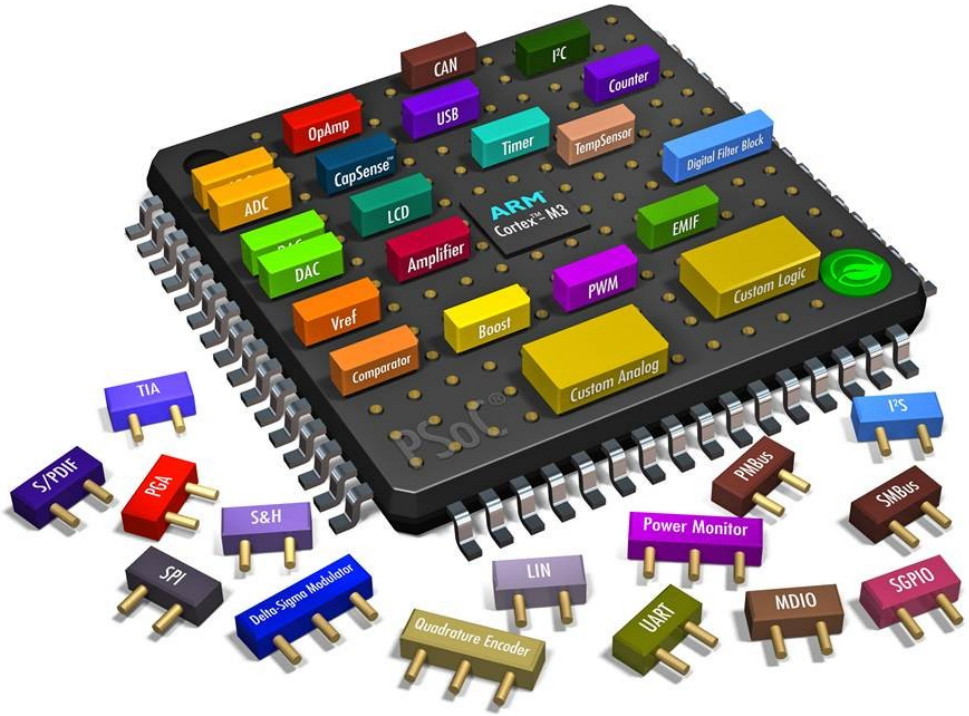
Újrakonfigurálható eszközök



7. Hétszegmenses LED kijelző multiplex vezérlése

Felhasznált irodalom és segédanyagok

- Icarus Verilog Simulator: http://iverilog.icarus.com/
- GtkWave wave viever: http://gtkwave.sourceforge.net/
- Verilog online tutorial: vol.verilog.com/VOL/main.htm
- Verilog tutorial for beginners: www.referencedesigner.com/tutorials/verilog/verilog_01.php
- University of Washington Computer Science & Engineering, CSE370 Lecture 23: Factoring FSMsDesign Examples
- ASIC world – Verilog tutorial: asic-world.com/verilog/veritut.html
- Végh János: Bevezetés a **Verilog** hardverleíró nyelvbe
- Végh János: Segédeszközök az **Altera DE2** tanulói készlethez
- Végh János: Bevezetés a **Quartus II V13** fejlesztő rendszerbe

Tartalom

- A **C-M240** fejlesztői kártya
- A felhasznált kivezetések listája

Mintaprogramok:

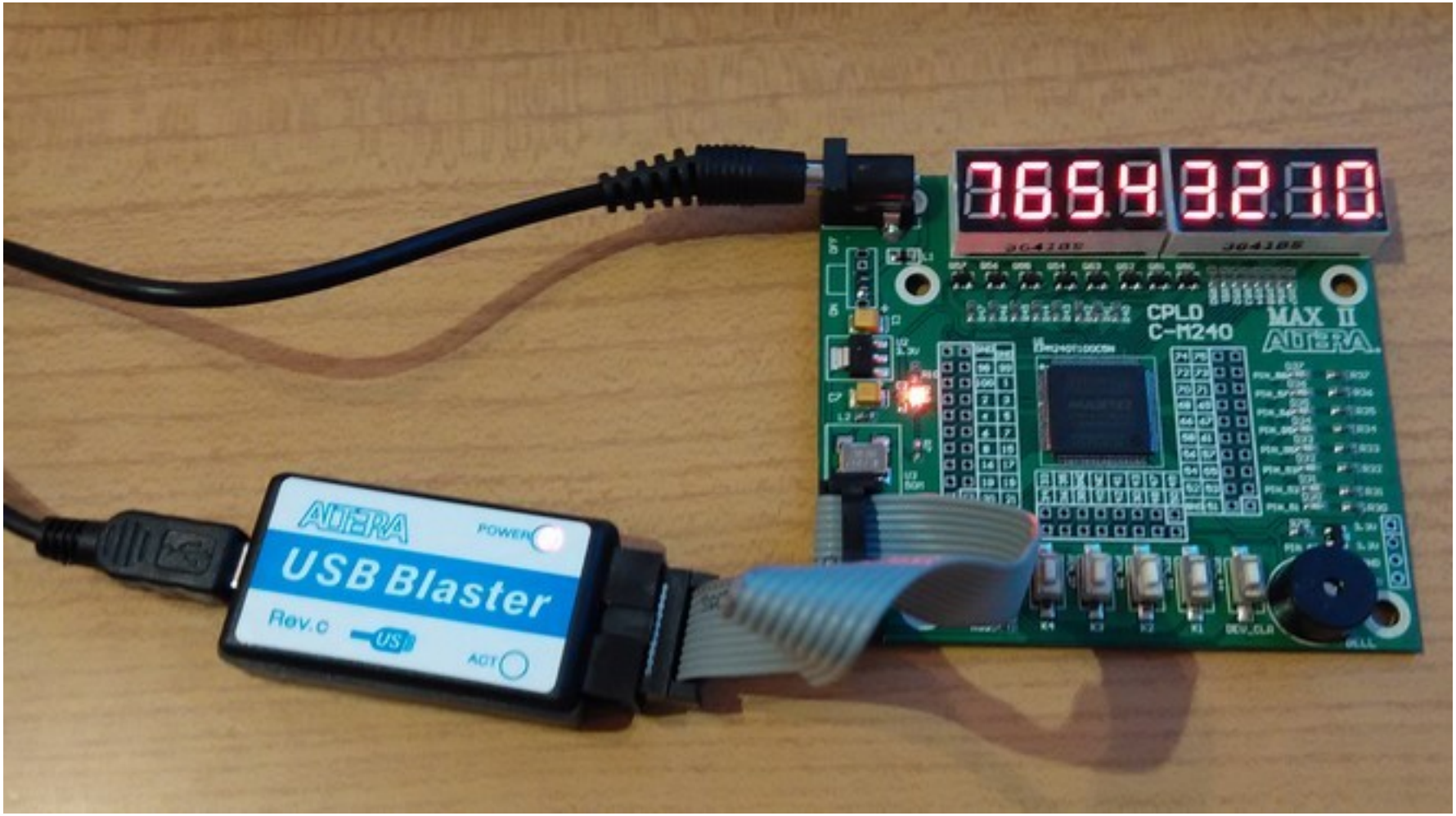
- **one_digit_display** – 1 számjegyű hexadecimális számláló
- **eight_digit_display** – 8 jegyű kijelzés időmultiplex verérlése
- **clock-display** – stopperóra hh-mm-ss kijelzéssel, időmultiplex vezérléssel és nyomógomb pergésmentesítéssel

