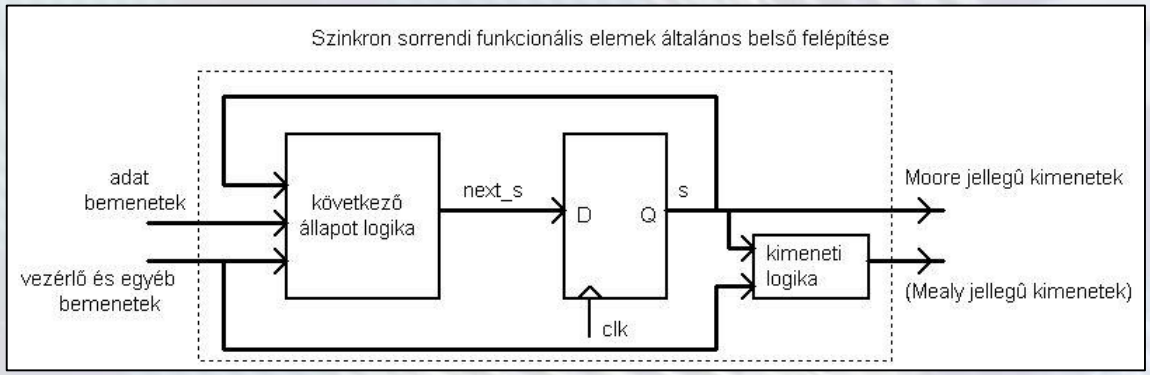
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Digitális technika VIMIAA02 5. EA

Fehér Béla, Benesóczky Zoltán
BME MIT

Szinkron sorrendi funkcionális egységek

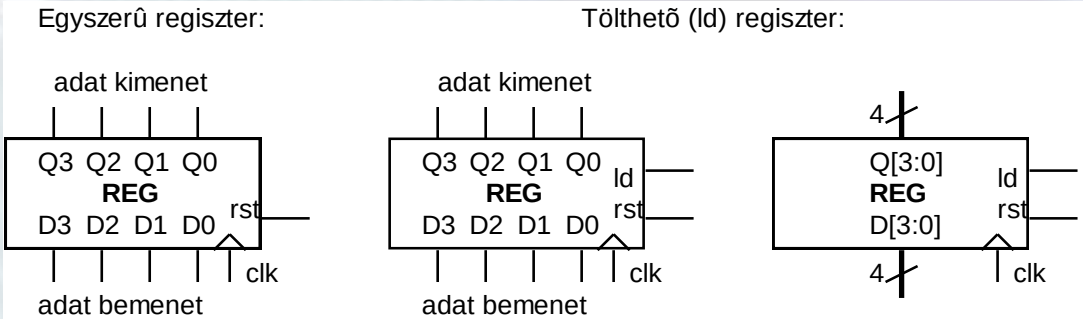
- A **szinkron funkcionális egységek** a kombinációs funkcionális elemekhez hasonlóan ***gyakran használt többnyire komplex sorrendi hálózatot igénylő funkciót valósítanak meg***.
- Ezek tipikus, ***a megvalósított funkcióhoz illeszkedő be és kimenetekkel rendelkeznek***
- Általános belső felépítésük:



- A fő kimeneteit közvetlenül a regiszter kimenete adja (Moore jellegű adat kimenetek)
- Egyéb kimenetei is lehetnek
- A működésük a vezérlő bemeneteikkel állítható be.

Szinkron sorrendi funkcionális egységek

- **REGISZTER** (már ismerjük)
- Kapcsolási rajz szimbólumai:



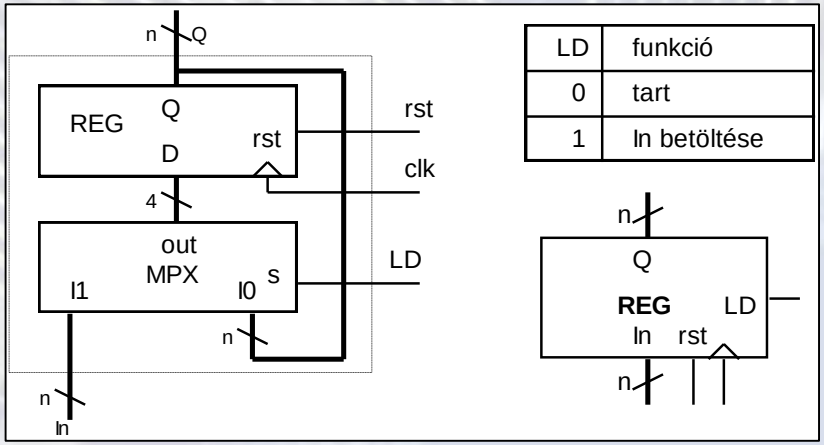
- **A regiszterek funkciói:**
  - ***Alaphelyzet beállítás*** (rst): lehet tetszőleges induló érték (nem csak 0). A többi vezérlő jelhez képest ennek a ***legnagyobb a prioritása***
  - ***Töltés*** (load, ld): Az adatbemeneti vonalak mintavételezése és beírása (D flip-flopok CE bemenetei: CE = 1 töltés)
  - ***Tartás***: A regiszter tartalom fenntartása ld = 0 (CE = 0)

Szinkron sorrendi funkcionális egységek tölthető regiszter

- **Tölthető regiszter**
- A regiszter betöltés funkciója CE nélküli DFF esetén MUX használatával is megoldható.
  - Általános séma: A regiszter a bemeneti adatot egy busz multiplexer kimenetéről kapja.
  - A MUX kiválasztó bemeneteire kapcsoljuk az adott vezérléshez szükséges kódot, ami kiválasztja a betöltendő adat forrását.
    Töltés (LD =1): forrás az adat bemenet: In[3:0], (MPX I1).
    Tartás (LD = 0): forrás az adat kimenet: Q[3:0], (MPX I0) az aktuális tartalom visszaíródik, tartás funkció).

Verilog kód
(a változó definícióktól eltekintünk):

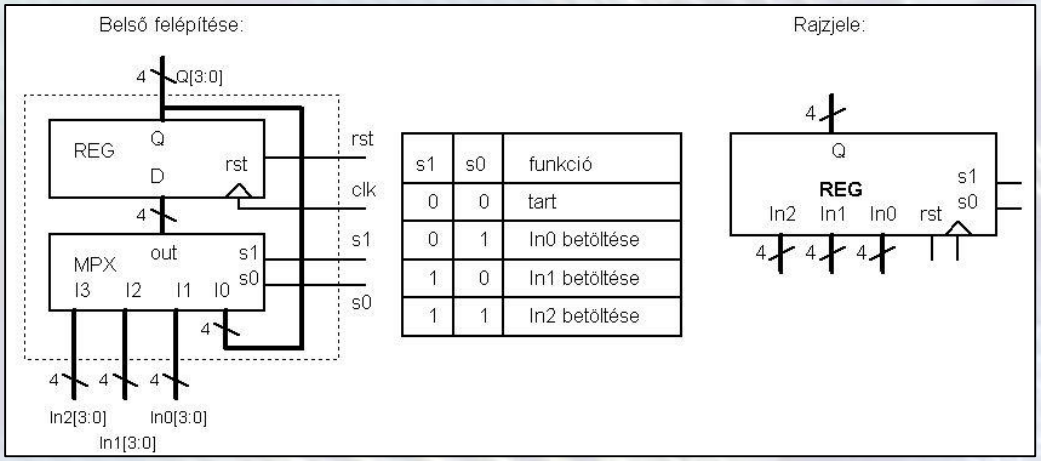
```
always@(posedge clk)
  if(rst) q <= 8'h00;
  else if(ld) q <= In;
  //egyéb esetben q megtartja értékét
```



LD	funkció
0	tart
1	In betöltése

Szinkron sorrendi funkcionális egységek multifunkciós regiszter

- **Multifunkciós regiszter**
  - ***A multifunkciós regiszter választhatóan több forrásból képes adat betöltésére*** (a tartás funkció mellett).
    Kódolt funkció választású verzió esetén a MPX kiválasztó jeleinek (s1, s0) kombinációja választja ki a betöltendő adat bemenetét. A különféle funkcionális elemeknél ide kapcsolódik a funkciót megvalósító kombinációs hálózat kimente.



s1	s0	funkció
0	0	tart
0	1	In0 betöltése
1	0	In1 betöltése
1	1	In2 betöltése

  - A példa regisztere 3 adat forrás adatát képes betölteni, tárolni
  - S1S1 00: tart, 01: load In0, 10: load In1, 11: load In2

Szinkron sorrendi funkcionális egységek multifunkciós regiszter

Verilog kód *kódolt funkció választású* multifunkciós regiszterhez (4 bites 3 forrásból tölthető regiszter)

