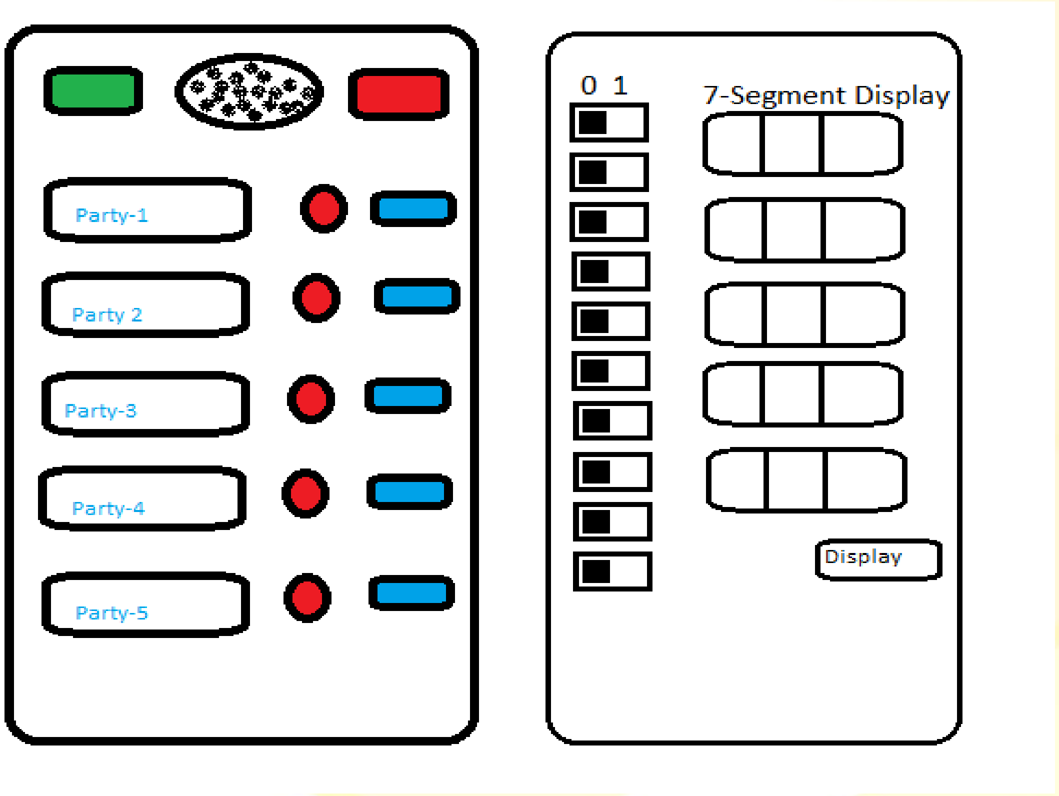
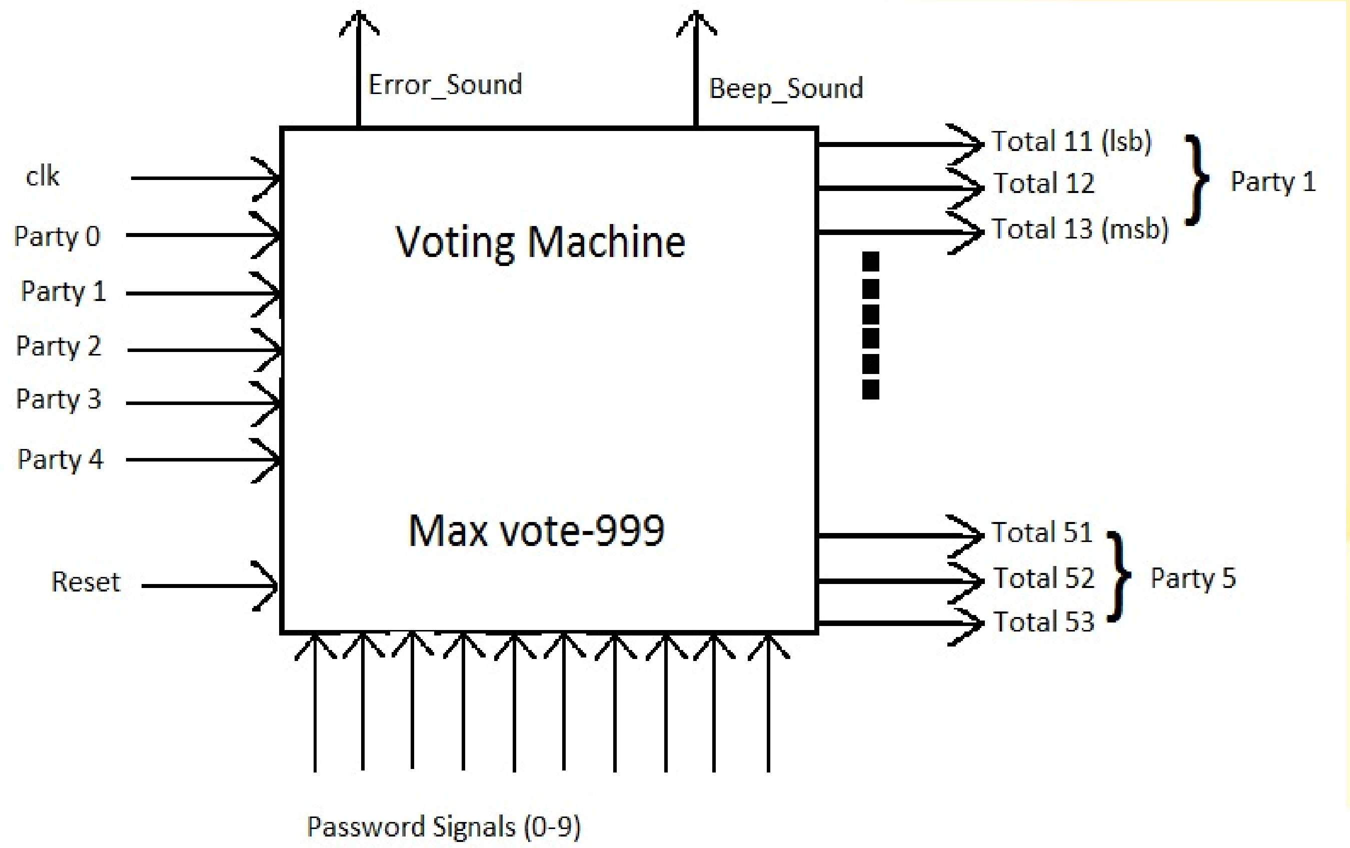
Design


Party-1
Party 2
Party-3
Party-4
Party-5
0 1
7-Segment Display
Display

Black- Box of Machine



Input Signals to Machine

1. Enable
2.Party(0-4)
3.Reset
4.Password

1.Enable

The function of enable input is to on and off the machine. It will be controlled manually by the officials. They can on and off the Machine as per their requirement .

