Logic and Computer Design Fundamentals Verilog

Part 5 – Chapter 8 – Algorithmic State Machine Example: Binary Multiplier

Charles Kime & Thomas Kaminski

© 2004 Pearson Education, Inc. <u>Terms of Use</u>

(Hyperlinks are active in View Show mode)

Overview

Part 5

- Conversion of ASM into Verilog description
 - Decompose into:
 - Sequencing synchronous always for state, combinational always for next state
 - Output values combinational always
 - Register transfers synchronous always
 - Use parameters to make state assignment
 - Use case for more than two states
 - Use if then else for scalar decisions
 - Use case for vector decisions
- Illustrate using binary multiplier ASM chart in Figure 8-8 of text

Logic and Computer Dealgn Fundamentals PowerPoint® Slides © 2004 Pearson Education, Inc.

```
//Alternative Binary Multiplier with n = 4
//See Figure 8-8 of text
module alt bin multiplier (CLK, RESET, G, LOADB,
  LOADQ, MULT IN, MULT OUT);
input CLK, RESET, G, LOADB, LOADQ;
input [3:0] MULT IN;
output[7:0] MULT OUT;
reg state, next state;
parameter IDLE = 0, MUL = 1;
reg [1:0] P;
req [3:0] A, B, Q;
wire Z;
// Test P for value 0
assign Z = ~| P; // ~| is reduction NOR
// that NORs together all bits of P
```

Logic and Computer Design Fundamentals PowerPoint[®] Slides © 2004 Pearson Education, Inc.

```
assign MULT OUT = \{A,Q\};
//state register
always@(posedge CLK or posedge RESET)
begin
   if (RESET == 1)
       state <= IDLE;</pre>
       else
       state <= next state;</pre>
end
//next state function
always@(G or Z or state)
begin
   if (state == IDLE)
       if (G == 1)
           next state <= MUL;</pre>
   Logic and Competer Design Fundamentals
```

© 2004 Pearson Education, Inc.

```
else
           next state <= IDLE;</pre>
else // state = MUL
      if (Z == 1)
          next state <= IDLE;</pre>
      else
          next state <= MUL;</pre>
end
//register transfers
always@(posedge CLK or posedge RESET)
begin
if (LOADB)
       B \leq MULT IN;
else
if (LOADQ)
      Q \leq MULT IN;
else
   Logic and Computer Design Fundamentals
```

© 2004 Pearson Education, Inc.

```
if (state == IDLE && G == 1)
begin
   P <= 2'b11;
   A \le 4'b0000;
end
else
if (state == MUL)
   begin
        P \le P - 1;
         if (0[0] == 1)
             \{A,Q\} \le \{\{1'b0,A\} + \{1'b0,B\},Q[3:1]\};
//1'b0 is concatenated on the left to acquire the Cout value.
        else
              \{A,Q\} \leq \{1'b0, A,Q[3:1]\};
```

end

end

endmodule

Logic and Computer Dealon Fundamentals PowerPoint[®] Slides © 2004 Pearson Education, Inc.

Terms of Use

- © 2004 by Pearson Education, Inc. All rights reserved.
- The following terms of use apply in addition to the standard Pearson Education <u>Legal Notice</u>.
- Permission is given to incorporate these materials into classroom presentations and handouts only to instructors adopting Logic and Computer Design Fundamentals as the course text.
- Permission is granted to the instructors adopting the book to post these materials on a protected website or protected ftp site in original or modified form. All other website or ftp postings, including those offering the materials for a fee, are prohibited.
- You may not remove or in any way alter this Terms of Use notice or any trademark, copyright, or other proprietary notice, including the copyright watermark on each slide.
- Return to Title Page