LAB REPORT

*****On A Dice Game*****

3220101111 洪晨辉

****2024.11.20****

1. Problem Description

The objective of this exercise is to design an electronic dice game similar to the traditional game of craps. The game involves two dice, each capable of producing values between 1 and 6, resulting in sums between 2 and 12. Two counters are used to simulate the rolling of the dice. The game progresses based on the sum of the dice rolls according to specific rules, and the design requires creating multiple hardware components and integrating them into a functional circuit.

Game Rules:

On the first roll of the dice:

The player wins if the sum of the dice is 7 or 11.

The player loses if the sum is 2, 3, or 12.

For any other sum, the result becomes the point, and the player must roll again.

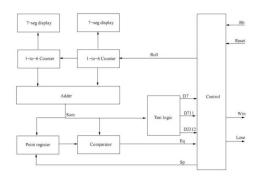
On subsequent rolls:

The player wins if the sum equals the previously stored point.

The player loses if the sum equals 7.

If neither condition is met, the player continues rolling until a win or loss occurs.

2. Design Formulation



The system uses the following key components to implement the dice game:

1-to-6 Counters:

Two counters simulate rolling two dice. The counters increment at high speed when the roll button (Rb) is pressed, making the values unreadable.

When the roll button is released, the counters hold the values, which represent the dice roll results.

Adder:

An adder calculates the sum of the outputs from the two counters, producing a value between 2 and 12.

Point Register:

The point register stores the sum of the dice after the first roll (if it is not a win or loss).

Test Logic:

This block determines specific game conditions:

D7 = 1 if the sum of the dice equals 7.

D711 = 1 if the sum of the dice is 7 or 11.

D2312 = 1 if the sum of the dice equals 2, 3, or 12.

Comparator:

The comparator checks whether the current sum of the dice equals the stored point from the point register, producing the signal Eq.

Control Circuit:

The control circuit manages the game logic based on the signals provided by the test logic and comparator. It:

Detects win/loss conditions (Win, Lose).

Controls the game progression (e.g., storing the point and triggering rolls).

7-Segment Displays:

The outputs of the two counters are displayed on 7-segment displays to show the dice values once the roll button is released.

3. Design Entry

Counters

```
    library IEEE;

 use IEEE.STD_LOGIC_1164.ALL;

 use IEEE.NUMERIC_STD.ALL;

4.
5. entity Counter is
6.
         Port (
7.
            clk : in std_logic; -- Clock signal
            rst : in std_logic; -- Reset signal
9.
            roll : in std_logic;
10.
            count : out std_logic_vector(2 downto 0); -- 3-bi
11.
            cycle : out std_logic
12.
end Counter;
14.
```

```
15. architecture Behavioral of Counter is
       16.
                 signal count_signal : std_logic_vector(2 downto 0) :=
             "001": -- Initial value
       17.
                signal cycle_signal : std_logic := '0'; -- Signal for
       18. begin
       19.
                process (clk, rst)
       2.0.
                    variable count_var : std_logic_vector(2 downto 0) :
             = "001"; -- Internal variable
       21.
       22.
                    if rst = '1' then
       23.
                         count_var := "001"; -- Reset to "001" (binary
       24.
                         cycle_signal <= '0';
       25.
                    elsif rising_edge(clk) then
       26.
                         if roll = '1' then
       27
                            if count_var = "110" then -- Check if it i
       28.
                                count_var := "001"; -- Wrap around to
              "001" (binary 1)
       29.
                                cycle_signal <= '1'; -- New clock to t
             he next counter
       30.
       31.
                                count_var := std_logic_vector(unsigned
             (count_var) + 1); -- Increment counter
       32.
                                cycle_signal <= '0'; -- Same
       33.
                            end if;
       34.
                         end if;
       35.
       36.
                     -- Update the signal outputs
       37.
                    count_signal <= count_var;</pre>
       38.
                 end process;
       39.
       40.
                 -- Assign signals to the output ports
       41.
                 count <= count_signal;</pre>
       42.
                cycle <= cycle_signal;</pre>
       43.
       44. end Behavioral;
Adder

    library IEEE;

            use ieee.std_logic_1164.all;
       3.
            Entity Adder is port(
```

