**EECS150 - Digital Design**  
**Lecture 8 - Hardware Description Languages**  

September 19, 2002  
John Wawrzynek

---

**Outline**
- Netlists
- Design flow
- What is a HDL?
- Verilog
  - history
  - examples

---

**Netlist**
- A key data structure (or representation) in the design process is the “netlist”:
  - Network List
- A netlist lists components and connects them with nodes:

  ![Netlist Diagram]

  ```
  n1 g1.in1
  n2 g1.in2
  n3 g2.in1
  n4 g2.in2
  n5 g1.out g3.in1
  n6 g2.out g3.in2
  n7 g3.out
  g1 "and"
  g2 "and"
  g3 "or"
  ```

  - Netlist is what is needed for simulation and implementation.
  - Could be at the transistor level, gate level, ...
  - Could be hierarchical or flat.
  - How do we generate a netlist?

---

**Design Flow**
- Design Entry
  - High-level Analysis
    - Technology Mapping
    - Low-level Analysis
Design Flow

- Circuit is described and represented:
  - Graphically (Schematics)
  - Textually (HDL)
- Result of circuit specification (and compilation) is a netlist of:
  - generic primitives - logic gates, flip-flops, or
  - technology specific primitives - LUTs/CLBs, transistors, discrete gates, or
  - higher level library elements - adders, ALUs, register files, decoders, etc.

Design Entry | High-level Analysis | Technology Mapping | Low-level Analysis

Fall 2002 EECS150 - Lec08-HDL Page 5

Design Flow

- High-level Analysis is used to verify:
  - correct function
  - rough:
    - timing
    - power
    - cost
- Common tools used are:
  - simulator - check functional correctness, and
  - static timing analyzer
  - estimates circuit delays based on timing model and delay parameters for library elements (or primitives).

Design Entry | High-level Analysis | Technology Mapping | Low-level Analysis

Fall 2002 EECS150 - Lec08-HDL Page 6

Design Flow

- Technology Mapping:
  - Converts netlist to implementation technology dependent details
    - Expands library elements,
    - performs:
      - partitioning,
      - placement,
      - routing
- Low-level Analysis
  - Simulation and Static Tools perform low-level checks with:
    - accurate timing models,
    - wire delay
  - For FPGAs this step could also use the actual device.

Design Entry | High-level Analysis | Technology Mapping | Low-level Analysis

Fall 2002 EECS150 - Lec08-HDL Page 7

Design Flow

Netlist: used between and internally for all steps.

Design Entry | High-level Analysis | Technology Mapping | Low-level Analysis

Fall 2002 EECS150 - Lec08-HDL Page 8
Design Entry

- Schematic entry/editing used to be the standard method in industry
- Used in EECS150 until last semester
- Schematics are intuitive. They match our use of gate-level or block diagrams.
- Somewhat physical. They imply a physical implementation.
- Require a special tool (editor).
- Unless hierarchy is carefully designed, schematics can be confusing and difficult to follow.

Hardware Description Languages are the new standard
- except for PC boards

HDLs

- “Structural” example:
  
  Decoder(output x0,x1,x2,x3; inputs a,b) 
  
  { 
    wire abar, bbar;
    inv(bbar, b);
    inv(abar, a);
    nand(x0, abar, bbar);
    nand(x1, abar, b);
    nand(x2, a, bbar);
    nand(x3, a, b);
  }

- “Behavioral” example:
  
  Decoder(output x0,x1,x2,x3; inputs a,b) 
  
  { 
    case [a b]
    00: [x0 x1 x2 x3] = 0x0;
    01: [x0 x1 x2 x3] = 0x2;
    10: [x0 x1 x2 x3] = 0x4;
    11: [x0 x1 x2 x3] = 0x8;
    endcase;
  }

