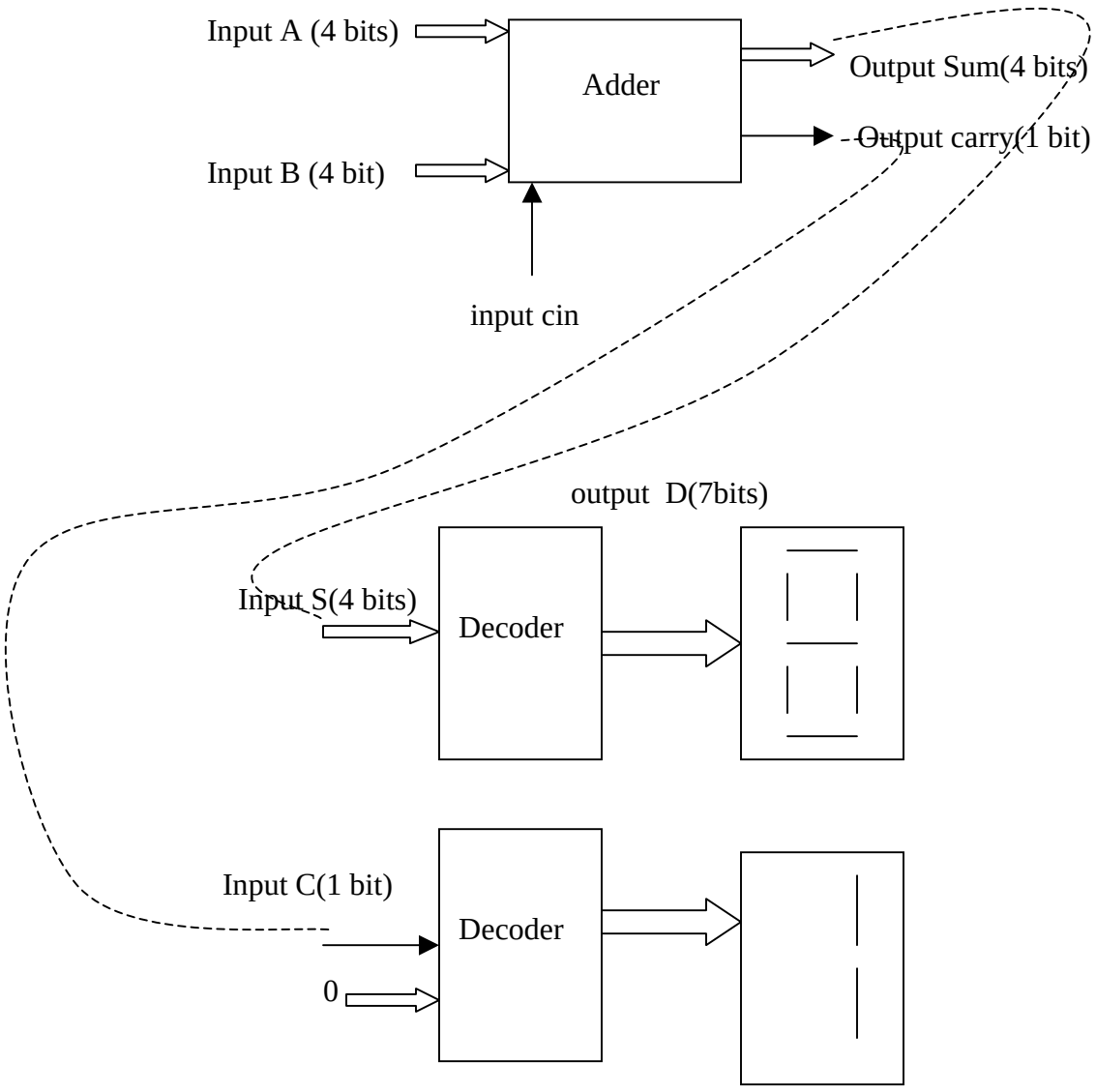
**Csc 343 Lab 2**
**Sep 28. 07**

**Objective:**

Design a 4 bit-adder. Then design a 4-7 decoder to show the outputs.

