Logisim
HDL language and Logisim

• Most real-world hardware design is done using a text-based hardware description language – VHDL, Verilog, etc.
  – Schematics can be compiled into a text description
  – Can use a simulator to test the circuit
  – Other back-end tools optimize, perform layout and wire routing, floorplan, etc.
  – Final spec is either downloaded onto a programmable device, or etched into silicon.

• We will use Logisim for all hardware design
  – Interactive, graphical schematic editor
  – Educational use, user-friendly
Verilog Example: Decoder

```verilog
// File Name : decoder_using_case.v
// Function  : decoder using case
// Coder      : Deepak Kumar Tala
//-------------------------------
module decoder_using_case(
    input [3:0] binary_in,
    decoder_out,
    // 4 bit binary input
    // 16 bit out
    enable); // Enable for the decoder

input [3:0] binary_in;
input enable;
output [15:0] decoder_out;
reg [15:0] decoder_out;

always @ (enable or binary_in)
begin
    decoder_out = 0;
    if (enable) begin
        case (binary_in)
            4'h0: decoder_out = 16'h0001;
            4'h1: decoder_out = 16'h0002;
            4'h2: decoder_out = 16'h0004;
            4'h3: decoder_out = 16'h0008;
            4'h4: decoder_out = 16'h0010;
            4'h5: decoder_out = 16'h0020;
            4'h6: decoder_out = 16'h0040;
            4'h7: decoder_out = 16'h0080;
            4'h8: decoder_out = 16'h0100;
            4'h9: decoder_out = 16'h0200;
            4'hA: decoder_out = 16'h0400;
            4'hB: decoder_out = 16'h0800;
            4'hC: decoder_out = 16'h1000;
            4'hD: decoder_out = 16'h2000;
            4'hE: decoder_out = 16'h4000;
            4'hF: decoder_out = 16'h8000;
        endcase
    end
end
```
Other CAD tools in circuit design

• Circuit level tool vendors
  – Cadence, Synopsys, etc.
  – other smaller players

• Board level tool vendors
  – Altium, Eagle and many more
HDL language and Logisim

• Most real-world hardware design is done using a text-based hardware description language – VHDL, Verilog, etc.
  – Schematics can be compiled into a text description
  – Can use a simulator to test the circuit
  – Other back-end tools optimize, perform layout and wire routing, floorplan, etc.
  – Final spec is either downloaded onto a programmable device, or etched into silicon.

• We will use Logisim for all hardware design
  – Interactive, graphical schematic editor
  – Educational use, user-friendly
To be covered...

• Pins and subcircuits
• Probes for debugging
• Bundles/splitters
• Logging
• Test vectors
• S-R latch, D latch, D flip-flop
• Examples
Example Circuit: 1 bit 2:1 Mux

- \( S = P \) if \( R == 0 \)
- \( S = Q \) if \( R == 1 \)
Example Circuit: 32-bit 2:1 Mux
Subcircuits: 2:1 Mux and Controller

- $S = Q$ if $R = 010$
- $S = P$ otherwise
Logging and Test Vectors

Log File

Test Vector Truth Table

<table>
<thead>
<tr>
<th>status</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pass</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>pass</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pass</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>pass</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>pass</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>pass</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>pass</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logisim Don’ts

• Leave wires floating
  – Works in logisim
  – Breaks in real life

• Use a multiplexor instead of a controlled buffer
Logisim Don’ts (cont.)

• Don’t reinvent the wheel
  – Waste time
  – Confusing to grade

• Almost every component is customizable
  – Number of inputs
  – Input bit width
Logisim Don’ts (cont.)

• Avoid Constant input
  – Constants are almost never necessary
  – Exception is supplying value to extra input

• Try to optimize away before using
  – Truth table
Logisim Don’ts (cont.)

- Don’t make trivial subcircuits
  - Sub-circuit needs to perform some meaningful logic functions
  - You will never have a C function just to add two numbers up, do you?

- Problems
  - Wasting time specifying inputs and outputs of small circuits
  - Big hierarchy hard to understand
Logisim Don’ts (cont.)

• Don’t use invisible splitters
  – All you really need is just a wire
  – It is really hard to see them when we grade
Logisim Don’ts (cont.)

• Don’t work from Right to Left
Some more information

• MIPS assignment:
  – 32-bit ALU
  – 32-bit pipelined processor

• Looking for help?
  – Course webpage http://www.cs.cornell.edu/courses/cs3410/2011sp/
  – Newsgroup: cornell.class.cs3410
  – Staff email list: cs3410-staff-l@cs.cornell.edu