Bringing the power of Eclipse to Digital Hardware Designers

Hendrik Eeckhaut
@heeckhau
Eclipse / Java

- Navigation
- Autocomplete
- Real-time errors
- Quick-assist/fix
- Refactoring
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
end entity testbench;

architecture STR of testbench is
begin

  clock_generator_instance: entity work.clock_generator(BEH)
  port map (  
    clock1 => clock,  
    clock2 => open  
  );

  dut_instance: entity work.dut(RTL)
  port map (  
    data_out => data_out,  
    data_in => data_in,  
    valid => valid,  
    start => start,  
    clock => clock,  
    rst => reset  
  );

end architecture STR;
Navigation ➔ Text based search
Autocomplete ➔ Syntax books
Real-time errors ➔ Scroll through logs
Quick-assist/fix ➔ Stare at code
Refactoring ➔ Live with ugly code
I needed this
Can we fix this?
But How?

self funded, two man team

Commercial quality Electronic Design Automation tool
But first: What is VHDL

VHDL (Very High Speed Integrated Circuit) Hardware Description Language

- 20 year old language (based on Ada)
- But all shortcomings of arcane languages
  - very verbose
  - limited expressiveness
  - irregular
  - most designers use small subset
  - ...
architecture STR of testbench is
  signal data_out : std_logic_vector(7 downto 0);
  signal data_in  : std_logic_vector(7 downto 0);
  signal clock   : std_logic;
  constant PERIOD : time := 50 ns; -- Half the clock period.

component clock_generator
  generic(PERIOD : time := 25 ns);
  port(clock : out std_logic);
end component clock_generator;

component dut
  port(clock : in std_logic;
        data_out: out std_logic_vector(7 downto 0);
        data_in : in std_logic_vector(7 downto 0));
end component dut;

begin
  clock_generator_instance : clock_generator
  generic map(PERIOD => PERIOD)
  port map(clock => clock);

  dut_instance : dut
  port map(
        clock  => clock,
        data_out => data_out,
        data_in  => data_in
  );
end architecture STR;
Irregular syntax

architecture STR of testbench is
signal data_out : std_logic_vector(7 downto 0);
signal data_in  : std_logic_vector(7 downto 0);
signal clock    : std_logic;
constant PERIOD : time := 50 ns;  -- Half the clock period.

component clock_generator
  generic(PERIOD : time := 25 ns);
  port(clock : out std_logic);
end component clock_generator;

component dut
  port(clock : in std_logic;
        data_out : out std_logic_vector(7 downto 0);
        data_in  : in std_logic_vector(7 downto 0))
end component dut;

begin
  clock_generator_instance : clock_generator
    generic map(PERIOD => PERIOD)
    port map(clock => clock);

  dut_instance : dut
    port map(
      clock => clock,
      data_out => data_out,
      data_in  => data_in
    );
end architecture STR;
No open community

- A few communities such as
  - OpenCores
  - Open Hardware Repository (CERN)
- But very little traction
Long feedback loops

Hey! Get back to work.
Oh, carry on.

Compiling!
Not everything is negative!

- Robust way to design HW
- You can build cool stuff
- Custom cpu’s, Mars rovers, ...
- A good IDE can resolve most issues
So How?

self funded, two man team

Commercial quality
Electronic Design Automation
tool
• Technology
• Business
Built on Eclipse

Not easy, but feasible

Xtext
Tycho
EMF
p2
JDT
Xtend
Xcore
PDE
JFace
...
Social problems

- People are reluctant to change (Emacs)
- Value is clear, but difficult to quantify
• RCP: one-click download
• Free starter edition:
  • all features for small projects
  • limited features for bigger projects
• Eclipse marketplace
• Scale business
• Verilog support
• Graphical views
Conclusions

• Eclipse is great platform: allows us to focus on VHDL specific part
• Getting started is easy
• But... you need a lot of patience