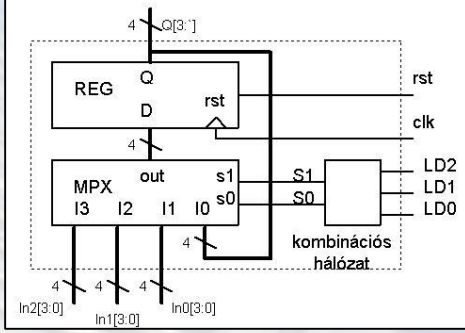
```
module MFREG3(input clk, rst, input [1:0] s,
                    input [3:0] in0, in1, in2, output reg [3:0] q);
always@(posedge clk)
if(rst) q <= 4'h0;
else
    case(s)
    2'b01: q <= in0;
    2'b10: q <= in1;
    2'b11: q <= in2;
    endcase          // egyéb s[1:0] bemenetnél (00) tartás, mert
endmodule            // nem változik meg q értéke
```

Szinkron sorrendi funkcionális egységek multifunkciós regiszter

- Sokszor egyszerűsíti a tervezést, ha ***dekódolt betöltő jelek*** állanak rendelkezésre, vagyis az egyes csatornák betöltését a csatornához rendelt egyetlen jel vezérli (a kiválasztó jelek kombinációi helyett). (Pl. LD0 =1 hatására In0 töltődik be, LD1=1 esetén In1...
- Az egyes ***betöltő jelek között prioritást kell biztosítani***. (Ha egyszerre több aktív, melyik legyen hatásos.) ***A legnagyobb prioritású mindig az rst alaphelyzetbe hozó jel***. Ez többnyire közvetlenül a regisztert vezérli, ahogy a rajz is mutatja.
- A dekódolt vezérlőjelek megvalósításához a betöltő jelek (LD0, LD1, LD2) és a multiplexer kiválasztó bemenetei (S1, S0) közé egy kombinációs hálózat kell. Ennek igazságtáblája kicsit hasonlít a prioritás enkóderéhez.
- A példában 3 betöltő jel (LD0, LD1, LD2) esetére rajzoltuk fel a belső felépítést és igazságtáblát. Az igazságtáblában x-el a bit közömbös (tetszőleges) értékét jelöltük. Az Ldi jelek közül az LD0 prioritása a legnagyobb. (Ha LD0=1, mindenképpen In0 töltődik be.) Utána LD1 (ha LD0=0, LD1=1, akkor In1 töltődik be.)
  Ha LD0=LD1=0, LD2=1 akkor In2 töltődik be.
  A prioritások:
  LD0 > LD1 > LD2

Multifunkciós regiszter dekódolt vezérlőjelekkel

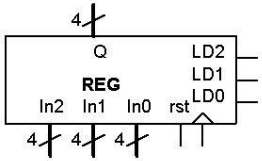
Belső felépítése



A multiplexer select jeleit előállító kombinációs hálózat igazságtáblája

dekódolt funkcióválasztás LD0	LD1	LD2	S1	S0	funkció
0	0	0	0	0	tart
0	0	1	1	1	In2 betöltése
0	1	x	1	0	In1 betöltése
1	x	x	0	1	In0 betöltése

Rajz szimbóluma



Szinkron sorrendi funkcionális egységek multifunkciós regiszter

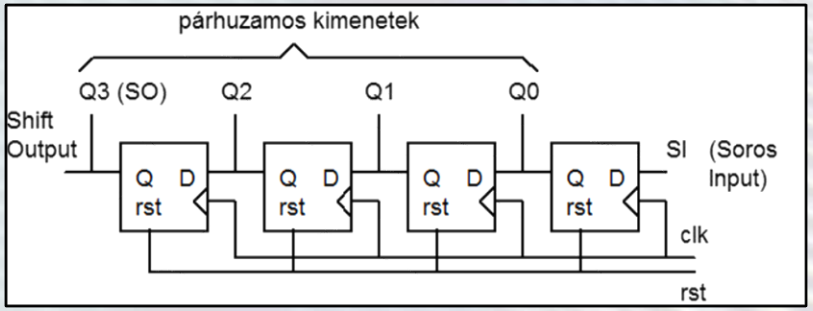
Verilog kód *dekódolt funkció választó* jelekkel multifunkciós regiszterhez (3db 4 bites forrás közül tölti be az LDi-vel kiválasztottat)

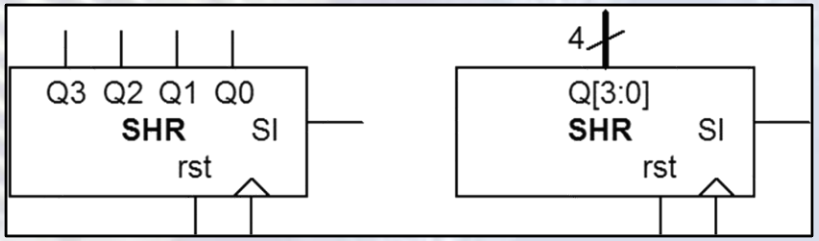
```
module MFREG3(input clk, rst, ld0, ld1, ld2,
                    input [3:0] in0, in1, in2, output reg [3:0] q);
always@(posedge clk)
    begin
    if(rst) q <= 4'h0;     // rst a legnagyobb prioritású jel
    else if(ld0)  q <= in0;          // ld0 a 2. legnagyobb prioritású
        else if(ld1) q <= in1;
            else if(ld2) q <= in2;
    end   // egyéb esetben tartás, q regiszter értéke nem változik
endmodule
```

Szinkron sorrendi funkcionális egységek, shiftregiszter

Shiftregiszter (SHR)

- ***Funkciója a regiszter tartalom elshiftelése*** (eltolása) és a ***shift input*** (SI) bemeneten levő adat bit beshiftelése. A reset (rst) törli.
- Ha DFF-okat sorba kötünk az alábbi ábra szerint, akkor az órajel él hatására a szélsőbe beíródik az új adat bit, a többi DFF tartalma pedig 1-el elshiftelődik (a szomszédos DFF-ba íródik).
- A shiftregiszter bemeneti flip-flopjával ellentétes oldali flip-flopjának kimenetét ***shift output***-nak nevezik. (Ennek adata elvész, felülírja az előző bit.)



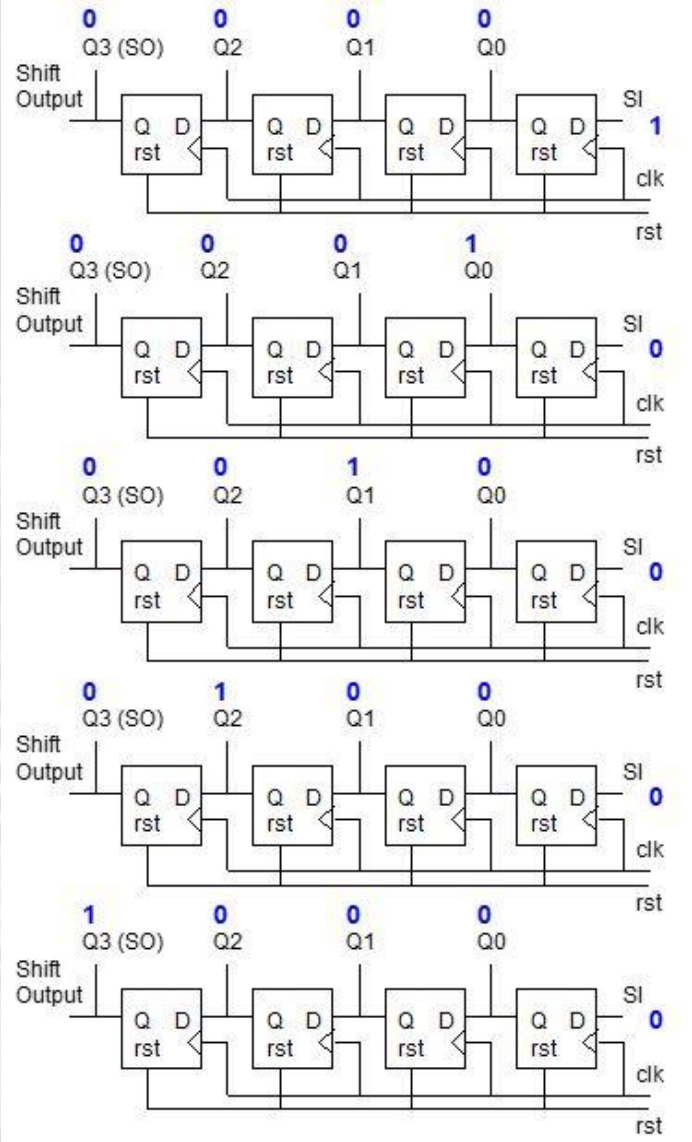


Szinkron sorrendi funkcionális egységek, shiftregiszter

Példa: Ha az eredetileg törölt SHR shift inputjára (SI) az 10000 sorozatot adjuk az órajellel szinkronban, akkor az 1-es az első órajelre beíródik, majd minden órajel után eggyel balra shiftelődik. (Az SI bemeneten most a további bitek mind 0-ák, ezért csak 1 db 1-es lép be.) **Verilogban a shiftelést konkatenálással adjuk meg. {q[2:0], si} egy 4 bites vektor, bitjei: q[2], q[1], q[0], si** (*A Verilog shift operátora 0-át léptet be, ezért nem megfelelő.*) 4 bites shiftregiszter Verilog kódja:

