EEC 116 Lecture: Hardware Description Languages

Rajeevan Amirtharajah

University of California, Davis

Outline

- Announcements
- HDL Overview
- A Verilog Example

Hardware Description Languages

- **HDL: Language for capturing behavior and/or implementation of digital (sometimes analog or mixed-signal) hardware**
  - Don’t think of it as a programming language…
  - …because concurrency (parallelism) is inherent!
  - Assuming sequential execution of statements can mislead you into bugs or poor synthesized hardware
- **Design Process**
  - Begin by planning hardware
  - Write code that implies that hardware to synthesis tool
- **Two major languages: VHDL and Verilog**
  - Will use Verilog in EEC119AB

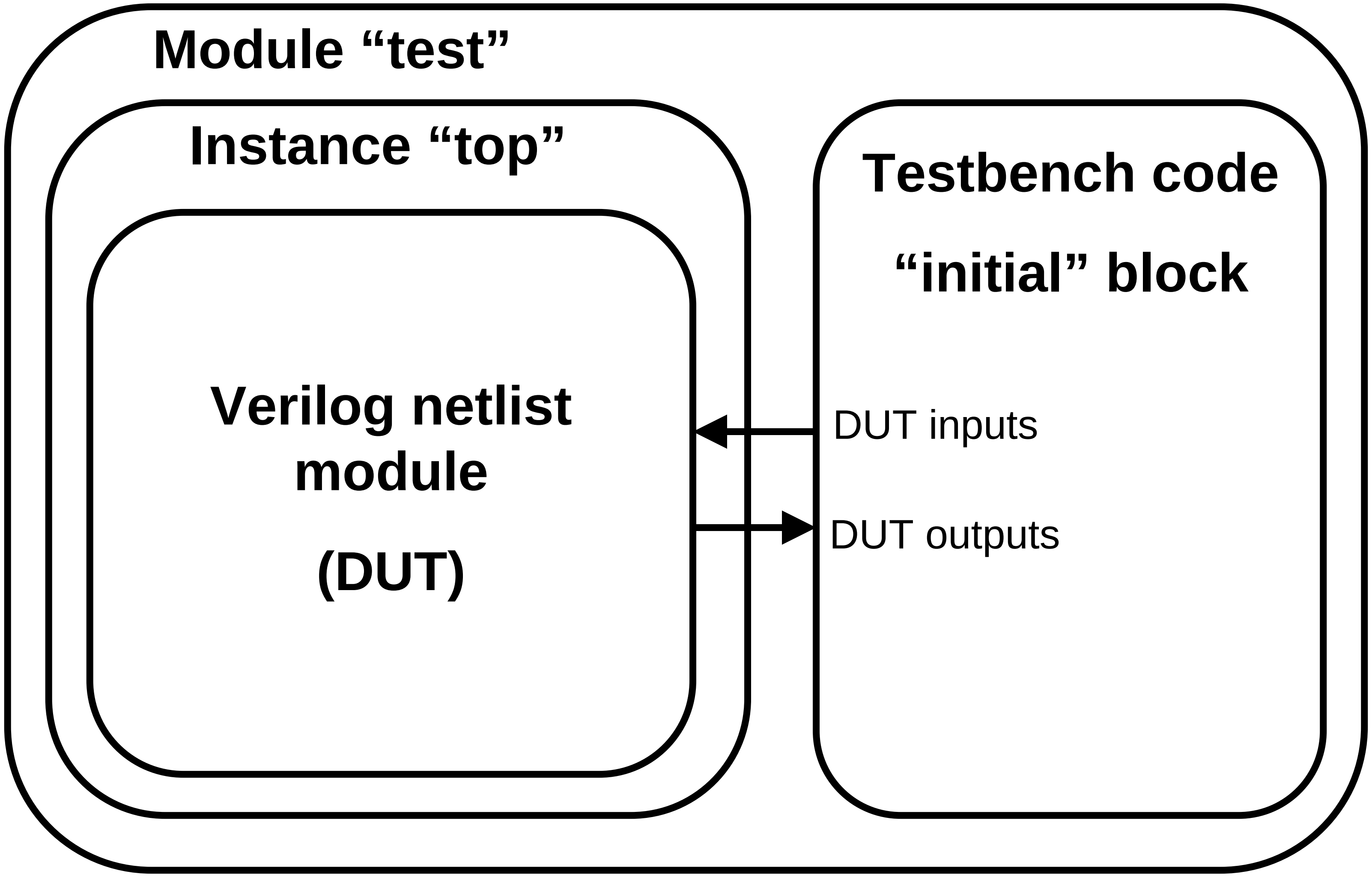
HDL and Simulation

- **A simulator that implements the event-driven semantics of an HDL is needed to evaluate Verilog code.**
  - Verilog-XL (Cadence): interpreted simulator; well-integrated but slow (esp. for large designs)
  - NC_Verilog (Cadence): compiled simulator; translates Verilog to C and compiles C to executable. Startup slow but execution fast.
  - VCS (Synopsys): compiled simulator; fast, but not integrated with Cadence
- **All simulators can be run standalone from command line for HDL-only designs.**

Hardware Description Styles

- **Two general styles for capturing hardware description**
- **Structural**
  - Modules composed of simpler modules or primitives (gates, transistors)
  - Describes a circuit schematic or netlist
- **Behavioral**
  - Describes how outputs are computed as function of inputs
  - Incorporates high-level programming language constructs (loops, if-then-else statements, case statements)
  - Behavioral descriptions can be *synthesized* to structural

Verilog Simulation Environment



Verilog Example: 4b Adder

```
`timescale 1ns / 1ns

module test;

