

CSSE 232

Computer Architecture I

Verilog

What it is

Verilog is hardware description language

- Describes the functions of a hardware circuit
- Can be used to test circuits also
- Looks similar to C (+, -, !=, ~, etc)
- Multiple parts can run in parallel (just like the datapath)

Basic syntax

- Verilog doesn't use braces (`{, }`), instead uses 'begin' and 'end'
- Components are enclosed in module tags
- A list of inputs and output follow the module name

Verilog:

```
module namegoeshere(an_input, an_output);  
    ... //logic  
endmodule
```

C:

```
int namegoeshere(int an_input, int* an_output)  
{  
    ... //logic  
}
```

Modules

- Functional unit in verilog
 - Module name followed by list of all inputs and outputs
 - Defined inputs and output (always wires)
 - Local variables for module
 - Module logic (initials, always, or assign)

```
module awesomeUnit(a, b, c);  
    input a;  
    input b;  
    output c;  
    assign c = a && b;  
endmodule
```

Execution blocks

- Two types of execution: initial and always
 - initial runs one time and stops
 - always runs over and over
 - can be triggered on an event with @ (see control example)
 - also 'assign' for continuous assignment (combinational logic)

```
module modtest;  
  always begin clk=~clk; #5; end  
  //or  
  initial begin  
    clk = 0;  
    #5;  
    clk = 1;  
    #5;  
    clk = 0;  
    #5;  
    clk = 1;  
  end  
endmodule
```

Variables

- Two types of variables: `wire` and `reg`
 - Wire types are just wires, used for combinational logic
 - Regs are similar to registers, used for combinational or sequential
 - Can be busses with `[x:y]` notation
- Two types of assignment: blocking (`=`) and non-blocking (`<=`)
 - Blocking: nothing else happens until the value is assigned
 - Non-blocking: the assignment happens while everything else is happening
- Values are specified as `s'bx` for size s , base b , value xx

Variables

Verilog

```
module modtest;
    ... setup goes here
    //blocking assign 255
    //blocking assign 1337
    //non-blocking assign 7
    reg a = 16'hff;
    reg b = 32'd1337;
    reg c <= 8'b00000111;
endmodule
```

C

```
int functest() {
    //all assigns block
    //C can't do binary
    short a = 0xff;
    int = 1337;
    char = 7;
}
```

Time

Unless you indicate otherwise, everything happens at the same time!

- Blocking assignments are serialized (block next action)
- Non-blocking assignments are parallelized (do not block)
- A single module can have multiple execution paths
 - All execution in a module happen in parallel!
- Indicate a delay with the # operator
 - `#5 //wait 5ns`

Tasks

Non-synthesizable concepts: cannot be expressed in hardware; host system will execute

- `$write` : like `printf`
- `$display` : like `printf` with an implied newline
- `$finish` : stop the simulation

Conditions and loops

- Similar to C
 - `if(i==0) begin ... end`
 - `for(i=0; i<16; i=i+1) begin ... end`
 - `while(i<10) begin ... end`
- Other constructs
 - forever: like always block
 - repeat: fixed number of repeats

```
initial begin
  clk = 0;
  forever begin          repeat(9) begin
    #5;                 $display("Everything is awesome!");
    clk = ~clk;        end
  end
end
```