C-M240 fejlesztői kártya

Gyártó: Shenzhen 21EDA Electronic Technology CPLD: Altera MAX II EPM240T100C5N

Órajel: 50 MHz VDD: 3,3 V Perifériák: 8-digit kijelző, 8 LED, 4 +1 nyomógomb, 1 csipogó

Fejlesztői környezet: **ALTERA QUARTUS Prime (Intel FPGA)**



C-M240 – felhasznált kivezetések

Bemenetek

CLK_50M az órajel bemenet
DEV_CLK a reset gomb

CLK_50M	input	PIN_12
K1 gomb	input	PIN_29
K2 gomb	input	PIN_28
K3 gomb	input	PIN_27
K4 gomb	input	PIN_26
DEV_CLK	input	PIN_44

Kimenetek

BELL	output	PIN_50
D30 LED8	output	PIN_51
D31 LED7	output	PIN_52
D32 LED6	output	PIN_53
D33 LED5	output	PIN_54
D34 LED4	output	PIN_55
D35 LED3	output	PIN_56
D36 LED2	output	PIN_57
D37 LED1	output	PIN_58

C-M240 – felhasznált kivezetések

Számjegyek

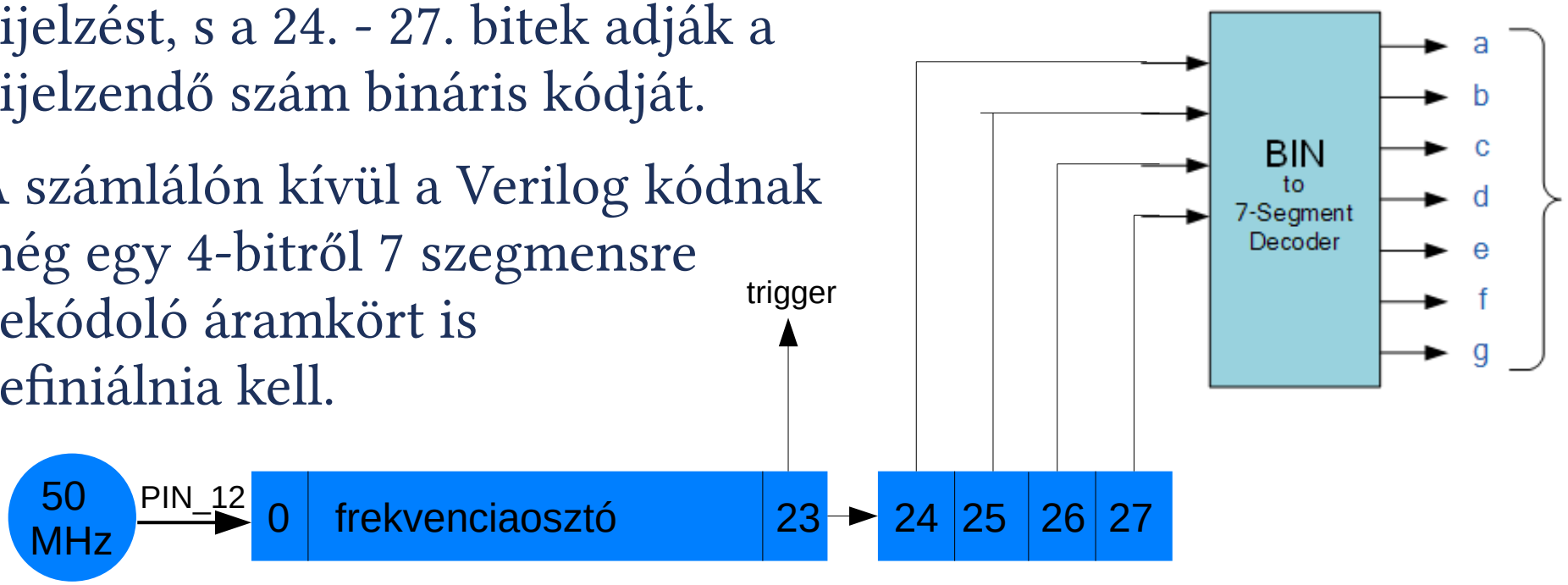
DIGIT1 (jobb szélső)	output	PIN_86
DIGIT2	output	PIN_87
DIGIT3	output	PIN_96
DIGIT4	output	PIN_89
DIGIT5	output	PIN_97
DIGIT6	output	PIN_91
DIGIT7	output	PIN_92
DIGIT8 (bal szélső)	output	PIN_95