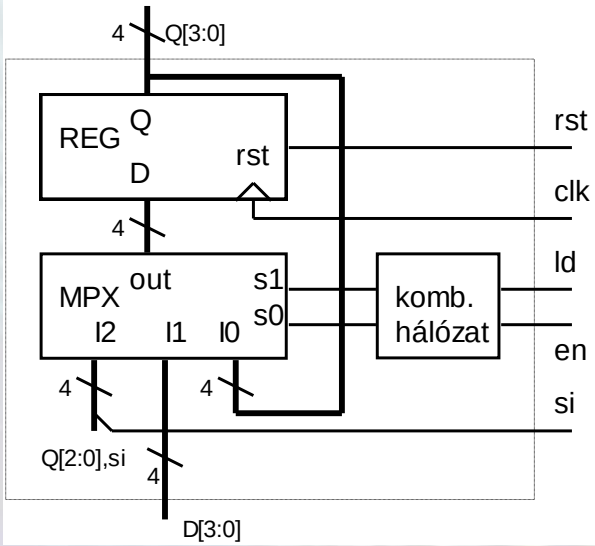
```
always@(posedge clk)
if(rst) q <= 4'b0000;
else   q <= {q[2:0], si}
```

A *q[2:0]* 1-el balra lép és jobbról belép *si*.
(q[3] = q[2], q[2]=q[1], q[1]=si)

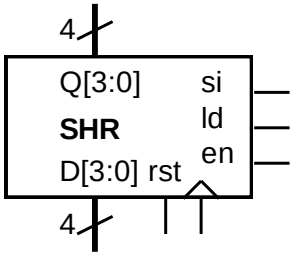


Szinkron sorrendi funkcionális egységek, shiftregiszter

- A shiftregiszter lehet ***engedélyezhető (en) és tölthető (ld)***. A töltésnek van nagyobb prioritása. ***A funkcionális elemek engedélyező jele mindig a fő funkciót*** (itt shiftelés) ***engedélyezi***. (A reset-re és betöltésre nincs hatása, azok nagyobb prioritásúak.) Prioritás: rst > ld > en
- Belső felépítése multifunkciós regiszterrel megvalósítva:



LD	en	S1	S0	funkció
0	0	0	0	tart (Q[3:0] betöltése)
0	1	1	0	shiftelés (Q[2:0],si betöltése)
1	x	0	1	betöltés (D[3:0] betöltése)



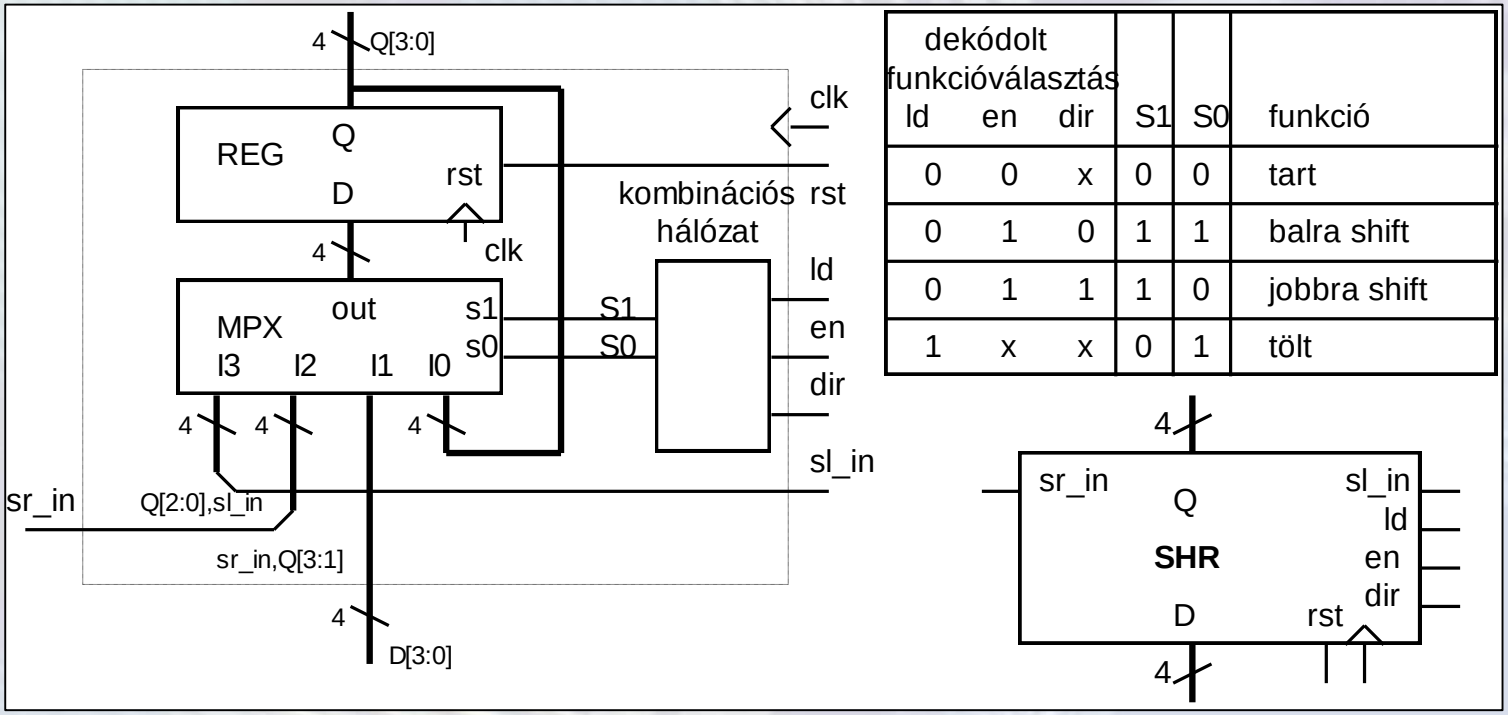
Szinkron sorrendi funkcionális egységek, shiftregiszter

Verilog kód egyszerű törölhető, tölthető, engedélyezhető SHR-hez

```
module SHR4EN(input clk, rst, ld, en, si,
                        input [3:0] d, output reg [3:0] q);
    always@(posedge clk)
    begin
      if (rst) q <= 4'h0;       // rst a legnagyobb prioritású jel
      else if (ld) q <= d;      // utána a betölés
            else if (en) q <= {q[2:0], si}; //utána a shift eng.
    end                                     // egyéb esetben tartás
endmodule
```

Szinkron sorrendi funkcionális egységek

- **Kétirányú shiftregiszter Verilog HDL** shiftelés engedélyező (**en**) és irányvezérlő jellel (**dir**)
- Töltés (ld), léptetés engedélyezés (en), irányvezérlés (dir) (en =1 és dir = 0 balra shift, en =1 és dir = 1 jobbra shift)
- **sl_in**: balra shiftelés bemenet, **sr_in**: jobbra shiftelés bemenet



dekódolt funkcióválasztás ld	en	dir	S1	S0	funkció
0	0	x	0	0	tart
0	1	0	1	1	balra shift
0	1	1	1	0	jobbra shift
1	x	x	0	1	tölt

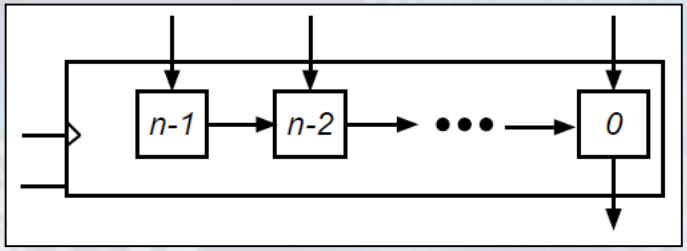
Szinkron sorrendi funkcionális egységek shiftregiszter

Veilog kód dekódolt vezérlésű (ld, en, dir) **kétirányú shiftregiszterhez**

```
module SHR4_RLC(input clk, rst, en, dir, sl_in, sr_in,
                        input [3:0] d, output reg [3:0] q);
    always@(posedge clk)
    if(rst) q <= 4'h0;
    else if(ld) q <= d;    //betöltés
        else if(en)
                if (dir) q <= {sr_in, q[3:1]};  // jobbra shift
                else     q <= {q[2:0], sl_in};  // balra shift
                                                // egyéb esetben tartás
endmodule
```

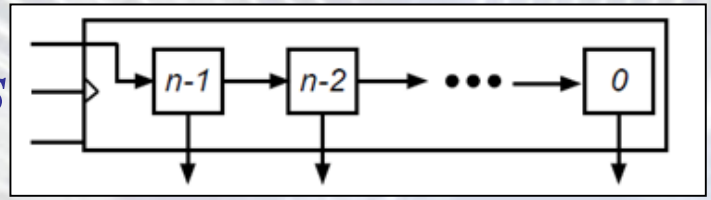
Szinkron sorrendi funkcionális egységek shiftregiszter

- **Shiftregiszter alkalmazási területek**
  - ***Párhuzamos-soros átalakítás***
    - A párhuzamosan rendelkezésre álló n bites adatot betöltjük egy n bites SHR-be



    - Kishifteljük az SHR-ből (a bitek időben egymás után jelennek meg az SOUT kimeneten)
    - Ha az összes bitet kishifteltük, kezdhetjük előlről

