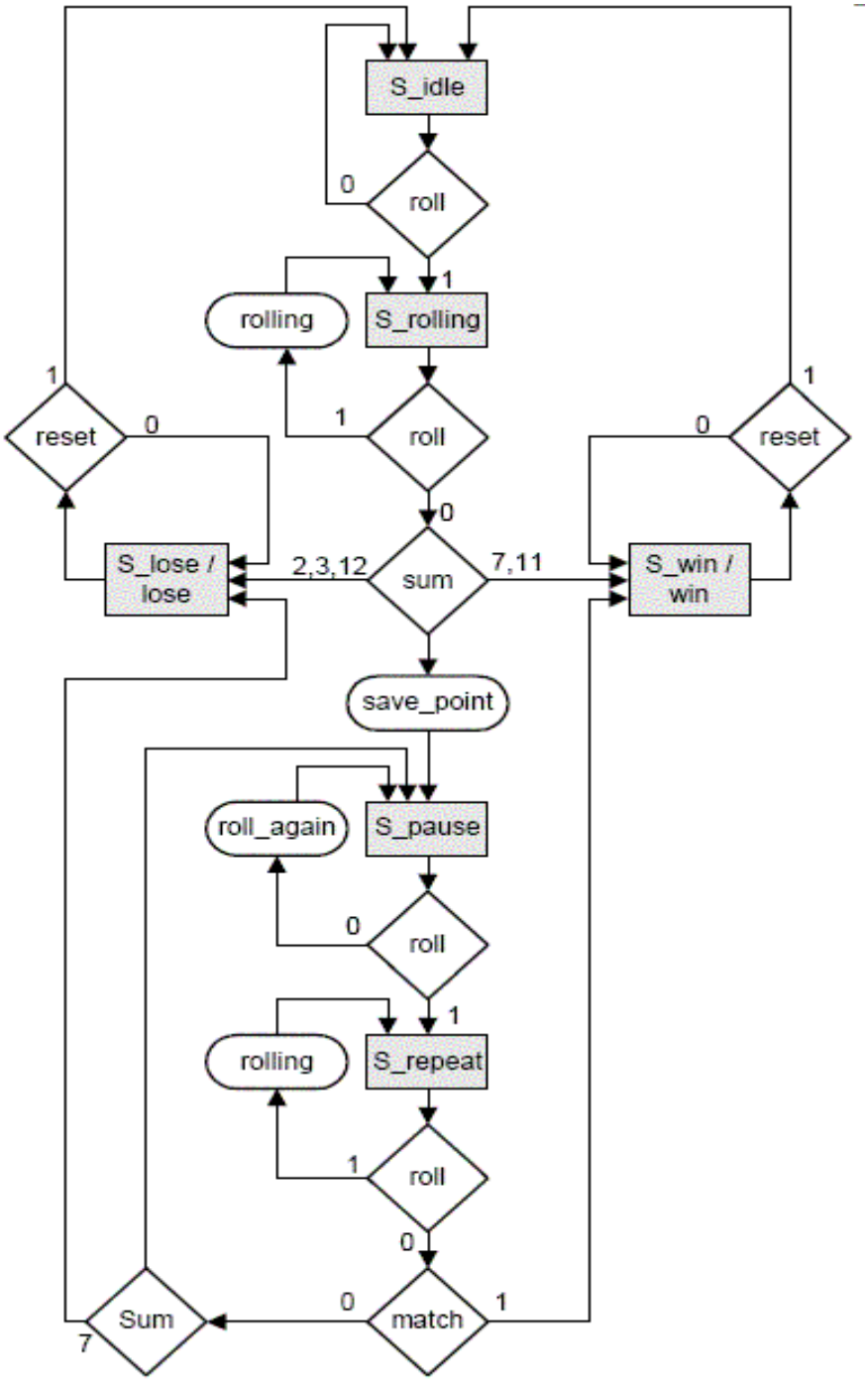