Szegmensek

DATA0	output	PIN_85	A
DATA1	output	PIN_84	B
DATA2	output	PIN_83	C
DATA3	output	PIN_82	D
DATA4	output	PIN_81	E
DATA5	output	PIN_78	F
DATA6	output	PIN_77	G
DATA7	output	PIN_76	DP

one_digit_display: hexadecimális számláló

- Az 50 MHz-es rendszer órajelet egy 24 bites számlálóval leosztjuk
- A leosztott, kb. 3 Hz-es jelet egy 4 bites számlálóval számláljuk, s az eredményt hexadecimálisan kijelezzük egy 7 szegmenses LED kijelzőn
- A feladatot úgy is felfoghatjuk, hogy egy 28 bites (0 – 27. bitek) bináris számlálóláncot használunk, melynek 23. bitje vezérli a kijelzést, s a 24. - 27. bitek adják a kijelzendő szám bináris kódját.
- A számlálón kívül a Verilog kódnak még egy 4-bitről 7 szegmensre dekódoló áramkört is definiálnia kell.



one_digit_display: hexadecimális számláló

- A C-M240 kártya számkijelzőjén a számjegyek szegmensei párhuzamosítva vannak, ezért egyszerre, egyidejűleg csak egy számjegy jeleníthető meg
- A szegmensek vezérlésével szinkronban a megfelelő számjegy anódját is aktív állapotba kell helyezni
- Ebben a mintaprogramban csak a jobb szélső számjegyet használjuk, ezért állandóan annak anódját tesszük aktívvá

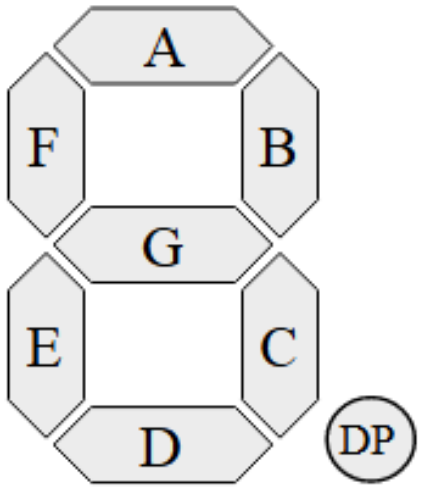
```
module display (clk_50M, digit, segment);
  input clk_50M;                    // 50MHz órajel (pin12)
  output wire [7:0] digit;          // számjegyek (anódok) vezérlése
  output reg [7:0] segment;         // DP és gfedcba szegmensek vezérlése
  reg [27:0] count;                 // 24 + 4 bites számláló

  assign digit = 8'b11111110;       // mindig az 1. számjegy aktív!

  always @ (posedge clk_50M)        // minden óraütéskor
    count <= count + 1;             // léptetjük a számlálót
```

Folytatás a következő oldalakon...

hexadecimális kijelző 7 szegmenssel

- Négybites számokkal a 7 szegmenses kijelzőt az alábbi táblázat alapján vezéreljük. A közös anódú módban a katódokat vezéreljük, így „0” jelszint esetén világít a szegmens.

Bemenetek				7 szegmens konfiguráció	Bemenetek				7 szegmens konfiguráció
I3	I2	I1	I0		I3	I2	I1	I0	
0	0	0	0	0	1	0	0	0	8
0	0	0	1	1	1	0	0	1	9
0	0	1	0	2	1	0	1	0	A
0	0	1	1	3	1	0	1	1	b
0	1	0	0	4	1	1	0	0	C
0	1	0	1	5	1	1	0	1	d
0	1	1	0	6	1	1	1	0	E
0	1	1	1	7	1	1	1	1	F



