Példa feladatok a kisZH3-hoz

Digitális technika (VIMIAA02)

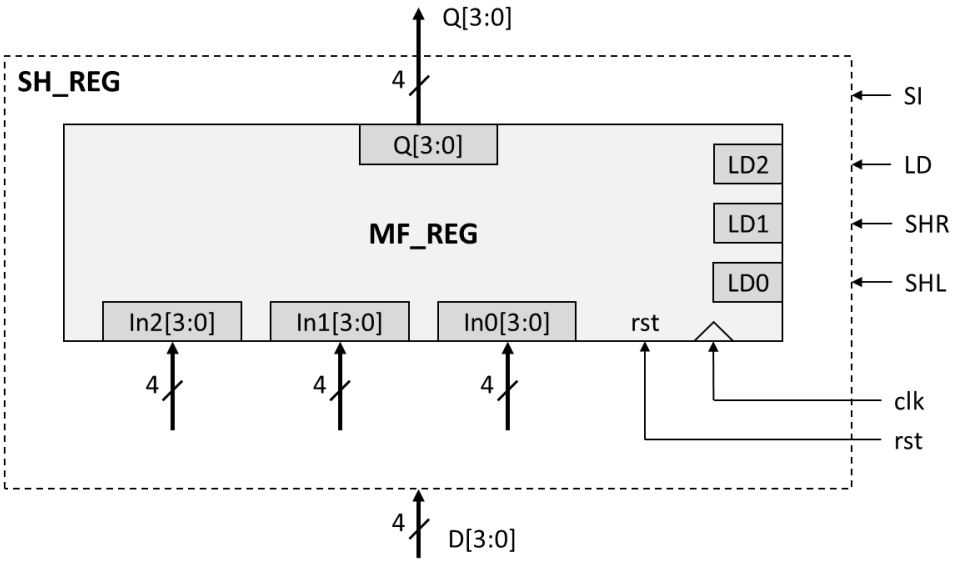
A példa kisZH feladatok célja, hogy a kisZH-ra történő felkészülés során segítséget nyújtson pár tipikus feladat ismertetésével, valamint hogy a hallgatók számára egyenlő esélyeket biztosítson azáltal, hogy a példa kisZH feladatokat a legkorábbi labor kurzus időpontja előtt pár nappal közzétesszük. (A kisZH-n nem pontosan ilyen feladatok lesznek, de némileg hasonlóak.)

Az alábbiakban a kisZH3 által számonkért témakörökhöz található egy-két példa feladat.

Sorrendi funkcionális elemek (shiftregiszter, számláló)

1. Multifunkciós regiszterből shiftregiszter

Az alábbi 4 bites multifunkciós (**MF_REG**) regiszter 3 forrásból képes betölteni adatot. Ha LD0 aktív, akkor In0-át tölti be, ha LD1, akkor In1-et, ha LD2, akkor In2-őt, ha pedig egyik sem aktív, akkor nem változik a tartalma. **Készítse el belőle az alább specifikált funkcionális elemet!**



Funkcionális elem: *kétirányú* (SHL, SHR), *tölthető* (LD) shiftregiszter (**SH_REG**).

Elvárt működés:

