/home/raffy/Dev/MouseDecoder/MouseDecoder.srcs/sources_1/new/Bin2Bcd.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/12/2022 10:22:06 PM
-- Design Name:
-- Module Name: Bin2Bcd - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use iEEE.numeric_std.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Bin2Bcd is
    Port(
        Clock:  in  STD_LOGIC;
        Reset:  in  STD_LOGIC;
        Number: in  STD_LOGIC_VECTOR(15 downto 0);
        BCD:    out STD_LOGIC_VECTOR(15 downto 0)
    );
end Bin2Bcd;

architecture Behavioral of Bin2Bcd is
type TState is (Take, Shift, Add, Provide);
signal State: TState := Take;
begin
    process(Reset, Clock, State, Number)
    variable vBuff: STD_LOGIC_VECTOR(31 downto 0);
    variable I: NATURAL;
    begin
        if Reset = '1' then
            State <= Take;
        elsif falling_edge(Clock) then
            case State is
```

```
                when Take =>
                    vBuff(15 downto 0) := Number;
                    vBuff(31 downto 16) := (others => '0');
                    I := 0;
                    State <= Shift;
                when Shift =>
                    vBuff := vBuff(30 downto 0) & '0';
                    if I = 15 then
                        State <= Provide;
                    else
                        State <= Add;
                    end if;
                when Add =>
                    if vBuff(19 downto 16) > 4 then
                        vBuff(19 downto 16) := vBuff(19 downto 16) + 3;
                    end if;
                    if vBuff(23 downto 20) > 4 then
                        vBuff(23 downto 20) := vBuff(23 downto 20) + 3;
                    end if;
                    if vBuff(27 downto 24) > 4 then
                        vBuff(27 downto 24) := vBuff(27 downto 24) + 3;
                    end if;
                    if vBuff(31 downto 28) > 4 then
                        vBuff(31 downto 28) := vBuff(31 downto 28) + 3;
                    end if;
                    I := I + 1;
                    State <= Shift;
                when Provide =>
                    BCD <= vBuff(31 downto 16);
            end case;
        end if;
    end process;

end Behavioral;
```

/home/raffy/Dev/MouseDecoder/MouseDecoder.srcs/sources_1/new/CounterUnit.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 03/19/2022 09:01:34 PM
-- Design Name:
-- Module Name: CommandUnit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use work.Mouse_Types.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity CounterUnit is
    Port(
        Reset:      in  STD_LOGIC;
        Clock:      in  STD_LOGIC;
        MouseClock: in  STD_LOGIC;
        MouseData:  in  STD_LOGIC;

        ReverseSw:  in  STD_LOGIC;
        DebugSw:    in  STD_LOGIC;
        LED:        out STD_LOGIC_VECTOR(15 downto 0);

        Segments:   out STD_LOGIC_VECTOR(6 downto 0);
        Anodes:     out STD_LOGIC_VECTOR(3 downto 0)
    );
end CounterUnit;

architecture Behavioral of CounterUnit is

component SSGDisplay is
    Port(
        Clock:      in STD_LOGIC;
```

```
        Number:     in STD_LOGIC_VECTOR(15 downto 0);
        Segments:   out STD_LOGIC_VECTOR(6 downto 0);
        Anodes:     out STD_LOGIC_VECTOR(3 downto 0)
    );
end component;

component MouseDecoder is
    Port(
        Reset:          in  STD_LOGIC;
        Clock:          in  STD_LOGIC;
        MouseClock:     in  STD_LOGIC;
        MouseData:      in  STD_LOGIC;
        MouseMessage:   out Mouse_Message;
        NewMessage:     out STD_LOGIC
    );
end component;

signal Number:      STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
signal M: Mouse_Message;
signal NewMessage: STD_LOGIC;
signal X, Y: STD_LOGIC := '0';

begin
    Count_Clicks: process(Reset, NewMessage, M, Number)
    variable Temp: STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
    begin
        if Reset = '1' then
            Number <= (others => '0');
            Temp := (others => '0');
            X <= '0';
            Y <= '0';
        elsif falling_edge(NewMessage) then
            if ReverseSw = '0' then
                if M.LeftClick = '0' and X = '1' then
                    Temp := Temp + 1;
                end if;
                if M.RightClick = '0' and Y = '1' and Temp > 0 then
                    Temp := Temp - 1;
                end if;
            else
                if M.LeftClick = '0' and X = '1' and Temp > 0 then
                    Temp := Temp - 1;
                end if;
                if M.RightClick = '0' and Y = '1' then
                    Temp := Temp + 1;
                end if;
            end if;
            X <= M.LeftClick;
            Y <= M.RightClick;
            Number <= Temp;
        end if;
    end process;

    LED_Control: process(Reset, Clock, DebugSw, ReverseSw, M, Number)
    begin
        if Reset = '1' then
            LED <= (others => '1');
```

```
        elsif falling_edge(Clock) then
            LED(15) <= ReverseSw;       -- Leftmost LED corresponds to IS_REVERSED
            if DebugSw = '1' then
                LED(14) <= '0';
                LED(13) <= M.LeftClick;
                LED(12) <= M.MiddleClick;
                LED(11) <= M.RightClick;
                LED(10) <= '0';
                LED(9) <= M.OverflowX;
                LED(8) <= M.OverflowY;

                if M.X = 0 then LED(7) <= '0'; else LED(7) <= '1'; end if;
                if M.Y = 0 then LED(6) <= '0'; else LED(6) <= '1'; end if;
                if M.Z = 0 then LED(5) <= '0'; else LED(5) <= '1'; end if;

                LED(4) <= '0';

                LED(3 downto 0) <= Number(3 downto 0);
            else
                LED(14 downto 0) <= (others => '0');
            end if;
        end if;
    end process;

    Decode_Message: MouseDecoder port map (
        Reset => Reset,
        Clock => Clock,
        MouseClock => MouseClock,
        MouseData => MouseData,
        MouseMessage => M,
        NewMessage => NewMessage
    );

    Display_Number: SSGDisplay port map (Clock, Number, Segments, Anodes);
end Behavioral;
```