  - ***Soros-párhuzamos átalakítás***

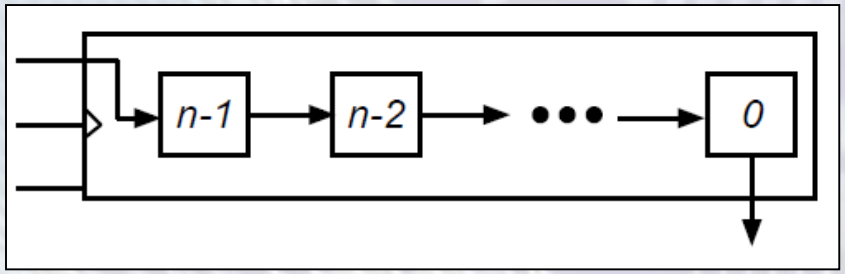


    - A shift inputon időben egymás után jövő biteket beléptetjük az SHR SIN bemenetén
    - Ha az összes bit bejött, átírjuk egy másik regiszterbe, ahol ezután az összes bit egyszerre rendelkezésre áll

Szinkron sorrendi funkcionális egységek shiftregiszter

- **Shiftregiszter alkalmazási területek**
- ***Adat késleltetés***

  A shiftregiszter SI bemenetére az órajellel szinkronban adott jel, az SHR Qi kimenetére T_{clk}*i késleltetéssel jut el. (T_{clk} az órajel periódusideje).

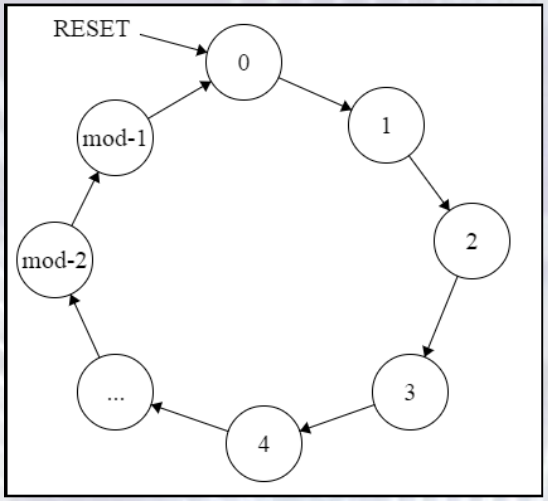


- ***Szorzás (osztás) 2 hatvánnyal***

  pl. 0110 *10 = 1100, 1100:100 = 0011
  - * 2^n → Töltés + n db Léptetés balra (jobbra)
  - Összesen (n+1) órajel

Szinkron sorrendi funkcionális egységek számlálók

- **Számlálók (CNT)**
  - A legfontosabb SSH komponensek
  - Közös jellemzőjük: ***Állapotdiagramjuk ciklikus (gyűrű alakú)***
  - **Állapotok száma: *MODULUS*** (MOD)
  - **A hagyományos felfele számlálók *egymást követő állapotainak kódja 1-gyel növekvő számok.*** Az i-edik állapotra következő kód: (i+1) modulo MOD
  - Pl. **8-as modulusú** felfele számlálónál (3 biten) 0,1,2,3,4,5,6,**7,0**,1,2,3,4,….
  - **Hagyományos lefele számlálókra *ugyanez fordított sorrendben.*** 7,6,5,4,3,2,1,**0,7**,6,5,4,....
  - Általános esetben egy N modulusú számláló állapot kódolása tetszőleges N darab azonos bitszámú, különböző érték lehet.



Szinkron sorrendi funkcionális egységek számlálók

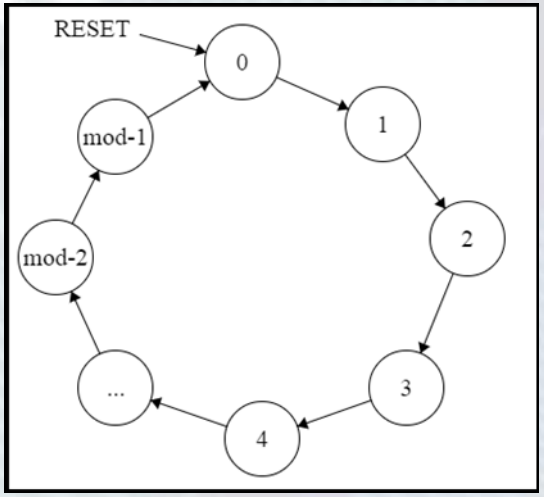
- **Számlálók**
  - Tágabb értelemben számlálóknak tekintjünk minden ciklikus állapotgépet (gyűrű alakú állapotgráf)
  - **Hagyományos számlálók osztályozása modulus alapján**
    - **Bináris számlálók** (modulus 2^n)
    - Nem bináris modulusú számlálók
      Pl: **BCD számláló** (10-es modulus, BDC kódban számol: 0...9)

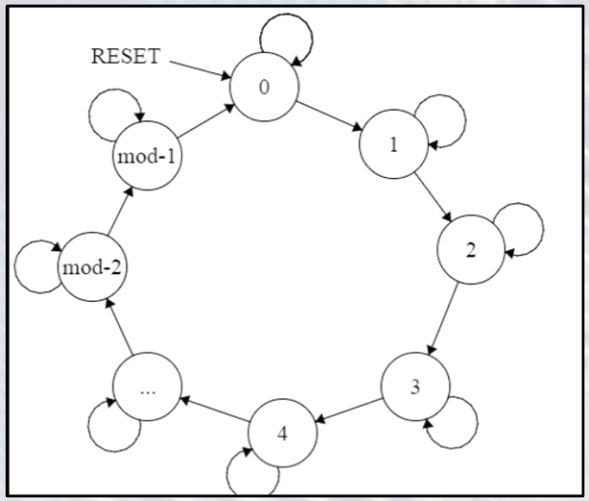
Szinkron sorrendi funkcionális egységek számlálók

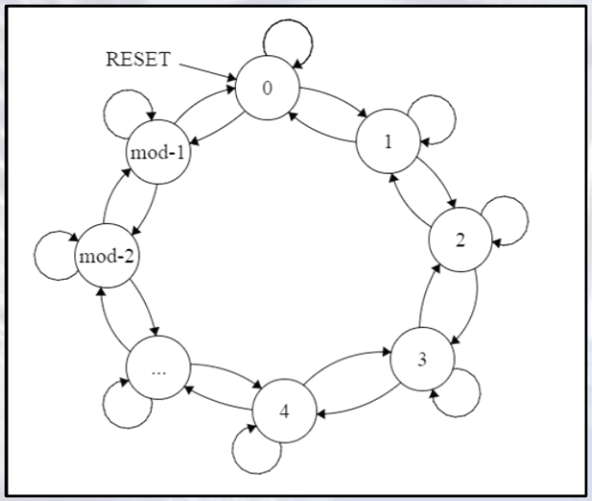
- **Bináris számlálók** (modulus = 2^n)
- **A legfontosabb számláló típus**
- **n bit 2^n állapot: 0, 1, ..., (2^n – 1)**
- **Állapotdiagramok a működési opciókra:**

Egyszerű felfele számláló | **Engedélyezhető felfele számláló** | **Engedélyezhető kétirányú (fel/le) számláló**

i-edik állapotban en=0, marad i-ben, en=1 lép (i+1)-be



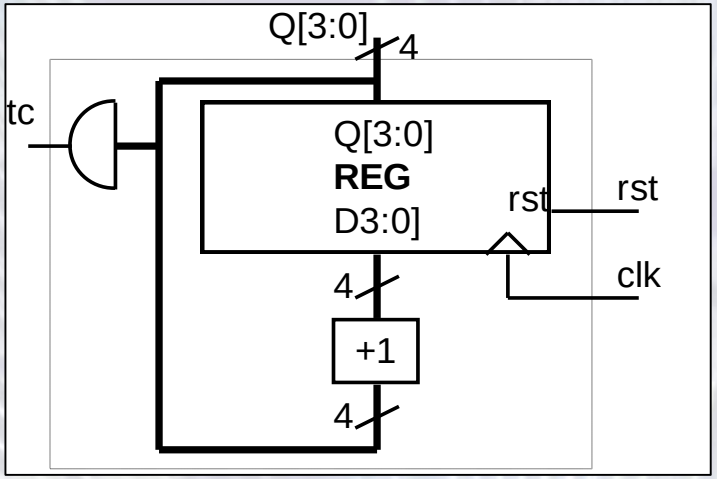




- A tölthetőséget nem rajzoljuk fel, a túl sok állapot átmenet miatt áttekinthetetlen lenne.