**Structure:**



**Truth table:**

**Adder**

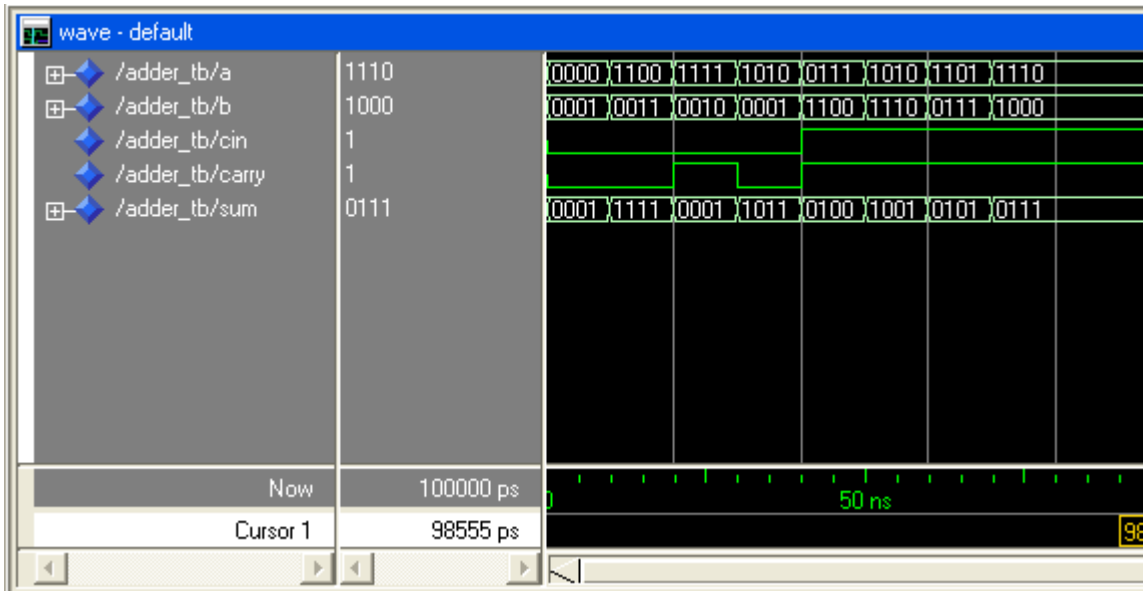
| Input | Output |
|---|---|
| A | Sum = A + B + cin |
| B | Carry = 0 if A+B+cin $\leq F_H\ (15_D)$<br>Carry = 1 Otherwise |
| cin | |

**4-7 decoder**

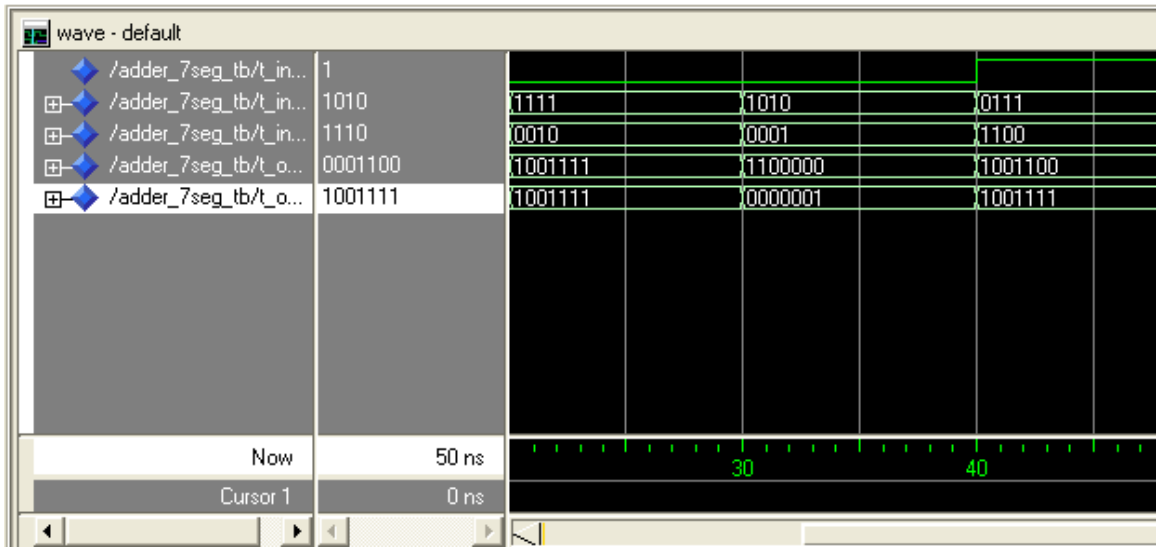
| input | output | Corresponding LED display |
|---|---|---|
| 0000 | 0000001 | 0 |
| 0001 | 1001111 | 1 |
| 0010 | 0010010 | 2 |
| 0011 | 0000110 | 3 |
| 0100 | 1001100 | 4 |
| 0101 | 0100100 | 5 |
| 0110 | 0100000 | 6 |
| 0111 | 0001111 | 7 |
| 1000 | 0000000 | 8 |
| 1001 | 0001100 | 9 |
| 1010 | 0001000 | a |
| 1011 | 1100000 | b |
| 1100 | 0110001 | c |
| 1101 | 1000010 | d |
| 1110 | 0110000 | e |
| 1111 | 0111000 | f |

