EE 459/500 – HDL Based Digital Design with Programmable Logic

Lecture 14
Electronic Dice Game:
From ASM Chart to Microprogrammed Control

References:

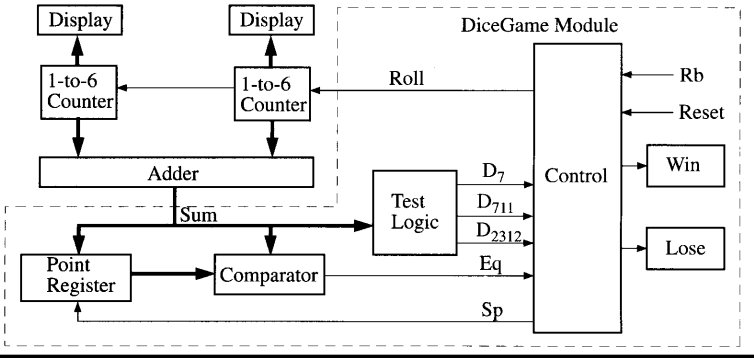
Chapter s 5 from textbook

Overview

- Dice Game Description
  - ASM chart
  - Controller Implementation 1: Behavioral
  - Controller Implementation 2: Equations
- Microprogrammed Control
  - Two address microcode

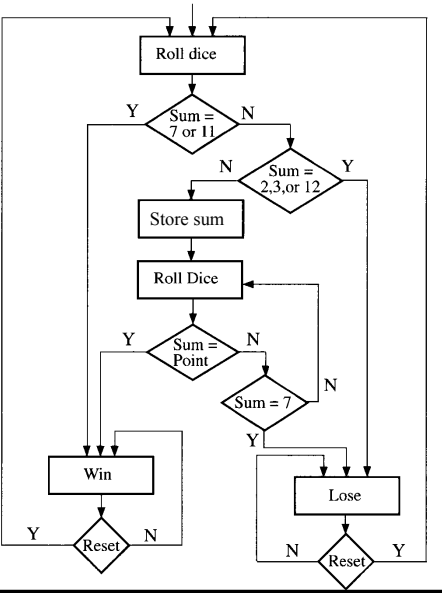
Electronic Dice Game: there are two dice to roll

- Rules of the game:
  - After the first roll of the dice, the player (P) wins if the sum is 7 or 11. P loses if the sum is 2, 3, or 12. Otherwise, the sum P obtained on the first roll is referred to as a point, and P must roll again.
  - On the second or subsequent roll of the dice, P wins if the sum equals the point, and loses if the sum is 7. Otherwise, P must roll again until finally wins or loses.



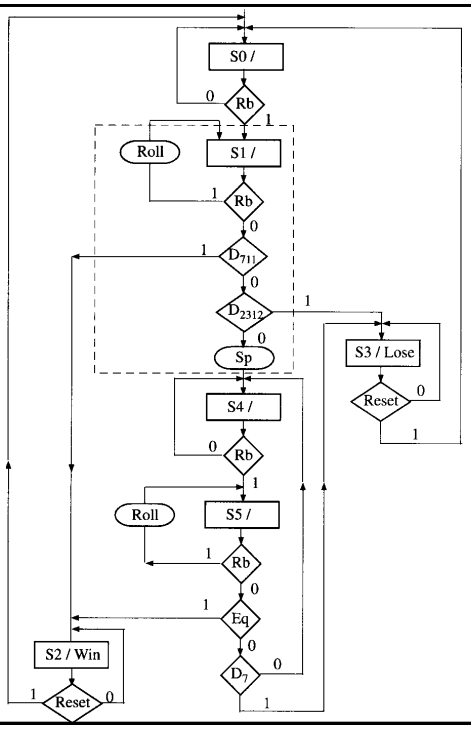
Electronic Dice Game: Flow Chart

- **Reset**: to initiate a new game
- **Rb** (Roll button):
  - If Rb is pushed, dice counters count at a high speed
  - When released, the values in the two counters are displayed, and the game proceeds



