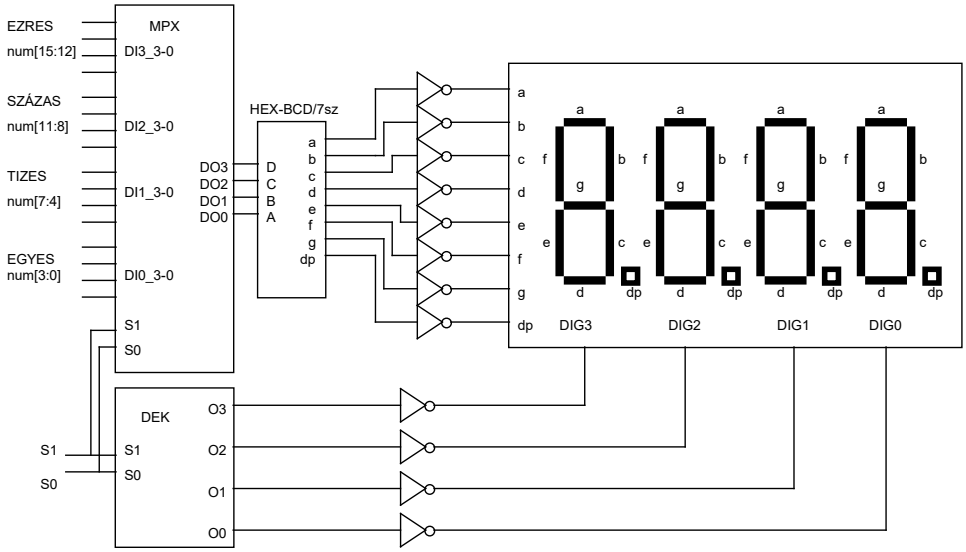
Funkcionális egységek használata a tervezésben

F1. A laborban használt LOGSYS SP3E FPGA kártyán található 4 digites 7 szegmenses kijelzőt kívánjuk számkijelzéses módban használni. A 4 digites számjegy lehet hexadecimális, vagy BCD kódolású. A kijelzés vezérlést ütemező 1kHz frekvenciával ciklikusan ismétlődő 2 bites bináris S[1:0] = 00-01-10-11 kódolású jelsorozat rendelkezésre áll. Tervezzük meg a kijelzést, ha a bemenet egy 16 bites, 4 digites numerikus érték! Ne felejtsük el a kijelző vezérlőjelek negatív logikájú meghajtását sem!

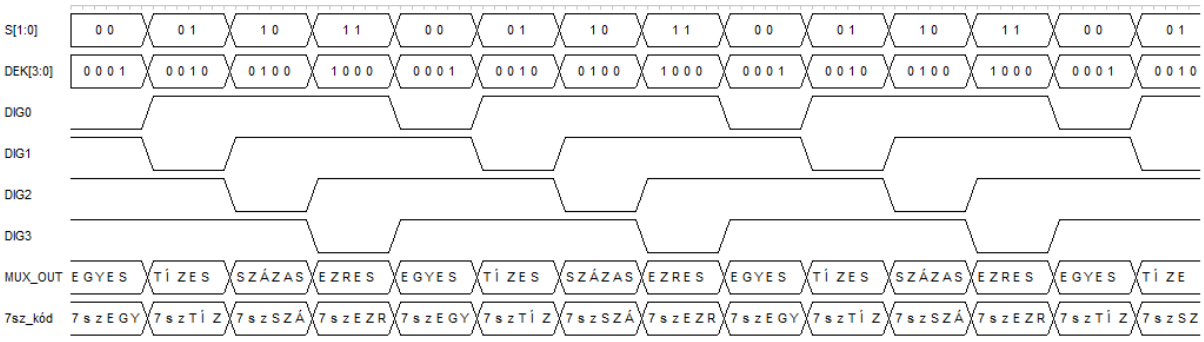
F1.a Először rajzoljuk fel a teljes rendszer blokkvázlatát! Milyen funkcionális egységeket használjunk a rendszerben?