wire  Co;

reg  Ci;

wire [3:0]  S;

reg [3:0]  B;
reg [3:0]  A;


adder4b top(Co, S, A, B, Ci);


`ifdef verilog

 //please enter any additional verilog stimulus in the testfixture.verilog
file
  `include "testfixture.verilog"
`endif

endmodule
```

Time Units

“test” Module Wrapper

Carry Out

Carry In

Sum

Signal Declarations

Summands

“top” Module Instance

Testbench Code File

Outputs

Inputs declared as type

`wire`.

`reg`.

Verilog Example: 4b Adder (cont.)

```
// Library - EEC119_ramirtha, Cell - adder4b, View - schematic
// LAST TIME SAVED: Dec 22 09:10:09 2010
// NETLIST TIME: Dec 22 09:59:21 2010
`timescale 1ns / 1ns

module adder4b ( Co, S, A, B, Ci );

output  Co;

input  Ci;

output [3:0]  S;

input [3:0]  B;
input [3:0]  A;


specify
    specparam CDS_LIBNAME  = "EEC119_ramirtha";
    specparam CDS_CELLNAME = "adder4b";
    specparam CDS_VIEWNAME = "schematic";
endspecify

FullAdder FA3 ( Co, S[3], A[3], B[3], net6);
FullAdder FA2 ( net6, S[2], A[2], B[2], net7);
FullAdder FA1 ( net7, S[1], A[1], B[1], net8);
FullAdder FA0 ( net8, S[0], A[0], B[0], Ci);

endmodule
```

← **"adder 4b" Module Declaration**

← **Signal Declarations**

← **Parameter Definitions**

← **Structural Description**

Verilog Example: 4b Adder (cont.)

```
// Library - EEC119_ramirtha, Cell - FullAdder, View - schematic
// LAST TIME SAVED: Dec 22 09:10:07 2010
// NETLIST TIME: Dec 22 09:59:21 2010
`timescale 1ns / 1ns

module FullAdder ( Co, S, A, B, Ci );
output  Co, S;

input  A, B, Ci;


specify
    specparam CDS_LIBNAME  = "EEC119_ramirtha";
    specparam CDS_CELLNAME = "FullAdder";
    specparam CDS_VIEWNAME = "schematic";
endspecify

nand2 I5 ( net21, net24, Ci);
nand2 I0 ( net22, A, B);
nor2 I1 ( net23, B, A);
xor2 I6 ( S, net25, Ci);
xor2 I2 ( net25, B, A);
inv I3 ( net24, net23);
nand2 I4 ( Co, net22, net21);

endmodule
```