Hex ⟶ 7 szegmens dekódoló

- A frekvenciaosztó 27 – 24. bitjei adják meg a kiírandó szám 4 bites kódját. A kiíratást a 23. bit 0-ba billenésekor végezzük (ekkor változik a kiírandó szám).

```
always @ (negedge count[23])
 begin
  case (count [27:24])            // A kiírandó számjegy
   0: segment <= 8'b11000000;     // 0 (sorrend: DP és gfedcba szegmensek)
   1: segment <= 8'b11111001;     // 1
   2: segment <= 8'b10100100;     // 2
   3: segment <= 8'b10110000;     // 3
   4: segment <= 8'b10011001;     // 4
   5: segment <= 8'b10010010;     // 5
   6: segment <= 8'b10000010;     // 6
   7: segment <= 8'b11111000;     // 7
   8: segment <= 8'b10000000;     // 8
   9: segment <= 8'b10010000;     // 9
  10: segment <= 8'b10001000;     // A
  11: segment <= 8'b10000011;     // B
  12: segment <= 8'b11000110;     // C
  13: segment <= 8'b10100001;     // D
  14: segment <= 8'b10000110;     // E
  15: segment <= 8'b10001110;     // F
  endcase
 end
```

one_digit_display: a teljes kód egyben

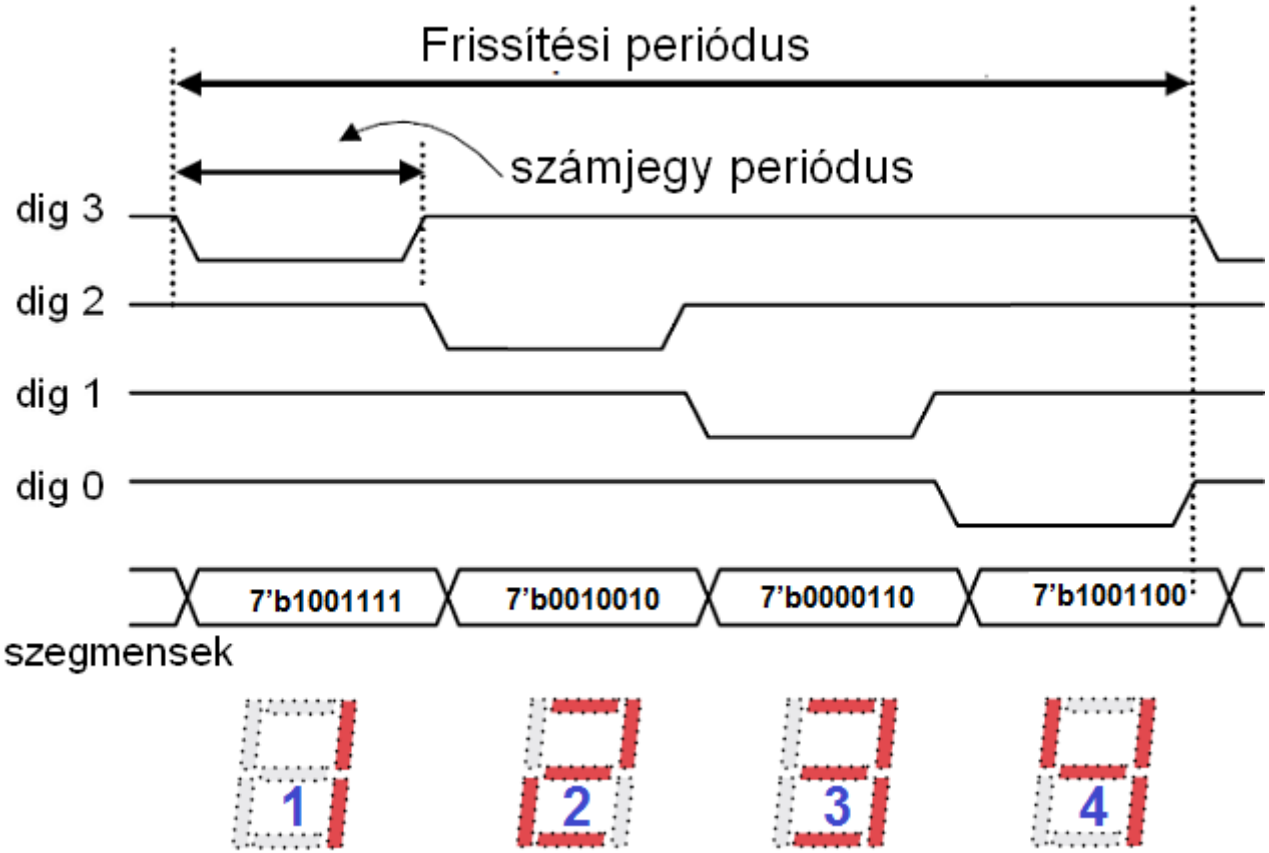
```
module display (clk_50M, digit, segment);
input clk_50M;                          // 50MHz órajel (pin12)
output wire [7:0] digit;                // számjegyek vezérlése
output reg [7:0] segment;               // DP és gfedcba szegmensek vezérlése
reg [27:0] count;                       // frekvencia-osztó számláló
assign digit = 8'b11111110;             // digital tube bit selection is on

always @ (posedge clk_50M)              // frekvencia leosztás
 begin
  count <= count + 1;                   // számláló léptetése
 end

always @ (negedge count[23])
 begin
  case (count [27:24])                  // A frekcenciaosztó felső bitjei
   0: segment <= 8'b11000000;           // Közös anódú szegmensek - negatív logika
   1: segment <= 8'b11111001;
   2: segment <= 8'b10100100;
   3: segment <= 8'b10110000;
   4: segment <= 8'b10011001;
   5: segment <= 8'b10010010;
   6: segment <= 8'b10000010;
   7: segment <= 8'b11111000;
   8: segment <= 8'b10000000;
   9: segment <= 8'b10010000;
  10: segment <= 8'b10001000;
  11: segment <= 8'b10000011;
  12: segment <= 8'b11000110;
  13: segment <= 8'b10100001;
  14: segment <= 8'b10000110;
  15: segment <= 8'b10001110;
  endcase
 end
endmodule
```