ASM Chart

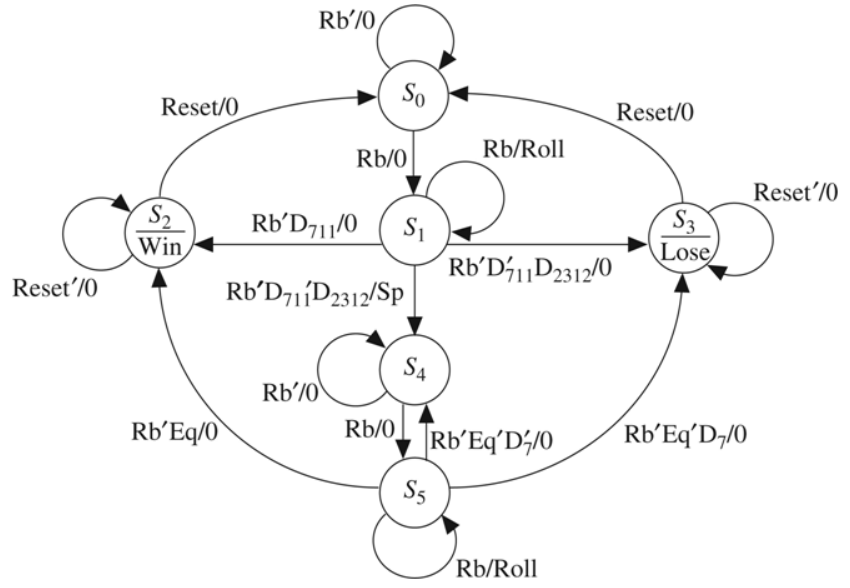
- Inputs to Control Unit:
  - Reset
  - Rb
  - D7 (‘1’ if sum of dice is 7)
  - D711 (‘1’ if sum is 7 or 11)
  - D2312
  - Eq (sum = Point)
- Outputs of Control Unit:
  - Roll
  - Sp (Sum to be stored)
  - Win
  - Lose



Overview

- Dice Game Description
  - ASM chart
  - Controller Implementation 1: Behavioral
  - Controller Implementation 2: Equations
- Microprogrammed Control
  - Two address microcode

State Graph of Control Unit (Mealy or Moore?)



Control Unit: Behavioral VHDL Code (1/2)

```
library BITLIB;
use BITLIB.bit_pack.all;

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;
```

Control Unit: Behavioral VHDL Code (2/2)

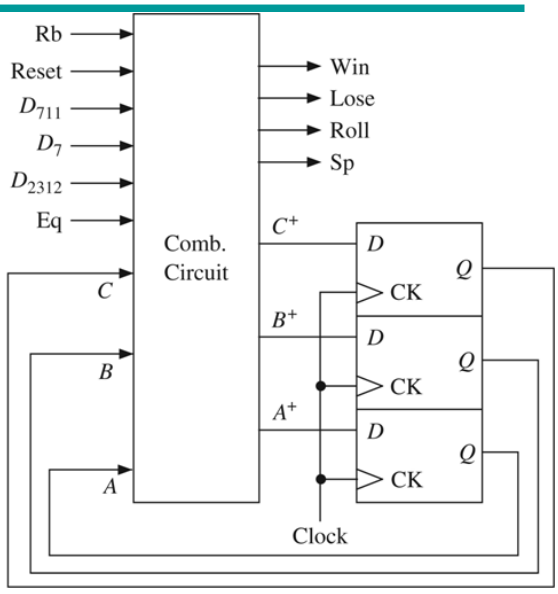
```
      when 2 => Win <= '1';
       if Reset = '1' then Nextstate <= 0; end if;

      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;
```

Overview

- Dice Game Description
  - ASM chart
  - Controller Implementation 1: Behavioral
  - Controller Implementation 2: Equations
- Microprogrammed Control
  - Two address microcode

Control Unit: Just a Sequential Circuit

- Typical block diagram of sequential circuit
- Need three Flip-Flops for State register
- Construct State Transition Table and then use K-maps to derive equations for: A+, B+, C+, Win, Lose, Roll, Sp



State Transition Table

- Derived from the ASM chart
- A row for each link path in the ASM chart

	ABC	Rb	Reset	D_7	D_{711}	D_{2312}	Eq	A^+	B^+	C^+	Win	Lose	Roll	Sp
1	000	0	—	—	—	—	—	0	0	0	0	0	0	0
2	000	1	—	—	—	—	—	0	0	1	0	0	0	0
3	001	1	—	—	—	—	—	0	0	1	0	0	1	0
4	001	0	—	—	0	0	—	1	0	0	0	0	0	1
5	001	0	—	—	0	1	—	0	1	1	0	0	0	0
6	001	0	—	—	1	—	—	0	1	0	0	0	0	0
7	010	—	0	—	—	—	—	0	1	0	1	0	0	0
8	010	—	1	—	—	—	—	0	0	0	1	0	0	0
9	011	—	1	—	—	—	—	0	0	0	0	1	0	0
10	011	—	0	—	—	—	—	0	1	1	0	1	0	0
11	100	0	—	—	—	—	—	1	0	0	0	0	0	0
12	100	1	—	—	—	—	—	1	0	1	0	0	0	0
13	101	0	—	0	—	—	0	1	0	0	0	0	0	0
14	101	0	—	1	—	—	0	0	1	1	0	0	0	0
15	101	0	—	—	—	—	1	0	1	0	0	0	0	0
16	101	1	—	—	—	—	—	1	0	1	0	0	1	0
17	110	—	—	—	—	—	—	—	—	—	—	—	—	—
18	111	—	—	—	—	—	—	—	—	—	—	—	—	—