A : In std_logic;

```
6. B : In std_logic;
                                                                                    29. Adder1: Adder port map(A=>A(1), B=>B(1), CI=>Carryout(0)
                                                                                        , Sum=>Sum(1) , CO=>Carryout(1) );

 CI : In std_logic;

      8. Sum : Out std_logic;
                                                                                    30. \quad \text{Adder2: Adder port } \mathop{\mathsf{map}}(\mathsf{A} \!\! > \!\! \mathsf{A(2)} \text{, } \mathsf{B} \!\! > \!\! \mathsf{B(2)} \text{ , } \mathsf{CI} \!\! > \!\! \mathsf{Carryout(1)}
                                                                                           , Sum=>Sum(2) , CO=>Sum(3) );
      9. CO : Out std_logic
      10. );
                                                                                    31.
      11. End Adder;
                                                                                    32.
      12.
                                                                                    33. end structural;
                                                                             Point Register
      13. \ \ \text{Architecture behavioral of Adder Is}
       14.

    library IEEE;

      15. Begin

 use IEEE.STD_LOGIC_1164.ALL;

      17. sum \leftarrow A xor B xor CI;
                                                                                    4. entity Point_register is
      18.\, CO <= (A and B) or (CI and A) or (CI and B) ;
                                                                                    5. generic ( width : integer := 3 );
      19.
                                                                                    6.

 Port ( clk : in std_logic;

      20. end behavioral;
MultiAdder
                                                                                              -- Clock input
       1. library IEEE;
                                                                                    8.
                                                                                                  rst : in std_logic;
                                                                                                 -- Synchronous reset
      2. use ieee.std_logic_1164.all;
                                                                                              Sp : in std_logic;
                                                                                               -- Load enable signal
      4. entity MultiAdder is
      5. port (
                                                                                    10.
                                                                                                  Data_In : in std_logic_vector(width downto 0)
      6. \hspace{0.5cm} \textbf{A} \hspace{0.5cm} : \hspace{0.1cm} \textbf{In} \hspace{0.1cm} \textbf{std\_logic\_vector(2 downto 0);} \\
                                                                                              -- Data input
      7. B : In std_logic_vector(2 downto 0);
                                                                                    11.
                                                                                                 Data_Out : out std_logic_vector(width downto 0)
      8. Sum : Out std_logic_vector(3 downto 0)
                                                                                               -- Data output
      9. );
                                                                                    12.
      end MultiAdder;
                                                                                    13. end Point_register;
                                                                                    14.
       12. Architecture structural of MultiAdder is
                                                                                    15. architecture slice of Point_register is
       13.
                                                                                    16. \hspace{1.5cm} {\tt signal \ Data} \, : \, {\tt std\_logic\_vector(width \ downto \ 0)} \, := \, ({\tt oth} \,
                                                                                          ers => '0'); -- Internal signal
      14. component Adder
       15. port(
                                                                                    17. begin
       16. A : In std_logic;
                                                                                    18. process(clk) -- Only clk in the sensitivity list for
                                                                                          synchronous logic
       17. B : In std_logic;
       18. cI : In std_logic;
                                                                                    19. begin
      19.
                                                                                    20.
                                                                                                 if rising_edge(clk) then
      20. Sum: Out std_logic;
                                                                                    21.
                                                                                                  if rst = '1' then
                                                                                    22.
      21. co : Out std_logic);
                                                                                                         -- Reset the register synchronously
      22. end component;
                                                                                    23.
                                                                                                     Data <= (others => '0');
      23.
                                                                                    24.
                                                                                                     elsif Sp = '1' then
      24. \quad {\tt signal \ Carryout: \ {\tt std\_logic\_vector(1 \ downto \ 0);}}
                                                                                    25.
                                                                                                      -- Load new data when Sp is high
      25.
                                                                                                       Data <= Data_In;
      26. Begin
                                                                                    27.
                                                                                                  end if;
                                                                                    28.
                                                                                                 end if;
      28. Adder0: Adder port \mbox{map}(\mbox{A=>A(0), B=>B(0) , CI=>'0'} , Su
                                                                                    29.
                                                                                              end process;
```

m=>Sum(0) , CO=>Carryout(0));

```
30.
31. -- Assign internal signal to output
32. Data_Out <= Data;</pre>
33.
34. end slice;
```