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 04/14/2022 01:10:02 PM
6  -- Design Name:
7  -- Module Name: MouseDecoder - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use Work.Mouse_Types.ALL;
25
26 -- Uncomment the following library declaration if using
27 -- arithmetic functions with Signed or Unsigned values
28 --use IEEE.NUMERIC_STD.ALL;
29
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34
35 entity MouseDecoder is
36     Port(
37         Reset:          in  STD_LOGIC;
38         Clock:          in  STD_LOGIC;
39         MouseClock:     in  STD_LOGIC;
40         MouseData:      in  STD_LOGIC;
41         MouseMessage:   out Mouse_Message;
42         NewMessage:     out STD_LOGIC
43     );
44 end MouseDecoder;
45
46 architecture Behavioral of MouseDecoder is
47 signal MouseBits:   NATURAL := 0;
48 signal MouseReg:    STD_LOGIC_VECTOR(42 downto 0) := (others => '0');
49 signal Trigger:     STD_LOGIC;
50 begin
51     Count_Bits: process(Reset, MouseClock)
52     begin
53         if Reset = '1' then
54             MouseBits  <= 0;
55         elsif rising_edge(MouseClock) then
56             -- Counter modulo 43
```

```
            if MouseBits <= 42 then
                MouseBits <= MouseBits + 1;
            else
                MouseBits <= 0;
            end if;
        end if;
    end process;

    Shift_And_Sync: process(Reset, MouseClock)
    begin
        if Reset = '1' then
            MouseReg <= (others => '0');
        elsif falling_edge(MouseClock) then
            Trigger <= '0';
            MouseReg <= MouseReg(41 downto 0) & MouseData;
            if MouseBits = 43 then
                if IsMouseDataValid(MouseReg) then
                    MouseMessage <= ParseMouseData(MouseReg);
                    Trigger <= '1';
                end if;
            end if;
        end if;
    end process;

    Pulse_Gen: process(Reset, Clock)
    variable Idle: Boolean := true;
    begin
        if Reset = '1' then
            Idle := true;
        elsif rising_edge(Clock) then
            NewMessage <= '0';
            if Idle then
                if Trigger = '1' then
                    NewMessage <= '1';
                    Idle := false;
                end if;
            else
                if Trigger = '0' then
                    Idle := true;
                end if;
            end if;
        end if;
    end process;

