Övningsuppgifter 2025-04-11

1. Du ska konstruera ett digitalt system innehållande två 7-segmentsdisplayer samt åtta slide-switchar. På respektive 7-segmentsdisplay ska ett hexadecimalt tal 0 – F skrivas ut. Talet som skrivs ut ska matas in binärt via fyra slide-switchar. För konstruktionen behövs därmed åtta slide-switchar (fyra för respektive 7-segmentsdisplay).

För att inte behöva skriva samma kod två gånger ska du skapa en delkomponent, dvs. en till modul i vårt VHDL-projekt, för att kunna kontrollera vad som skrivs ut på en 7-segmentsdisplay genom att ange ett binärt tal 0000 - 1111. Du ska sedan skapa två instanser av denna delkomponent i toppmodulen för att styra respektive 7-segmentsdisplay.

Projektet ska döpas *hex_display* och delkomponenten ska döpas *display.* En visuell överblick över konstruktionen på komponentnivå finns i bilaga A på nästa sida. Motsvarande hårdvarubeskrivande kod skriven i SystemVerilog finns i bilaga C eller kan laddas ned här.

Tips: *Ladda ned och öppna .qar-filen för motsvarande projekt i SystemVerilog via länken ovan, kompilera och testa sedan konstruktionen på ett FPGA-kort, så får ni en överblick över hur den ska fungera.*

a) Skapa ett projekt döpt *hex_display* i Quartus. Välj FPGA-kort Terasic DE0 (enhet 5CEBA4F23C7).
Lägg till toppmodulen *hex_display* innehållande följande portar:
- **input[7:0] :** Insignaler från slide-switchar.
- **hex1[6:0] & hex0[6:0]:** Utsignaler till var sin 7-segmentsdisplay.

Hårdvaran ska implementeras så att:
- Det 4-bitars binära tal som matas in via slide-switchar input[7:3] skrivs ut hexadecimalt på hex1.
- Det 4-bitars binära tal som matas in via slide-switchar input[3:0] skrivs ut hexadecimalt på hex0.

Efter att du har lagt till *entity* och *architecture,* kompilera koden och korrigera eventuella fel. Öppna sedan *Pin Planner* och anslut portarna enligt nedan (se databladet för FPGA-kort Terasic DE0 för PIN-nummer):
- Anslut input[7:0] till slide-switchar SW[7:0].
- Anslut hex1 till 7-segmentsdisplay HEX1[6:0] samt hex0 till 7-segmentsdisplay HEX0[6:0].

b) Skapa en delkomponent döpt *display* i en fil döpt *display.vhd.* Denna delkomponent ska kunna användas för att kunna styra en 7-segmentsdisplay genom att ange ett 4-bitars binärt tal. Använd följande portar:
- **number[3:0]:** Insignal som utgörs av det tal som ska skrivas ut på ansluten 7-segmentsdisplay.
- **hex[6:0] :** Utsignal som tilldelas binärkoden för motsvarande tal (vilket skrivs till ansluten display).

Talet som ska skrivas ut på 7-segmentsdisplayen ska anges på 4-bitars binär form och matas till number[3:0].
Detta tal ska då skrivas ut hexadecimalt på ansluten 7-segmentsdisplay. Som exempel:
- Om number[3:0] matas med talet 0111_2 ska binärkoden för talet 7_{16} skrivas till hex[6:0].
- Om number[3:0] matas med talet 1110_2 ska binärkoden för talet E_{16} skrivas till hex[6:0].

För att åstadkomma detta ska 7-segmentsdisplayen matas med motsvarande binärkod. Ni har tillgång till binärkoder för siffror 0x0 – 0x9 i bilaga B, resten (för 0xA – 0xF) måste ni lägga till själva.