Az ismert kombinációs fukcionális egységeket használjuk. Általános célú egységek a dekóder, multiplexer, ezek paraméterei (bemenetek, kimenetek, engedélyezhetőség) megválaszthatók. Továbbá beépíthetünk olyan alkalmazás specifikus áramköröket, mint pl. egy hexadecimális számértékből 7 szegmenses kijelző kódot előállító modul. Amennyiben szükséges, elemi logikai egységek (pl. inverter) is beépíthető. A blokkvázlat tulajdonképpen a felépítési tervünk a teljes digitális áramkörről.



F1.b. A helyes működéshez azonos időben kell kiválasztanunk egy numerikus számjegyet (pl. BCD kódolásnál az ezres, százas, tízes, egyes helyiértéket) a 4 digites 16 bites értékből, és ugyanekkor a megfelelő pozíciójú digit aktiválásáról is gondoskodni kell. Tervezzük meg a számjegyek közül választó egységet és a digitek kiválasztását vezérlő modult.

A fenti ábra alapján és előzetes sejtéseink szerint is a számjegyek kiválasztását egy 4 bites, 4:1-be (4 db 4 bites adatbemenetű, 1 db 4 bites adatkimenetű) busz multiplexer tudja biztosítani. Ha az S[1:0] kijelzés ütemező kétbites jelsorozat 00 értéke az EGYES, 01 értéke a TÍZES, 10 értéke a SZÁZAS és 11 értéke az EZRES helyiértéket választja ki, akkor a MUX bemeneteire ilyen sorrendben kell bevezetni a num[15:0] 16 bites érték azon 4 bites részeit, amelyek az egyes számjegyeket kódolják (fenti ábrán EGYES = num[3:0], TÍZES = num[7:4]……). A jelek időbeli változásait az alábbi diagram szemlélteti. A 4 digites kijelzőt a negált logikájú 8 bites 7sz_kód[a,b,c,d,e,f,g,dp] és 4 bites DIG[3:0] vezérli.




```
////////////////////////////////////////////////////////////////////////////////
// 4 bites 4:1-be buszmultiplexer, azaz nem egyedi jelek,
// hanem több bites (4) bitvektorok közül választ egyet
////////////////////////////////////////////////////////////////////////////////

module BUS_MUX_4_4_1(
    input      [15:0] num,                  // 16 bites, 4 digites számérték
    input       [1:0] s,                    // Kijelzés ütemező jel
    output reg  [3:0] digit                 // A kiválasztott számjegy
    );

always @(*)
   case (s)                                 // s a két bites kiválasztó érték
      2'b00:   digit <= num[ 3: 0];         // Ha s=0, a kimenet EGYES
      2'b01:   digit <= num[ 7: 4];         // Ha s=1, a kimenet TÍZES
      2'b10:   digit <= num[11: 8];         // Ha s=2, a kimenet SZÁZAS
      2'b11:   digit <= num[15:12];         // Ha s=3, a kimenet EZRES
   endcase

endmodule
```

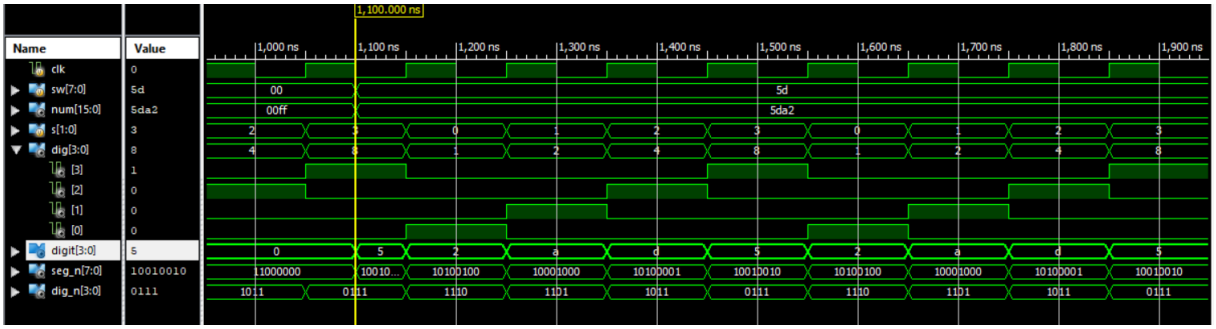
Ugyanezen ütemben működtethető a 7 szegmenses kijelző digitjeinek vezérlését végző 2:4-be bináris dekóder is.

```
////////////////////////////////////////////////////////////////////////////////
// 2:4 dekóder, ami a kijelző EZRES-TÍZES-SZÁZAS-EGYES pozícióit kiválasztja
////////////////////////////////////////////////////////////////////////////////

module DEK_2_4(
    input       [1:0] s,                    // Kijelzés ütemező jel
    output reg  [3:0] dig                   // A kiválasztott helyiérték pozíció
    );

always @(*)
   case (s)
      2'b00:   dig <= 4'b0001;              // Ha s=0, az EGYES kimenet kapcsol be
      2'b01:   dig <= 4'b0010;              // Ha s=1, az TÍZES kimenet kapcsol be
      2'b10:   dig <= 4'b0100;              // Ha s=2, az SZÁZAS kimenet kapcsol be
      2'b11:   dig <= 4'b1000;              // Ha s=3, az EZRES kimenet kapcsol be
   endcase

endmodule
```