When Machine will be turned off the vote count will be stored in its memory and can be accessed later on.

When the input to enable is “0” then the machine is on i.e it will count the votes and when the input is “1” then it is off and will not count any vote.

2. Party(0-4)

These are the input buttons on the machine to vote for a particular party.

When button is pressed the vote will be counted.

It doesn’t matter how longer you press the button it will count the single vote for one time press as it will count on the falling edge.

Party 0- First Party
Party 1- Second Party
Party 2- Third Party
Party 3- Fourth Party
Party 4- Fifth Party

3.Reset

This input will not be accessed from the outer body of the machine as it will reset the whole machine i.e it will be used to renew the machine for further elections so it will be accessed by inside the machine.

Vote count will become zero when the input to reset will “1”.

4.Password

(Pass 0-9)

This is password input of the voting machine. Pass key (0-9) will take input of the password by which the officials will gain access to vote count.

When password is will be entered and display button will be pressed the the machine will show vote count on the 7-segment display.

Output Signals

1.Error sound
2.Beep sound
3.7-segment display(total)

1.Error Sound

It will produce error sound when there will be any in-discrepancy in the machine such as when someone will press 2 party buttons simultaneously.

When Error-sound is produced then that vote will not be counted.

2. Beep sound

This output will produce a beep sound i.e when their will be normal input to machine and everything else is good. The vote with beep sound will be counted.

3. 7-Segment display(total)

This is the output to the 7 segment LED display which will be accessed by entering password .It will display the total vote count of all the parties.

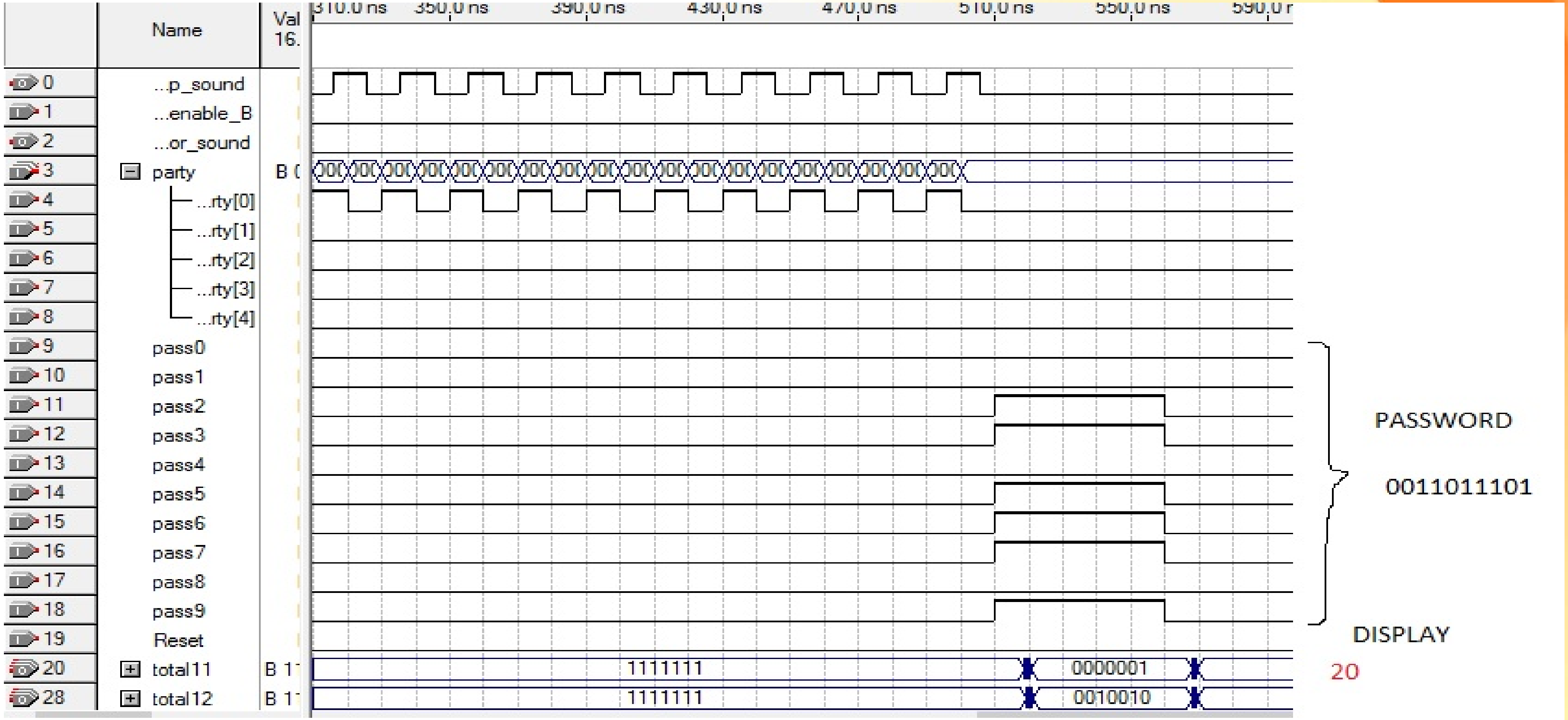
$$\left.\begin{array}{l}\text{total11}\\ \text{total12}\\ \text{total13}\end{array}\right\}\begin{array}{l}\text{->LSB}\\ \\ \text{->MSB}\end{array}$$

