2014 Fall CSE140L

Digital Systems Laboratory

Lecture #4

by

Dr. Choon Kim

CSE Department, UCSD
crk034@eng.ucsd.edu
**Real-time clock signal**

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

Can we use Verilog delay construct, `<n>`, (e.g., assign #2 n1 = ab; )?  

==> No! It is ignored during synthesis process.

**Solution:** If a 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 (e.g., 1Hz, 0.1Hz, etc)
1Hz clock example:
counter chain: digital clock example

modulo 60 counter
modulo 60 counter
modulo 24 counter
modulo 365 counter

second minute hour Day...Year...
How to design a counter?

**Structural** level design

or

**Behavioral** level design
Structural level Counter Design
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)

<table>
<thead>
<tr>
<th>Time Steps</th>
<th>A B C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>1 0 0</td>
</tr>
<tr>
<td>3</td>
<td>1 1 0</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1</td>
</tr>
<tr>
<td>5</td>
<td>0 1 1</td>
</tr>
<tr>
<td>6</td>
<td>0 0 1</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>
Odd Length Walking – Ring Counter
A Shifter with Twisted Feedback Loops

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

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

\( n \) JK F-Fs => 2n-1 states

The counter works itself back to the proper sequence.
Pseudo Random Number Sequencer (Generator)

$n = 4$, length = 15

$D_0 = Q_3 \text{ XNOR } Q_2$

4-bit output ($Q_3 Q_2 Q_1 Q_0$) is a random number,
where, $Q_3 = \text{MSB, } Q_0 = \text{LSB}$
Behavioral level Counter Design

(****NOTE***  All the Verilog codes in any slide are an example only, not necessary a working solution.)
Up counter example code

***NOTE*** All the Verilog codes in any slide are an example only, not necessary a working solution.

```verilog
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
```
Down counter

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
Parity Checker

module EVEN_PARITY_CHECKER (input [8:0] INCOMING_SIG, output reg PARITY_BIT);

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

endmodule
module EVEN_PARITY_CHECKER (input [8:0] INCOMING_SIG,
output PARITY_BIT);  // it's wire, not reg!

reg temp;

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

assign PARITY_BIT = temp;

endmodule