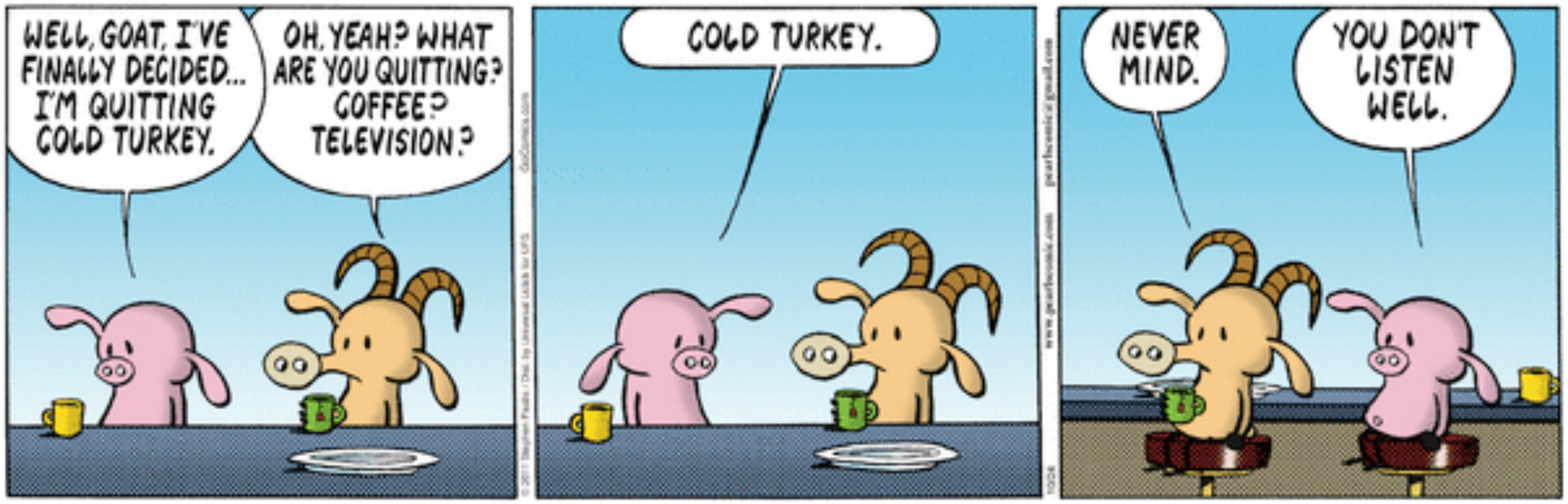
WELL, GOAT. I'VE FINALLY DECIDED... I'M QUITTING COLD TURKEY.
OH, YEAH? WHAT ARE YOU QUITTING? COFFEE? TELEVISION?
COLD TURKEY.
NEVER MIND.
YOU DON'T LISTEN WELL.

Lecture 10: Verilog

- Verilog resources
- Verilog vs VHDL
- HDL Design questions
- Verilog examples

Verilog Resources

Sakai: folder in Resources

Pacific Library (electronic books):

- Verilog digital system design : RT level synthesis, testbench, and verification
  by Zainalabedin Navabi
- FSM-based digital design using Verilog HDL
  by Peter D. Minns
- Verilog Quickstart : a practical guide to simulation and synthesis in Verilog
  by James M. Lee
- The Verilog hardware description language
  by D. E. Thomas

Online:

- http://en.wikipedia.org/wiki/Verilog
- http://www.asic-world.com/verilog/veritut.html
- http://www.altera.com/support/examples/verilog/verilog.html

Verilog vs VHDL

Verilog	VHDL
C-like language	More explicit typing
Easier coding	Less error prone
US Industry supported (sort of)	Government/research and external US supported (sort of)

Fundamentally, best to know both with serious expertise in at least one!

HDL Design

How do we:

- Define the ports?
- Implement sequential logic?
  - Where do we place the code?
  - How do we define the clock and initiate a block of logic?
  - How do we declare variables?
- Design hierarchically?

HDL Design: Port Declarations

VHDL:

```
ENTITY blah IS
  PORT (a : in STD_LOGIC;
        b : out STD_LOGIC);
END blah;
```

HDL Design: Port Declarations

Verilog:

```
module blah
(input a,
  output b);

endmodule;
```

HDL Design: Port Declarations

VHDL:

```
ENTITY blah IS
  PORT (a : in STD_LOGIC_VECTOR(2 DOWNTO 0);
        b  :  out STD_LOGIC_VECTOR(0 TO 1));
END blah;
```

HDL Design: Port Declarations

Verilog:

```
module blah
(input [2:0] a,
 output [0:1] b);

endmodule;
```

HDL Design: Port Declarations

Verilog:

```
module blah (a,b);
  input [2:0] a;
  output [0:1] b;

endmodule;
```

HDL Design: Sequential – Code?

Where do we place the code?

VHDL:

```
ENTITY blah IS
  PORT (a : in STD_LOGIC_VECTOR(2 DOWNTO 0);
        b  :  out STD_LOGIC_VECTOR(0 TO 1));
END blah;

ARCHITECTURE Behavior OF blah IS
BEGIN
  CODE GOES HERE
END Behavior;
```

HDL Design: Sequential – Code?

Where do we place the code?

Verilog:

```
module blah (a,b);
  input [2:0] a;
  output [0:1] b;

  CODE GOES HERE
endmodule;
```

HDL Design: Sequential – Clocks?

How do we define the clock and initiate a block of logic?