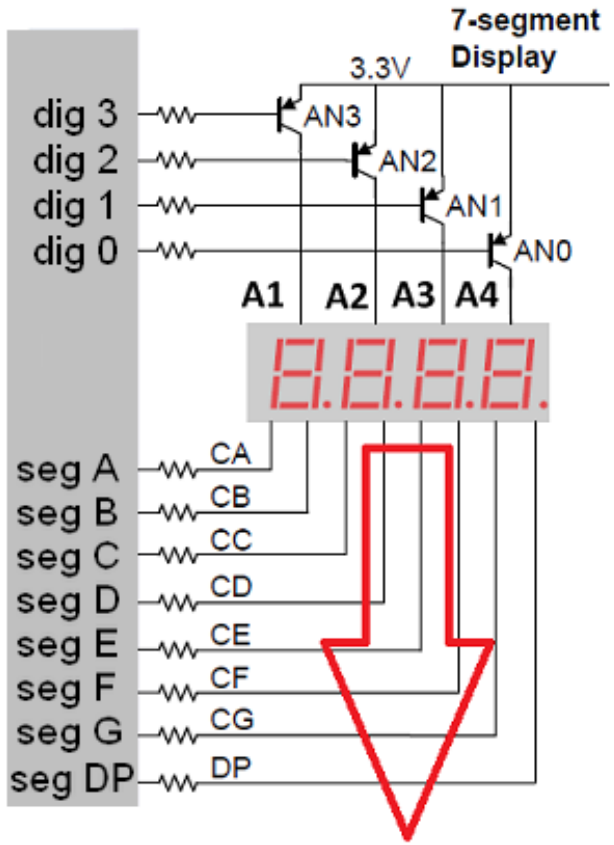
display.v

```
+-----------------------------------------------------------------------+
; Fitter Summary                                                        ;
+---------------------+-------------------------------------------------+
; Fitter Status       ; Successful - Mon Dec 04 11:02:15 2017           ;
; Quartus Prime Version ; 17.0.0 Build 595 04/25/2017 SJ Lite Edition   ;
; Revision Name       ; display                                         ;
; Top-level Entity Name ; display                                       ;
; Family              ; MAX II                                          ;
; Device              ; EPM240T100C5                                    ;
; Timing Models       ; Final                                           ;
; Total logic elements ; 35 / 240 ( 15 % )                             ;
; Total pins          ; 17 / 80 ( 21 % )                                ;
; Total virtual pins  ; 0                                               ;
; UFM blocks          ; 0 / 1 ( 0 % )                                   ;
+---------------------+-------------------------------------------------+
```

8 számjegy időmultiplex kijelzése

- Az ábrákon csak 4 számjegy kijelzését mutatjuk, de a módszer könnyen bővíthető
- Egyszerre csak egy számjegy aktív, s az ehhez tartozó kódot vezetjük a szegmensekre





8 számjegyhez
16 kivezetés kell:
8 számjegyválasztó és
8 szegmensvezérlő jel
(7 szegmens+tizedespont)

Az „eight_digit_display” projekt

- Nyolc számjegyet jelenítünk meg időmultiplex módon (a nyolc számjegyet felváltva jelenítjük meg)
- Mindegyik helyiértéken a számjegy sorszámát jelezzük ki
- A megjelenítés ütemezését az 50 MHz-es órajelből leosztott jelekkel vezéreljük

```
module display (clk_50M, digit, segment);
  input clk_50M;                          // 50MHz órajel (pin12)
  output reg [7:0] digit;                 // számjegyek vezérlése
  output reg [7:0] segment;               // DP és gfedcba szegmensek

  reg [24:0] count;                       // frekvenciaosztó számláló
  reg [3:0] data;                         // a kiírandó számjegy

  always @ (posedge clk_50M)              // frekvencia leosztás
    count <= count + 1;                   // számláló léptetése
  ...
endmodule
```

Az „eight_digit_display” projekt

- A kijelzendő adat (data) a számláló 16 – 18. bitjei, 4-bitre bővítve
- Egy-egy számjegy kiírása a számláló 15. bitjének lefutásánál történik (2^{16}/50MHz, azaz kb. 1,3 ms-onként)
- A frissítési periódus ennek 8-szorosa: 2^{19}/50MHz ≈ 10 ms

