;
        key_word     : in std_logic_vector(63 downto 0);
        plain_text   : in std_logic_vector(31 downto 0);
        ENC_EN       : in std_logic;
        cipher_text  : out std_logic_vector(31 downto 0);
        done_flag    : out std_logic;
        running_flag : out std_logic
      );
end encrypt;

...
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity encrypt is
    port ( clk : in std_logic;
           reset : in std_logic;
           key_word : in std_logic_vector(63 downto 0);
           plain_text : in std_logic_vector(31 downto 0);
           ENC_EN : in std_logic;
           cipher_text : out std_logic_vector(31 downto 0);
           done_flag : out std_logic;
           running_flag : out std_logic );
end encrypt;

...
SIMON Encryption in VHDL
Declarative Section

...}
architecture Behavioral of encrypt is
component round_cipher is
    port (  
        blockcipher : in std_logic_vector(31 downto 0);
        key_word    : in std_logic_vector(15 downto 0);
        cipher_text : out std_logic_vector(31 downto 0)
    );
end component round_cipher;

...

signal block_half1, block_half2 : std_logic_vector(15 downto 0);
signal text_holder, temp_block : std_logic_vector(31 downto 0);
signal key_expanded : std_logic_vector(15 downto 0);
signal rnd : std_logic_vector(7 downto 0);
signal enc_run : std_logic;

begin
    ...

Declare components to be used in design

Declare internal signals (wires between components)
SIMON Encryption in VHDL
Main Architecture Body

begin
  mux_left: mux2to1 port map(ENC_EN, plain_text(31 downto 16),
                              temp_block(31 downto 16),
                              block_half2);
  mux_right: mux2to1 port map(ENC_EN, plain_text(15 downto 0),
                               temp_block(15 downto 0),
                               block_half1);
  reg16_left: reg16 port map (clk, reset, block_half2,
                             text_holder(31 downto 16));
  reg16_right: reg16 port map (clk, reset, block_half1,
                             text_holder(15 downto 0));
  cipher_text <= text_holder;
  round: round_cipher port map (text_holder, key_expanded, temp_block);
  keys: key_expansion port map (rnd, ENC_EN, key_word, clk, reset, key_expanded);
...

Instantiate components used in design
Use internal signals to define connections
SIMON Encryption in VHDL
Main Architecture Body

...  
rnd_ctl: process(clk)
begin
  if (clk'event and clk = '1') then
    if (reset = '1') then
      rnd <= x"00";
      enc_run <= '0';
      done_flag <= '0';
    else
      if (rnd = x"1F") then
        rnd <= x"00";
        enc_run <= '0';
        done_flag <= '1';
      elsif (ENC_EN = '1' and enc_run = '0') then
        rnd <= x"00";
        enc_run <= '1';
      elsif (enc_run = '1') then
        rnd <= std_logic_vector(unsigned(rnd) + 1);
      else
        rnd <= x"00";
        done_flag <= '0';
        enc_run <= '0';
      end if;
    end if;
  end if;
end process;
...