TestLogic

```
    library IEEE;

    use IEEE.STD_LOGIC_1164.ALL;

    use IEEE.NUMERIC_STD.ALL;

entity TestLogic is
       Port (
        Sum : in std_logic_vector(3 downto 0); -- Sum
of the two dice
8.
          D7 : out STD_LOGIC;
9. D711 : out STD_LOGIC;
          D2312 : out STD_LOGIC
11. );
12. end TestLogic;
14. architecture Behavioral of TestLogic is
15. begin
16.
       PROCESS(Sum)
17. BEGIN
18.
          -- Default outputs
19.
          D7 <= '0';
20.
           D711 <= '0';
21.
           D2312 <= '0';
22.
23.
24.
              WHEN "0111" =>
25.
              D7 <= '1';
26.
                D711 <= '1';
              WHEN "1011" =>
28.
                 D711 <= '1';
              WHEN "0010" | "0011" | "1100" =>
30.
                D2312 <= '1';
31.
              WHEN OTHERS =>
32.
                 -- Do nothing, all signals remain '0'
33.
                NULL;
34.
           END CASE;
       END PROCESS;
```

Comparator

36. end Behavioral;

```
    library IEEE;

    use IEEE.STD_LOGIC_1164.ALL;

4. \quad \text{entity comparator is} \\
generic(
6.
         WIDTH : integer := 3
8.
       Port (
clk : in STD_LOGIC;
             CompA : in STD_LOGIC_VECTOR (WIDTH downto 0);
          CompB : in STD_LOGIC_VECTOR (WIDTH downto 0);
12.
             Eq: out STD_LOGIC);
13. end comparator;
14.
15. architecture Behavioral of comparator is
16. begin
17. process(clk) begin
        if (CompA = CompB) then
19. Eq <= '1';
20.
21. Eq <= '0';
22. end if;
23. end process;
24. end Behavioral;
```

Control

1.	library IEEE;	
2.	use IEEE.STD_LOG	GIC_1164.ALL;
3.	use IEEE.NUMERIO	_STD.ALL;
4.		
5.	entity Control i	S
6.	Port (
7.	clk	: in STD_LOGIC;
8.	rst	: in STD_LOGIC;
9.	Rb	: in STD_LOGIC; Roll button
10.	D7	: in STD_LOGIC;
11.	D711	: in STD_LOGIC;
12.	D2312	: in STD_LOGIC;
13.	Eq	: in STD_LOGIC;
14.		
15.	Outpu	rts
16.	Win	: out STD_LOGIC;
17.	Lose	: out STD_LOGIC;
18.	Sp	: out STD_LOGIC;
19.	Roll	: out STD_LOGIC

```
20.
        );
21. end Control;
22.
23. architecture Behavioral of Control is
24.
        -- State Declaration
25.
     TYPE STATE_TYPE IS (Start, Play1, Play2Interval, Play2,
      EndGame);
26.
        SIGNAL state : STATE_TYPE := Start;
27. SIGNAL Roll0 : std_logic;
28.
29. begin
30.
         PROCESS(clk, rst)
31.
        BEGIN
32.
33.
            Roll0 <= Rb;
34.
            Roll <= Roll0;
35.
36.
       IF rst = '1' THEN
37.
                state <= Start;
38.
                Win <= '0';
39.
               Lose <= '0';
40.
                      <= '1';
41.
               Roll <= '0';
42.
43.
           ELSIF state = EndGame THEN
44.
        Roll <= '0';
45.
46.
            ELSIF rising_edge(clk) THEN
47.
             -- Default outputs
48.
49.
                CASE state IS
50.
                   WHEN Start =>
51.
                       IF Rb = '1' THEN
52.
                          state <= Play1;
53.
                       END IF;
54.
55.
         WHEN Play1 =>
56.
         IF Rb = '0' THEN
57.
          Sp <= '0';
58.
          IF D711 = '1' THEN
59.
           Win <= '1';
60.
           state <= EndGame;</pre>
61.
          ElSIF D2312 = '1' THEN
62.
           Lose <= '1';
```