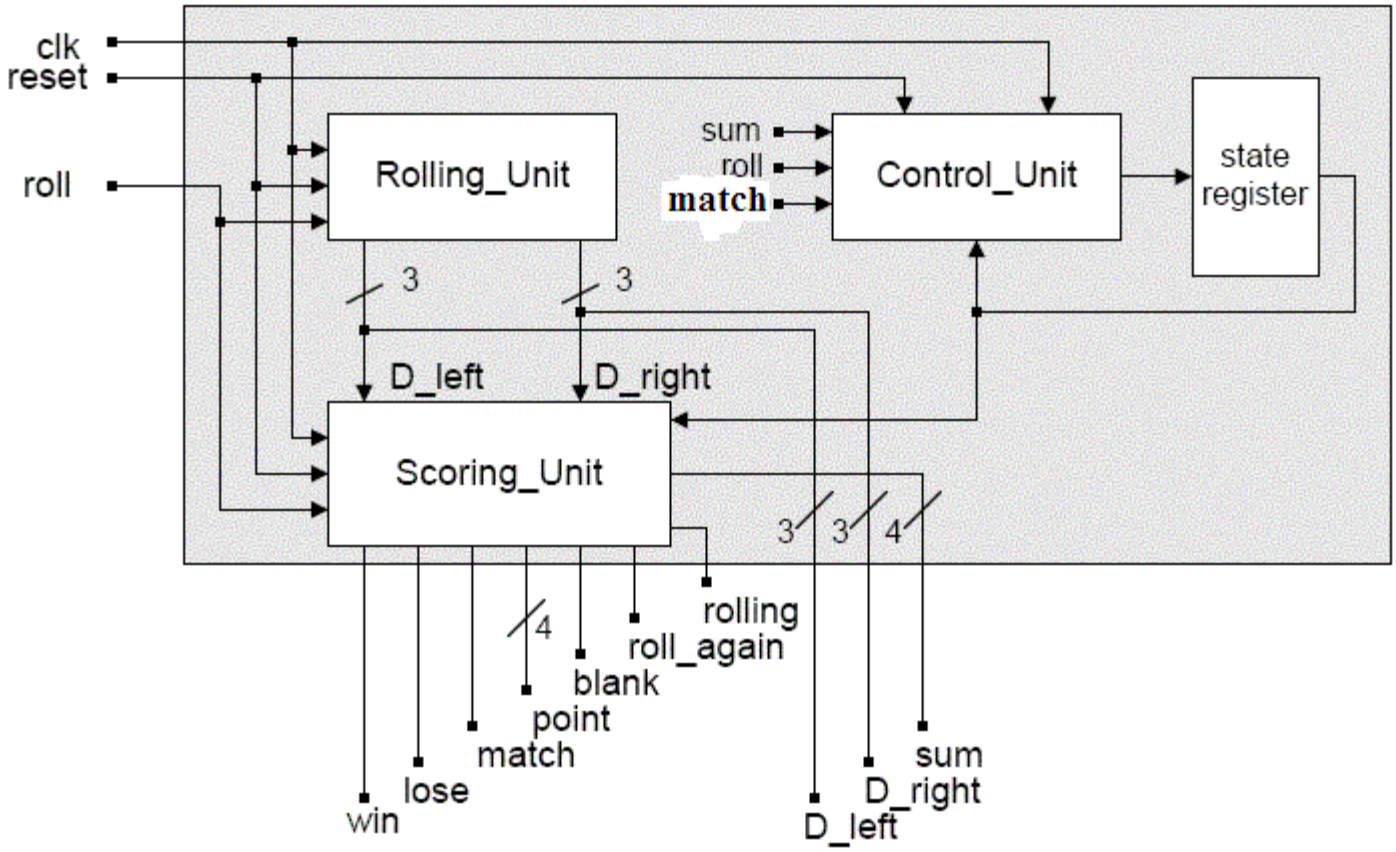
Dice Game Example

