Logic and Computer Design Fundamentals

Verilog

Part 3 – Chapter 6 – Finite State Machines

Charles Kime & Thomas Kaminski

Overview

- **Part 3**
  - **Process (Procedural) Description**
  - **Verilog Keywords and Constructs**
  - **Process Verilog for a Positive Edge-triggered D Flip-flop**
  - **Process Verilog for Figure 6-19(a)**

Process (Procedural) Description

- So far, we have done dataflow and behavioral Verilog using continuous assignment statements (`assign`)
- Continuous assignments are limited in the complexity of what can be described
- A process can be viewed as a replacement for a continuous assignment statement that permits much more complex descriptions
- A process uses procedural assignment statements much like those in a typical programming language

Verilog Keywords & Constructs - 1

- **Because of the use of procedural rather than continuous assignment statements, assigned values must be retained over time.**
  - **Register type: reg**
  - **The reg in contrast to wire stores values between executions of the process**
  - **A reg type does not imply hardware storage!**
- **Process types**
  - **initial – executes only once beginning at t = 0.**
  - **always – executes at t = 0 and repeatedly thereafter.**
  - **Timing or event control is exercised over an always process using, for example, the @ followed by an event control statement in ( ).**

Verilog Keywords & Constructs - 2

- **Process begins with begin and ends with end.**
- **The body of the process consists of procedural assignments**
  - **Blocking assignments**
    - **Example: C = A + B;**
    - **Execute sequentially as in a programming language**
  - **Non-blocking assignments**
    - **Example: C <= A + B;**
    - **Evaluate right-hand sides, but do not make any assignment until all right-hand sides evaluated. Execute concurrently unless delays are specified.**

Verilog Keywords & Constructs - 3

- **Conditional constructs**
  - **The if-else**
    - **If (condition)**
    - **begin procedural statements end**
    - **{else if (condition)**
    - **begin procedural statements end}**
    - **else**
    - **begin procedural statements end**
  - **The case**
    - **case expression**
    - **{case expression : statements}**
    - **endcase;**

Examples

```
always
  begin
     B  = A;
     C  = B;
   end
```

- **Suppose initially A = 0, B = 1, and C = 2. After execution, B = 0 and C = 0.**

```
always
  begin
       B <= A;
       C <= B;
  end
```

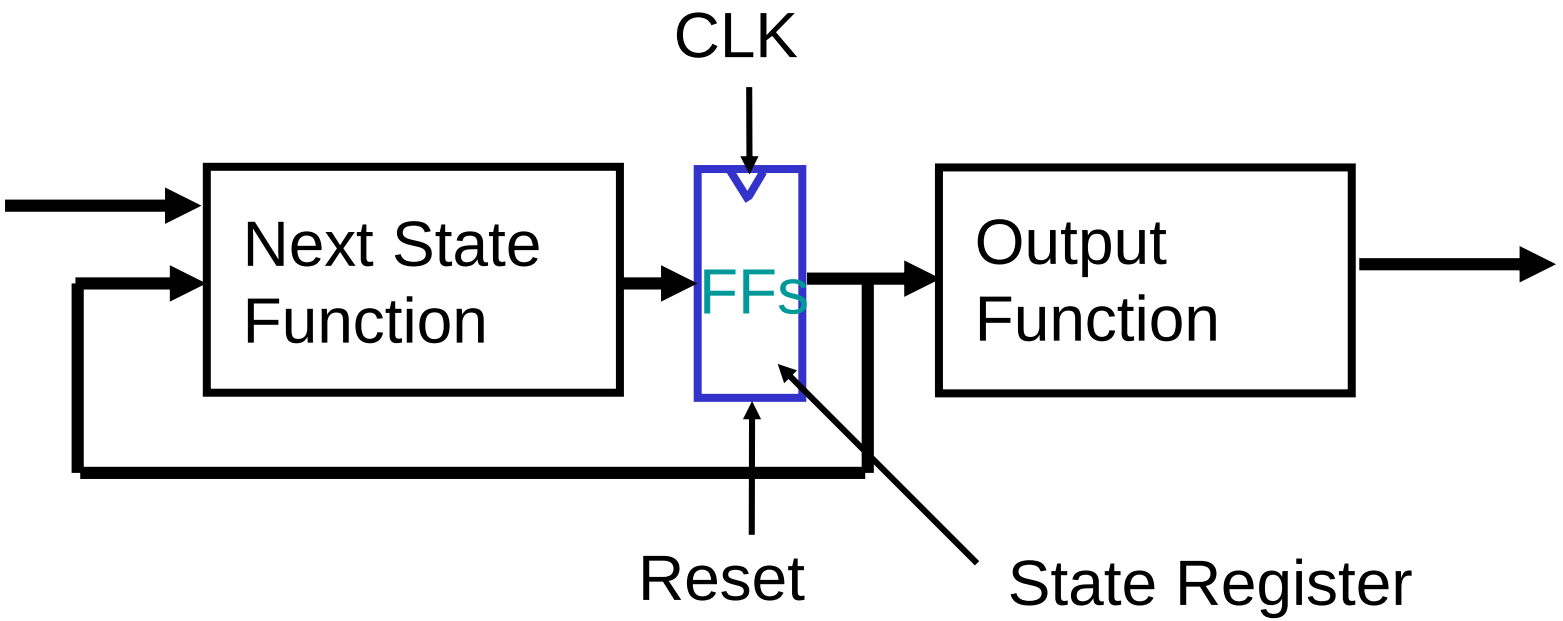
- **Suppose initially A = 0, B = 1, and C = 2. After execution, B = 0 and C = 1.**

Verilog for Positive Edge-Triggered D Flip-Flop

```
module dff (CLK, RESET, D, Q)
  input CLK, RESET, D;
  output Q;
  reg Q;
  always@ (posedge CLK or posedge RESET)
      begin
          if (RESET)
             Q <= 0;
          else
             Q <= D;
      end
endmodule
```

Describing Sequential Circuits

- **There are many different ways to organize models for sequential circuits. We will use a model that corresponds to the following diagram:**



- **A process corresponds to each of the 3 blocks in the diagram.**

Verilog for Figure 6-19(a) State Diagram

```
module fig_619 (CLK, RESET, X, Z);
    input CLK, RESET, X;
    output Z;
    reg[1:0] state, next_state;
    parameter S0 = 2'b00, S1 = 2'b01,
              S2 = 2'b10, S3 = 2'b11;
//state register
   always@(posedge CLK or posedge RESET)
   begin
      if (RESET == 1)
         state <= S0;
      else
         state <= next_state;
   end
```

Verilog State Diagram (continued)

```
//next state function
   always@(X or state)
   begin
      case (state)
         S0: if (X == 1) next_state <= S1;
             else next_state <= S0;
         S1: if (X == 1) next_state <= S3;
             else next_state <= S0;
         S2: if (X == 1) next_state <= S2;
             else next_state <= S0;
         S3: if (X == 1) next_state <= S2;
             else next_state <= S0;
         default: next_state <= 2'bxx;
      endcase
   end
```

Verilog State Diagram (continued)

```
reg Z;
//output function
    always@(X or state)
    begin
       case (state)
         S0: Z <= 1'b0;
         S1: if (X == 1) Z <= 1'b0;
             else Z <= 1'b1;
         S2: if (X == 1) Z <= 1'b0;
             else Z <= 1'b1;
         S3: if (X == 1) Z <= 1'b0;
             else Z <= 1'b1;
         default: Z <= 1'bx;
       endcase
    end
endmodule
```

Terms of Use

-
- The following terms of use apply in addition to the standard Pearson Education Legal Notice.
- 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