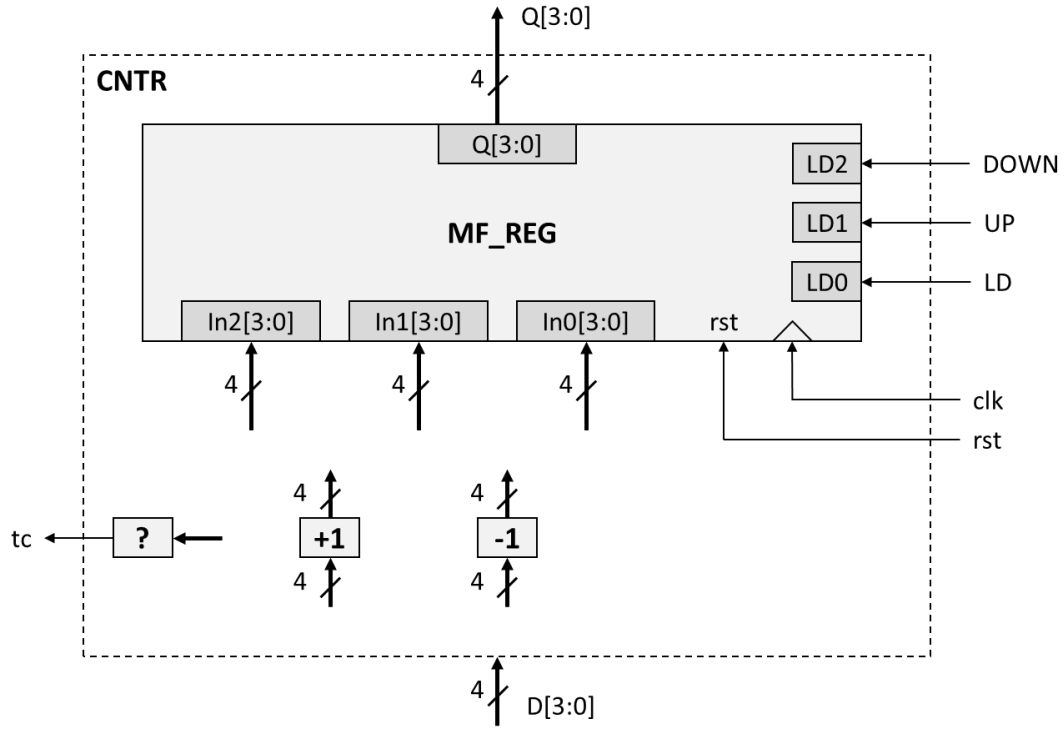
- Ha az LD jel aktív, **töltse be** a D[3:0] adatot.
- Különben, ha az SHR jel aktív, **shiftelje el jobbra** a tartalmát, és a megfelelő bitjébe **léptesse be** SI értékét.
- Különben, ha az SHL jel aktív, **shiftelje el balra** a tartalmát, és a megfelelő bitjébe **léptesse be** SI értékét.
- Egyébként ne változzon a tartalma.

Kérdések:

1. Tételezzük föl, hogy az **MF_REG** töltő bemenetei között az alábbi a prioritás sorrend: LD0 > LD1 > LD2. Ebben az esetben a megvalósítandó **SH_REG** shiftregiszter SHL, SHR és LD vezérlő jeleit az **MF_REG** mely töltő bemeneteire kell kötni?
2. A töltő bemenetek vezérlő jelekhez történő hozzárendelése után a következő kérdés, hogy az elvárt funkció megvalósítása érdekében az **MF_REG** In0, In1 és In2 bemeneteire **mely jel(ek), mely bitjeit, milyen sorrendben** kell kötni?

2. Multifunkciós regiszterből számláló

Az alábbi 4 bites multifunkciós (**MF_REG**) regiszter 3 forrásból képes betölteni adatot. Ha LD0 aktív, akkor In0-át tölti be, ha LD1, akkor In1-et, ha LD2, akkor In2-őt, ha pedig egyik sem aktív, akkor nem változik a tartalma (egyidejűleg több aktív töltő jel esetén a fontossági sorrend a következő: LD0 > LD1 > LD2). **Készítse el belőle az alább specifikált funkcionális elemet!**



Funkcionális elem: *kétirányú* (UP, DOWN), *tölthető* (LD) számláló (**CNTR**).

Elvárt működés:

- Ha az LD jel aktív, **töltse be** a D[3:0] adatot.
- Különben, ha az UP jel aktív, **számoljon felfele** (1-et).
- Különben, ha a DOWN jel aktív, **számoljon lefele** (1-et).
- Egyébként ne változzon a tartalma.