Verilog

- A brief history:
  - Invented as simulation language. Synthesis was an afterthought. Many of the basic techniques for synthesis were developed at Berkeley in the 80’s and applied commercially in the 90’s.
  - Around the same time as the origin of Verilog, the US Department of Defense developed VHDL. Because it was in the public domain it began to grow in popularity.
  - Afraid of losing market share, Cadence opened Verilog to the public in 1990.
  - An IEEE working group was established in 1993, and ratified IEEE Standard 1394 in 1995.
  - Verilog is the language of choice of Silicon Valley companies, initially because of high-quality tool support and its similarity to C-language syntax.
  - VHDL is still popular within the government, in Europe and Japan, and some Universities.
  - Most major CAD frameworks now support both.
  - Latest HDL: C++ based. OSCI (Open System C Initiative).

Basic Example: 2-to1 mux

//2-input multiplexor in gates
module mux2 (in0, in1, select, out);
  input in0, in1, select;
  output out;
  wire w0,w1,w2;
  not (s0, select);
  and (w0, s0, in0),
  (w1, select, in1);
  or (out, w0, w1);
endmodule // mux2

Notes:
- comments
- "module"
- port list
- declarations
- wire type
- primitive gates
Announcements

Hierarchy & Bit Vectors

2-to-1 mux behavioral description

Behavioral 4-to1 mux
Behavioral with Bit Vectors

// Behavioral model of 32-bit wide 2-to-1 multiplexer.
module mux32 (in0, in1, select, out);
  input [31:0] in0, in1;
  input select;
  output [31:0] out;
  //
  reg [31:0] out;
  always @ (in0 or in1 or select)
    if (select) out = in1;
    else out = in0;
endmodule // Mux

// Behavioral model of 32-bit adder.
module add32 (S, A, B);
  input [31:0] A, B;
  output [31:0] S;
  //
  always @ (A or B)
    S = A + B;
endmodule // Add

“Dataflow” Descriptions of Logic

// Dataflow description of mux
module mux2 (in0, in1, select, out);
  input in0, in1, select;
  output out;
  assign out = (~select & in0) | (select & in1);
endmodule // mux2

Alternative:
assign out = select ? in1 : in0;

Sequential Logic

// Parallel to Serial converter
module ParToSer (LD, X, out, CLK);
  input [3:0] X;
  input LD, CLK;
  output out;
  reg out;
  reg [3:0] Q;
  assign out = Q[0];
  always @(posedge CLK)
    if (LD) Q = X;
    else Q = Q >> 1;
endmodule // mux2

module FF (CLK, Q, D);
  input D, CLK;
  output Q;
  reg Q;
  always @(posedge CLK)
    Q = D;
endmodule // FF

Testbench

module testmux;
  reg a, b, s;
  wire f;
  reg expected;

  mux2 myMux (.select(s), .in0(a), .in1(b), .out(f));
  initial begin
    s = 0; a = 0; b = 1; expected = 0;
    #10 a = 1; b = 0; expected = 1;
    #10 a = 1; b = 1; expected = 1;
    end

  initial $monitor("select=%b in0=%b in1=%b out=%b, expected out=%b time=%d"
    , s, a, b, f, expected, $time);
endmodule // testmux
**Verilog does not turn hardware design into writing programs!**

- The fact that Verilog looks similar to programming languages fools some people into thinking that they can design hardware by writing programs. Not so.
- Verilog is a hardware description language. The best way to use it is to first figure out the circuit you want, then figure out how to express it in Verilog.
- The behavioral constructs hide a lot of the circuit details but you as the designer must still manage the structure, data-communication, parallelism, and timing of your design. Not doing so leads to very inefficient designs.

**More Verilog Help**

- The lecture notes only cover the very basics of Verilog and mostly just the conceptual issues.
- The Mano textbook covers Verilog with many examples.
- The Bhasker book is a good tutorial. On reserve in the Engineering library (starting Friday).

- The complete language specification from the IEEE is available on the class website under “Refs/Links”