Tutorial on Hardware Description Language

CAD Tools Tutorials (Example with class project)

Release HW#6 + Q&A Section
Design and Coding Style for HDL

Zhaori Bi

9/24/2014
- Modeling and Synthesis
- HDL Design Flow
- HDL Coding Style
- Synthesizable Coding
- Summary && Reference
Modeling and Synthesis

HDL Design Flow

HDL Coding Style

Synthesizable Coding

Summary && Reference
Modeling and Synthesis:

- The Hardware Description Languages were designed for modeling hardware behavior.
- Then tools have been developed for synthesis.
Not all HDL Code can be synthesized - limited by compiling tool

Synthesizable HDL is a subset of HDL that can be synthesized by tools
- Modeling and Synthesis
- HDL Design Flow
- HDL Coding Style
- Synthesizable Coding
- Summary & Reference
HDL Synthesis Flow:

- HDL Code
- HDL Simulator
- HDL Compiler
- Design Compiler

Functional verification
- Modelsim/Synopsys VCS

Translation, block-level Optimization
- Synopsys Design Vision

Logical Optimization, generate netlist
- Synopsys Design

Technology Library

Netlist & Schematic
Example for HDL Compiler (Operator reordering):

This code implies the following initial operator ordering:

\[ \text{SUM} \leftarrow A \times B + C \times D + E + F + G \]

Operators are re-ordered to reduce critical path delays.
HDL Based Design Flow

Example for Design Compiler (Logic optimization):

The process of selecting the best combinational logic gates from the target library to generate a design that meets timing and area goals.
HDL Based Design Flow

- HDL Code
- Testbench
- HDL Simulator
- Compare Output
- Synopsys HDL Compiler
- Synopsys Design Compiler
- Gate-level Netlist
- HDL Simulator
HDL Based Design Flow

a. Understand the specification completely

b. Write HDL code

c. Write HDL test bench

d. Run simulation

e. Synthesize HDL code to gate level

f. Check the timing (Done by CAD tool)

g. Re-Run gate level netlist simulation and compare with d
- Modeling and Synthesis
- HDL Design Flow
- HDL Coding Style
- Synthesizable Coding
- Summary & Reference
HDL Coding Style

- **Behavioral Style**
  Express the function of the system as algorithms

- **Dataflow Style**
  Express the function of the system as the data dependencies

- **Structural Style**
  Express the function of the system in terms of interconnection of components
entity MUX4TO1 is
  port( A, B, C, D : in std_logic;
       S0, S1 : in std_logic;
       Y     : out std_logic);
end MUX4TO1;
Behavioral Style:

Architecture BEHAVIORAL of MUX4T01 is
Begin
    process(A,B,C,D,S1,S0)
    begin
        case S1&S0 is
            when "00" => Y <= A;
            when "01" => Y <= B;
            when "10" => Y <= C;
            when "11" => Y <= D;
            when others => null;
        end case;
    end process;
end BEHAVIORAL;
Dataflow Style:

Architecture DATAFLOW of MUX4TO1 is
begin
    Y <= (A and not S1 and not S0) or
         (B and not S1 and S0) or
         (C and S1 and not S0) or
         (D and S1 and S0);
end DATAFLOW;
Structural Style:

Architectural STRUCTURAL of MUX4TO1 is
Signal I0,I1,P,Q,R,S: std_logic;
-- Component Declarations
...
begin
  INV1: INV port map (S0,I0);
  INV2: INV port map (S1,I1);
  A1: AND3 port map (A,I1,I0,P);
  A2: AND3 port map (B,I1,S0,Q);
  A3: AND3 port map (C,S1,I0,R);
  A4: AND3 port map (D,S1,S0,S);
  U1: OR3 port map (P,Q,R,S,Y);
end STRUCTURAL;
✓ Behavioral style is more natural and easier to code for complex system

✗ Difficult for synthesis tools to handle. May not synthesizable

✓ Structural style is similar to hardware and easy to synthesize

✗ Difficult in coding especially for complex system
Hardware designers prefer to structural design:

1. Reusable

2. Trick the synthesis tool to generate exactly the hardware structure as design

3. Easy to debug & test each component from bottom to up
- Modeling and Synthesis
- HDL Design Flow
- HDL Coding Style
- Synthesizable Coding
- Summary & Reference
Synthesizable Coding

HDL Compiler Unsupported

- delay
- initial
- repeat
- wait
- fork ... join
- event
- deassign
- force
- release
- primitive -- User defined primitive
- time

- triand, trior, tri1, tri0, trireg
- nmos, pmos, cmos, rnmos, rpmos, rcmos
- pullup, pulldown
- rtran, tranif0, tranif1, rtranif0, rtranif1
- case identity and not identity operators
- Division and modulus operators
  - division can be done using DesignWare instantiation
Synthesizable Coding

• Not supported: Ignored
  – The construct is ignored by the synthesis tool
  – Examples:
    • Initialization of variables or signals
    • Assert statements
    • Physical Type: Time
      – The synthesizer ignores the AFTER clause in expressions
        Example: \( x \leq y \) AFTER 100ns

• Not Supported: Illegal
  – Data types
    • Files, real, generic that are not integers (Altera allows strings)
- Modeling and Synthesis
- HDL Design Flow
- HDL Coding Style
- Synthesizable Coding
- Summary && Reference
Thanks!