end Behavioral;
```

/home/raffy/Dev/MouseDecoder/MouseDecoder.srcs/sources_1/new/MouseTypes.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/05/2022 12:51:38 PM
-- Design Name:
-- Module Name: MouseTypes -
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package Mouse_Types is

    type Mouse_Message is record
        LeftClick:  STD_LOGIC;
        RightClick: STD_LOGIC;
        MiddleClick:STD_LOGIC;
        OverflowX:  STD_LOGIC;
        OverflowY:  STD_LOGIC;

        X: STD_LOGIC_VECTOR(8 downto 0);
        Y: STD_LOGIC_VECTOR(8 downto 0);
        Z: STD_LOGIC_VECTOR(8 downto 0);
    end record;

    function ParseMouseData(signal Buf: STD_LOGIC_VECTOR(42 downto 0)) return
Mouse_Message;
    function IsMouseDataValid(signal Buf: STD_LOGIC_VECTOR(42 downto 0)) return
Boolean;

end package;

package body Mouse_Types is
    function ParseMouseData(signal Buf: STD_LOGIC_VECTOR(42 downto 0)) return
Mouse_Message is
    variable Msg: Mouse_Message;
    begin
        Msg.LeftClick   := Buf(41);
        Msg.RightClick  := Buf(40);
        Msg.MiddleClick := Buf(39);
        -- Bit #3 <==> #38 is always 1
        Msg.X(8) := Buf(37);
        Msg.Y(8) := Buf(36);
```

```
        Msg.OverflowX   := Buf(35);
        Msg.OverflowY   := Buf(34);
        -- Parity, Stop, Start

        Msg.X(7 downto 0) := Buf(30 downto 23);
        -- Parity, Stop, Start
        Msg.Y(7 downto 0) := Buf(19 downto 12);
        -- Parity, Stop, Start
        Msg.Z := Buf(8 to 1);
        -- Parity bit

        return Msg;
    end function;

    function ParityOf(signal Buf: STD_LOGIC_VECTOR) return STD_LOGIC is
    variable P: STD_LOGIC := '1';
    begin
        for I in Buf'RANGE loop
            P := P xor Buf(I);
        end loop;
        return P;
    end function;

    function IsMouseDataValid(signal Buf: STD_LOGIC_VECTOR(42 downto 0)) return Boolean
is
    begin
        if Buf(42) /= '0' then return false; end if;

        if Buf(38) /= '1' then return false; end if;
        if ParityOf(Buf(41 downto 34)) /= Buf(33) then return false; end if;
        if Buf(32) /= '1' or Buf(31) /= '0' then return false; end if;

        if ParityOf(Buf(30 downto 23)) /= Buf(22) then return false; end if;
        if Buf(21) /= '1' or Buf(20) /= '0' then return false; end if;

        if ParityOf(Buf(19 downto 12)) /= Buf(11) then return false; end if;
        if Buf(10) /= '1' or Buf(9) /= '0' then return false; end if;

        if ParityOf(Buf(8 downto 1)) /= Buf(0) then return false; end if;

        return true;
    end function;
end;
```

/home/raffy/Dev/MouseDecoder/MouseDecoder.srcs/sources_1/new/SSGDisplay.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 03/15/2022 01:22:28 PM
-- Design Name:
-- Module Name: SSGDisplay - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use iEEE.numeric_std.all;

entity SSGDisplay is
    Port(
        Clock:      in STD_LOGIC;
        Number:     in STD_LOGIC_VECTOR (15 downto 0);
        Segments:   out STD_LOGIC_VECTOR(6 downto 0);
        Anodes:     out STD_LOGIC_VECTOR(3 downto 0)
    );
end SSGDisplay;

architecture Behavioral of SSGDisplay is

signal Digit: STD_LOGIC_VECTOR(3 downto 0);
-- 1 / (100MHz / 2^20) = 10.48ms refresh rate
signal Refresh: STD_LOGIC_VECTOR(19 downto 0);
-- 2 bits for the 4 to 1 MUX
signal NextAnode: STD_LOGIC_VECTOR(1 downto 0);

component Bin2Bcd is
    Port(
        Clock:  in  STD_LOGIC;
        Reset:  in  STD_LOGIC;
        Number: in  STD_LOGIC_VECTOR(15 downto 0);
        BCD:    out STD_LOGIC_VECTOR(15 downto 0)
    );
end component Bin2Bcd;

signal ResetB2B:    STD_LOGIC;
signal BCD:         STD_LOGIC_VECTOR(15 downto 0);

begin
    Segment_Pattern: process(Digit)
    begin
```

```
        case Digit is
            when "0000" => Segments <= "0000001"; -- 0
            when "0001" => Segments <= "1001111"; -- 1
            when "0010" => Segments <= "0010010"; -- 2
            when "0011" => Segments <= "0000110"; -- 3
            when "0100" => Segments <= "1001100"; -- 4
            when "0101" => Segments <= "0100100"; -- 5
            when "0110" => Segments <= "0100000"; -- 6
            when "0111" => Segments <= "0001111"; -- 7
            when "1000" => Segments <= "0000000"; -- 8
            when "1001" => Segments <= "0000100"; -- 9
            when others => Segments <= "0000001"; -- 0
        end case;
    end process;

    Refresh_Interval: process(Clock)
    begin
        if rising_edge(Clock) then
            Refresh <= Refresh + 1;
        end if;
    end process;

    -- Select the next anode pattern
    Anode_Mux: NextAnode <= Refresh(19 downto 18);

    ResetB2B <= Refresh(18);

    Binary_To_BCD: Bin2Bcd port map(
        Number => Number,
        Clock => Clock,
        Reset => ResetB2B,
        BCD => BCD
    );

    Digit_Selection: process(NextAnode, BCD)
    begin
        -- Select the actual digit to show
        case NextAnode is
            when "00" =>    -- First digit
                Anodes <= "0111";
                Digit <= BCD(15 downto 12);
            when "01" =>    -- Second digit
                Anodes <= "1011";
                Digit <= BCD(11 downto 8);
            when "10" =>    -- Third digit
                Anodes <= "1101";
                Digit <= BCD(7 downto 4);
            when "11" =>    -- Fourth digit
                Anodes <= "1110";
                Digit <= BCD(3 downto 0);
            end case;
    end process;

end Behavioral;
```