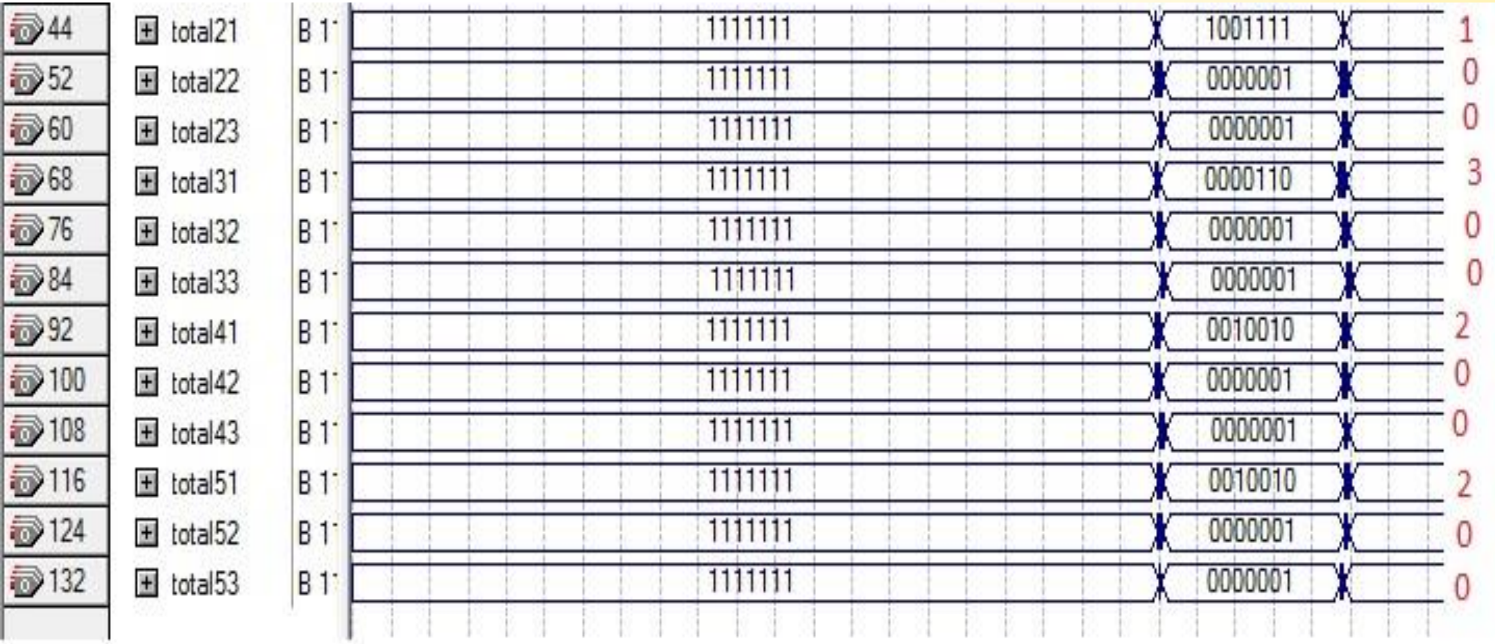
VHDL-code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity voting_machine is
 port( Clock_enable_B: in std_logic;
                party :in std_logic_vector(0 to
4);
                beep_sound :out std_logic ;
                error_sound: out std_logic;
                Reset: in std_logic;
                pass0:in std_logic;
                pass1:in std_logic;
                pass2:in std_logic;
                pass3:in std_logic;
                pass4:in std_logic;
                pass5:in std_logic;
                pass6:in std_logic;
pass7:in std_logic;
                pass8:in std_logic;
                pass9:in std_logic;

                total11 :out  std_logic_vector(0 to 6);
                total12:out std_logic_vector(0 to 6);
                total21 :out  std_logic_vector(0 to 6);
                total22:out std_logic_vector(0 to 6);
                total31 :out  std_logic_vector(0 to 6);
                total32:out std_logic_vector(0 to 6);
                total41 :out  std_logic_vector(0 to 6);
                total42:out std_logic_vector(0 to 6);
                total51 :out  std_logic_vector(0 to 6);
                total52:out std_logic_vector(0 to 6);
                total13:out std_logic_vector(0 to 6);
                total23 :out  std_logic_vector(0 to 6);
                total33:out std_logic_vector(0 to 6);
total43 :out  std_logic_vector(0 to 6);
                  total53:out std_logic_vector(0 to 6));

end voting_machine;
architecture voting_archi of voting_machine is
component BCD_party is
port( Clock_enable_B: in std_logic;
                   party: in std_logic;
                   Reset: in std_logic;
                   display: in std_logic;
                  total1 :out  std_logic_vector(0 to 6);
                  total3 :out  std_logic_vector(0 to 6);
                  carry: out std_logic;
                  total2:out std_logic_vector(0 to 6));
end component;

signal carry1,carry2,carry3,carry4,carry5,display: std_logic;
signal tparty: std_logic_vector(0 to 4);
begin
Stage1: BCD_party port map(clock_enable_B,tparty(0),Reset,display,total11,total13,carry1,total12);
Stage2: BCD_party port
```

```
map(clock_enable_B,tparty(1),Reset,display,total21,total23,carry2,total22);
Stage3: BCD_party port
map(clock_enable_B,tparty(2),Reset,display,total31,total33,carry3,total32);
Stage4: BCD_party port
map(clock_enable_B,tparty(3),Reset,display,total41,total43,carry4,total42);
Stage5: BCD_party port
map(clock_enable_B,tparty(4),Reset,display,total51,total53,carry5,total52);

display <=((not pass0)and (not pass1) and( pass2)and ( pass3)and(not pass4)and (
pass5)and( pass6)and( pass7)and(not pass8)and( pass9));

process(party)
begin
case party is
when "10000" => beep_sound <='1';
when "01000" => beep_sound <='1';
when "00100" => beep_sound <='1';
when "00010" => beep_sound <='1';
when "00001" => beep_sound <='1';
when others => beep_sound<='0';
end case;
end process;
process(party)
begin
case party is
when "10000" => error_sound <='0';
when "01000" => error_sound <='0';
when "00100" => error_sound <='0';
when "00010" => error_sound <='0';
when "00001" => error_sound <='0';
when "00000" => error_sound <='0';
--- if some one press more button the error sound
will given
when others => error_sound<='1';
end case;
end process;
process(party)
begin
case party is
when "10000" => tparty <="10000";
when "01000" => tparty <="01000";
when "00100" => tparty <="00100";
when "00010" => tparty <="00010";
when "00001" => tparty <="00001";
--- if  error came or no button pushed then no vote will
calculated
when others => tparty<="00000";
end case;
end process;