63.	<pre>state <= EndGame;</pre>
64.	ELSE
65.	<pre>state <= Play2Interval;</pre>
66.	END IF;
67.	END IF;
68.	
69.	When Play2Interval =>
70.	IF Rb = '1' THEN
71.	state <= Play2;
72.	END IF;
73.	
74.	WHEN Play2 =>
75.	IF Rb = '0' THEN
76.	IF D7 = '1' THEN
77.	Lose <= '1';
78.	<pre>state <= EndGame;</pre>
79.	ELSIF Eq = '1' THEN
80.	Win <= '1';
81.	<pre>state <= EndGame;</pre>
82.	ELSE
83.	<pre>state <= Play2Interval;</pre>
84.	END IF;
85.	END IF;
86.	
87.	WHEN EndGame =>
88.	Wait for reset
89.	NULL;
90.	END CASE;
91.	END IF;
92.	END PROCESS;
93.	
	end Behavioral;
r 7	seg

Ch

ıa	r_7s	seg
	1.	library ieee;
	2.	use ieee.std_logic_1164.all;
	3.	
	4.	entity char_7seg is
	5.	port (
	6.	c : in std_logic_vector(2 downto 0); 4-bit inpu
		t character code
	7.	display : out std_logic_vector(7 downto 0) 7-se
		gment output
	8.);
	9.	end char_7seg;

```
10.
                                                                                                                                                                                               26. \hspace{0.5cm} \textbf{A} \hspace{0.5cm} : \hspace{0.1cm} \textbf{In} \hspace{0.1cm} \hspace{0.1cm} \textbf{std\_logic\_vector(width -1 downto 0);} \\
                                                                                                                                                                                               27. B : In std_logic_vector(width -1 downto 0);
                11. architecture behavior of char_7seg is
                12. begin
                                                                                                                                                                                               28. \hspace{0.5cm} \texttt{Sum} \hspace{0.5cm} : \hspace{0.5cm} \texttt{Out} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \texttt{std\_logic\_vector}(\texttt{width} \hspace{0.5cm} \hspace{0.1cm} \hspace{0.1c
               13. process(c)
                                                                                                                                                                                               29. );
                14. begin
                                                                                                                                                                                               30. end component;
                15. case c is
                16.
                                                                                                                                                                                               32. component Counter
                                                      when "000" => display <= "11000000"; -- 0
                                                                                                                                                                                               33. Port ( clk : in std_logic;
                17.
                                                  when "001" => display <= "11111001"; -- 1
                                                      when "010" => display <= "10100100"; -- 2
                                                                                                                                                                                                                                       rst : in std_logic;
                19.
                                                                                                                                                                                               35.
                                                      when "011" => display <= "10110000"; -- 3
                                                                                                                                                                                                                      roll : in std_logic;
               20.
                                                      when "100" => display <= "10011001"; -- 4
                                                                                                                                                                                                                     count : out std_logic_vector (width -1 d
               21.
                                                                                                                                                                                                      ownto 0);
                                                     when "101" => display <= "10010010"; -- 5
                                                                                                                                                                                               37. cycle : out std_logic
                                                     when "110" => display <= "10000010"; -- 6
               23. when others => display <= "11111111"; -- Defau
                                                                                                                                                                                               38. );
                  Lt to blank
                                                                                                                                                                                               39. end component;
               24.
                                         end case;
               25. end process;
                                                                                                                                                                                               41. component Point_register
                                                                                                                                                                                               42. Port (
               26. end behavior;
                                                                                                                                                                                               43. clk : in std_logic;
Dice(Main function)

    library IEEE;

                                                                                                                                                                                                                                             rst : in std_logic;
                                                                                                                                                                                               45. Sp : in std_logic;
               2. use ieee.std_logic_1164.all;
                                                                                                                                                                                               46.
                                                                                                                                                                                                                                          Data_In : in std_logic_vector (width down
               4. entity Dice is
                                                                                                                                                                                                        to 0);
                                                                                                                                                                                               47. Data_Out : out std_logic_vector (width down
               generic(
                                 width : Natural := 3 --
                                                                                                                                                                                                       to 0)
                                                                                                                                                                                               48. );
                                      port (
                                                                                                                                                                                               49. end component;
               9. clk : in std_logic;
                10.
                                                                                                                                                                                               51. component comparator
                                      Rb : in std logic;
                                  rst : in std_logic;
                11.
                                                                                                                                                                                               52. Port ( clk : in std_logic;
                                                                                                                                                                                               53. CompA : in STD_LOGIC_VECTOR (width d
                12.
                                                                                                                                                                                                     ownto 0);
                13. win : out std_logic;
                                                                                                                                                                                               54.
                14.
                             lose : out std_logic;
                                                                                                                                                                                                                                          CompB : in STD_LOGIC_VECTOR (width d
                15.
                                                                                                                                                                                                        ownto 0);
               16.
                                                                                                                                                                                               55. Eq : out STD_LOGIC
                                   display0 : out std_logic_vector(7 downto 0);
                17. display1 : out std_logic_vector(7 downto 0)
                                                                                                                                                                                               56. );
                18.
                19. );
               20. end Dice;
                                                                                                                                                                                               59. component TestLogic
               21.
                                                                                                                                                                                               60. Port (
                                                                                                                                                                                               61. Sum : in std_logic_vector(3 downto 0); -- Sum
               22. Architecture structural of Dice is
                                                                                                                                                                                                   of the two dice
```