**Outputs:**

**Adder:**




**adder + decoder:**



**--% ==========add.vhd==========**

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


entity ADDER is

port(   A:      in std_logic_vector(3 downto 0);
        B:      in std_logic_vector(3 downto 0);
        cin:            in std_logic;
             carry:       out std_logic;
        sum:    out std_logic_vector(3 downto 0)
);

end ADDER;


architecture behv of ADDER is

signal result: std_logic_vector(4 downto 0);

begin

   result <= ('0' & A)+('0' & B)+ cin ;                 --0A + 0B +cin  (result is 5 bits)
   sum <= result(3 downto 0);
   carry <= result(4);

end behv;
```

**--%==========dec_7seg_D.vhd==========**

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

-- Hexadecimal to two 7 Segment Decoder for LED Display

ENTITY dec_7seg_D IS
PORT(       hex_digit : in std_logic_vector(3 downto 0);
        -- enable:     in std_logic;
```

```
        Segment1  : out std_logic_vector(6 downto 0)    --segments a, b,c,d,e,f,g

);
END dec_7seg_D;

ARCHITECTURE behav OF dec_7seg_D IS
        signal segment_data1 : std_logic_vector(6 downto 0);


        BEGIN
                PROCESS (hex_digit)
                        -- HEX to two 7 Segment Decoder for LED Display
                begin
                --if ( enable ='0' ) then
                 --   segment_data1 <= "0000001";
                --else

                case  Hex_digit   is
                WHEN "0000"=>     segment_data1 <= "1111110";   -- 0
                WHEN "0001"=>     segment_data1 <= "0110000";   -- 1
                WHEN "0010"=>     segment_data1 <= "1101101";   -- 2
                WHEN "0011"=>     segment_data1 <= "1111001";   -- 3
                WHEN "0100"=>     segment_data1 <= "0110011";   -- 4
                WHEN "0101"=>     segment_data1 <= "1011011";   -- 5
                WHEN "0110"=>     segment_data1 <= "1011111";   -- 6
                WHEN "0111"=>     segment_data1 <= "1110000";   -- 7
                WHEN "1000"=>     segment_data1 <= "1111111";   -- 8
                WHEN "1001"=>     segment_data1 <= "1110011";   -- 9
                WHEN "1010"=>     segment_data1 <= "1110111";   -- A
                WHEN "1011"=>     segment_data1 <= "0011111";   -- b
                WHEN "1100"=>     segment_data1 <= "1001110";   -- C
                WHEN "1101" =>    segment_data1 <= "0111101";   -- d
                WHEN "1110"=>     segment_data1 <= "1001111";   -- e
                WHEN "1111" =>    segment_data1 <= "1000111";   -- f
                WHEN OTHERS =>     segment_data1 <= "1111110" ;  --error
                END CASE;
                END PROCESS;

                -- extract segment data bits and invert
                -- LED driver circuit is inverted

        segment1 <=   NOT segment_data1;

         END behav;
```

```
--% ==========tb_adder_7seg.vhd==========================
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity ADDER_7seg_TB is
end ADDER_7seg_TB;

architecture TB of ADDER_7seg_TB is

   component Adder_7seg is
   port(

        Input:       in std_logic;
        InputA:     in std_logic_vector(3 downto 0);
        InputB:      in std_logic_vector(3 downto 0);
        outputA:    out std_logic_vector(6 downto 0);
        outputB:    out std_logic_vector(6 downto 0)

   );
   end component;

   signal t_Input:             std_logic;
   signal t_InputA:          std_logic_vector(3 downto 0);
   signal t_InputB:       std_logic_vector(3 downto 0);
   signal t_OutputA:        std_logic_vector(6 downto 0);
   signal t_OutputB:        std_logic_vector(6 downto 0);


   begin

      U_ADDER_7seg:   Adder_7seg  port map (t_Input, t_InputA, t_InputB,
t_OutputA,t_OutputB);

   process

        variable err_cnt: integer :=0;