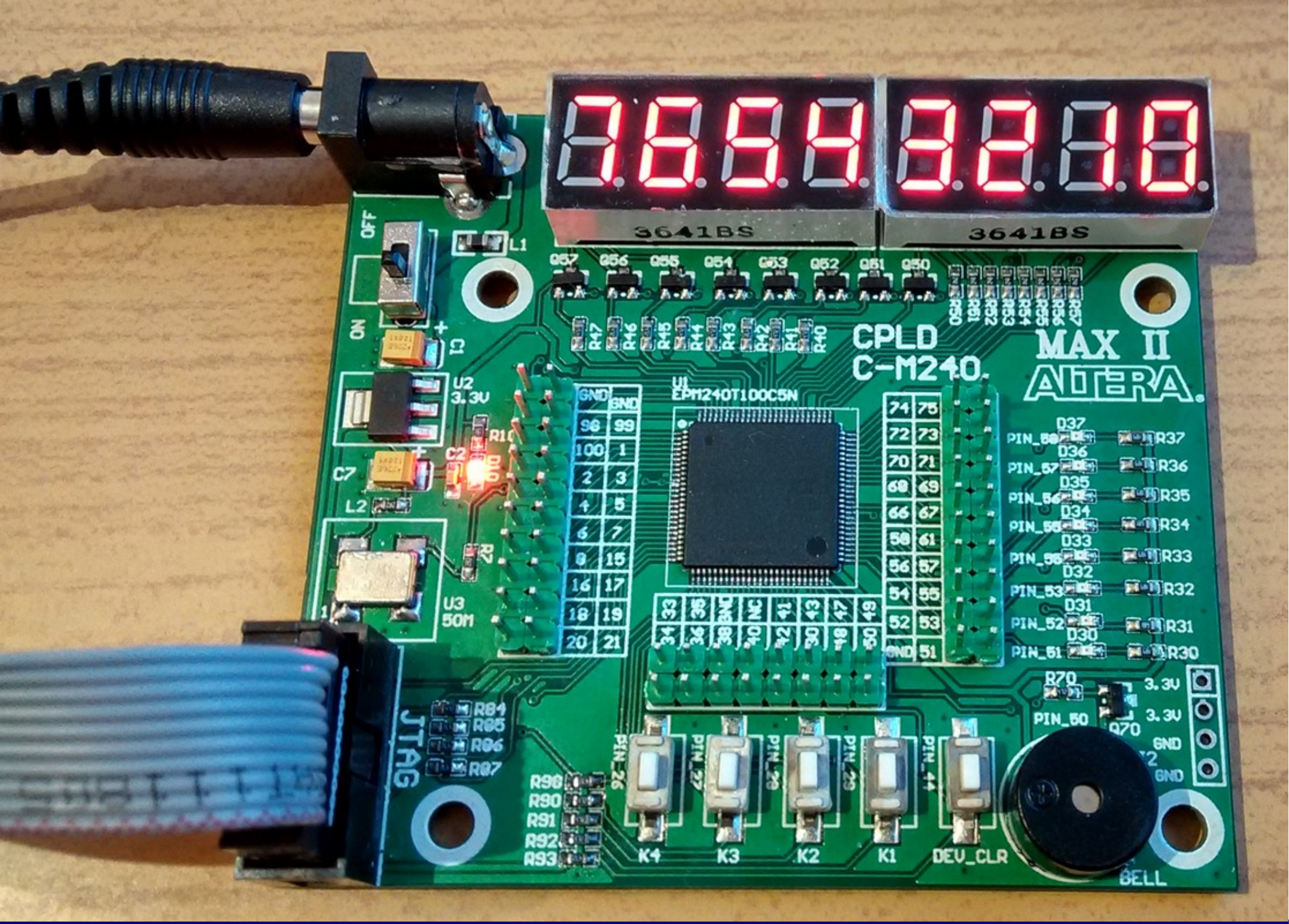
```
always @ (negedge count[15])
 begin
  data <= {1'b0, count[18:16]};          // a számjegy sorszámát íratjuk ki
  case (count[18:16])                    // a számláló vezérli a kijelzést
       3'b000: digit <= 8'b1111_1110;    // jobb szélső számjegy aktiválása
       3'b001: digit <= 8'b1111_1101;
       3'b010: digit <= 8'b1111_1011;
       3'b011: digit <= 8'b1111_0111;
       3'b100: digit <= 8'b1110_1111;
       3'b101: digit <= 8'b1101_1111;
       3'b110: digit <= 8'b1011_1111;
       3'b111: digit <= 8'b0111_1111;    // bal szélső számjegy aktiválása
       default:
               digit <= 8'b1111_1110;
  endcase
 end
```

Az „eight_digit_display” projekt

- A megjelenítendő szám (***data*** regiszter) hét szegmensre dekódolása hasonlóan történik, mint az előző projektben