62. D7 : out STD_LOGIC;

63. D711 : out STD_LOGIC;

24. component MultiAdder

25. port (

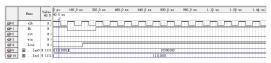
```
64.
           D2312 : out STD_LOGIC
    ):
66. end component;
69.
     Port (
70.
                  : in STD_LOGIC;
71.
                : in STD_LOGIC;
72.
                  : in STD_LOGIC; -- Roll button
73.
               : in STD_LOGIC;
                 : in STD LOGIC:
           D711
75.
           D2312 : in STD_LOGIC;
76.
                  : in STD LOGIC;
77.
78.
           -- Outputs
79.
           Win
                 : out STD_LOGIC;
80
                : out STD_LOGIC;
81.
                : out STD_LOGIC;
82.
                : out STD_LOGIC
83. );
84. end component;
85. -----
86. component char_7seg
87. Port (
           c : in std_logic_vector(2 downto 0); -- 4-bit inpu
     t character code
           display : out std logic vector(7 downto 0) -- 7-se
     gment output
90.
93. signal Cnt0 : std_logic_vector(width -1 downto 0):= (
     others => '0');
94.
       signal Cnt1 : std_logic_vector(width -1 downto 0):= (
95. signal DatIn : std_logic_vector(width downto 0):= (
     others => '0');
96.
      signal DatOut: std_logic_vector(width downto 0):= (
     others => '0');
97. signal Sum0 : std_logic_vector(width downto 0);
98.
       signal Sp0 : std_logic;
99. signal Eq0 : std_logic;
100.
       signal D70 : std_logic;
101. signal D7110 : std_logic;
```

```
102.
        signal D23120 : std_logic;
103. signal Roll0 : std_logic;
104.
       signal cycle0 : std_logic;
105.
      signal cycle1 : std_logic;
106.
107.
108. Begin
109.
                                        port map (clk => c
     lk , rst => rst , roll => Roll0 , count => cnt0, c
     ycle => cycle0);
111. Counter1:
                                        port map (clk => c
     ycle0 , rst => rst , roll => Roll0 , count => cnt1,
     cycle => cycle1 );
112. MultiAdder0:
                      MultiAdder
     nt0, B => cnt1 , Sum => Sum0 );
113. Point_register0:
                        Point_register
      Data Out => DatOut);
114. comparator0:
                        comparator
                                          port map (clk =>
      clk , CompA => DatOut, CompB => Sum0 , Eq => Eq0 );
115. TestLogic0: TestLogic
                                     port map (Sum => Sum0,
   D7 => D70 , D711 => D7110, D2312 => D23120);
116. Control0:
                      Control
      clk , rst => rst , Rb => Rb, D7 => D70, D711 => D7110, D
     2312 => D23120 , Eq => Eq0, Win => win, Lose => lose, Sp
     => Sp0, Roll => Roll0);
117. Char_7seg0:
     cnt0, display => display0 );
118. Char_7seg1:
      cnt1, display => display1 );
119.
120. end structural;
121.
122. end behavior;
```

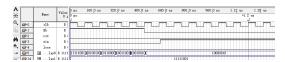
4. Simulation and Synthesis Results

On the first roll of dice:

Loses when D2312 = 1(e.g. 3)



Wins when D711 = 1(e.g. 7)Else proceeds.