An electronic version of a dice game played by one player is proposed. Counters are used to emulate the rolling of dice. The sum of the counter values is the input to the game.
Here are the rules:
1. 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.
Othewise, the sum the player obtained on the first roll is referred to as a point, and he or she must roll the dice again.
2. 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.

Dice Game Design -- ASM Chart



Dice Game Design -- Block diagram



Dice Game Design -- VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity dicegame is
port(
   clk, reset, roll: in std_logic;
   win, lose, match_out: out std_logic;
   roll_again, rolling, blank: out std_logic;
   point_out: out std_logic_vector(3 downto 0);
   D_left_out, D_right_out :out std_logic_vector(2 downto 0);
   sum_out: out std_logic_vector(3 downto 0)
);
end dicegame;
```

```
architecture dicegame_arch of dicegame is
type statetype is (S_idle, S_rolling, S_pause,S_repeat,S_lose,S_win);
signal state, next_state: statetype;
signal match, save_point: std_logic;
signal D_left, D_right: std_logic_vector(2 downto 0);
signal sum, point: std_logic_vector(3 downto 0);

begin

match_out <= match;
D_left_out <= D_left;
D_right_out <= D_right;
sum_out <= sum;
point_out <= point;



---Rolling_unit:
sum <= ('0'& D_left) + ('0'& D_right);

Rollingproc: process(clk,reset)
      begin
      if (reset='1') then
            D_left <= (0=>'1', others=>'0');
            D_right <= (0=>'1', others=>'0');
      elsif(rising_edge(clk)) then
      if (D_left < 6) then
            D_left <= D_left + 1;
      else
            D_left <= (0=>'1', others=>'0');
      end if;
      if ((D_left=6)and(D_right<6)) then
            D_right <= D_right +1;
      elsif ((D_left=6)and(D_right=6))then
            D_right <= (0=>'1', others=>'0');
      end if;
      end if;
end process;
```

```
-- scoring_unit
match <= '1' when (sum = point) else '0';
roll_again <= '1' when ((state=S_pause)and(roll='0'))
                else '0';
rolling <= '1' when((state=S_rolling)and (roll='1'))
                else '1' when ((state=S_repeat) and (roll='1'))
                else '0';
save_point <= '1' when ((state=S_rolling)
                    and (roll='0')
                    and (sum /= 2)
                    and (sum /= 3)
                    and (sum /= 12)
                    and (sum /= 7)
                    and (sum /= 11))
                  else '0';
win <= '1' when (state = S_win) else '0';
lose <= '1' when (state = S_lose) else '0';
blank <= '1' when (point < 2) else '0';

-- control_unit
controlproc: process(save_point,reset)
            begin
             if (reset='1') then
                    point <= (others=>'0');
             elsif(rising_edge(save_point)) then
                    point <= sum;
             end if;
            end process;

            process(clk,reset)
            begin
                    if (reset='1') then
                          state <= S_idle;
                    elsif(rising_edge(clk)) then
                          state <= next_state;
                    end if;
            end process;
```

```
process(state,sum, roll, match)
begin
        case (state) is
        when S_idle=> if (roll='1')then
                            next_state <= S_rolling;
                      else
                        next_state <= S_idle;
                      end if;

        when S_rolling=>
                       if (roll='1') then
                          next_state <= S_rolling;
                      elsif ((sum = 2) or(sum=3)or(sum=12)) then
                          next_state <= S_lose;
                      elsif ((sum = 7) or(sum=11)) then
                        next_state <= S_win;
                      else
                        next_state <= S_pause;
                      end if;

        when S_pause=>
                      if (roll='1') then
                        next_state <= S_repeat;
                      else
                        next_state <= S_pause;
                      end if;

        when S_repeat=>
                     if (roll='1') then
                        next_state <= S_repeat;
                      elsif (match='1') then
                        next_state <= S_win;
                      elsif (sum = 7) then
                        next_state <= S_lose;
                      else
                        next_state <= S_pause;
                      end if;
```

```
                when S_win=> if (reset='1') then
                               next_state <= S_idle;
                             else
                               next_state <= S_win;
                             end if;

                when S_lose=> if (reset='1') then
                               next_state <= S_idle;
                              else
                               next_state <= S_lose;
                              end if;
                when others=> next_state <= S_idle;

                end case;
                end process;

end dicegame_arch;
```