Figure 5-1 Components of an SM Chart

(a) state box

(b) decision box

(c) conditional output box

- xxx state code
- (true branch) 1
- (false branch) 0
- conditional output list
Figure 5-2 Example of an SM Block

one entrance path

one state

link path a

link path b

n exit paths

SM block
Figure 5-3 Equivalent SM Blocks

(a) S1 / Z1
   0
   X1 1
   Z2
   0
   X2 1
   S2 /

(b) S1 / Z1
   0
   X2 1
   S2 /
   1
   X1 0
   Z2
   0
   X1 1
   S3 /
Figure 5-4: Equivalent SM Chart for a Combinational Network
Figure 5-5 SM Block with Feedback

(a) incorrect

(b) correct
Figure 5-6 Equivalent SM Blocks

(a) Parallel form

(b) Serial form
Figure 5-7 Conversion of State Graph to an SM Chart
Figure 5-8 Timing Chart for Figure 5-7

<table>
<thead>
<tr>
<th>Clock</th>
<th>S0</th>
<th>S1</th>
<th>S2</th>
<th>S2</th>
<th>S0</th>
<th>S0</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z_a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z_b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z_c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 5-9 SM Chart for Binary Multiplier
entity Mult is
    port(CLK, St, K, M: in bit;
        Load, Sh, Ad, Done: out bit);
end mult;

architecture SMbehave of Mult is
    signal State, Nextstate: integer range 0 to 3;
begin
    process(St, K, M, State) -- start if state or inputs change
    begin
        Load <= '0'; Sh <= '0'; Ad <= '0';
        case State is
            when 0 => if St = '1' then -- St (state 0)
                Load <= '1'; Nextstate <= 1;
            else Nextstate <= 0; -- St'
            end if;
            when 1 => if M = '1' then -- M (state 1)
                Ad <= '1'; Nextstate <= 2;
            else Sh <= '1'; if K = '1' then Nextstate <= 3; -- K
                else Nextstate <= 1; -- K'
                end if;
            end if;
            when 2 => Sh <= '1'; -- (state 2)
                if K = '1' then Nextstate <= 3; -- K
                else Nextstate <= 1; -- K'
                end if;
            end when;
        end case;
    end process;
end SMbehave;
Figure 5-10(b) VHDL for SM Chart of Figure 5-9

```vhdl
when 3 => Done <= '1'; -- (state 3)
    Nextstate <= 0;
end case;
end process;
process(CLK)
begin
    if CLK = '1' then
        State <= Nextstate; -- update state on rising edge
    end if;
end process;
end SMbehave;
```
Figure 5-11 Block Diagram for Dice Game

DiceGame Module

- Roll
- Rb
- Reset
- Win
- Lose

Point Register

Comparatorder

Adder

1-to-6 Counter

1-to-6 Counter

Display

Display

Sum

Test Logic

D_7

D_7_{11}

D_{2312}

Eq

Sp
Figure 5-12 Flowchart for Dice Game

1. Roll dice
2. If Sum = 7 or 11, go to Win
   - If Y, Win
   - If N, Roll Dice
3. If Sum = 2, 3, or 12, go to Lose
   - If Y, Lose
   - If N, Roll Dice
4. Store sum in point register
5. Roll Dice
6. If Sum = Point, go to Win
   - If Y, Win
   - If N, Roll Dice
7. If Sum = 7, go to Lose
   - If Y, Lose
   - If N, Roll Dice
8. If Roll again, go to Roll Dice
9. If Reset is Y, go to Roll dice
10. If Reset is N, go to Roll dice
Figure 5-13 SM Chart for Dice Game
Figure 5-14 State Graph for Dice Game Controller
Figure 5-15(a) Behavioral Model for Dice Game

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

library BITLIB;
use BITLIB.bit_pack.all;

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;
        when
Figure 5-15(b) Behavioral Model for Dice Game

```
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;
```
Figure 5-16 Dice Game with Test Bench
Figure 5-17 SM Chart for Dice Game Test

T0 / Rb

i = i + 1

T1 /

T2 /

T3 / (Stop)

Win or Lose

Reset

Roll

Sum = Sumarray(i)
i = i + 1

i ≥ N
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;
library BITLIB;
use BITLIB.bit_pack.all;
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'; -- wait for Roll
      Reset <= '0';
    if i>=12 then Tnext <= 3;
    elsif Roll = '1' then
      Sum <= Sumarray(i);
      i:=i+1;
      Tnext <= 1;
    end if;
    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;
Figure 5-19  Tester for Dice Game

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;
**Figure 5-20 Simulation and Command File for Dice Game Tester**

```plaintext
cmd /dicetest/ trig1 -NOTrigger sum1 win1 lose1 /dice/point
run 2000

<table>
<thead>
<tr>
<th>ns</th>
<th>delta</th>
<th>trig1</th>
<th>sum1</th>
<th>win1</th>
<th>lose1</th>
<th>point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>+3</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>260</td>
<td>+3</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>420</td>
<td>+3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>580</td>
<td>+2</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>740</td>
<td>+3</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>900</td>
<td>+2</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1060</td>
<td>+2</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1220</td>
<td>+3</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1380</td>
<td>+2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1540</td>
<td>+2</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1700</td>
<td>+2</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1860</td>
<td>+3</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>
```
Table 5-1  PLA Table for Multiplier Control