Control Unit: VHDL Code

```
architecture Dice_Eq of DiceGame is
signal Sp,Eq,D7,D711,D2312: bit:= '0';
signal DA,DB,DC,A,B,C: bit:='0';
signal Point: integer range 2 to 12;
begin
  process(CLK)
  begin
    if CLK = '1' and CLK'event then
      A <= DA; B <= DB; C <= DC;
      if Sp = '1' then Point <= Sum; end if;
    end if;
  end process;
  Win <= B and not C;
  Lose <= B and C;
  Roll <= not B and C and Rb;
  Sp <= not A and not B and C and not Rb and not D711 and not D2312;
  D7 <= '1' when Sum = 7 else '0';
  D711 <= '1' when (Sum = 11) or (Sum = 7) else '0';
  D2312 <= '1' when (Sum = 2) or (Sum = 3) or (Sum = 12) else '0';
  Eq <= '1' when Point = Sum else '0';
  DA <= (not A and not B and C and not Rb and not D711 and not D2312) or
        (A and not C) or (A and Rb) or (A and not D7 and not Eq);
  DB <= ((not A and not B and C and not Rb) and (D711 or D2312)) or
        (B and not Reset) or ((A and C and not Rb) and (Eq or D7));
  DC <= (not B and Rb) or (not A and not B and C and not D711 and D2312) or
        (B and C and not Reset) or (A and C and D7 and not Eq);
end Dice_Eq;
```

Overview

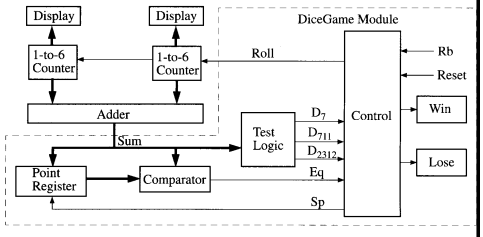
- Dice Game Description
  - ASM chart
  - Controller Implementation 1: Behavioral
  - Controller Implementation 2: Equations
  - Complete game
- Microprogrammed Control
  - Two address microcode

Counters + Adder of Datapath: VHDL Code

```
entity Counter is
  port(Clk, Roll: in bit;
       Sum: out integer range 2 to 12);
end Counter;

architecture Count of Counter is
signal Cnt1, Cnt2: integer range 1 to 6:= 1;
begin
  process(Clk)
  begin
    if Clk = '1' then
      if Roll = '1' then
        if Cnt1 = 6 then Cnt1 <= 1; else Cnt1 <= Cnt1 + 1; end if;
        if Cnt1 = 6 then
          if Cnt2 = 6 then Cnt2 <= 1; else Cnt2 <= Cnt2 + 1; end if;
        end if;
      end if;
    end if;
  end process;
  Sum <= Cnt1 + Cnt2;
end Count;
```

Complete Dice Game: VHDL Code

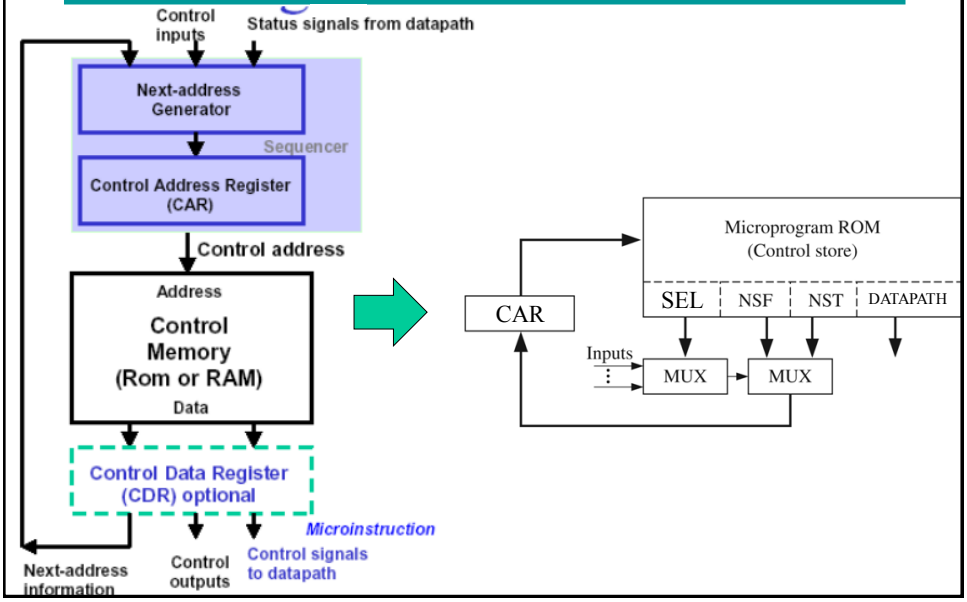
```
entity Game is
  port(Rb, Reset, Clk: in bit;
       Win, Lose: out bit);
end Game;

architecture Play1 of Game is
component Counter
  port(Clk, Roll: in bit;
       Sum: out integer range 2 to 12);
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 roll1: bit;
signal sum1: integer range 2 to 12;
begin
  Dice: Dicegame port map (Rb, Reset, Clk, sum1, roll1, Win, Lose);
  Count: Counter port map (Clk, roll1, sum1);
end Play1;
```