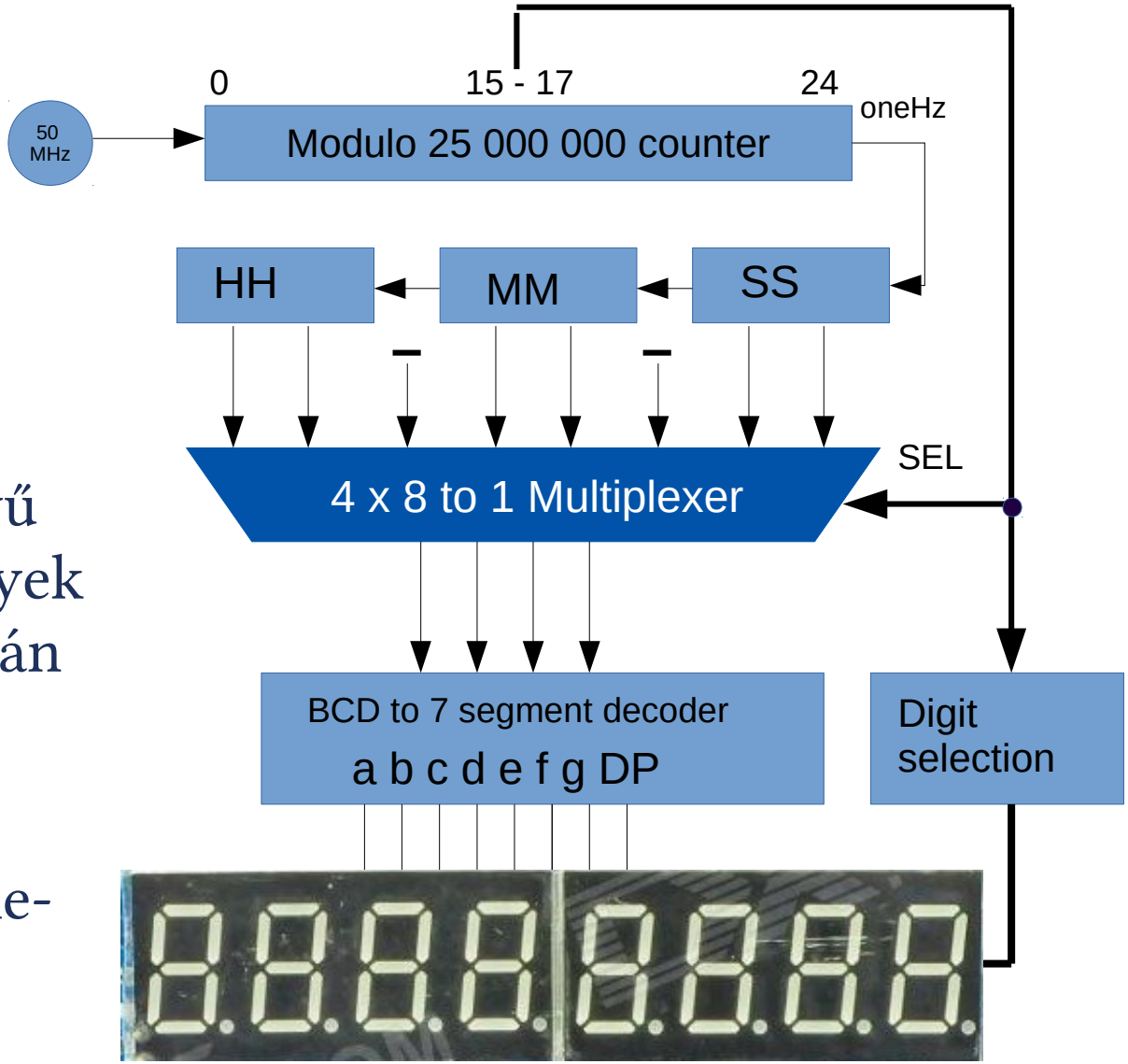
```
always @ (data)
 begin
  case (data])           // A kiírandó számjegy
   0: segment = 8'b11000000;   // 0 (sorrend: DP és gfedcba szegmensek)
   1: segment = 8'b11111001;   // 1
   2: segment = 8'b10100100;   // 2
   3: segment = 8'b10110000;   // 3
   4: segment = 8'b10011001;   // 4
   5: segment = 8'b10010010;   // 5
   6: segment = 8'b10000010;   // 6
   7: segment = 8'b11111000;   // 7
   8: segment = 8'b10000000;
   9: segment = 8'b10010000;
  10: segment = 8'b10001000;
  11: segment = 8'b10000011;
  12: segment = 8'b11000110;
  13: segment = 8'b10100001;
  14: segment = 8'b10000110;
  15: segment = 8'b10001110;
  endcase
 end
```

```
+-----------------------------------------------------------------------+
; Fitter Summary                                                        ;
+---------------------+-------------------------------------------------+
; Fitter Status       ; Successful - Mon Dec 04 14:47:42 2017            ;
; Quartus Prime Version ; 17.0.0 Build 595 04/25/2017 SJ Lite Edition    ;
; Revision Name       ; display                                         ;
; Top-level Entity Name ; display                                       ;
; Family              ; MAX II                                          ;
; Device              ; EPM240T100C5                                    ;
; Timing Models       ; Final                                           ;
; Total logic elements ; 34 / 240 ( 14 % )                             ;
; Total pins          ; 17 / 80 ( 21 % )                               ;
; Total virtual pins  ; 0                                               ;
; UFM blocks          ; 0 / 1 ( 0 % )                                   ;
+---------------------+-------------------------------------------------+
```


3641BS
3641BS
CPLD
C-M240
MAX II
ALTERA
EPM240T100C5N
JTAG
K4
K3
K2
K1
DEV_CLR
BELL
3.3V
GND

Stopperóra tervezése

- Az óra 1 Hz-es időalapját az 50 MHz rendszer órajelből osztjuk le
- A multiplex kijelzéshez a számjegy kiválasztása és az adatmultiplexer vezérlőjele a számláló 15 – 17. bitjei (kb. 1 ms időközönként vált)
- Az időt 3 db két számjegyű BCD számláló méri, amelyek 60, 60 illetve 24 egység után csordulnak túl
- Az időszámlálót **törlés** és **engedélyezés/tiltás** bemenettel is elláttuk, ezeket nyomógombok vezérlik



clock_display projekt: clock.v

```
module clock (clk_50M,key_in,digit,segment);
  input clk_50M;                                  // 50 MHz órajel (pin 12)
  input [1: 0] key_in;                            // K1 és K2 nyomógombok állapota
  output reg [7: 0] digit;                        // számjegy választó regiszter
  output reg [7: 0] segment;                      // szegmensvezérlő regiszter

  reg [ 3: 0] disp_dat;                           // a kiírandó számjegy kódja
  reg [24: 0] count;                              // frekvenciaosztó számláló
  reg [23: 0] hour;                               // az idő számlálója
  reg oneHz, keyen;                               // 1 Hz jel és az állapotjelző
  reg [1: 0] dout1, dout2, dout3;                 // pergésmentesítő regiszterek
  wire [1: 0] key_done;                           // gomblenyomások könyvelése


  always @ (posedge clk_50M)                      // oneHz jel generálása
    begin
      count = count + 1'b1;
      if (count == 25'd25000000)                  // 0.5 s eltelt?
        begin
          count = 25'd0;                          // számláló törlése
          oneHz = ~ oneHz;                        // az 1 Hz-es jel előállítása
      end
    end
```

Folytatás a következő oldalakon ...