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>St</th>
<th>M</th>
<th>K</th>
<th>A+</th>
<th>B+</th>
<th>Load</th>
<th>Sh</th>
<th>Ad</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ A^+ = A'B'M'K + A'BM + AB'K = A'B(M + K) + AB'K \]

\[ B^+ = A'B'St + A'BM'(K'+K) + AB'(K'+K) = A'B'St + A'BM' + AB' \]

\[ Sh = A'BM'(K'+K) + AB'(K'+K) = A'BM' + AB' \]

\[ Load = A'B'St \quad Ad = A'B M \quad Done = A \; B \]
Figure 5-21 PLA Realization of Dice Game Controller
Table 5-2  PLA Table for Dice Game

<table>
<thead>
<tr>
<th>ABC</th>
<th>Rb</th>
<th>Reset</th>
<th>D7</th>
<th>D711</th>
<th>D2312</th>
<th>Eq</th>
<th>A+</th>
<th>B+</th>
<th>C+</th>
<th>Win</th>
<th>Lose</th>
<th>Roll</th>
<th>Sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>111</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 5-22 Maps Derived from Table 5-2

\[ E_1 = D'_711 D'_2312 \]
\[ E_2 = D'_7 Eq' \]

\[ R = \text{Reset} \]
\[ E_3 = D_711 + D'_711 D'_2312 = D_711 + D'_2312 \]
\[ E_4 = Eq + D_7 Eq' = Eq + D_7 \]
library BITLIB;
use BITLIB.bit_pack.all;

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 rising_edge(Clk) 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;
Figure 5-24 Counter for Dice Game

**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;
Figure 5-25 Complete Dice Game

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: bit 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;
Figure 5-26  Control Network Using an input Mux to Select the Next State
Figure 5-27(a) SM Chart with Moore Outputs and One Test per State
Figure 5-27(b) Chart with Moore Outputs and One Test per State
Figure 5-28  MUX for SM Chart of Figure 5-27
Table 5-3  PLA/ ROM Table for Figure 5-27

<table>
<thead>
<tr>
<th>State</th>
<th>ABCD</th>
<th>TEST</th>
<th>NSF</th>
<th>NST</th>
<th>ROLL</th>
<th>Sp</th>
<th>Win</th>
<th>Lose</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0000</td>
<td>001</td>
<td>0000</td>
<td>0001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>0001</td>
<td>001</td>
<td>0010</td>
<td>0001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S11</td>
<td>0010</td>
<td>010</td>
<td>0011</td>
<td>0100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S12</td>
<td>0011</td>
<td>011</td>
<td>0101</td>
<td>0110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>0100</td>
<td>110</td>
<td>0100</td>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S13</td>
<td>0101</td>
<td>xxx</td>
<td>0111</td>
<td>0111</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>0110</td>
<td>110</td>
<td>0110</td>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S4</td>
<td>0111</td>
<td>001</td>
<td>0111</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>1000</td>
<td>001</td>
<td>1001</td>
<td>1000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S51</td>
<td>1001</td>
<td>100</td>
<td>1010</td>
<td>0100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S52</td>
<td>1010</td>
<td>101</td>
<td>0111</td>
<td>0110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 5-29 Control Network Using a Counter for the State Register
Figure 5-30(a) SM chart with Serial State Assignment and Added X-states

Diagram includes state transitions labeled with states such as S0/0000, S1/Roll 0001, S11/, S2/Win 1111, Reset', S12/0011, and D711 0010. Connections are made between these states with arrows indicating transitions based on input conditions.
Figure 5-30(b) SM Chart with Serial State Assignment and Added X-state
Figure 5-31 MUX for SM chart of Figure 5-30
## Table 5-4  PLA Table for Figure 5-31

<table>
<thead>
<tr>
<th>State</th>
<th>ABCD</th>
<th>TEST</th>
<th>NST</th>
<th>ROLL</th>
<th>Sp</th>
<th>Win</th>
<th>Lose</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0000</td>
<td>000</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>0001</td>
<td>001</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S11</td>
<td>0010</td>
<td>010</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S12</td>
<td>0011</td>
<td>011</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S13</td>
<td>0100</td>
<td>111</td>
<td>010</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>0101</td>
<td>000</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>0110</td>
<td>001</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S51</td>
<td>0111</td>
<td>100</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S52</td>
<td>1000</td>
<td>101</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>1001</td>
<td>110</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sx</td>
<td>1010</td>
<td>111</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>1111</td>
<td>110</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Test(2) = B C'D' + B C D + A
Test(1) = B'C + B C'D' + A D
Test(0) = A'B'D + B D' + A D'
NST(3) = A'B'C + C D + A D
NST(2) = A'C D' + B + A C'D'
NST(1) = A'C D' + B C
NST(0) = D + A'B'C + B C' + A C'
Roll = A'B'C'D + B C D'
SP = B C'D'
Win = A B
Lose = A B'D
Figure 5-32  SM Charts for Serially Linked State Machine

Machine A (calling machine)

- SOME STATES
- SA/ZA
- ZB
- OTHER STATES

Machine B (called machine)

- IDLE
- ZA
- OTHER STATES
- SB/ZB
Figure 5-33a  Linked SM Charts for Dice Game

(a) Main control
Figure 5-33b   Linked SM Charts for Dice Game

(b) Roll Control