end voting_archi;
```

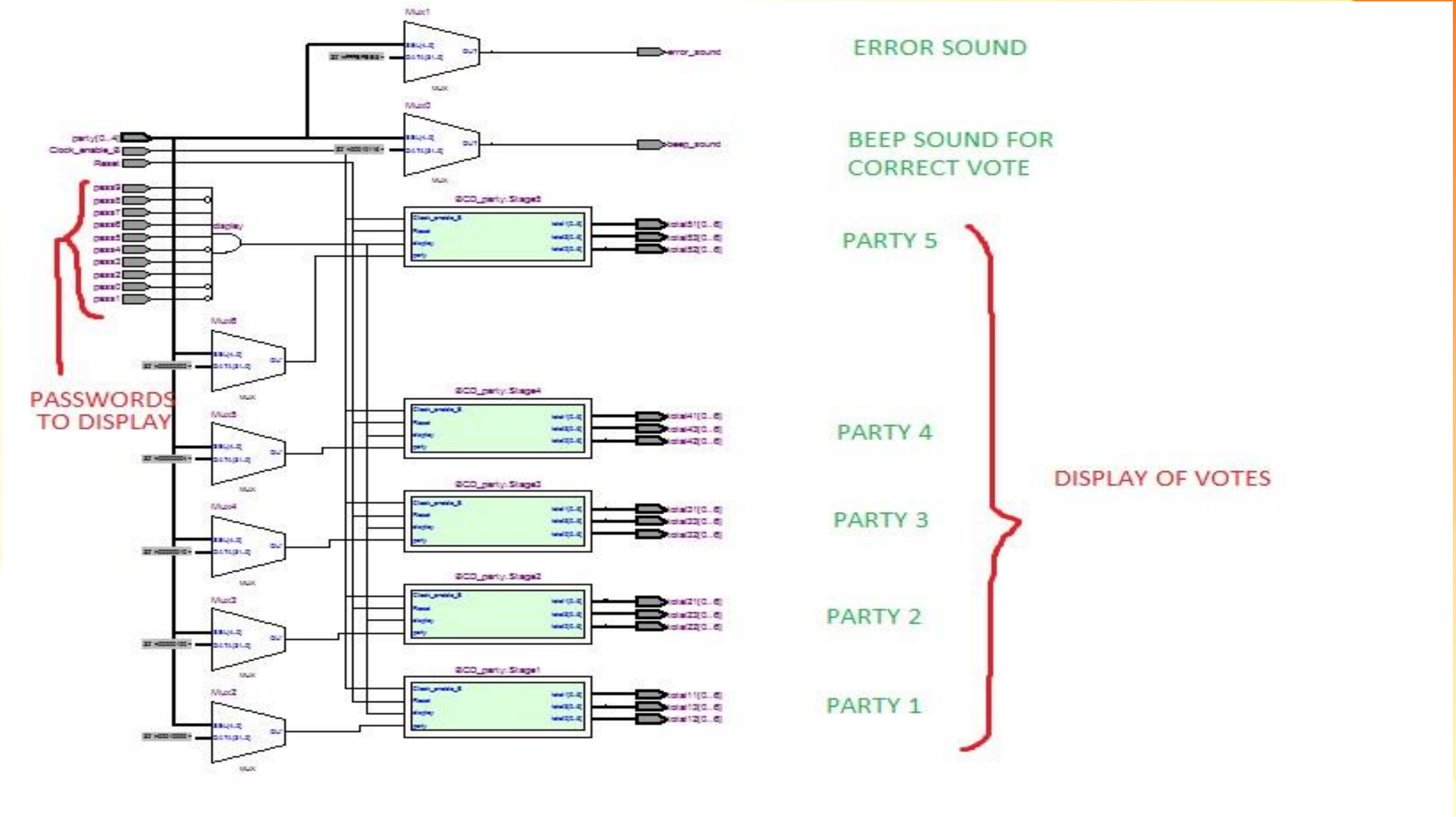
Waveform of Machine



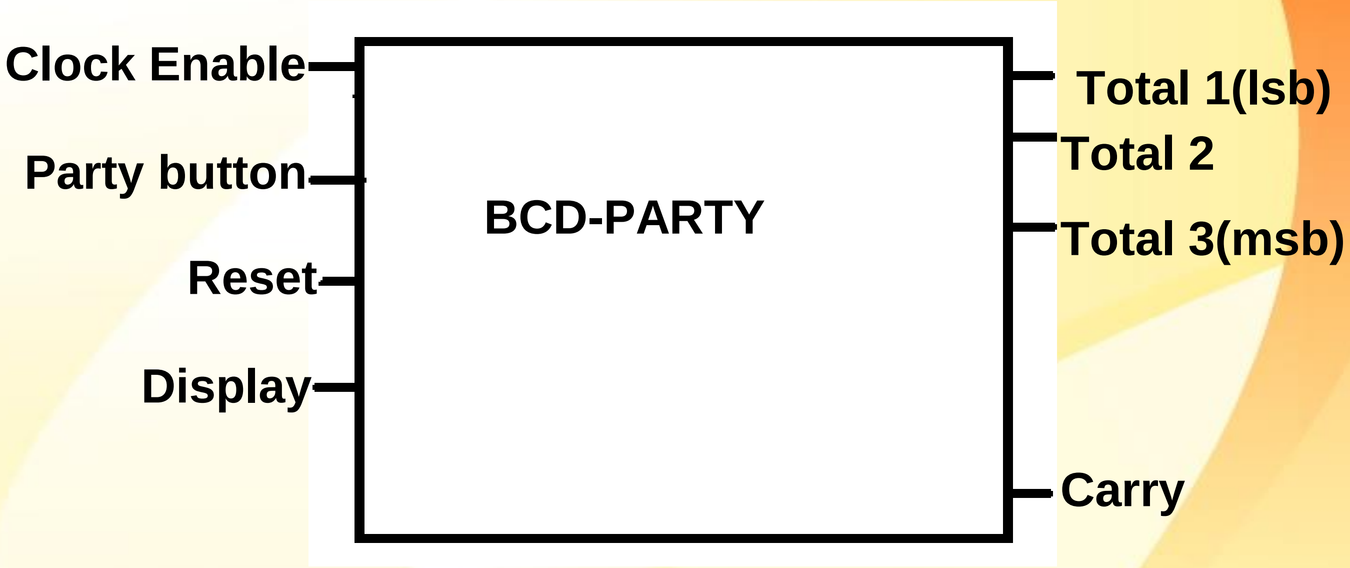


