Hardware Description Languages

- VHDL
  - VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
  - Taught in EEC 180B
- Verilog
  - Generally more common in industry
  - Similarities with C

HDL

- Hardware Description Language
- Design process
  - Think
  - Draw diagram of hardware, figure out where logic and registers go
  - Think
  - Then…
    - Write verilog
    - Test it

HDL

- You'll write far better HDL code if you think of it differently than a standard programming language
  - a way to code an *algorithm*
- Hardware description language
  - a way to code *hardware*!

HDL Simulation Tools

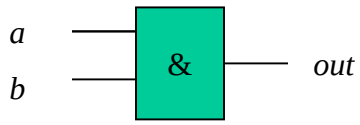
- Synopsys
  - VCS – simulator
  - Virsim – waveform viewer and environment
- Cadence
  - Verilog XL – simulator
  - NC Verilog – simulator
  - Simvision – waveform viewer and environment
- Many others…

Verilog

- *Modules* are basic building blocks
- Main ways to do logic
  - *wires, assign* statements
  - *registers, always* blocks

wire, assign

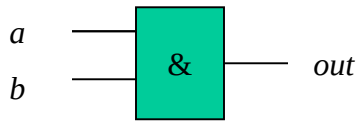
- Picture "always active" hardwired logic
- Declare all wires



wire, assign

- Example:

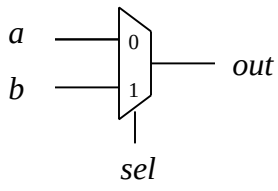
```
wire out;
assign out = a & b;
```



wire, assign

- Example:

```
wire out;
assign out = sel ? b : a;
```



reg, always

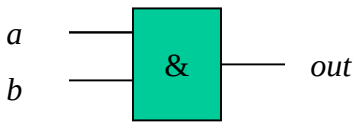
- Picture a much more general way of assigning "wires" or "nodes"
- Use "if/else" and "case" statements
- Can, but don't use "for loops" in logic blocks (ok for testing code)
- Procedural execution – statements execute in order

reg, always

- Example:

```
reg out;
always @(a or b) begin
   out = a & b;
   end
```

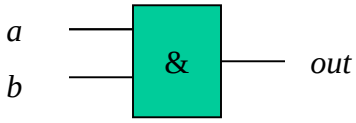
"Sensitivity list" declares when to execute the code within the *always* block.



reg, always

- Example:

```
reg out;
always @(a or b) begin
   out = a & b;
   end
```

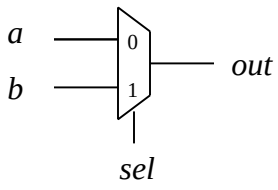


Operation goes like this:
Anytime *a* or *b* changes, re-evaluate *out = a & b;* otherwise, keep *out* the same.
Notice that there is the possibility of a problem (memory) if all inputs are not included in the sensitivity list.

wire, assign

- Example:

```
reg out;
always @(a or b or sel) begin
   if (sel == 1'b1) out = b;
   else             out = a;
end
```

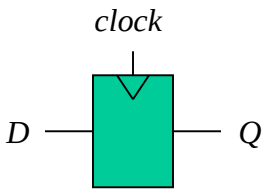


reg, always

- Building a flip-flop

```
reg Q;
always @(posedge clock) begin
  Q = D;
end
```

- The only time the flip-flop output *Q* gets updated is at the positive edge of the clock signal -> exactly what we want for a flip-flop.

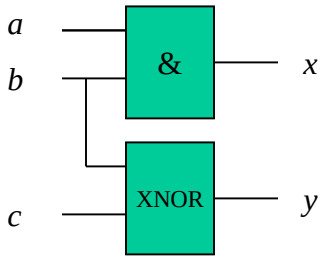


Concurrent Operation

- You can think of these as operating on independent circuits (remember *hardware* orientation).

```
always begin
   x = a & b;
   #5;        // 5-unit delay
   end

always begin
   y = ~(b ^ c);
   #7;        // 7-unit delay
   end
```



- Both functions compute during the same "simulation" time, just as with real hardware

Typical Design Flow

- Synthesis tool converts HDL into a netlist of appropriate components
  - FPGA: "CLB" or "slice" components
  - Custom chip: standard cells