VHDL:

```
ENTITY blah IS
   PORT (clk : in STD_LOGIC;
          a : in STD_LOGIC_VECTOR(2 DOWNTO 0);
          b : out STD_LOGIC_VECTOR(0 TO 1));
END blah;

ARCHITECTURE Behavior OF blah IS
BEGIN
  PROCESS(clk)
  BEGIN
    IF(rising_edge(clk)) THEN
    END IF;
  END PROCESS;
END Behavior;
```

HDL Design: Sequential – Clocks?

How do we define the clock and initiate a block of logic?

Verilog:

```
module blah (clk,a,b);
  input clk;
  input [2:0] a;
  output [0:1] b;

  always @(posedge clk) begin
  end
endmodule;
```

HDL Design: Sequential – Variables?

How do we declare variables?

VHDL:

```
ENTITY blah IS
   PORT (clk : in STD_LOGIC;
            a : in STD_LOGIC_VECTOR(2 DOWNTO 0);
            b : out STD_LOGIC_VECTOR(0 TO 1));
END blah;

ARCHITECTURE Behavior OF blah IS
   SIGNAL y : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
  PROCESS(clk)
    VARIABLE resync : STD_LOGIC_VECTOR(1 DOWNTO 0);
  BEGIN
    IF(rising_edge(clk)) THEN
    END IF;
  END PROCESS;
END Behavior;
```

HDL Design: Sequential – Variables?

How do we declare variables?

Verilog:

```
module blah (clk,a,b);
  input clk;
  input [2:0] a;
  output [0:1] b;

  reg [0:1] b;   // storage elements
  wire [1:0] y;  // literally wires

  always @(posedge clk) begin
  end
endmodule;
```

HDL Design: Hierarchy

VHDL:

```
ENTITY blah IS
   PORT (clk : IN STD_LOGIC;
         a : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
         b : OUT STD_LOGIC_VECTOR(0 TO 1));
END blah;

ARCHITECTURE Behavior OF blah IS
   SIGNAL y : STD_LOGIC_VECTOR(1 DOWNTO 0);
   SIGNAL outclk : STD_LOGIC;

   COMPONENT masterclk
    PORT (iclk : IN STD_LOGIC;
          oclk : OUT STD_LOGIC);
   END COMPONENT;
BEGIN
  clkdivide : masterclk PORT MAP(iclk => clk, oclk => outclk);

  PROCESS(outclk)
    VARIABLE resync : STD_LOGIC_VECTOR(1 DOWNTO 0);
  BEGIN
    IF(rising_edge(outclk)) THEN
    END IF;
  END PROCESS;
END Behavior;
```

HDL Design: Hierarchy

Verilog:

```
module blah (clk,a,b);
  input clk;
  input [2:0] a;
  output [0:1] b;

  reg [0:1] b;
  wire [1:0] y;
  reg outclk;

  masterclk clkdivide(.iclk(clk),.oclk(outclk));

  always @(posedge outclk) begin
  end
endmodule;
```

Verilog: Numbers

Verilog has four different bit values:

- 0, logic zero
- 1, logic one
- z, high impedance
- x, unknown

The general syntax is:

{bit width}’{base}{value}

For example:

– 4’d14	// 4-bit value, specified in decimal
– 4’he	// 4-bit value, specified in hex
– 4’b1110	// 4-bit value, specified in binary
– 4’b10xz	// 4-bit value, with x and z, in binary

Verilog Examples

```
module mux4(a,b,c,d,sel,y);
  input a, b, c, d;
  input [1:0] sel;
  output y;

  always @(a or b or c or d or sel)
    case (sel)
      2’b00: y <= a;
      2’b01: y <= b;
      2’b10: y <= c;
      2’b11: y <= d;
      default : y <= 1’bx;
    endcase

endmodule
```

Verilog Examples

```
module dff(clk,reset,d,q);
  input clk, reset, d;
  output q;

  always @(posedge clk)
    if (~reset) begin
      q <= 1’b0;
    end else begin
      q <= d;
    end

endmodule
```

Verilog Examples

```
module dff(clk,reset,d,q);
  input clk, reset, d;
  output q;

  always @(posedge clk or negedge reset)
    if (~reset) begin
      q <= 1’b0;
    end else begin
      q <= d;
    end

endmodule
```

Verilog Examples

```
module fsm1(clk,w,z);
  parameter idle = 2’b00;
  parameter state1 = 2’b01;
  parameter state2 = 2’b10;

  input clk, w;
  output z;

  reg [1:0] state;

  always @(posedge clk)
  begin
    case(state)
      idle:
        if(w == 1’b1)
          state <= state1;
        else
          state <= state2;
      state1:  state <= idle;
      state2:  state <= idle;
      default:  state <= idle;
    endcase
  end

  always @*
  begin
    z <= (state == state1);
  end

endmodule
```

Verilog Examples

```
module fsm1(clk,w,z);
  parameter idle = 2’b00;
  parameter state1 = 2’b01;
  parameter state2 = 2’b10;

  input clk, w;
  output z;

  reg [1:0] state;

  always @(posedge clk)
  begin
    case(state)
      idle:
        if(w == 1’b1)
          state <= state1;
        else
          state <= state2;
      state1:  state <= idle;
      state2:  state <= idle;
      default:  state <= idle;
    endcase
  end

  assign z = (state == state1);

endmodule
```

Quizzes, Project, and Questions?

- Quizzes delayed one week each:
  - Next one is two Tuesdays from today
  - Other one is Tuesday after Thanksgiving
- Project:
  - Handout posted
  - Need to form groups of 2-3
  - Need to meet with me to discuss topics
- Need all boards back by Nov. 30!