```
44   total21  B 1  1111111  1001111   1
52   total22  B 1  1111111  0000001   0   001
60   total23  B 1  1111111  0000001   0
68   total31  B 1  1111111  0000110   3
76   total32  B 1  1111111  0000001   0   003
84   total33  B 1  1111111  0000001   0
92   total41  B 1  1111111  0010010   2
100  total42  B 1  1111111  0000001   0   002
108  total43  B 1  1111111  0000001   0
116  total51  B 1  1111111  0010010   2
124  total52  B 1  1111111  0000001   0   002
132  total53  B 1  1111111  0000001   0
```

RTL View


ERROR SOUND
BEEP SOUND FOR CORRECT VOTE
PARTY 5
PARTY 4
PARTY 3
PARTY 2
PARTY 1
DISPLAY OF VOTES
PASSWORDS TO DISPLAY
Mux1
Mux0
error_sound
beep_sound
party[0..4]
Clock_enable_B
Reset
display
Mux6
Mux5
Mux4
Mux3
Mux2
BCD_party:Stage5
BCD_party:Stage4
BCD_party:Stage3
BCD_party:Stage2
BCD_party:Stage1

BCD-PARTY


Clock Enable
Party button
Reset
Display
BCD-PARTY
Total 1(lsb)
Total 2
Total 3(msb)
Carry

INPUT

Clock enable:

When the input is low it will start counting otherwise it will stop working but our previous data will be conserved

Party Button:

Actually a permanent clock signal(low) is generated and when the first voter will press the button the clock will be high and when the voter will leave the button the clock will be low again and again when the second voter will press the button the clock will be high again and the voter will leave the button clock will be low again and this process will continue and each falling edge of the clock will be counted as 1 vote by the counter

Reset:

When input will be high all data will reset

Display:

When input will be high it will display the total with the help of three

7-segment display.

Output

total1(msb), total2, total3(lsb) will display the total votes of a particular party

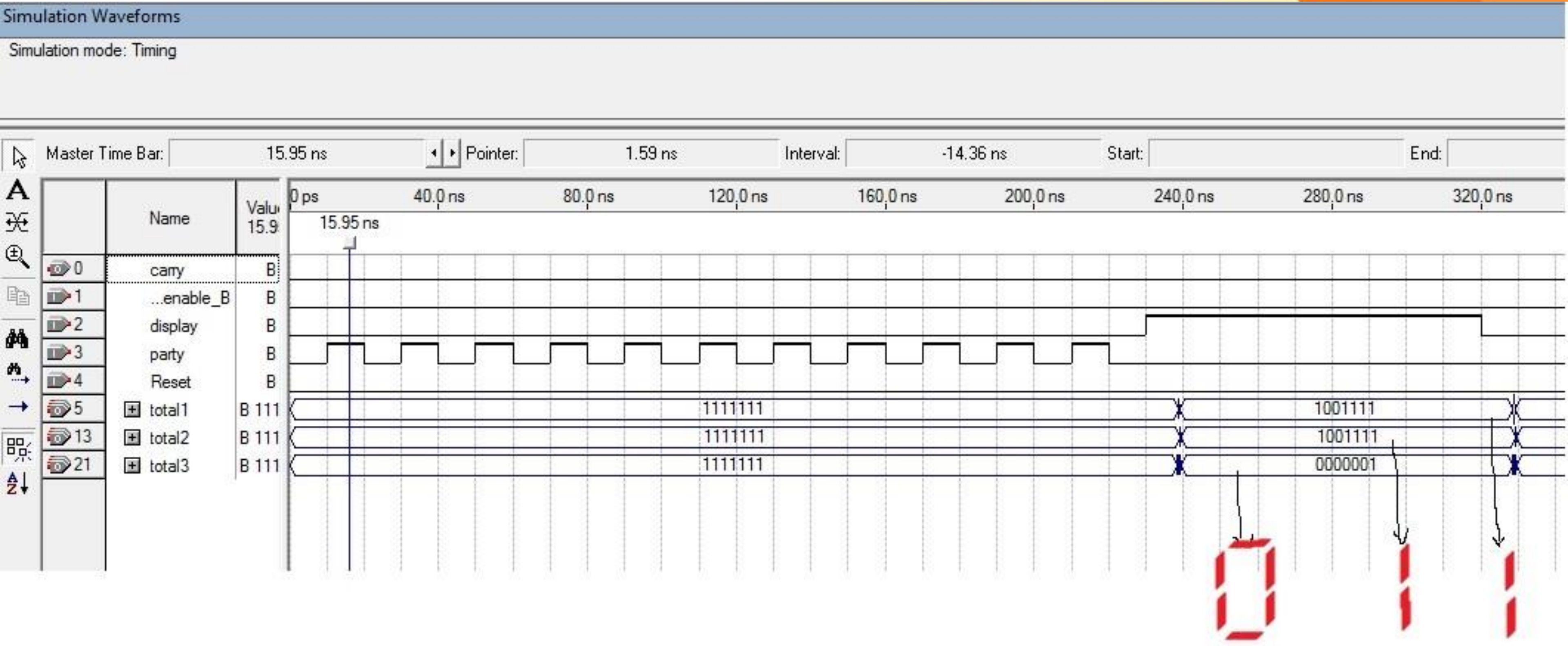
VHDL CODE FOR BCD PARTY

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity BCD_party is

 port( Clock_enable_B: in std_logic;

                   party: in std_logic;

                   Reset: in std_logic;

                   display: in std_logic;
```

```
total1 :out  std_logic_vector(0 to 6);
                  total3 :out  std_logic_vector(0 to 6);
                  carry: out std_logic;
                  total2:out std_logic_vector(0 to 6));


end BCD_party;
architecture party_archi of BCD_party is
component BCD_display is
port( Clock_enable_B: in std_logic;
                   party: in std_logic;
                   Reset: in std_logic;
```

