Logic and Computer Design Fundamentals

Verilog

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

Charles Kime & Thomas Kaminski

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**

Verilog for Alternative Binary Multiplier – 1

```
//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;
reg [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
```

Verilog for Alternative Binary Multiplier – 2

```
assign MULT_OUT = {A,Q};
//state register
always@(posedge CLK or posedge RESET)
begin
   if (RESET == 1)
      state <= IDLE;
      else
      state <= next_state;
end
//next state function
always@(G or Z or state)
begin
   if (state == IDLE)
      if (G == 1)
         next_state <= MUL;
```

Verilog for Alternative Binary Multiplier – 3

```
    else
          next_state <= IDLE;
else // state = MUL
     if (Z == 1)
        next_state <= IDLE;
     else
        next_state <= MUL;
end
//register transfers
always@(posedge CLK or posedge RESET)
begin
if (LOADB)
       B <= MULT_IN;
else
if (LOADQ)
     Q <= MULT_IN;
else
```

Verilog for Alternative Binary Multiplier – 4

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

Terms of Use

- Return to Title Page