M.C. Smith

VHDL Guidelines

Synthesizable VHDL coding basics

- Here, we describe a set of basic guidelines for coding in VHDL
- The objective is to produce readable VHDL source that is targeted for synthesis
- Many factors affect the ultimate performance of a VHDL-based design including
  - VHDL coding style
  - EDA tool(s) used: synthesis and place/route
  - Characteristics of the target device
  - User/designer awareness of how the above 3 items interact and depend on each other

Sources of information

- Various sources of information exist that should help you learn and use good coding practice
  - Altera online help in Quartus II gives a set of guidelines for VHDL coding
  - Altera website provides numerous design examples
  - Internet
    - Here you can see questions people typically have about VHDL and programmable devices. Look at the answers/hints that other people post to help you learn
    - DO NOT post questions directly related to your homework/project assignments to these groups!!!!!!
    - DO NOT copy and submit code from the internet as your own!!!!

Simulation accuracy and speed

- Accuracy
  - Ensure the sensitivity list of **process** statements is complete
- Speed
  - Use a **process** statement in preference to concurrent signal assignments
    - Reduces the number of signals a simulator must continually monitor for changes and so improves simulation speed
  - Design models to minimize the number of signals in a **process** statement
    - Less signals to monitor will improve **simulation** speed
- Do not model many small **process** statements
    - If there are many registers being clocked from the same clock source, it is better to put them in one process

Synthesis modeling recommendations

- When modeling purely combinational logic, ensure signals are assigned in **every** branch of conditional assignment statements
- Use **case** statements in preference to **if** statements containing **else-if** clauses
- The **others** default **case** branch ensures all branch values are covered
- There is no real need to use a wait statement to infer flip-flops
  - The *if* statement can do all that wait does and has the added advantage that combinational logic can be inferred

Source: D. J. Smith, HDL Chip Design, Doone Publications, 2001

General HDL modeling recommendations

- Before attempting to code a model, know and understand what it is you are modeling
- Divide your design into sub-designs when possible
  - Promotes short, easy to read, reusable code
- Make models as generic as possible
  - Parameterize bus widths
- Use meaningful signal names
  - For active low signals use **<signal_name>_n** for clearer understanding and easier debugging
- Use comments liberally
  - A header should describe each module and each signal declaration should have a comment
  - Dated, inconsistent comments are worse than none at all

Assigning Values in Different Number Formats

- Assignment statements (with VHDL 1993 syntax) can use different number formats as appropriate
  - X (hex), O (Octal)
- Assume **reg** is STD_LOGIC_VECTOR(7 DOWNTO 0)

  reg <= “10101111” ; is the same as reg <= x”AF”;

- Underscores can be used to separate fields in long strings to enhance readability

  instruction <= “00000000010000100001000000100000”;

  instruction <= “000000_00001_00001_00001_00000_100000”;

VHDL Aggregates

- An **aggregate** is a collection of items that are gathered together to form a total quantity

```
SIGNAL V : STD_LOGIC_VECTOR(7 DOWNTO 0);

V <= (others => '0');                    -- "00000000"
V <= ('1', '0', others => '0');            -- "10000000"
V <= (4 => '1', others => '0');            -- "00010000"
V <= (3 DOWNTO 0 => '0', others => '1');  -- "11110000"
-- V <= ("0000", others => '1');           -- illegal!
```

VHDL Generics

- VHDL generics may be used to parameterize code making components as modular and hence reusable as possible
- For example:

```
ENTITY counters IS
GENERIC ( WIDTH : INTEGER := 8 );
   PORT(
      d : IN STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0);
      clk : IN STD LOGIC;
      clear : IN STD_LOGIC;
      load : IN STD_LOGIC;
      up_down : IN STD_LOGIC;
      qd : OUT STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0));
END counters;
```