VHDL/Verilog intro & Tooling

prepared for Programmable Logic Lessons / 1.2 by ~skmp

Kindly hosted by hackerspace.gr

HDL

- Hardware Description Language
  - Initially used to describe and simulate
  - As transistors became smaller (thus cheaper) synthesis became more popular
    - Some companies still do a lot of manual work (eg, Intel)
- History
  - ISPL/ISPS – first attempts (DEC, 70s)
  - KARL/ABL (University of Kaiserslautern, EU)
    - Became popular for PLDs
    - Extended to VLSI in 80s
  - Verilog
    - Introduced in 85, by Gateway Design Automation
    - First “modern”, designed for synthesis
    - Similar with C syntax
    - Popular with designs in the USA
  - VHDL
    - Initiated by DoD in 87
    - Syntax inspired from Ada
    - Popular in EU

Design “Levels”

- VHDL/Verilog
  - High level .. well kind of
- RTL
  - Low level “assembly” for synchronous digital logic
  - Describes data movement between abstract registers
- Netlist
  - Connection level
  - Describes connections between parts (transistors, etc)
  - Lowest level

HDL vs Typical programming

- Hardware works in parallel
  - You have concurrent and sequential logic
  - There are also time based constructs
    - Do x after y time
    - Most of these aren't synthesizable

- You can't debug, not in the same way anyway
  - You can simulate your design and stare at the waveforms
    - Feels pretty primitive if you're used to rich debugging tools

VHDL Example

(I'm sleepy, show me some code nao)

```
-- (this is a VHDL comment)

-- import std_logic from the IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- this is the entity
entity ANDGATE is
  port (
    I1 : in std_logic;
    I2 : in std_logic;
    O  : out std_logic);
end entity ANDGATE;

-- this is the architecture
architecture RTL of ANDGATE is
begin
  O <= I1 and I2;
end architecture RTL;

– More examples at wikipedia
```

Verilog Example

(I'm sleepy, show me some code nao)

```
module AND2gate(A, B, F);

   input A;
   input B;
   output F;

   assign F = A & B;

Endmodule

//http://en.wikipedia.org/wiki/Verilog for more
```

Not all code is synthesizable

- Synthesizable means that it can be actually implemented in fpga/vlsi
  - Some parts of the language are just for documentation/verification/simulation
  - A pretty complete list here
- We will only care for the synthesizable parts

Some definitions

- Top level design
  - The main “entity” of the design. During the synthesis, all required submodules will be synthesized.
    - ANYTHING not directly referenced will be REMOVED (optimized-out)
- Schematic
  - A diagram with connections
- Symbol
  - The image shown for a part on the schematic
- Ucf
  - User Constrains file. Used to map pin names to pin i/o on xilinx ISE

Basic Verilog structure

- Wire
  - A connection (like a wire)
- Register
  - Buffered output
- Module
  - A block of logic with inputs and outputs
- Always blocks
  - Logic that gets updated when the input changes
  - Two assignment operators, = (blocking) and <= (non-blocking, updates on next cycle)
- Assign
  - Combinatorial logic
- Basic operators
  - & | ^ ~ , && ||, < > != ==, << >> <<< >>>, much more
- Wikipedia has a pretty good article :)

Let's get Xilinx ISE installed

- ftp.hsgr.awmn/upload

Xilinx ISE

- Make a basic project
- Add a source file
- Implement AND
- Use simulator
- Try to program it to papilio

Thanks !

Next week we'll get into more into more complicated synchronous logic!

Feel free to drop by #hsgr @ freenode

... or the hsgr mailing list

... and use the wiki !

(or, send direct feedback – skmp@emudev.org)