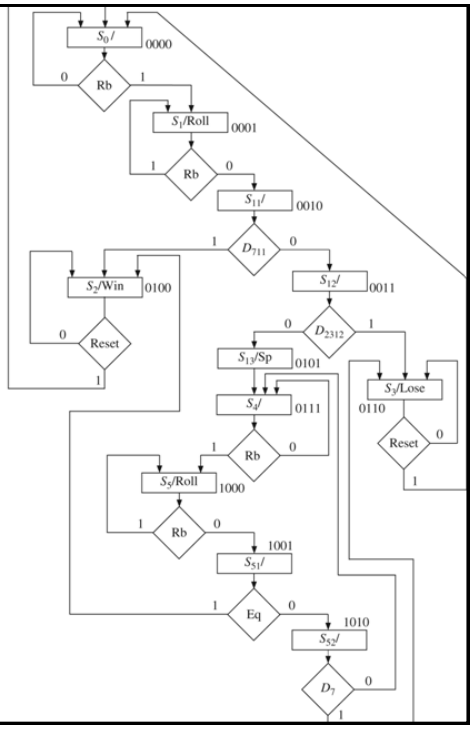
Overview

- Dice Game Description
  - ASM chart
  - Controller Implementation 1: Behavioral
  - Controller Implementation 2: Equations
  - Complete game
- Microprogrammed Control
  - Two address microcode

Hardware arrangement for microprogramming



ASM chart with Moore outputs and one qualifier per state

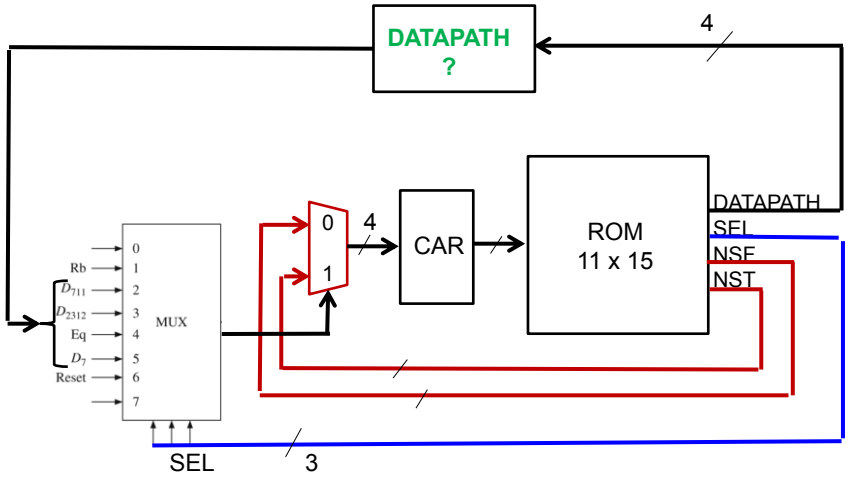


- ASM chart modifications:
  - All output converted to Moore outputs
  - Only one input variable must be tested in each state

Two-address microprogram for Dice Game

State	ABCD	TEST	NSF	NST	Roll	Sp	Win	Lose
S_0	0000	001	0000	0001	0	0	0	0
S_1	0001	001	0010	0001	1	0	0	0
S_{11}	0010	010	0011	0100	0	0	0	0
S_{12}	0011	011	0101	0110	0	0	0	0
S_2	0100	110	0100	0000	0	0	1	0
S_{13}	0101	xxx	0111	0111	0	1	0	0
S_3	0110	110	0110	0000	0	0	0	1
S_4	0111	001	0111	1000	0	0	0	0
S_5	1000	001	1001	1000	1	0	0	0
S_{51}	1001	100	1010	0100	0	0	0	0
S_{52}	1010	101	0111	0110	0	0	0	0

Final hardware arrangement



Summary

- Electronic Dice Game
- Multiple ways to design the control unit
- Microprogrammed control is a more structured approach for complex systems