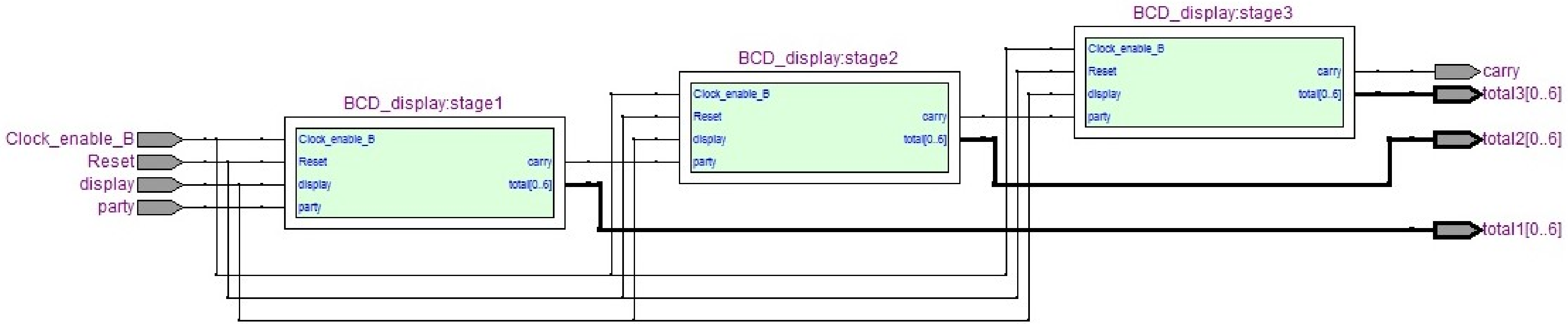
```
display: in std_logic;
                  carry:out std_logic;
                 total :out  std_logic_vector(0 to 6));
                 end component;
signal party_next:std_logic;
signal party_next1:std_logic;
begin
stage1 : BCD_display port map(Clock_enable_B ,party ,Reset,display,party_next,total1);
stage2 : BCD_display port map(Clock_enable_B ,party_next ,Reset,display,party_next1,total2);
stage3 : BCD_display port map(Clock_enable_B ,party_next1 ,Reset,display,carry,total3);
end  party_archi;
```

BCD PARTY WAVE FORM

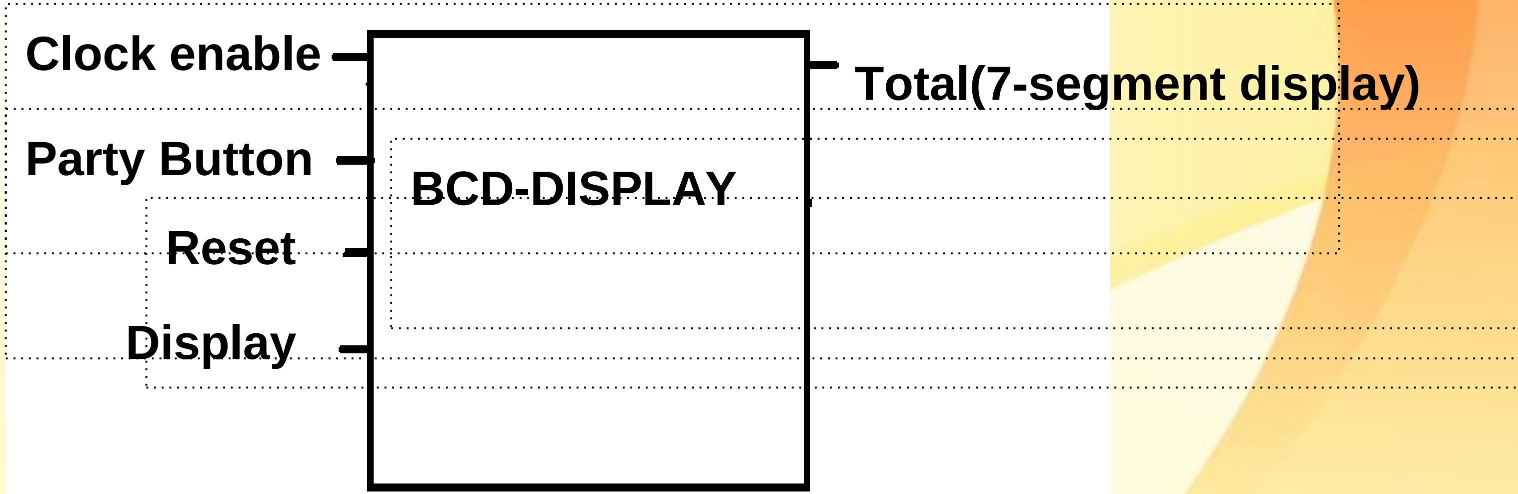


The party (clock signal) is high 11 times that means 11 voters has voted to that party So total1(lsb) will be 1001111 that indicates 1 on 7 segment display i.e 1 and similarly total2 will display 1 and toatl 3 will display 0 on three 7-segment display

BCD PARTY RTL VIEW



BCD-DISPLAY


Clock enable
Party Button
Reset
Display
BCD-DISPLAY
Total(7-segment display)

The output total1 will show LSB of the resultant vote and if the LSB is 10 then LSB will become 0 and 1 will be added to the total2 i.e is the second bit.Since maximum voters are 999 so we will use three seven segment display to show the final votes of a particular party