Kérdések:

1. Az **MF_REG** multifunkciós regiszteren felül rendelkezünk még egy inkrementer (**+1**) és egy dekrementer (**-1**) áramkörrel is. Az elvárt funkció megvalósítása érdekében az **MF_REG** In0, In1 és In2 bemeneteit, valamint az inkrementer (**+1**) és a dekrementer (**-1**) ki- ill. bemeneteit **hogyan kéne összekötögetni**?
2. A **?**-el jelölt kombinációs hálózat állítja elő a számlálónk végérték jelzését (tc). Tételezzük föl, hogy ehhez először előállítja az alábbi két segéd jelet:
   EN: aktív, ha felfele vagy lefele számolunk
   DIR: logikai 1 felfele számlálás esetén, egyébként 0
   Verilog nyelven hogyan lehetne leírni azt a kifejezést, ami EN és DIR jeleket is felhasználva előállítja tc értékét?

3. Shiftregiszter Verilog nyelvű implementációja

Egészítse ki az alábbi Verilog kódot úgy, hogy az egy *törölhető*, *tölthető*, *engedélyezhető*, *kétirányú* shiftregisztert írjon le! Válaszában ne használjon szóközöket!

```
module SHR4(
    input clk,
    input rst,
    input ld,
    input en,
    input dir,
    input si,
    input [3:0] d,
    output reg [3:0] q);

    always @(posedge clk)
        if (rst)
            q <= 4'h0;
        else if (      )
            q <=      ;
        else if (      )
            if (      )
                q <=      ; // Shift to right
            else
                q <=      ; // Shift to left
endmodule
```

4. Számláló Verilog nyelvű implementációja

Egészítse ki az alábbi Verilog kódot úgy, hogy az egy *törölhető*, *tölthető*, *engedélyezhető*, *kétirányú* számlálót írjon le! (A számlálónak most nincs tc kimenete.) Verilog konstanst a kiadott kódban már látható formátumban adjon meg! Válaszában ne használjon szóközöket!

```
module CNTR16(
    input clk,
    input rst,
    input ld,
    input en,
    input dir,
    input [3:0] d,
    output reg [3:0] q);

    always @(posedge clk)
        if (rst)
            q <= 4'h0;
        else if (      )
            q <=      ;
        else if (      )
            if (      )
                q <=      ; // Count up
            else
                q <=      ; // Count down
endmodule
```

Újabb kombinációs funkciók (összeadó, kivonó)

Tippek az igazságtáblás feladatokhoz:

1. Az igazságtáblákat természetesen nem celláról-cellára célszerű "bemagolni", hanem a bináris összeadás és kivonás szabályait célszerű megtanulni a jegyzetből.
2. A logikai függvényekhez hasznos lehet továbbá egy rövid, természetes nyelven megfogalmazott mondatot is bememorizálni, hogy fejből is gyorsan ki lehessen értékelni őket (pl.: "Ez a függvény páros számú 1-es esetén igaz.").

5. Teljes összeadó kimeneteinek igazságtáblája

Töltse ki a teljes összeadó kimeneteinek igazságtábláját!

a	b	ci	s	co
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

6. Teljes kivonó kimeneteinek igazságtáblája

Töltse ki a teljes kivonó kimeneteinek igazságtábláját (a: kisebbítendő, b: kivonandó, ci: átvitel bemenet)!

a	b	ci	s	co
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

7. Kétbites teljes összeadó 1 bites teljes összeadók példányosításával

Adott egy 1 bites teljes összeadó implementációja az alábbi módon deklarált Verilog modulban:

```
module fadd(
   input  a, b, ci,
   output s, co
);
```

Két 1 bites teljes összeadó példányosításával készítsen egy kétbites teljes összeadót az alábbi kódrészlet kiegészítésével:

```
module fadd2(
    input [1:0] a,
    input [1:0] b,
    input ci,
    output [1:0] s,
    output co);

    wire [2:0] c;

    assign c[0] = [     ];

    fadd [     ]( .a([     ]), .b([     ]), .ci([     ]), .s([     ]), .co([     ]) );
    fadd [     ]( .a([     ]), .b([     ]), .ci([     ]), .s([     ]), .co([     ]) );

    assign co = [     ];

endmodule
```