F1.c. A numerikus kódok közvetlenül nem alkalmasak a 7 szegmenses kijelzőn számjegyek megjelenítésére. Szükség van egy hexadecimális → 7 szegmenses kódátalakítóra. Tervezzük meg ezt a 4 bemenetű, 7 (vagy 8) kimenetű áramkört! Milyen formában érdemes elkészíteni a specifikációt?

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// HEXA - 7 szegmenses dekóder
//////////////////////////////////////////////////////////////////////////////////

module HEX_7SEG(
    input       [3:0] digit,                // Kijelzendő számjegy numerikus kódja
    output      [7:0] seg                   // A kijelzendő szegmenskód
    );

//////////////////////////////////////////////////////////////////////////////////
// A kiválasztott számértéket 7 szegmenses kijelző kóddá konvertáljuk
// Az átkódoló logikát pozitív logika szerint írjuk fel,
//////////////////////////////////////////////////////////////////////////////////

reg [6:0] segment;                          // Kimeneti bitvektor változó deklaráció
wire decp = 1'b0;                           // A tizedespont nem aktív

always @(*)                                 // Kódkonverter kombinációs hálózat,
   begin                                    // Lényegében egy 16*7 bites táblázat (ROM memória)
      case (digit)              //gfedcba  szegmensek
         4'b0000 : segment = 7'b0111111;  // 0
         4'b0001 : segment = 7'b0000110;  // 1
         4'b0010 : segment = 7'b1011011;  // 2
         4'b0011 : segment = 7'b1001111;  // 3
         4'b0100 : segment = 7'b1100110;  // 4
         4'b0101 : segment = 7'b1101101;  // 5
         4'b0110 : segment = 7'b1111101;  // 6
         4'b0111 : segment = 7'b0000111;  // 7
         4'b1000 : segment = 7'b1111111;  // 8
         4'b1001 : segment = 7'b1101111;  // 9  Idáig a decimális számjegyek
         4'b1010 : segment = 7'b1110111;  // A  Innen a hexadecimális kiegészítés
         4'b1011 : segment = 7'b1111100;  // b
         4'b1100 : segment = 7'b0111001;  // C
         4'b1101 : segment = 7'b1011110;  // d
         4'b1110 : segment = 7'b1111001;  // E
         4'b1111 : segment = 7'b1110001;  // F
      endcase
   end

assign seg = {decp,segment};        // Tizedes pont és szegmenskód együtt a kimeneti teljes kód