Szinkron sorrendi funkcionális egységek számlálók

- **Bináris számlálók**
  - Egyszerű bináris felfele számláló: REG + ***INCREMENTER***
  - ***Az INCREMENTER az n bites bemenetére adott számhoz hozzáad 1-et.*** A rajzon +1-el jelölt doboz.
  - Inkremenete összeadóból is készíthető, pl. az egyik operandusa 0, és a cin bemenetére 1-et kötünk, a másikra a növelendő számot (IN).
    Így a kimenete: SUM = IN + 1
  - A REG értéke minden órajel után REG+1 lesz



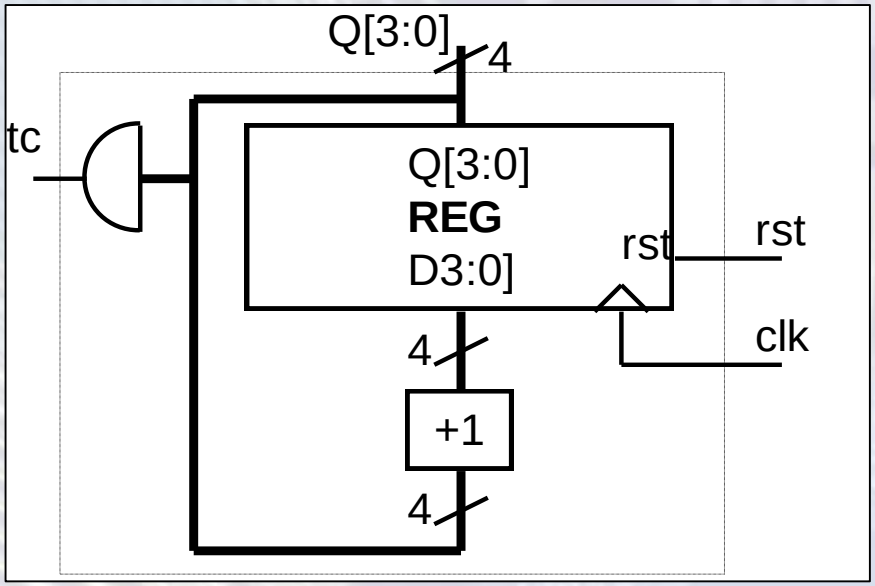
  - A ***tc*** (terminal count) ***a végérték jelzés tc = 1, ha a számláló elérte a végértéket***. Bináris számlálónál minden bitje 1 (itt tc = 1, ha CNT = 1111, tc = Q[3]*Q[2]*Q[1]*Q[0])
  - A bináris számláló a csupa 1 kimeneti érték után átfordul és újra indul 0-ról.

Szinkron sorrendi funkcionális egységek számlálók

- **Egyszerű felfele számláló Verilog kódja:**

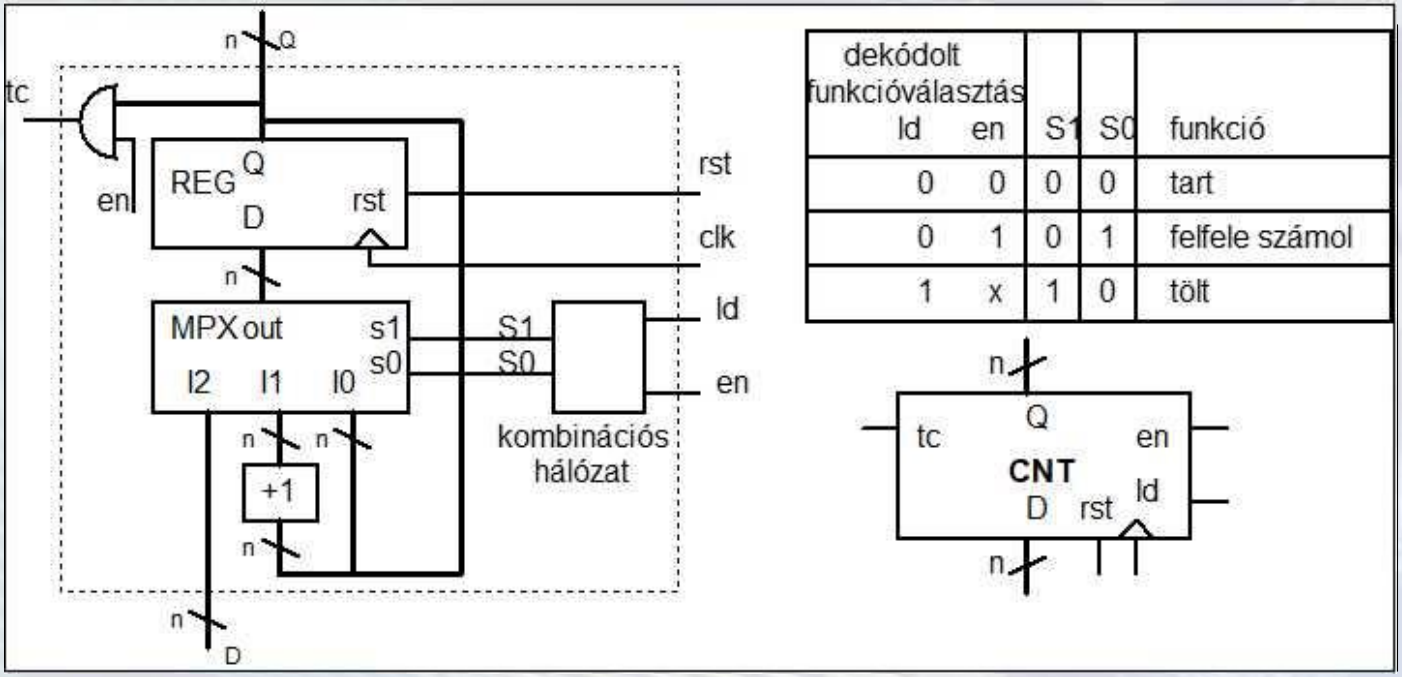
```
module CNT_4(input clk, rst, output [3:0]q, output tc);
 assign tc = &q;              // végértékjelzés

always@(posedge clk)
if(rst) q <= 4'h0;
else q <= q + 4'h1;
endmodule
```



Szinkron sorrendi funkcionális egységek számlálók

- **Bináris felfele számláló, törölhető, tölthető**
- A belső felépítés a tölthető törölhető shiftregiszteréhez hasonlít, csak a számlálónál a funkció az inkrementálás.



dekódolt funkcióválasztás ld	en	S1	S0	funkció
0	0	0	0	tart
0	1	0	1	felfele számol
1	x	1	0	tölt

- ***Engedélyezhető számlálók esetén végérték jelzés csak en=1 esetén van*:**

tc = Q[3]*Q[2]*Q[1]*Q[0]*en

Szinkron sorrendi funkcionális egységek számlálók

Veilog kód (törölhető, tölthető bináris felfele számláló)

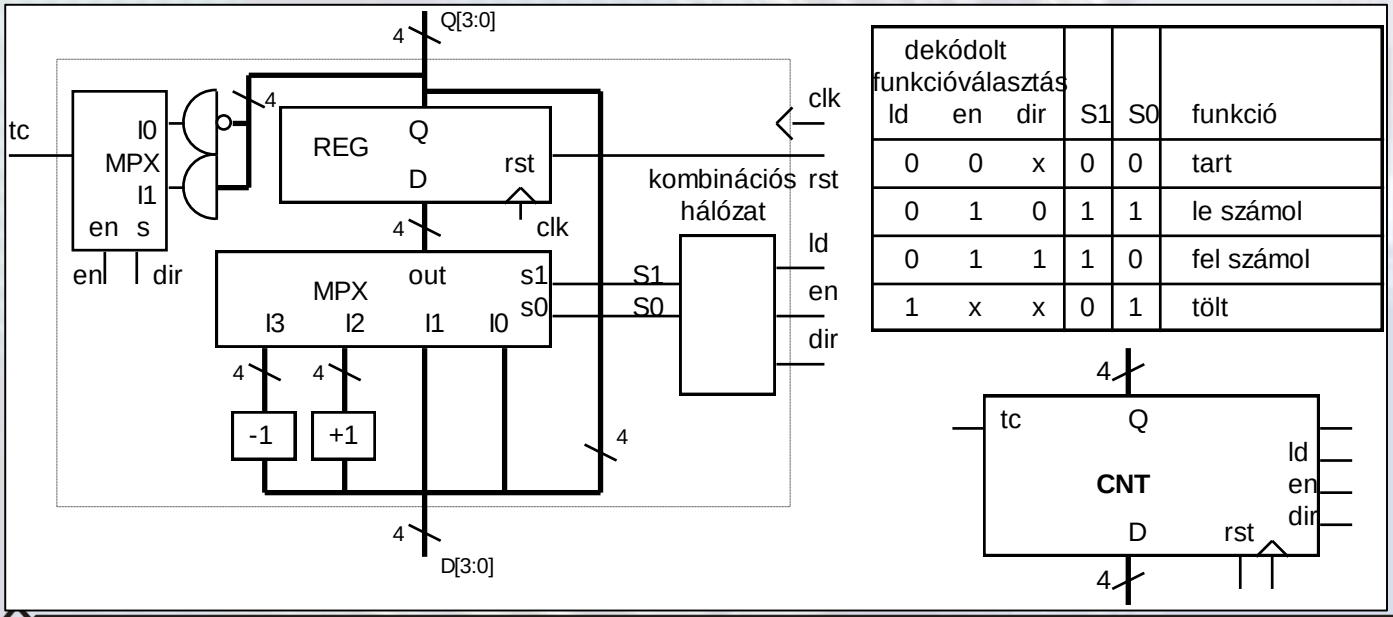
```
module CNT16EN(input clk, rst, ld, en, input [3:0] d,
                        output reg [3:0] q, output tc);
   assign tc = (&q)&en;
    always @(posedge clk)
    begin
    if(rst) q <= 4'h0;   // rst a legnagyobb prioritású jel
    else if(ld) q <= d;
          else if(en)  q <= q + 4'h1;  //felfele számlálás
    end                        // egyéb esetben tartás
endmodule
```

