Useful Things to Know

Norm
Administrative

• Midterm Grading Finished
  – Stats on course homepage
  – Pickup after this lab lec.
  – Regrade requests within 1wk of posted solution

• Homework deadline extended to next friday
Design Flow

- Description
- Design Conception
- Implementation
- Verification (Debugging)
Design Flow

- Description
- Design Conception
- Implementation
- Verification (Debugging)
## Classification of Elements in a Digital Circuits

<table>
<thead>
<tr>
<th></th>
<th>Storage (Sequential)</th>
<th>Combinational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Datapath</strong></td>
<td>Flipflops, Counter, Shift-reg…</td>
<td>Arith: Add, Sub, Mult, Comp… Logic: And, Or, Xor… Routing?: Shifter, Mux, Tri-st…</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>States for FSM (flip-flops)</td>
<td>Random logic (FSM, next state, output)</td>
</tr>
</tbody>
</table>

*Memory is an exception to the above chart. They can operate with or without clock and can store values*
Reminder From Lecture

• Sequential, or Storage
  – These are **the only things**\(^*\) that can store values. They are controlled by a clock and their output value can only change on a clock-edge.

• Combinational
  – These **cannot** hold state, output is purely a function of input.
Verilog...

- IS a Hardware Description Language
- Is NOT a programing language
  - Design your circuit first then write the code.
- Was initially designed for simulating hardware
- Was NOT initially designed for generating hardware
  - Not all “valid” verilog turns into hardware.
  - Some verilog turns into inefficient hardware implementation (too many CLBs….)
Verilog for Sequential Elements

always @ (posedge CLK or posedge a)
if (a)
    Q <= R;
else if (b)
    Q <= R;
else if (c)
    Q <= D;

Express any sequential logic by substituting different values, variables for a, b, R, D

a = asynchronous set-reset line
b = synchronous set-reset line
R = set-reset value (should be constant)
D = next value for Q (could be expression)

Counter

rst→Q  a = rst, c = en, d = Q + 1;
en→

Note: it is okay to write Q <= Q…, in this case because Q is a storage element. This is not always the case.
Verilog for Combinational logic

Can be written in two ways

1. `assign O = Y;`
2. `always @ (all inputs) begin 
   ...
   ...
   end`

Example:

```
Case (A) 
1 : begin 
   if (B) 
     O <= C; 
   end 
```

Be careful!

A, B, C are all inputs!

Notice:

- O must be of type `wire` for `assign` statement and type `reg` for `always` statement.
- However after synthesis O will physically be a wire in the circuit.

Assign O = E ? I : 1'bZ

Always @ (E or I) if (E) 
   O <= I;
else 
   O <= 1'bZ;
always @(posedge clk)
begin
    c <= a+1;
    b <= c+1;
end

always @(posedge clk)
begin
    c = a+1;
    b = c+1;
end
Common Pitfall

1. Not assigning a wire outputs. (incomplete truth table therefore variable must remember previous values)

2. Assigning a variable to itself. (same as not assigning since reg types remembers it’s value if it is not assigned)

```
1 always @(A or B)
     begin
       if (A)
         D = B;
     End

2 always @(A or B)
     begin
       if (A)
         D = B;
       else
         D = D;
     End
```

Template for A latch (look familiar?)

```
always @(GATE or DIN)
begin
  if (GATE)
    DOUT = DIN;
End
```

This circuit is not combinational! Output is not just a function of inputs
Tri-State or Mux?

- Desired functionality
  - To read one of many results depending on come control information

- Notice the tri state Has 2 control lines where as the mux has only one
Xilinx 4000 CLB Structure

LUTs cannot output high-impedence “Z” therefore each CLB also has a pair of Tri-states
Tri-State or Mux on Xilinx?
Using Tri-states

• Pros
  – Saves LUTS so you can use them for other things
  – Drives Long lines (Might be faster than other types of routing)
  – If you don’t use them then is just a wasted resource

• Cons
  – Decoded select signal (make sure only one select line is high at anytime!!)
  – Using up fast transmission long lines