Verem (Stack), FIFO

8. Verem

Adja meg egy 4 bites verem interfészének jeleit az alábbi Verilog kód kiegészítésével:

```
module stack(
    input clk,
    input rst,
    input [    ],      // Vezérlő jel adat betételéhez
    input [    ],      // Vezérlő jel adat kivételéhez
    input [    ] din,  // Adat bemenet
    output [    ] dout // Adat kimenet
);
```

Tételezzük föl, hogy a verembe az alábbi három adatot írták be (a megadott sorrendben):
DATA0, DATA1, DATA2.

Ezt követően kiolvasunk két adatot. Mi lesz az elsőnek, és mi a másodiknak kiolvasott adat?

9. FIFO

Adja meg egy 4 bites FIFO interfészének jeleit az alábbi Verilog kód kiegészítésével:

```
module FIFO(
    input clk,
    input rst,
    input [    ],      // Vezérlő jel adat betételéhez
    input [    ],      // Vezérlő jel adat kivételéhez
    output [    ],     // Státusz jel annak jelzésére, hogy üres a FIFO
    output [    ],     // Státusz jel annak jelzésére, hogy tele van a FIFO
    input [    ] din,  // Adat bemenet
    output [    ] dout // Adat kimenet
);
```

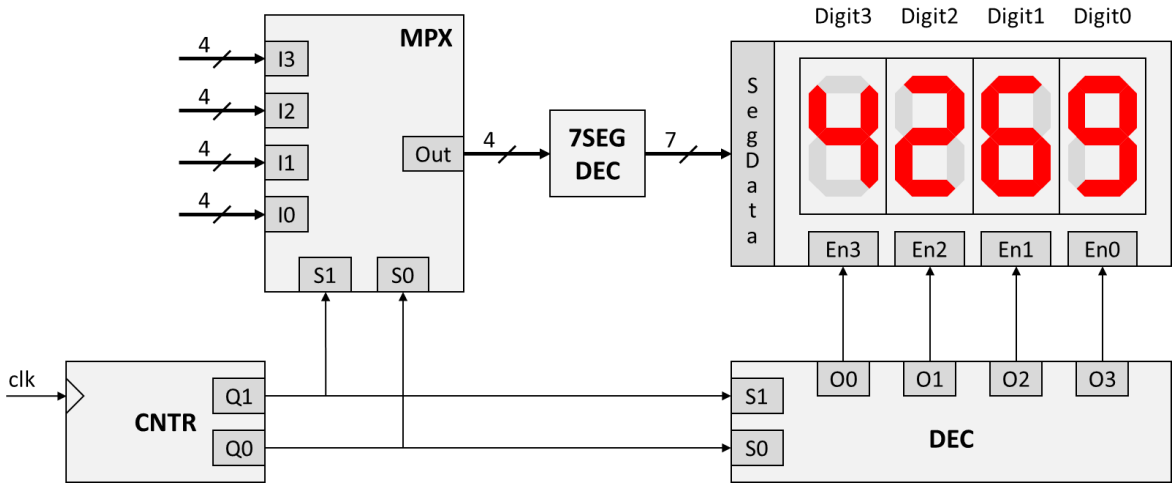
Tételezzük föl, hogy a FIFO-ba az alábbi három adatot írták be (a megadott sorrendben):
DATA0, DATA1, DATA2.

Ezt követően kiolvasunk két adatot. Mi lesz az elsőnek, és mi a másodiknak kiolvasott adat?

A megismert funkcionális elemekből felépített logika

10. Időmultiplexált kijelző

Adott az alábbi, időmultiplexált kijelző. A kijelző időzítésének szerepét ellátó számláló kellően gyorsan számol ahhoz, hogy az egy időben egyszerre csak egy digit felvillantása mellett is az emberi szem állóképet lásson.



Ha az ábrán látható kijelzési kép jelenik meg, milyen bináris értékek találhatóak az **MPX** multiplexer I0, I1, I2 és I3 bemenetein?