A nyomógombok pergésmentesítése

- A **K1** és **K2** nyomógombok állapotát 3 fokozatú léptető regiszterbe mintavételezzük kb. 5 ms-onként
- Ha három egymás utáni mintavétel alacsony szintet mutat, akkor elnyomottnak tekintjük a gombot (key_done[0], ill. key_done[1])
- A **K1** gomb a ***keyen*** engedélyező/tiltó jel állapotát billenti át
- A **K2** gomb lenyomásakor nullázzuk az időszámlálót és felengedésig nullán is tartjuk

```
assign key_done = (dout1|dout2|dout3);  // pergésmentesített gomb állapotok

 always @ (posedge count [17]) begin    // mintavételezés 5 ms-onként
     dout1 <= key_in;
     dout2 <= dout1;
     dout3 <= dout2;
 end

 always @ (negedge key_done [0])
     keyen = ~ keyen;                   // a K1 gomb lenyomásakor átbillen
```

Az időmultiplex kijelzés vezérlése

- A kijelzendő adat vagy a 24 bites időszámláló egy-egy 4-bites szelete, vagy egy tagoló karakter (hh – mm – ss formátum)

```
always @ (posedge clk_50M)
  begin
      case (count [17:15])              // a kijelzendő adat kiválasztása
          3'd0: disp_dat = hour [3: 0];   // másodpercek
          3'd1: disp_dat = hour [7: 4];   // 10 másodpercek
          3'd2: disp_dat = 4'ha;          // "-" tagolójel
          3'd3: disp_dat = hour [11: 8];  // percek
          3'd4: disp_dat = hour [15:12];  // 10 percek
          3'd5: disp_dat = 4'ha;          // "-" tagolójel
          3'd6: disp_dat = hour [19:16];  // órák
          3'd7: disp_dat = hour [23:20];  // 10 órák
      endcase
      case (count [17:15])             // a számjegyválasztó jel előállítása
          3'd0: digit = 8'b11111110;      // jobb szélső számjegy
          3'd1: digit = 8'b11111101;      // 2. számjegy
          3'd2: digit = 8'b11111011;      // 3. számjegy
          3'd3: digit = 8'b11110111;      // 4. számjegy
          3'd4: digit = 8'b11101111;      // 5. számjegy
          3'd5: digit = 8'b11011111;      // 6. számjegy
          3'd6: digit = 8'b10111111;      // 7. számjegy
          3'd7: digit = 8'b01111111;      // bal szélső számjegy
      endcase
  end
```

A 7-szegmens dekóder

- Csak decimális számjegyeket vagy „-” tagolójelet jelenítünk meg (utóbbinak decimális 10 a kódja)
- A második tagolójelet az ***oneHz*** jellel kapuzzuk (villogtatás)

```
always @ (posedge clk_50M)
    begin
     case (disp_dat)                        // 7-szegmens dekóder
         4'h0: segment = 8'hc0;             // 0 kijelzése
         4'h1: segment = 8'hf9;             // 1 kijelzése
         4'h2: segment = 8'ha4;             // 2 kijelzése
         4'h3: segment = 8'hb0;             // 3 kijelzése
         4'h4: segment = 8'h99;             // 4 kijelzése
         4'h5: segment = 8'h92;             // 5 kijelzése
         4'h6: segment = 8'h82;             // 6 kijelzése
         4'h7: segment = 8'hf8;             // 7 kijelzése
         4'h8: segment = 8'h80;             // 8 kijelzése
         4'h9: segment = 8'h90;             // 9 kijelzése
         4'ha: segment = 8'hbf;             // - kijelzése
         default:
               segment = 8'hff;             // ne mutasd!
     endcase
     if ((count [17:15] == 3'd2) & oneHz)
         segment = 8'hff;          // a 2. elválasztójel villogtása
   end
```

Az időszámláló kezelése

```
always @ (negedge oneHz or negedge key_done [1])            // az időszámláló kezelése
  begin
    if (! key_done [1])                                     // is the reset key?
       hour = 24'h0;                                        // Yes, then cleared
    else if (! keyen) begin
        hour [3: 0] = hour [3: 0] + 1'b1;                   // Add 1 second
        if (hour [3: 0] == 4'ha) begin
            hour [3: 0] = 4'h0;
            hour [7: 4] = hour [7: 4] + 1'b1;               // Overflow at 10th s
            if (hour [7: 4] == 4'h6) begin
                hour [7: 4] = 4'h0;
                hour [11: 8] = hour [11: 8] + 1'b1;         // Overflow at 60th s
                if (hour [11: 8] == 4'ha) begin
                    hour [11: 8] = 4'h0;
                    hour [15:12] = hour [15:12] + 1'b1;     // Overflow at 10th m
                    if (hour [15:12] == 4'h6) begin
                        hour [15:12] = 4'h0;
                        hour [19:16] = hour [19:16] + 1'b1; // Overflow at 60th m
                        if (hour [19:16] == 4'ha) begin
                            hour [19:16] = 4'h0;
                            hour [23:20] = hour[23:20]+1'b1;// Overflow at 10th h
                        end
                            if (hour [23:16] == 8'h24)      // Overflow at 24th h
                                hour [23:16] = 8'h0;
                    end
                end
            end
        end
    end
end
```