endmodule
```

A fenti modulokat megfelelően összekapcsolva, a KIJELZŐ egység működési szimulációja az alábbi. A bemeneti adat a sárga kurzor jelzéstől SW=8'h5d, ez 16 bitre kiegészítve (az adatbitek invertálásával 8'h5d -> INV -> 8'ha2) num = 16'h5da2, a kijelző ütemezés s[1:0] vezérlő jele az órajel ütemében számol 0-1-2-3-0-1-2… és ebből a DEK_2_4 modul generálja az 1-a-4-ből kódolású dig jelet. Amikor a dig egy-egy helyiértékel kiválaszt, akkor a digit 4 bites jel éppen azt a számjegyet tartalmazza, ezt a BUS_MUX_4_4_1 biztosítja. Ennek az értéknek a 7 szegmenses kódját pedig a HEX_7SEG konverter generálja. Ez nem látszik jól, mert a szimulátor ezt nem tudja visszakonvertálni, de pl. a seg_n értéke a T=1500ns időpontban seg_n=8'b10010010, amit ha vissza invertálunk és elhagyjuk a tizedespontot jelentő első bitet, éppen megkapjuk az 1101101 bitmintát, ami az 5 szegmensképe. (A kijelzés idősorrendje EGYES-TÍZES-SZÁZAS-EZRES.)



A teljes KIJELZŐ projekt top-level modulja a következő:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// GYAK_3 KIJELZŐ projekt top-level Verilog fájl
// Tartalmazza a számjegykiválasztó buszmultiplexert 4 számjegyre,
// a kiejlzést ütemező helyiértékpozíciót kiválasztó dekódert,
// egy 2 bites számlálót, ami 0-1-2-3-0-1-2-3-0 sorrendben számol (s jel)
// Az egész kijelző működését a clk órajel bemenet ütemezi (1Hz - 1000Hz)
// A kijelzendő értéket az SW kapcslókon állítjuk be, de mivel csak 8 kapcsoló
// van, a 8 bitet 2x használjuk. A normál érték adja az EZRES-SZÁZAS, a bitenként
// invertált érték a SZÁZAS-EGYES helyiértéken megjelenő számjegeket.
// Példa: Ha SW= 0101_1101, akkor a kijelzendő 16 bit  0101_1101__1010_0010,
// tehát a kijelzőn az 5dA2 érték fog megjelenni.
// Megjegyzés: A 7 szegmenses kijelzőkön alkalmazott szegmensképet a HEX_7SEG
// modul ismerteti
//////////////////////////////////////////////////////////////////////////////////
module KIJELZO(
    input clk,
    input [7:0] sw,
    output [7:0] seg_n,
    output [3:0] dig_n,
    output [4:0] col_n
    );

//////////////////////////////////////////////////////////////////////////////////////////
// Ez a részlet még nem ismert, de a működtetéshez szükséges 2 bites számláló,
// ami a vezérlő s jelet állítja elő, az órajel ütemében 0,1,2,3,0,1,2,3,0,1
//////////////////////////////////////////////////////////////////////////////////////////
wire [1:0] s;
CNT2   SZAMLALO(.clk(clk), .q(s));
//////////////////////////////////////////////////////////////////////////////////////////
// A 8 bites SW kapcsoló beállítása alapján generálunk egy 16 bites számértéket
//////////////////////////////////////////////////////////////////////////////////////////
wire [15:0] num;
assign num = {sw, ~sw};             // Az SW kapcsoló normál és invertált bitjei
//////////////////////////////////////////////////////////////////////////////////////////
// Belső jelek a rendszer felépítéséhez, a modulok összekapcsolásához
//////////////////////////////////////////////////////////////////////////////////////////
wire [3:0] digit;                   // A kiválasztott éppen kijelzendő számjegy
wire [3:0] dig;                     // Az éppen kijelzendő helyiérték (1-a-4-ből)kód
wire [7:0] seg;                     // A 7 szegmenses kijelzési kép (normál polaritás)
//////////////////////////////////////////////////////////////////////////////////////////
// Beépítjük a megtervezett modulokat. A beépítés leírási módja:
// MODUL_TÍPUSNEVE   MODUL_AZONOSÍTÓ(be és kimeneti jelek/portok felsorolása)
//////////////////////////////////////////////////////////////////////////////////////////
BUS_MUX_4_4_1  MPX(.num(num), .s(s), .digit(digit));
DEK_2_4        DEK(.s(s), .dig(dig));
HEX_7SEG       HEX_7SEG(.digit(digit), .seg(seg));
//////////////////////////////////////////////////////////////////////////////////////////
// A kijelzőhöz invertált vezérlés szükséges
//////////////////////////////////////////////////////////////////////////////////////////
assign seg_n = ~seg;
assign dig_n = ~dig;
assign col_n = 5'b11111;   // Mátrix kijelző letiltása

endmodule
```