   begin


        -- case 1
        t_InputA <= "0000";
        t_InputB <= "0001";
        t_Input <= '0';
```

```
        wait for 10 ns;
           assert (t_OutputA="1001111") report "fist 7 segment_1 Error!" severity error;
           assert (t_OutputB= "0000001") report "second 7 segment_1 Error!" severity error;
           if ( t_OutputA /= "1001111" or t_OutputB /=  "0000001") then
              err_cnt:=err_cnt+1;
           end if;

           -- case 2
           t_InputA <= "1100";
           t_InputB <= "0011";
           t_Input <= '0';
        wait for 10 ns;

           assert (t_OutputA="0111000") report "first 7 segment_2 Error!" severity error;
           assert (t_OutputB="0000001") report "second  7 segment_2  Error!" severity
error;
           if (t_OutputA/="0111000" or  t_OutputB/="0000001") then
              err_cnt:=err_cnt+1;
           end if;

           -- case 3
           t_InputA <= "1111";
           t_InputB <= "0010";
           t_Input <= '0';
        wait for 10 ns;

           assert (t_OutputA="1001111") report " first 7 segment_3 Error!" severity error;
           assert (t_OutputB="1001111") report "second 7 segment_3 Error!" severity error;
           if (t_OutputA/="1001111" or t_OutputB/="1001111") then
              err_cnt:=err_cnt+1;
           end if;

           -- case 4
           t_InputA <= "1010";
           t_InputB <= "0001";
           t_Input <= '0';
        wait for 10 ns;

           assert (t_OutputA="1100000") report "first 7 segment_4 Error!" severity error;
           assert (t_OutputB= "0000001" ) report "second 7 segment_4  Error!" severity
error;
           if (t_OutputA/="1100000" or t_OutputB/="0000001") then
              err_cnt:=err_cnt+1;
           end if;

           -- case 5
```

```
      t_InputA <= "0111";
      t_InputB <= "1100";
      t_Input <= '1';
   wait for 10 ns;
      assert (t_OutputA="1001100") report "first 7 segment_5 Error!" severity error;
      assert (t_OutputB="1001111") report "second 7 segment_5 Error!" severity error;
      if (t_OutputA/="1001100" or t_OutputB/="1001111") then
         err_cnt:=err_cnt+1;
      end if;

-- case 6
      t_InputA <= "1010";
      t_InputB <= "1110";
      t_Input <= '1';
   wait for 10 ns;
      assert (t_OutputA="0001100") report "first 7 segment_6 Error!" severity error;
      assert (t_OutputB="1001111") report "second 7 segment_6 Error!" severity error;
      if (t_OutputA/="0001100" or t_OutputB/="1001111") then
         err_cnt:=err_cnt+1;
      end if;

-- case 7
      t_InputA <= "1101";
      t_InputB <= "0111";
           t_Input<= '1';
      wait for 10 ns;
      assert (t_OutputA="0100100") report "first 7 segment_7 Error!" severity error;
      assert (t_OutputB="1001111") report "second 7 segment_7 Error!" severity error;
      if (t_OutputA/="0100100" or t_OutputB/="1001111") then
         err_cnt:=err_cnt+1;
      end if;

 -- case 8
      t_InputA <= "1110";
      t_InputB <= "1000";
      t_Input <= '1';
   wait for 10 ns;
      assert (t_OutputA="0001111") report "first 7 segment_8 Error!" severity error;
      assert (t_OutputB="1001111") report "second 7 segment_8 Error!" severity error;
      if (t_OutputA/="0001111" or t_OutputB/="1001111") then
         err_cnt:=err_cnt+1;
      end if;

      -- summary of testbench
      if (err_cnt=0) then
         assert false
```

```
            report "Testbench of Adder_7 segment dec completed successfully!"
            severity note;
        else
            assert true
            report "Something wrong, try again"
            severity error;
        end if;

    wait;
   end process;
end TB;

configuration CFG_TB of ADDER_7seg_TB is
        for TB
        end for;
end CFG_TB;
```