

Lesson 22

Good VHDL Coding Style

Mohd Saufy Rohmad

Technical Trainer and Consultant

Comment

- Comment is essential to write good codes in any programming language.
- VHDL is already a complicated language.
- So, without comments, it becomes more cryptic and impossible to understand.
- There can be 3 types of comments
 - (1) Comment should be written before every entity that states the author name, creation date etc. A template of such comment is given here
 - (2) Every process should start with a comment, that summarizes what the process do. This is a very good practice as it is quite easy to forget why the process was written.
 - (3) Finally, inline comments. You must have them everytime you feel they are necessary.

(1)

-- Title : Real Arithmetic Library

-- Project : Comit

-- File : realArithmetic.vhd

-- Author : Shahriar Shahabuddin (shahriar.shahabuddin@oulu.fi)

-- Company : Centre for Wireless Communications, University of Oulu

-- Description: This package contains a set of functions that carries out
-- real arithmetic operations

-- Copyright (c) Centre for Wireless Communications, University of Oulu

-- Revisions :

-- Date	Version	Author	Description
---------	---------	--------	-------------

-- 2012	1.0	sshahabd	Created
---------	-----	----------	---------

-- 11/02/2012	1.1	sshahabu	First Release
---------------	-----	----------	---------------

-- 07/01/2012	1.2	sshahabu	bug fixes
---------------	-----	----------	-----------

-- 11/05/2012	1.3	sshahabu	Major revision
---------------	-----	----------	----------------

(2)


```
-- comb_proc : this process creates a multiplexer
```

```
comb_proc : process(state)
begin
  if state = "001" then
    cs <= "01";
  elsif state = "010" then
    cs <= "10";
  elsif state = "100" then
    cs <= "11";
  end if;
end process;
```

Indentation

- This is also a general rule for any programming language.
- Use an editor that can do proper indentation automatically.
- Here is an example of good indentation taken from Xilinx's coding style guideline.

```
-- purpose: to show proper indentation
sample_proc : process (clk, reset)
    variable muxed_data_v : std_logic_vector (1 downto 0);
begin -- process sample_proc
    if reset = '0' then
        for i in data'range loop
            data(i) <= (others => '0'); -- data is a 4x2 array
        end loop; -- i
        muxed_data <= '0'
    elsif clk'event and clk = '1' then
        muxed_data_v := data(conv_integer(addr));
        case sel is
            when '0' =>
                muxed_data <= mux_data_v(0);
            when '1' =>
                muxed_data <= mux_data_v(1);
        end case; -- case sel is
    end if; -- if reset = '0'...
end process sample_proc;
```

A vertical double-headed arrow on the left side of the code block indicates the indentation levels. It has several horizontal tick marks pointing to the start of each line of code, showing how the indentation increases and decreases as the code structure changes (e.g., entering and exiting loops, if-statements, and case blocks).

Naming Convention

- The naming of signals, entities, constants or variables should follow a guideline rather than having arbitrary names. This makes a VHDL code more readable.
- Hubert Kaeslin explained a naming convention in his book.
- He compiled a number of attributes for making a signal's nature evident.
- `signal_identifier ::= signal_name "x" signal_attributes`
- `signal_name ::= upper_case_letter { letter | digit }`
- `signal_attributes ::= class_char [signal_waveform] state_char trist_mode active_low_char io_mode`
- `class_char ::= "R" | "A" | "C" | "S" | "D" | "T"`
- `signal_waveform ::= "Q" | "M" | "G"`
- `state_char ::= "N" | "P" | ""`
- `trist_mode ::= "Z" | ""`
- `active_low_char ::= "B" | ""`
- `io_mode ::= "I" | "O" | "IO" | ""`

Example

- Examples
 - FOOxC identifies a clock signal.
 - BARxD indicates this is a regular data signal within some clock domain. As opposed to this, the name BARxA refers to the same signal before its being synchronized to the local clock and, hence, subject to toggle at any time.
 - REXTFxRB refers to some asynchronous active-low reset signal.
 - ADDR CNTxSP and ADDR CNTxSN are names for the present and next state of an address counter.
 - IRQxAMI identifies an interrupt request input that emanates from a foreign clock domain and that is meant to be stored in a
- positive-edge-triggered flop-flop until it is serviced.
 - CarryxDB denotes an active-low carry signal.
 - GeCntxDZO denotes an output-only data signal with high impedance capability.
 - ScanModexT stands for an active-high scan enable signal within some clock domain.

Ten Commandments of VHDL

- All state machine outputs shall always be registered
- Thou shalt use registers, never latches
- Thy state machine inputs, including resets, shall be synchronous
- Beware fast paths lest they bite thine ankles
- Minimize skew of thine clocks
- Cross clock domains with the greatest of caution. Synchronize thy signals!
- Have no dead states in thy state machines
- Have no logic with unbroken asynchronous feedback lest the fleas of myriad Test Engineers infest thee
- All decode logic must be crafted carefully—eschew asynchronicity
- Trust not thy simulator—it may beguile thee when thy design is just

Thank You