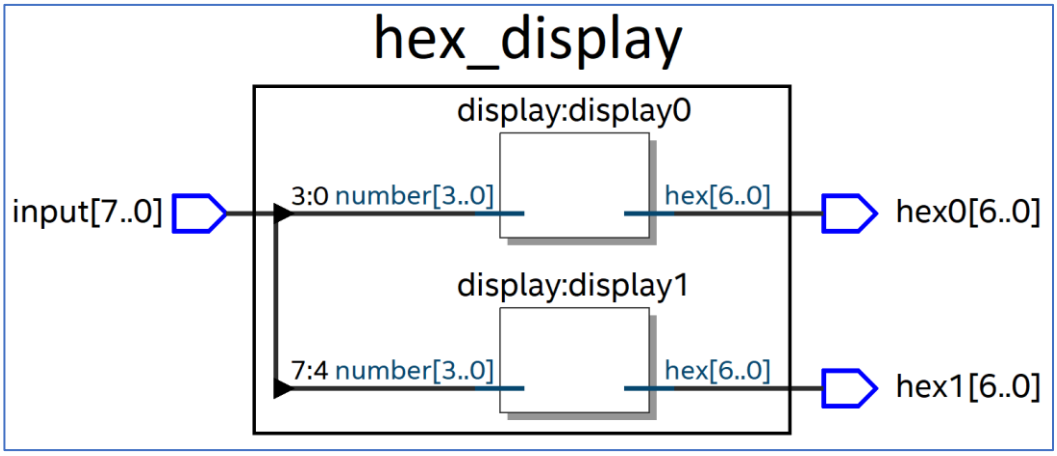
Tips: Använd en case-sats, som exekverar vid förändring av input[3:0].

c) Skapa två instanser av delkomponenten *display* i toppmodulen. Döp instanserna till display1 respektive display0. För respektive instans ska fyra 7-segmentsdisplayer anslutas till input[3:0] och en 7-segmentsdisplay anslutas till hex[6:0]:
- Anslut switch[7:4] samt hex1 till instansen display1.
- Anslut switch[3:0]] samt hex0 till instansen display0.

d) Verifiera konstruktionen på ett FPGA-kort.

e) Skapa en testbänk för modulen *display* och döp denna *display_tb.* Testa samtliga 16 kombinationer 0000 – 1111 av *number* och kontrollera att binärkoden som skrivs till *hex* är korrekt (jämför med de framtagna binärkoderna).

Bilaga A – Kretsschema på komponentnivå



Figur 1:Kretsschema som presenterar konstruktionen innehåll som separata komponenter.

Bilaga B - Binärkoder för 7-segmentsdisplayerna

Notering: Binärkoderna bör förslagsvis implementeras i form konstanter, som placeras i arkitekturen tillhörande entiteten *display*.

OBS! Ni måste själva implementera binärkoder för A – F.

```
--------------------------------------------------------------------------------
-- @brief Binary codes for displaying digits 0x0 - 0xF on hex displays.
--------------------------------------------------------------------------------
constant DISPLAY_0  : std_logic_vector(6 downto 0) := "1000000";
constant DISPLAY_1  : std_logic_vector(6 downto 0) := "1111001";
constant DISPLAY_2  : std_logic_vector(6 downto 0) := "0100100";
constant DISPLAY_3  : std_logic_vector(6 downto 0) := "0110000";
constant DISPLAY_4  : std_logic_vector(6 downto 0) := "0011001";
constant DISPLAY_5  : std_logic_vector(6 downto 0) := "0010010";
constant DISPLAY_6  : std_logic_vector(6 downto 0) := "0000010";
constant DISPLAY_7  : std_logic_vector(6 downto 0) := "1111000";
constant DISPLAY_8  : std_logic_vector(6 downto 0) := "0000000";
constant DISPLAY_9  : std_logic_vector(6 downto 0) := "0010000";
-- Add DISPLAY_A - DISPLAY_F here!
constant DISPLAY_OFF: std_logic_vector(6 downto 0) := "1111111";
```

Bilaga C – Implementering av projektet *hex_display* i SystemVerilog

Filen *hex_display.sv*

```
/*****************************************************************************
 * @brief System for displaying two hexadecimal numbers 0 - F via hex displays.
 *        The number displayed on each display is entered in 4-bit binary form
 *        via four slide switches.
 *
 * @param inputs[in] Eight slide-switches used for entering two 4-bit numbers.
 * @param hex1[out]  The first hex display, controlled by inputs[7:4].
 * @param hex0[out]  The second hex display, controlled by inputs[3:0].
 *****************************************************************************/
module hex_display(input logic[7:0] inputs,
                   output logic[6:0] hex1, hex0);

    /*************************************************************************
     * @brief Implements hardware for hex1, which is controlled by inputs[7:4].
     *************************************************************************/
    display display1(inputs[7:4], hex1);

    /*************************************************************************
     * @brief Implements hardware for hex0, which is controlled by inputs[3:0].
     *************************************************************************/
    display display0(inputs[3:0], hex0);

endmodule
```

OBS! Vänd blad!