On the second roll of dice:

Wins when Eq = 1

D	Master T	ime Bar:	0 p	o Pointer:	0 pc	Interva		0 po	Stat:	E	nd	
A ⊛		Nano	Value 0 p	0 ps 160,0 ns 0 ps	320 _. 0 as	480.0 as	640.0 ns	800.0 as	960.0 as	1.12 us	1.28 us	1.44 us
Œ.	130°0	elk	3		штш							
80	ii≥1	3h										
**	<u>m≥2</u>	rst	8 1									
44	6993	vin	3 :									
٠,	€94	lose	8 1									
-+	⊚ 5	₩1u ₀ 0	B 1111	11111001 1010010CX	10110000	001100100010	10000001	01111001		10100100		
894	⊚ 14	■leyt	B 1111		11111001			х		10100100		шш

Loses when D7 = 1

		Value) ps 160.0 ns :	320. 0 as	480. 0 ax	640.0 ax	800.0 ns	960.0 ns	1.12 us	1.28 us	1.44 ux
	Yune	0 1	ps						+1.2	65	
□ 0	clk	В			1 1				- m		\Box
	Eh.	В									
□ 2	rst	B (
€93	vin	B (
◆ 4	lose	B 1									
		B 1111	11111001101001001	10110000	200110	0010010010	000001(111100	001001001001	000000000000000000000000000000000000000	1001001	0
€9 14	⊞layt	B 1111		111110	01		X		10100100		

Else proceeds.

On any subsequent roll:

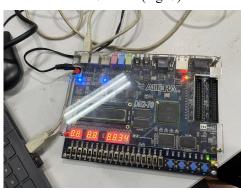
The criteria remains, e.g. 1^{st} Point = 5 and the 3^{rd} roll triggers.

		Value	D ps	160.0 as	320.0 ns	480.0 ns	640.0 ms	800. 0 as	960.0 ns	1.12 ws	1.28 us
	Nune	0.1	0 ps							*1.2	us
⊞ •0	elk	В		ПШТ		пшп			лшт	ww	
∰-1 ∰-2 ∰-3 ∰-4 ∰-6	Nb.	В				П					
⊞ 2	rst	B 1									
23 ≥3	win	B 1									
·23 4	lore	B 1									
⊚ 5	⊞1xy0	B 1111	1111100:	10100100100	11000CX 100	11001 (1001)	001000000100	1111001 10	100100 2001	10000 001100	10010010
€9/14	⊞ …layi	B 1111			1111100		X			10100100	

5. Experimental Results

On the first roll of dice:

Wins when D711 = 1 (e.g. 7)



Lose when D2312 = 1 (e.g. 3)



Else proceeds.

On the elaborating roll:

If the first roll reads 6,



...the green light would be ablaze,



...else if D7 = 1, one loses.



6. Discussion and Conclusion

This humble implementation, though seemingly running smoothly, is a result of sheer dedication and countless trials and errors. One of the most notable glitches encountered during the early stages of development was a peculiar one-cycle lag. This issue would have been painstaking to resolve without abandoning the 'Rb in, Roll out' structure of the Control block and directly plugging the

Roll Button signal into the subsequent stage, which is the first stage of the cascaded counter.

	Hune	Value 30.55	5,8 us	5.88 us	5.96 us	6.04 us	6.12 us	6.2 _, us	6.28 us
i 0 €	elk	В :	h						
ii)• 1	Rb	В							
<u>11</u> 2	rst	В							
⊕ 3	win	BI							
⊕4	lose	В 1							
⊚ 6	■1ay0	B 1111	10000010	11111100	01 10	100100 X			10110000
⊚ 14	■layi	B 1111	10110000	K T				10011001	

The root cause of this issue lies in the timing sequence. When the Rb signal is turned off, it

isn't processed until the next cycle. This delay turns off the roll signal but prevents the counter from addressing it because both are operating within the same asynchronous structure. Since the counter cannot predict the next state of the roll button, the issue remains unsolvable unless the roll signal is deliberately moved forward. This requires re-routing it directly to the roll button input.