VHDL CODE FOR BCD-DISPLAY

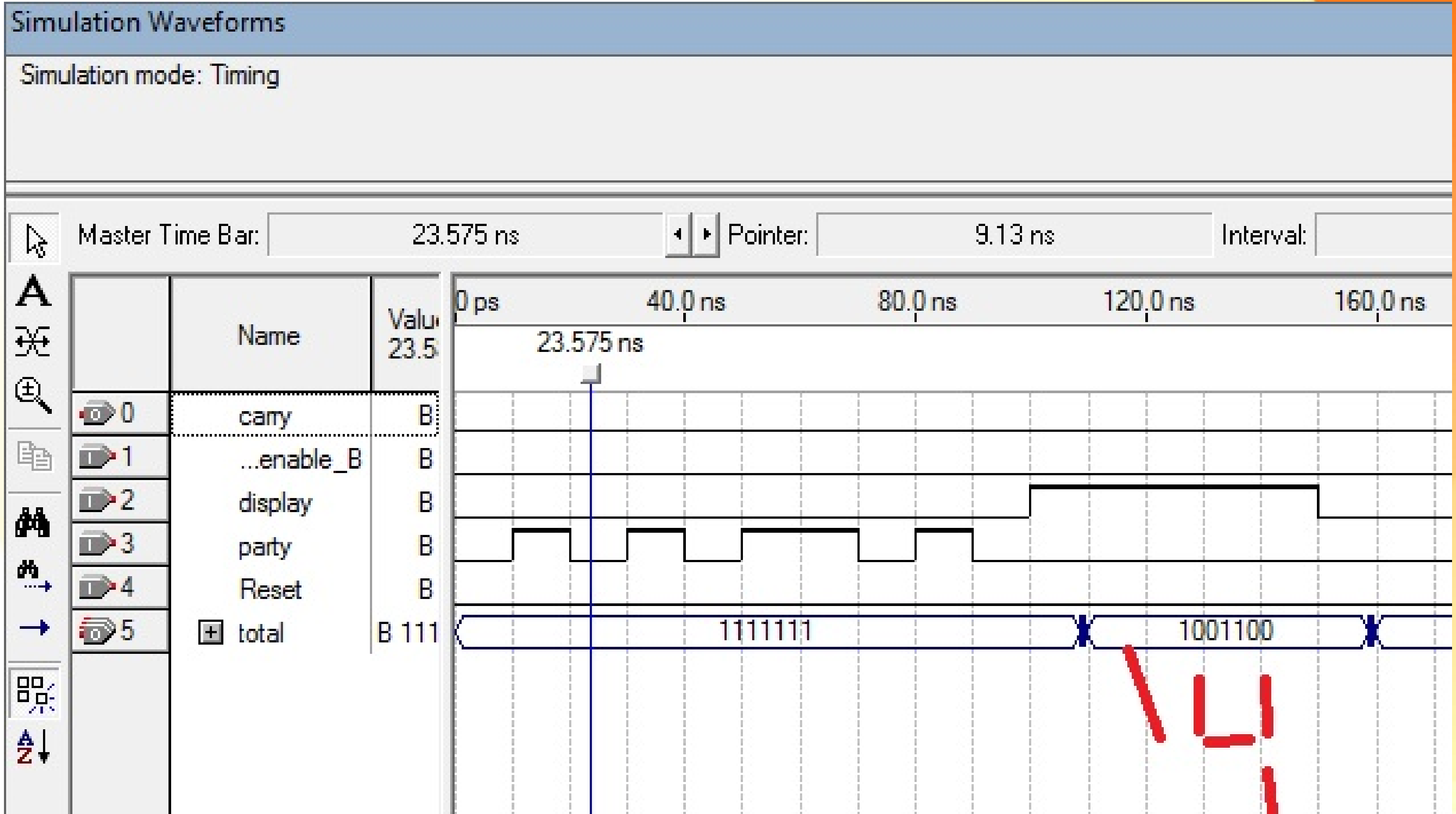
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity BCD_display is
  port( Clock_enable_B: in std_logic;
                  party: in std_logic;
                  Reset: in std_logic;
                  display: in std_logic;
                  carry:out std_logic;
                 total :out  std_logic_vector(0 to 6));
```

```
end BCD_display;

architecture BCDARCHI of BCD_display is

   signal temp: std_logic_vector(0 to 3);

   signal segment7 :std_logic_vector(0 to 6);

     begin

  process(party,Reset)

   begin

      if Reset='1' then

         temp <= "0000";elsif(falling_edge(party)) then

                    if Clock_enable_B='0' then

                       if temp="1001" then

                          temp<="0000";
```

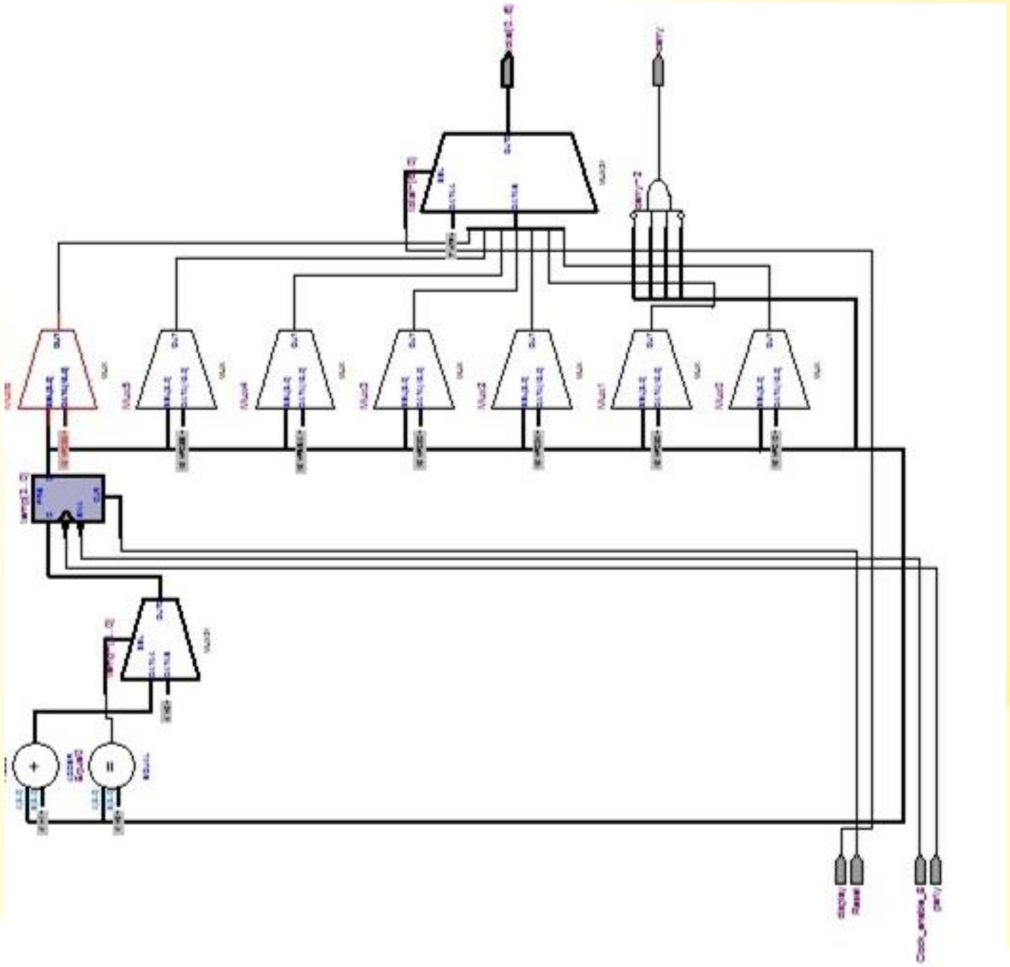
```
Else

 temp <= temp + 1;

 end if;

end if;

end if;

end process;

process (party,temp,display)

BEGIN

case  temp is

when "0000"=> segment7 <="0000001";  -- '0'

when "0001"=> segment7 <="1001111";  -- '1'

when "0010"=> segment7 <="0010010";  -- '2'
when "0011"=> segment7 <="0000110";  -- '3'

when "0100"=> segment7 <="1001100";  -- '4'

when "0101"=> segment7 <="0100100";  -- '5'

when "0110"=> segment7 <="0100000";  -- '6'

when "0111"=> segment7 <="0001111";  -- '7'

when "1000"=> segment7 <="0000000";  -- '8'

when "1001"=> segment7 <="0000100";  -- '9'