“FullAdder” Module Declaration

Signal Declarations

Parameter Definitions

Structural Description

Verilog Example: 4b Adder (cont.)

```
// Loop-Based Adder4b Stimulus
integer i,j,k;
initial
begin

   A[3:0] = 4'b0000;
   B[3:0] = 4'b0000;
   Ci = 1'b0;

$display("Starting simulation...");

for(i=0;i<16;i=i+1)
 begin
  for(j=0;j<16;j=j+1)
   begin
    for(k=0;k<=1;k=k+1)
     begin
      #20
      $display("A=%b B=%b Ci=%b, Co-Sum=%b%b", A, B, Ci, Co, S);
      if ({Co,S} != A + B + Ci)
          $display("ERROR: Co-Sum should equal %b, is %b",
                    (A + B + Ci), {Ci,S});
      Ci=~Ci; // invert Cin
     end
    B[3:0] = B[3:0] + 4'b0001; // add the bits
   end
  A = A+1; // shorthand notation for adding
 end

$display("Simulation finished... ");
end
```

Integer Declarations

Initialization of register signals

Useful console messages

Nested `for` loops to generate input vectors and test output results.

Verilog Example: 4b Adder (cont.)

```
// Loop-Based Adder4b Stimulus
integer i,j,k;
initial
begin

   A[3:0] = 4'b0000;
   B[3:0] = 4'b0000;
   Ci = 1'b0;

$display("Starting simulation...");

for(i=0;i<16;i=i+1)
 begin
  for(j=0;j<16;j=j+1)
   begin
    for(k=0;k<=1;k=k+1)
     begin
      #20
      $display("A=%b B=%b Ci=%b, Co-Sum=%b%b", A, B, Ci, Co, S);
      if ({Co,S} != A + B + Ci)
          $display("ERROR: Co-Sum should equal %b, is %b",
                   (A + B + Ci), {Ci,S});
      Ci=~Ci; // invert Cin
     end
   B[3:0] = B[3:0] + 4'b0001; // add the bits
   end
  A = A+1; // shorthand notation for adding
 end

$display("Simulation finished... ");
end
```

Print inputs and outputs

Check output against calculation

Print error if mismatch

Toggle Carry In

Increment B

Increment A

Verilog Example: 4b Adder (cont.)

```
// Loop-Based Adder4b Stimulus
integer i,j,k;
initial
begin

   A[3:0] = 4'b0000;
   B[3:0] = 4'b0000;
   Ci = 1'b0;

$display("Starting simulation...");

for(i=0;i<16;i=i+1)
 begin
  for(j=0;j<16;j=j+1)
   begin
    for(k=0;k<=1;k=k+1)
     begin
      #20
      $display("A=%b B=%b Ci=%b, Co-Sum=%b%b", A, B, Ci, Co, S);
      if ({Co,S} != A + B + Ci)
          $display("ERROR: Co-Sum should equal %b, is %b",
                    (A + B + Ci), {Ci,S});
      Ci=~Ci; // invert Cin
     end
    B[3:0] = B[3:0] + 4'b0001; // add the bits
   end
  A = A+1; // shorthand notation for adding
 end

$display("Simulation finished... ");
end
```

Behavioral Verilog code!

Summary

- **Think about hardware when coding HDL!**
  - Initial module development using behavioral code
  - Translated to structural code, schematic, layout as design proceeds
- **Testbench implemented using behavioral code – doesn’t change**
  - Write once, use many times, so worth it to design well
- **Test each refinement of design using same testbench (Regression Testing)**
  - Catch bugs at each step of design process

Other Verilog Resources

- **H. Wang, "Introduction to Verilog Hardware Description Language", PDF slides on SmartSite**
- **Aldec, Inc., Verilog Interactive Tutorial (AldecVerilogEvita.exe)**
  - Windows executable on SmartSite; no warranty and standard disclaimer applies
- **Weste and Harris, "CMOS VLSI Design", 3rd. ed., Appendix A.**
- **… many other resources on the web.**