Szinkron sorrendi funkcionális egységek

Törölhető, tölthető bináris fel/le számláló dir irányvezérléssel

- A belső felépítés a kétirányú shiftregiszteréhez hasonló, de a funkciók az inkrementálás és dekrementálás.
- ***A DECREMENTER az n bites bemenetére adott számból kivon 1-et.***
- Felfele számlál, ha **en =1** és ***dir = 1.*** Lefele számlál, ha **en =1** és ***dir = 0.***
- ***A tc most irányfüggő*** (lefele számolásnál a végérték a csupa 0):

tc = en*(dir*Q[3]*Q[2]*Q[1]*Q[0] + /dir*/Q[3]*/Q[2]*/Q[1]*/Q[0])



dekódolt funkcióválasztás ld	en	dir	S1	S0	funkció
0	0	x	0	0	tart
0	1	0	1	1	le számol
0	1	1	1	0	fel számol
1	x	x	0	1	tölt

Szinkron sorrendi funkcionális egységek számlálók

- **Verilog HDL kód törölhető, tölthető engedélyezhető 8 bites bináris fel/le számláló esetén**

```
////////////////////////////////////////////////////////////////////////////////
// A számláló működését viselkedési stílusban írjuk le
// Sok lehetséges üzemmód, összetett vezérlési logika
////////////////////////////////////////////////////////////////////////////////
always @ (posedge clk)                          // Felfutó élre működik
   begin
      if (rst)                q <= 8'b0;        // Jellemző alapérték 0, de
         else                                   // lehetne bármi más is
            if (load)         q <= d;           // Töltés a d bemenetről
               else                             // Ha nincs RST és LOAD,
                  if (en)                       // de a számolás engedélyezett
                     if (dir) q <= q + 8'b1;    // akkor DIR=1 esetén felfelé
                        else  q <= q - 8'b1;    // egyébként lefelé számol
   end

////////////////////////////////////////////////////////////////////////////////
// Végérték jelzés, ha a számlálás engedélyezett és teljesül a végérték feltétel,
// azaz felfelé számlálásnál Q=8'hFF, ill. lefelé számlálásnál Q = 8'h00, akkor tc=1
////////////////////////////////////////////////////////////////////////////////
assign tc = en & (dir ? (&q) : (~|q));
```

- Végérték jelzés: A Verilog HDL redukciós operátorral

tc = en & (dir ? (&q) : &(~q));

Szinkron sorrendi funkcionális egységek számlálók

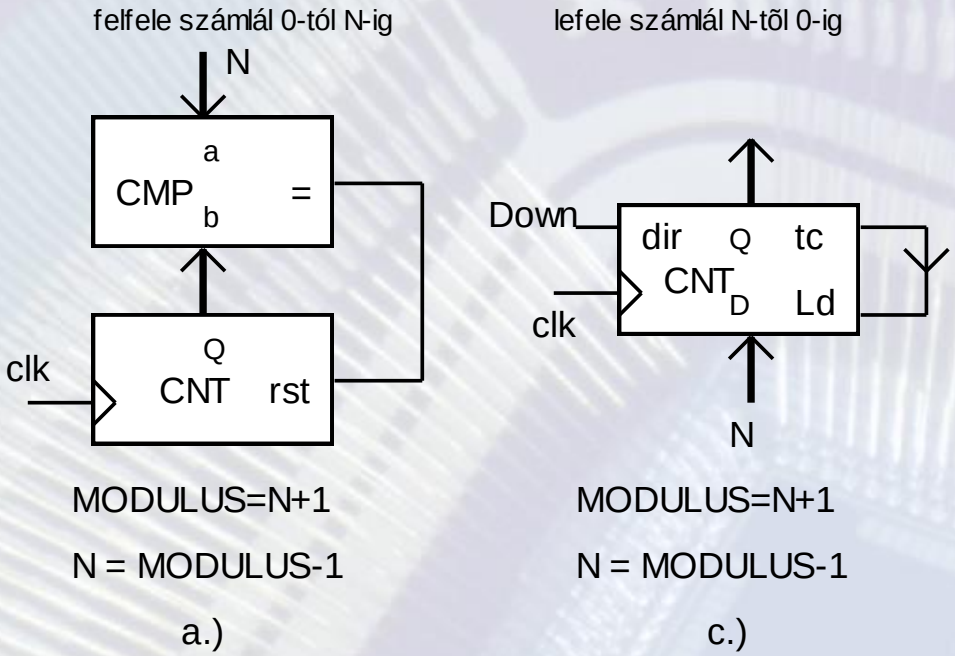
Bináris számlálók modulusának módosítása

- Modulus csökkentés N értékre a 2^n – ről

a. Fefele számláló, ha eléri N-et, töröljük.

b. Lefele számláló, ha eléri a 0-át, betöltünk N= MODULUS-1-et.

- Ha a modulust működés közben nem kell változtatni, akkor N értéke konstans.
- Az első esetben a komparátor ilyenkor egy AND kapura és inverterekre egyszerűsödik (konstans komparátor).

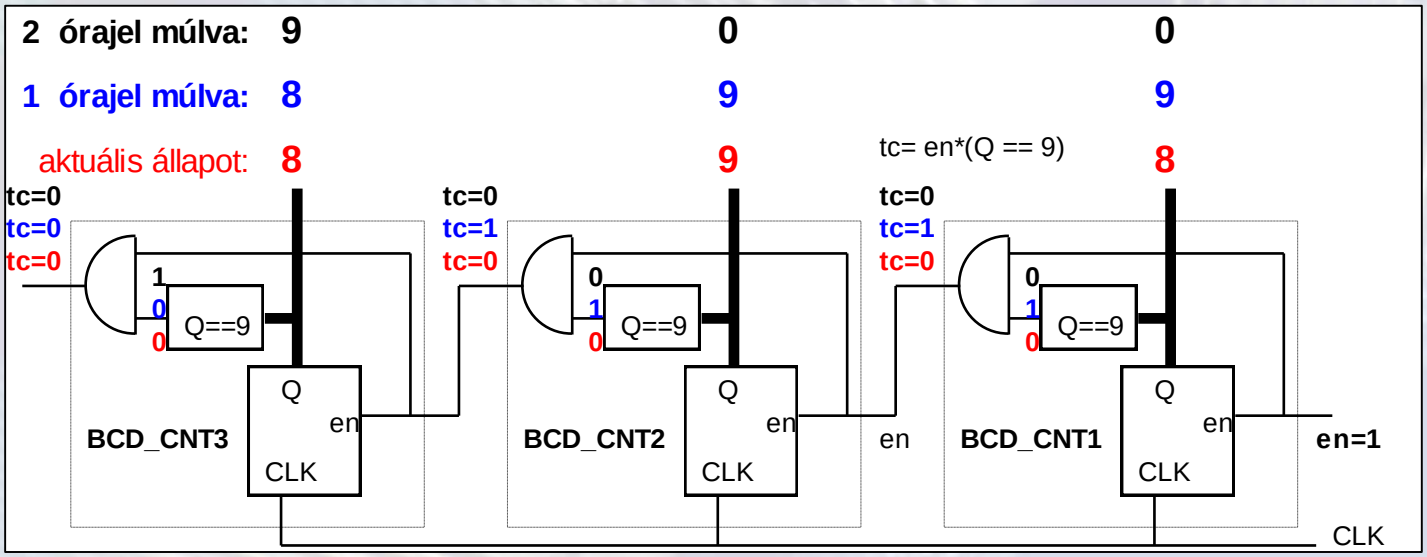


Szinkron sorrendi funkcionális egységek számlálók

- **Modulus növelés**
  - Bináris számlálók esetén Verilog HDL realizációnál egyszerűen msb megfelelő megválasztásával annyi bitszámú számálót csinálunk, amekkorára szükség van. (Nem kisebb modulusúakból állítjuk össze.)
  - A nem bináris számlálók esetén kaszkádosítással (több számláló sorba kapcsolásával) növelhetjük a modulust. Az i-edik számláló akkor kap 1 órajel hosszú engedélyező jelet, amikor az összes előző egység végállapotban van és a számlánc engedélyezett.

