CPE/EE 422/522
Spring 2004
Chapter 5 Supplement

Dr. Rhonda Kay Gaede

UAH

Electrical and Computer Engineering

Electronic Dice Game

- Two counters are used to simulate the roll of the dice
- Rules of the game
  - After the first roll of the dice, the player wins if the sum is 7 or 11. The player loses if the sum is 2, 3, or 12. Otherwise, the sum the player obtained on the first roll is referred to as a point, and he or she must roll the dice again.
  - On the second or subsequent roll of the dice, the player wins if the sum equals the point, and he or she loses if the sum is 7. Otherwise, the player must roll again until he or she finally wins or loses.

Electronic Dice Game Block Diagram

Electronic Dice Game Flowchart

Electronic Dice Game
Inputs and Outputs

- Inputs
  - D7 -
  - D711 -
  - D2312 -
  - Eq -
  - Rb -
  - Reset -
- Outputs
  - Roll -
  - Sp -
  - Win -
  - Lose -

Electronic Dice Game SM

Electronic Dice Game
Behavioral VHDL Model

```
entity DiceGame is
   port (Rb, Reset, CLK: in bit;
         Sum: in integer range 2 to 12;
         Roll, Win, Lose: out bit);
end DiceGame;

architecture DiceBehave of DiceGame is
   signal State, Nextstate: integer range 0 to 5;
   signal Point: integer range 2 to 12;
   signal Sp: bit;
begin
   process(Rb, Reset, Sum, State)
   begin
      Sp <= '0'; Roll <= '0'; Win <= '0'; Lose <= '0';
      case State is
        when 0 => if Rb = '1' then Nextstate <= 1;  end if;
        when 1 =>
          if Rb = '1' then Roll <= '1';
            elsif Sum = 7 or Sum = 11 then Nextstate <= 2;
            elsif Sum = 2 or Sum = 3 or Sum =12 then Nextstate <= 3;
            else Sp <= '1'; Nextstate <= 4;
          end if;
        when 2 => Win <= '1';
          if Reset = '1' then  Nextstate <= 0; end if;
```


Electronic Dice Game
Behavioral VHDL Model

```
when 3 => Lose <= '1';
          if Reset = '1' then Nextstate <= 0; end if;
        when 4 => if Rb = '1' then Nextstate <= 5; end if;
        when 5 =>
          if Rb = '1' then Roll <= '1';
            elsif Sum = Point then Nextstate <= 2;
            elsif Sum = 7 then Nextstate <= 3;
            else Nextstate <= 4;
          end if;
      end case;
    end process;
    process(CLK)
    begin
      if rising_edge(CLK) then
         State <= Nextstate;
         if Sp = '1' then Point <= Sum; end if;
      end if;
    end process;
end DiceBehave;
```

Dice Game Test - SM Chart

Dice Game Test Module VHDL

```
entity GameTest is
  port(Rb, Reset:  out bit;
       Sum: out integer range 2 to 12;
       CLK: inout bit;
       Roll, Win, Lose: in bit);
end GameTest;

architecture dicetest of GameTest is
  signal Tstate, Tnext: integer range 0 to 3; signal Trig1: bit;
  type arr is array(0 to 11) of integer;
  constant Sumarray:arr := (7,11,2,4,7,5,6,7,6,8,9,6);
  begin
    CLK <= not CLK after 20 ns;
  process(Roll, Win, Lose, Tstate)
    variable i: natural;          -- i is initialized to 0
    begin
    case Tstate is
        when 0 => Rb <= '1'; Reset <='0';
             if i>=12 then Tnext <= 3;
             elsif Roll = '1' then
               Sum <= Sumarray(i);
               i:=i+1;
               Tnext <= 1;
             end if;
```

Dice Game Test Module VHDL

```
       when 1 => Rb <= '0'; Tnext <= 2;
       when 2 => Tnext <= 0;
            Trig1 <= not Trig1;   -- toggle Trig1
             if (Win or Lose) = '1' then
                Reset <= '1';
             end if;
       when 3 => null;            -- Stop state
   end case;
  end process;
  process(CLK)
    begin
    if CLK = '1' then
       Tstate <= Tnext;
    end if;
  end  process;

end dicetest;
```


Two Component Test Bench Style

```
entity tester is
end tester;

architecture test of tester is
component GameTest
  port(Rb, Reset:  out bit;
       Sum: out integer range 2 to 12;
       CLK: inout bit;
       Roll, Win, Lose: in bit);
end component;
component DiceGame
   port (Rb, Reset, CLK: in bit;
         Sum: in integer range 2 to 12 ;
         Roll, Win, Lose: out bit);
end component;
signal rb1, reset1, clk1, roll1, win1, lose1: bit;
signal sum1: integer range 2 to 12;
begin
  Dice: Dicegame port map(rb1,reset1,clk1,sum1,roll1,win1,lose1);
  Dicetest: GameTest port map(rb1,reset1,sum1,clk1,roll1,win1,lose1);
end test;
```