Filen *display.sv*

```
/*****************************************************************************
 * @brief Component for displaying a hexadecimal digit 0 - F on a hex display.
 *
 * @param number The number to display (in 4-bit binary form).
 * @param hex    Hex display used for displaying the corresponding digit.
 *****************************************************************************/
module display(input logic[3:0] number,
               output logic[6:0] hex);

    /*************************************************************************
     * @brief Binary codes for displaying digits 0x0 - 0xF on hex displays.
     *************************************************************************/
    const logic[6:0] DISPLAY_0   = 7'b1000000;
    const logic[6:0] DISPLAY_1   = 7'b1111001;
    const logic[6:0] DISPLAY_2   = 7'b0100100;
    const logic[6:0] DISPLAY_3   = 7'b0110000;
    const logic[6:0] DISPLAY_4   = 7'b0011001;
    const logic[6:0] DISPLAY_5   = 7'b0010010;
    const logic[6:0] DISPLAY_6   = 7'b0000010;
    const logic[6:0] DISPLAY_7   = 7'b1111000;
    const logic[6:0] DISPLAY_8   = 7'b0000000;
    const logic[6:0] DISPLAY_9   = 7'b0010000;
    const logic[6:0] DISPLAY_A   = 7'b0001000;
    const logic[6:0] DISPLAY_B   = 7'b0000011;
    const logic[6:0] DISPLAY_C   = 7'b1000110;
    const logic[6:0] DISPLAY_D   = 7'b0100001;
    const logic[6:0] DISPLAY_E   = 7'b0000110;
    const logic[6:0] DISPLAY_F   = 7'b0001110;
    const logic[6:0] DISPLAY_OFF = 7'b1111111;

    /*************************************************************************
     * @brief Sets the binary code of the hex display depending on the input number.
     *        This process is run at every change of the input number.
     *************************************************************************/
    always_comb
    begin: output_process
        case (number)
            4'b0000: hex <= DISPLAY_0;
            4'b0001: hex <= DISPLAY_1;
            4'b0010: hex <= DISPLAY_2;
            4'b0011: hex <= DISPLAY_3;
            4'b0100: hex <= DISPLAY_4;
            4'b0101: hex <= DISPLAY_5;
            4'b0110: hex <= DISPLAY_6;
            4'b0111: hex <= DISPLAY_7;
            4'b1000: hex <= DISPLAY_8;
            4'b1001: hex <= DISPLAY_9;
            4'b1010: hex <= DISPLAY_A;
            4'b1011: hex <= DISPLAY_B;
            4'b1100: hex <= DISPLAY_C;
            4'b1101: hex <= DISPLAY_D;
            4'b1110: hex <= DISPLAY_E;
            4'b1111: hex <= DISPLAY_F;
            default: hex <= DISPLAY_OFF;
        endcase
    end
endmodule
```

OBS! Vänd blad!

Filen *display_tb.sv*

```
/*****************************************************************************
 * @brief Test bench for the display module. Each combination 0000 - 1111 of
 *        the number to display is tested during 10 ns, hence the simulation
 *        time is 160 ns.
 *
 *        The correlation between the input number and the binary code
 *        displayed on the corrected hex display is shown below:
 *
 *        -------------------
 *        | input |  code   |
 *        |-----------------|
 *        | 0000  | 1000000 |
 *        | 0001  | 1111001 |
 *        | 0010  | 0100100 |
 *        | 0011  | 0110000 |
 *        | 0100  | 0011001 |
 *        | 0101  | 0010010 |
 *        | 0110  | 0000010 |
 *        | 0111  | 1111000 |
 *        | 1000  | 0000000 |
 *        | 1001  | 0010000 |
 *        | 1010  | 0001000 |
 *        | 1011  | 0000011 |
 *        | 1100  | 1000110 |
 *        | 1101  | 0100001 |
 *        | 1110  | 0000110 |
 *        | 1111  | 0001110 |
 *        | other | 0000000 |
 *        -------------------
 *****************************************************************************/
module display_tb();

    /*************************************************************************
     * @brief Signals used to test the display module.
     *************************************************************************/
     logic[3:0] number = 4'b0;
     logic[6:0] hex    = 7'b0;

     /*************************************************************************
     * @brief Creates an instance of the display module and connects signals with
     *        the same name as the corresponding ports for simulation.
     *************************************************************************/
     display sim_instance(number, hex);

     /*************************************************************************
     * @brief Tests each combination of the inputs during 10 ns each.
     *************************************************************************/
     initial
     begin: sim_process
         for (int i = 0; i < 16; ++i)
         begin
             number = i;
             #10ns;
         end
     end

endmodule
```