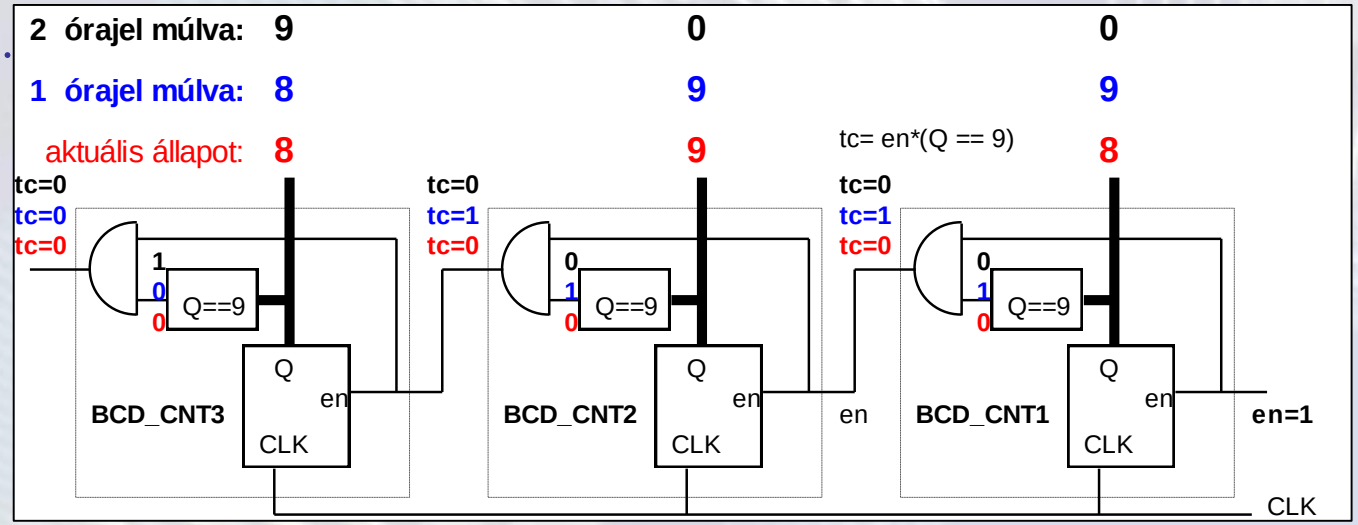
Szinkron sorrendi funkcionális egységek számlálók

- **Példa: BCD számlálók kaszkádosítása**
- Alább 3 db BCD (10-es modulusú számlálót) kaszkádosítottunk.
- Minden számláló **tc (végérték jelző) kimenetét a következő számláló en bemenetére** kötjük és a **számlálók órajele (és ha van a törlő és betöltő jele) közös**.
- Egy tc kimenet akkor aktív, ha a számláló állapota a végérték és engedélyezve van: (Q==9)*en. **A legelső számláló en bemenetével tudjuk engedélyeznei a teljes számláncot.**



Szinkron sorrendi funkcionális egységek számlálók

- A példában a számlánc a 898-es állapotban van.
- Az első számláló nincs végállapotban de engedélyezett, ezért tc kimenete 0. Ezért a 2. és 3. számláló nincs engedélyezve. A következő (1.) órajelre csak az 1. számláló lép, a számlánc tehát a 899-es állapotba kerül.
- Így az 1. és a 2. számláló is végállapotba került, ezért a 1. számláló tc kimenet is 1 lesz ez engedélyezi a 2. számlálót ami végállapotban van így annak tc kimenete is 1 ezért a 3. számláló is engedélyezetté válik. A következő (2.) órajelre mindhárom számláló lép, a számlánc a 900-as állapotba kerül.

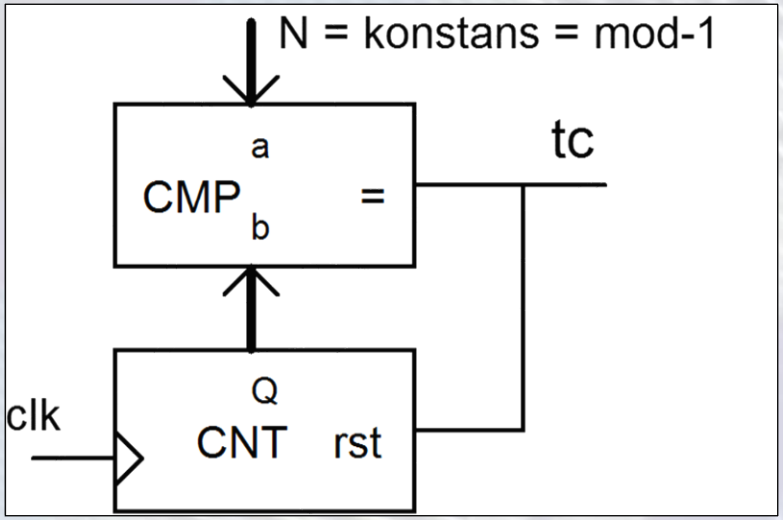


Szinkron sorrendi funkcionális egységek számlálók

- **Bináris számlálók alkalmazásai**
  - Frekvenciaosztás
    - Pulzus kimenet ütemezett működtetésre.
      Pl. egy SHR minden 8-adik órajelre lépjen.
  - Időzítés
    - Adott késleltetés/időzítés után egy pulzus kiadás
    - Adott időtartamú/szélességű pulzus előállítása
  - Pulzusszámlálás
    - Él detektálással az engedélyező bemeneten
  - Általános vezérlési feladatok

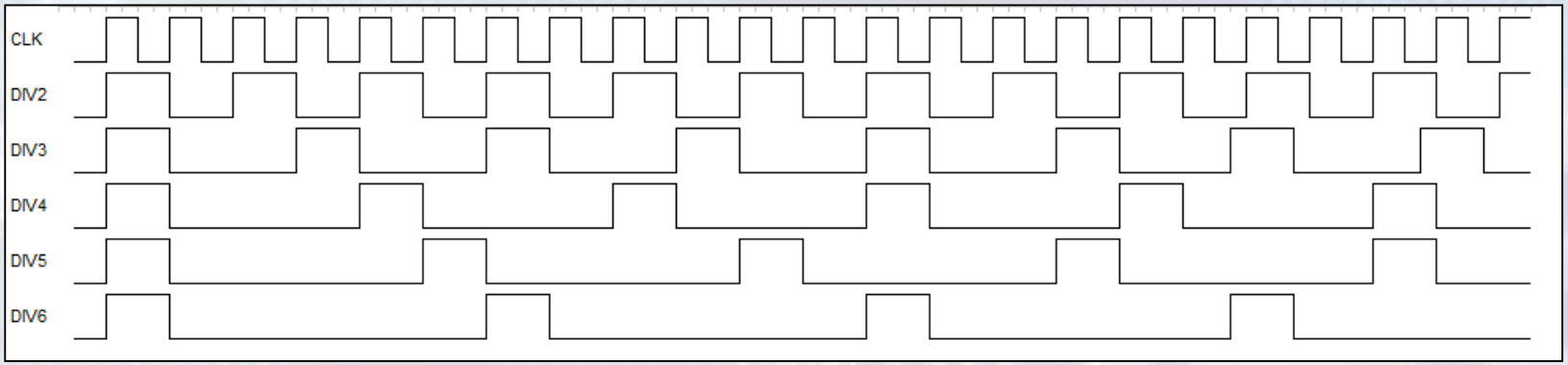
Szinkron sorrendi funkcionális egységek számlálók

- **Frekvenciaosztás bináris felfele számlálóval**
  - Pulzus kimenet előállítása ütemezett vezérléshez
  - A ***mod*** modulushoz a végérték dekódolást ***N=mod-1***-re állítjuk (0-tól *mod-1*-ig számol)
  - TC egy 1 órajel széles pulzus lesz, frekvenciája az órajel frekvencia N+1-ed része: $f_{tc} = f_{clk}/(N+1)$



Szinkron sorrendi funkcionális egységek számlálók

- Tipikus hullámformák kis osztásviszonyra



- **Verilog kód decmális felfele számlálóra (mod = 10) a következő oldalon.**

  A számláló így számol (q[3:0]): 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 0000, ...

Szinkron sorrendi funkcionális egységek számlálók

```
module CNT10EN(input clk, rst, ld, en, input[3:0] d,
                        output reg [3:0] q, output tc);
   assign ill = q > 4'h9;              // illegális állapotok
   assign tc = (q==4'd9) & en;   //modulus = 10 (0-tól 9-ig számol)
   always @(posedge clk)
   begin
         if (rst | ill) q <= 4'd0; // reset és illegállis állapot esetén törlés
         else if (ld) q <= d;
                  else if (en)
                           if (tc) q <= 4'd0;    //modulus csökkentés
                           else q <= q + 4'd1; //felfele számálás
    end
endmodule
```

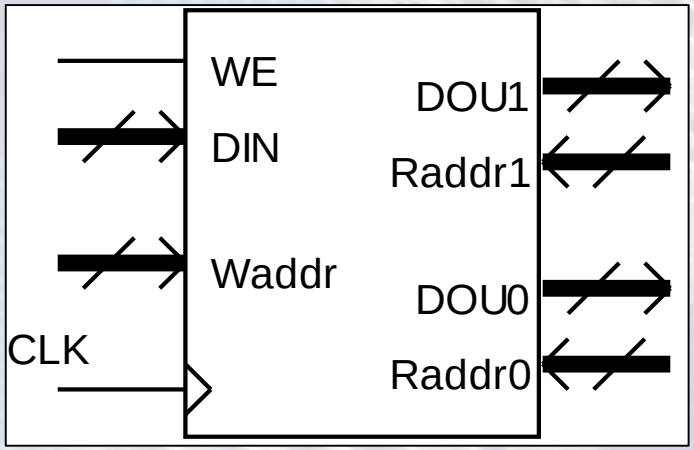
Speciális tároló funkcionális egységek

- **Eddig csak egyetlen adat tárolására alkalmas funkcionális elemekkel foglalkoztunk.**
- **Vannak több *regisztert* tartalmazó, gyakorlatban fontos adattárolók.** (Technológilag ténylegesen regiszeterekből vannak kialakítva. Viszonylag kevés 4...64 regisztert tartalmaznak.)
- **Regisztertömbök**
- Memóriák (RAM, ROM)
- STACK (LIFO)
- FIFO

A 6. előadás elmaradása miatt a **memóriák, STACK, FIFO** a 7. laborgyakorlat elején lesz ismertetve.

