2012 Fall CSE140L

Digital Systems Laboratory

Lecture #5

by

Dr. Choon Kim

CSE Department, UCSD
chk034@cs.ucsd.edu
Issues to be covered

About pop quiz

Some hints, suggestions, guides, etc. for LAB#2

Verilog data types... net type(e.g., wire) vs. register type(e.g., reg)

always block-----> one of key parts in Verilog!

Understand the Blocking vs. the Non-blocking assignments
Introduction to Verilog HDL

Home work (Reading materials)

Warning:
Some information contained in the following reading materials (or any material in general) may be incorrect and not working. Only way to verify the correctness of the information is to test it by simulating and synthesizing using our Altera Quartus SW & HW kit.

1. [Quartus II example with Verilog]: Note: You can do this without understanding of Verilog. Follow the instructions described in tut_quartus_intro_verilog_de1. This is the same homework in Lecture#1 except Verilog is used as a design entry instead of Schematic diagram.

2. [Tutorial for beginner]: Follow Verilog short tutorial to make yourself familiar with this new design entry methodology.

3. [Manual]: intro_verilog_manual

4. [Short Reference Card]: Two-page Card  Multiple-page Card

5. In addition, there are numerous Verilog information & tutorials available on the web in doc, book, & video format. The Verilog on wiki is an excellent place to learn the background. This Verilog Tutorial is one of tutorial sites. You can use whatever you feel comfortable.
Up counter

NOTE: All the Verilog codes in any slide are not necessary working solution.

module UP_COUNT (input RESET, ENABLE, INCOMING_SIG,
output reg [15:0] Z);

always @ (negedge RESET, posedge INCOMING_SIG) begin
  if (~RESET) Z = 0;
  else if (ENABLE) begin
    if (Z == 16'hFFFF) Z = 0; // max value, or modulo operation
    else Z = Z + 1;
  end
end // of always
endmodule
Real-time clock signal

- How to make a real-time clock of arbitrary period on DE1 board? e.g., 1 Hz clock

How about using Verilog delay construct, #N, (e.g., assign #2 n1 = ab; )?

If a fast real-time clock is available on board (e.g., 24Mhz), we can count it up to certain times and generates a signal of our interest.
counter chain: digital clock example

modulo 60 counter
Input  Output
second

modulo 60 counter
Input  Output
minute

modulo 60 counter
Input  Output
hour
Structural-level Counter description....
Johnson Counter: A Shifter with An Inverted Feedback Loop

1) Given n flip-flops, we have 2n states. Much less than previous counters. But Johnson is fast!
2) Only one output changes (low power).
3) Each output has n clock width (symmetrical).
4) Reset is needed. (ie, starts with 010, the counter ends up as 010->101->010->101)
Odd Length Walking – Ring Counter
A Shifter with Twisted Feedback Loops

The counter works itself back to the proper sequence.

n JK F-Fs => 2n-1 states

Time Steps | A B C
---|---|---|---
0 | 0 0 0
1 | 1 0 0
2 | 1 1 0
3 | 0 1 1
4 | 0 0 1
5 | 0 0 0
6 | 1 0 0
7 | 1 1 0

\[
\begin{array}{ccc|c|c}
J_A & K_A & A(t+1) \\
0 & 0 & A(t) \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & A'(t)
\end{array}
\]
Pseudo Random Sequencer

\[ D_0 = Q_3 \text{ XNOR } Q_2 \]

\[ n = 4, \text{ length } = 15 \]
module DOWN_COUNT (input RESET, ENABLE, INCOMING_SIG, output reg [15:0] Z);

always @ (negedge RESET, posedge INCOMING_SIG) begin
    if (!RESET) Z = 16'hFFFF;
    else if (ENABLE) begin
        if (Z == 0) Z = 16'hFFFF; // max value, or modulo operation
        else Z = Z - 1;
    end
end // of always
endmodule
module RING_COUNT (  input  RESET,  ENABLE,  INCOMING_SIG,  
     output  reg [15:0]  Z);

    always @ (negedge RESET,  posedge  INCOMING_SIG)  begin
        if (~RESET)       Z = 16'h8000;
        else  if (ENABLE) begin
            Z      <=  Z << 1;
            Z[0]  <=  Z[15];
        end
    end  // of always
endmodule
module EVEN_PARITY_CHECKER ( input [8:0] INCOMING_SIG,
output PARITY_BIT);

always @ (negedge RESET, posedge INCOMING_SIG) begin
  if (~RESET)
    PARITY_BIT = 0;
  else
    PARITY_BIT = ^(INCOMING_SIG);
end // of always

dendmodule
Various ADDER designs

module PARALLEL_ADDER (
    input [7:0] A, B,
    input Cin,
    output [7:0] Z,
    output Cout
);

assign { Cout, Z } = A + B + Cin;      // design #1

or,

always @(A, B, Cin)
    { Cout, Z } = A + B + Cin;      // design #2

endmodule
Incoming LAB#3 Preparation:

Vending Machine Controller Design

• It has many states – reset, 5 cents, 10 cents, ....
• How to move from one state to other state?
• How to detect & handle dispensing?
• How to detect & handle credit-card and coin input?
• How to generate moving message?
• ....
Mealy vs. Moore FSM

Mealy Machine: \( y(t) = f(x(t), s(t)) \)
Moore Machine: \( y(t) = f(s(t)) \)

\( s(t+1) = g(x(t), s(t)) \)

Mealy Machine

Moore Machine
A circuit which removes one 1 from every string of 1s on the input:

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>..... 00000......</td>
<td>=&gt; ..... 00000......</td>
</tr>
<tr>
<td>..... 01010......</td>
<td>=&gt; ..... 00000......</td>
</tr>
<tr>
<td>..... 00110......</td>
<td>=&gt; ..... 00100......</td>
</tr>
<tr>
<td>..... 01110......</td>
<td>=&gt; ..... 01100......</td>
</tr>
<tr>
<td>..... 11011......</td>
<td>=&gt; ..... 10010......</td>
</tr>
</tbody>
</table>
FSM design example – Moore vs. Mealy

- Remove one 1 from every string of 1s on the input
Moore FSM Verilog model example: *(Not necessary working solution)*

```verilog
module reduce (input clk, in,
               output reg out);

parameter zero = 2'b00, one1 = 2'b01, two1s = 2'b10;
reg[2:1] state, next_state;

always @(posedge clk) begin
  state = next_state;
end

always @(in, state) begin
  case (state)
    zero:    begin
      out = 0;
      if (in) next_state = one1;
      else    next_state = zero;
    end
    one1:    begin
      out = 0;
      if (in) next_state = two1s;
      else    next_state = zero;
    end
    two1s:   begin
      out = 1;
      if (in) next_state = two1s;
      else    next_state = zero;
    end
  endcase
end
end module
```

add default: statement here...
**Mealy FSM Verilog model example:** (Not necessary working solution)

```verilog
module reduce (input clk, in,
               output reg out);

parameter zero = 1'b0, one = 1'b1;
reg state, next_state;

always @(posedge clk) begin
    state = next_state;
end

always @(in, state) begin
    case (state)
        zero: begin
            out = 0;
            if (in) next_state = one;
            else next_state = zero;
        end
        one: begin
            out = 0;
            if (in) begin
                out = 1;
                next_state = one;
            end
            else begin
                out = 0;
                next_state = zero;
            end
    endcase
end
endmodule
```

Lecture #6

add default: statement here...