 --nothing is displayed when a number more than 9 is given as input.

when others=> segment7 <="1111111";

end case;

if display='1' then

total <=segment7 ;
```

```
else total <= "1111111";

end if;



end process;

carry <=(( temp(0))and (not temp(1))and( temp(2))and(not temp(3)));




end BCDARCHI;
```

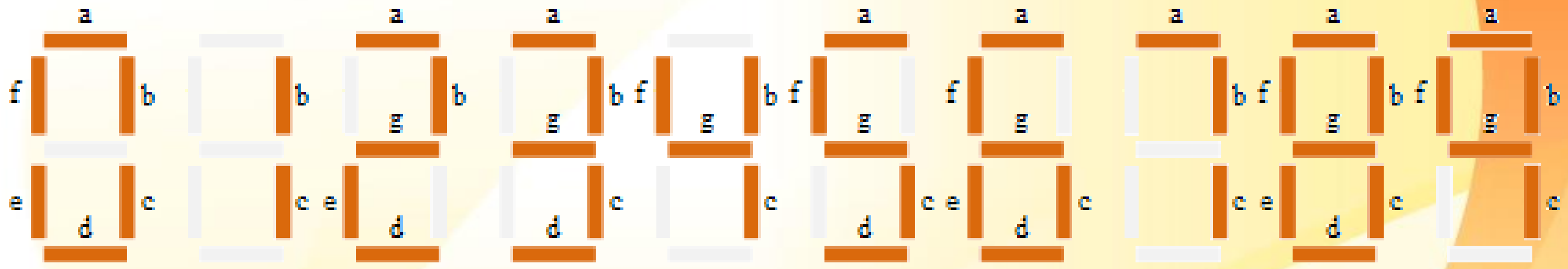
BCD DISPLAY WAVEFORM



RTL VIEW of DISPLAY



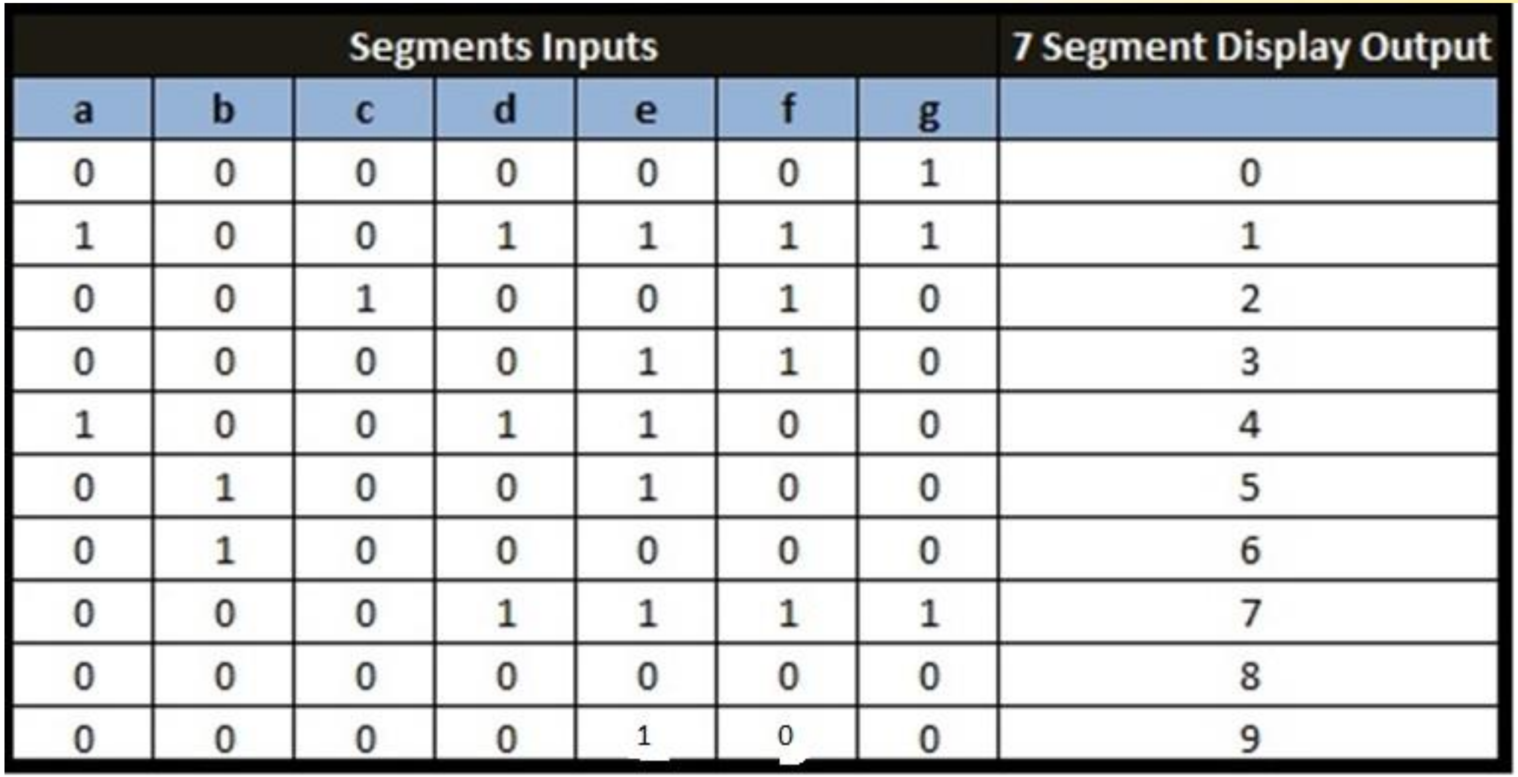
7-segment display



7-segment display for decimal no's 0 to 9

Input 1 = OFF & Input 0 = ON

TRUTH TABLE FOR 7-SEGMENT DISPLAY



Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
0	0	0	0	0	0	1	0
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2
0	0	0	0	1	1	0	3
1	0	0	1	1	0	0	4
0	1	0	0	1	0	0	5
0	1	0	0	0	0	0	6
0	0	0	1	1	1	1	7
0	0	0	0	0	0	0	8
0	0	0	0	1	0	0	9

END