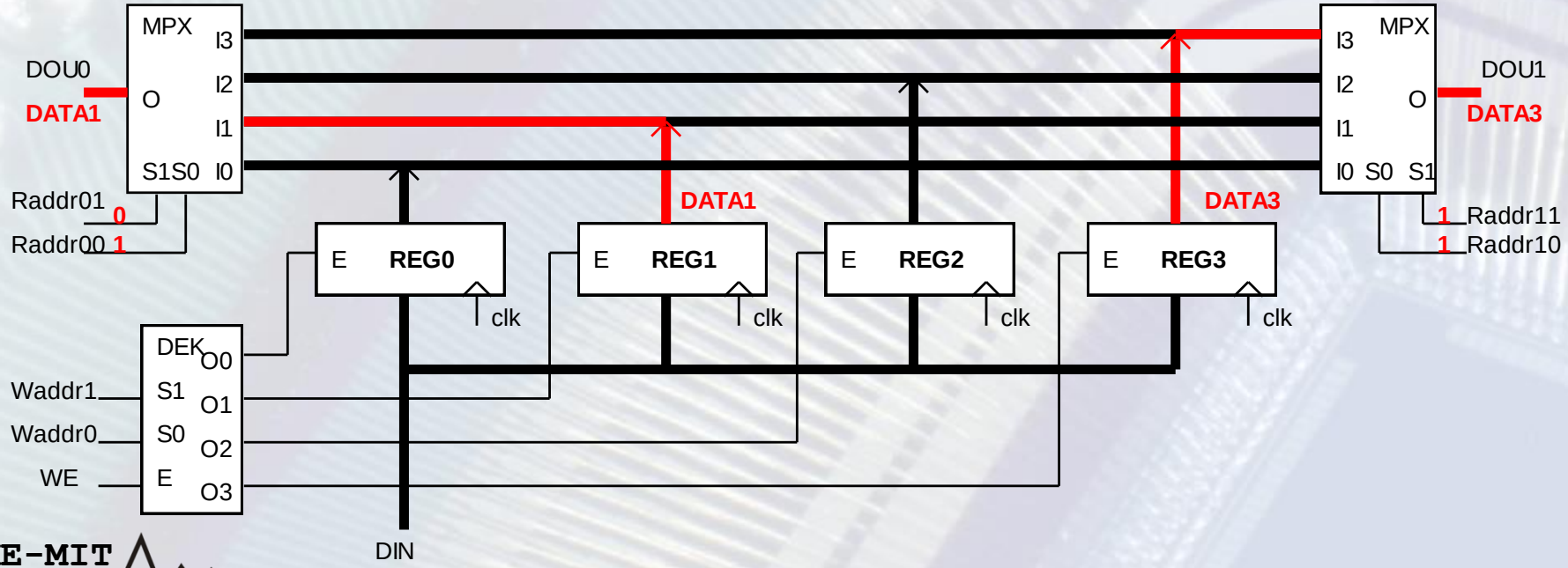
Funkcionális egységek regiszter tömb

- A regiszter tömb adat bemente a **DIN**, erre kel ráadni a beírandó adatot. Az adat a **Waddr** bemenetre adott című rekeszbe íródik. A beírást a **WE** jel magas szintje engedélyezi és a clk felfutó élére íródik be. A kiolvasandó adat címét az Raddri bemenetre kell ráadni. A megcímzett adat a DOUi kimeneten jelenik. Az adat kimenet (és olvasási címek) száma 1 vagy 2. Alább egy 2 címes változat kapcsolási rajz szimbóluma látható.



Funkcionális egységek

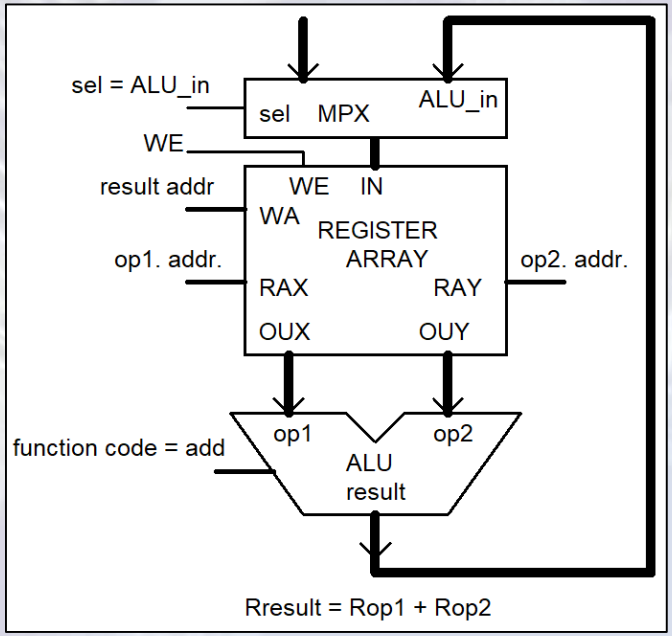
- **Regisztertömb belső felépítése egy írási és két olvasási port esetén**
- Egyszerre 2 regiszter olvasása
- Pl. Ha Raddr1 = 1, Raddr2 = 3. akkor Dou0-on a REG1, DOU1-en pedig REG3 tartalma jelenik meg.



Funkcionális egységek regiszter tömb

- **Regisztertömb alkalmazási példa:** processzor műveletvégző egysége
- A műveletet az ALU (Arithmetic Logic Unit) végzi el.
- Az **ALU** olyan kombinációs hálózat, mely a ***funkció*** **bemenetén kiválasztott műveletet végzi el az *op1* és *op2* operandusokon és a művelet eredménye a *result* kimenetén jelenik meg.** A funkció bemenet annyi bites, ahánnyal az összes művelet kiválasztható ($\log_2$(műveletek száma) felkerekítve).
- A 2 oprendus egyszerre kiválasztható a 2 olvasási címmel rendelkező regisztertömbből. A művelet eredménye egy tetszőleges 3. regiszterbe beírható.
- Ezt nevezik ***3 regiszter címes architektúrának***. Pl. op1 addr = 1, op2 addr =2, result addr =3, Funkció: összeadás

R3 = R1 + R2



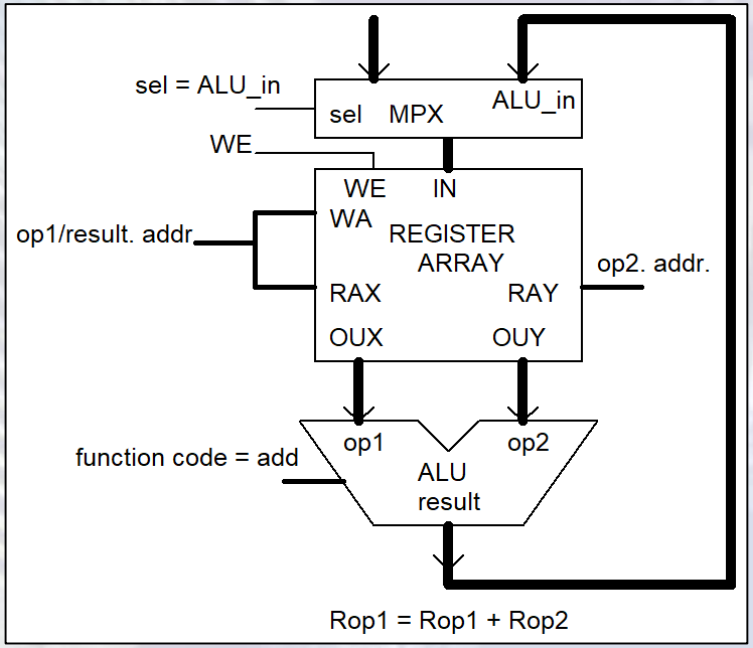
Funkcionális egységek regiszter tömb

- Ha az ***op1 addr*** és ***result addr*** bemeneteket párhuzamosan kötjük, akkor az eredmény felülírja az egyik operandust, azonban kevesebb bemenet szükséges a vezérléséhez.
- Ezt nevezik ***2 regiszter címes architektúrának***.

Pl. op addr/result addr = 1, op2 addr =2

Funkció: összeadás

R1 = R1 + R2



Digitális technika
5. EA vége