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

Digitális technika
VIMIAA02
5. EA

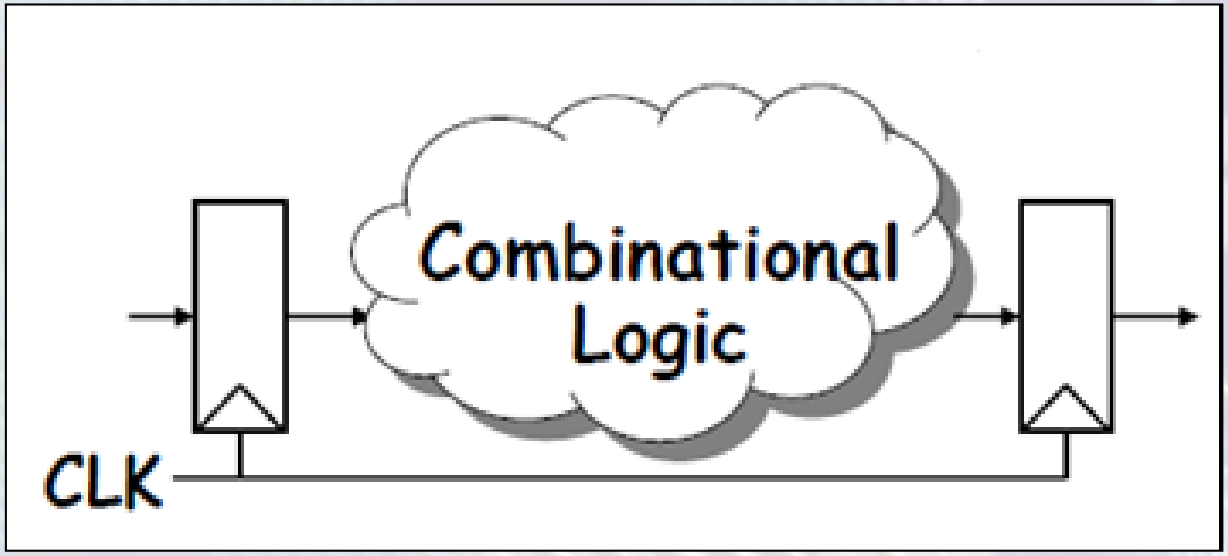
Fehér Béla, Benesóczky Zoltán
BME MIT

COVID-19 előadások

- **Ebben az évben az előadások további két teremben is, kivetítőről követhetők, a kapcsolat így nem elég közvetlen**
- **Ennek javítására beiktatunk visszajelzési lehetőséget**
- **Bármelyik teremből kérdések, megjegyzések emailben küldhetők a benes@mit.bme.hu címre, de csak ... @edu.bme.hu címről, tárgy: DIGIT Résztémák végén a kérdéseket megnyitjuk és próbálunk válaszolni**

Szinkron sorrendi logikák működési feltételei

- **Adott számú DFF párhuzamos működtetése, közös, globális, időben és térben egyidejű CLK órajellel történik** (az összes DFF órajele nagyon pontosan azonos, az aktív él egyszerre vált).

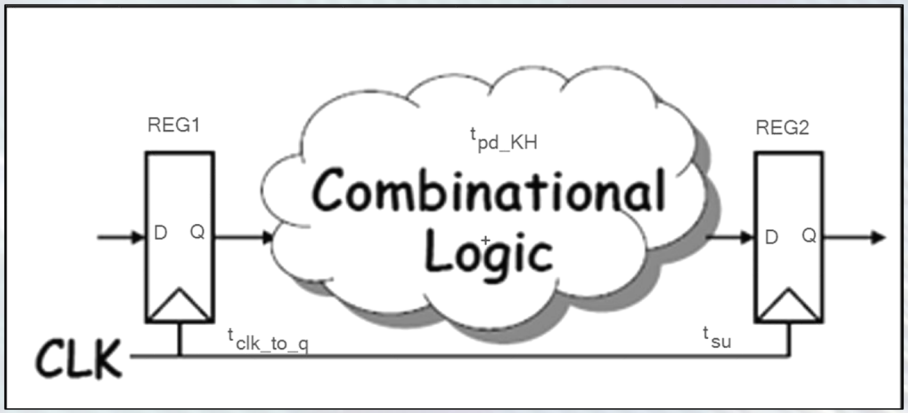


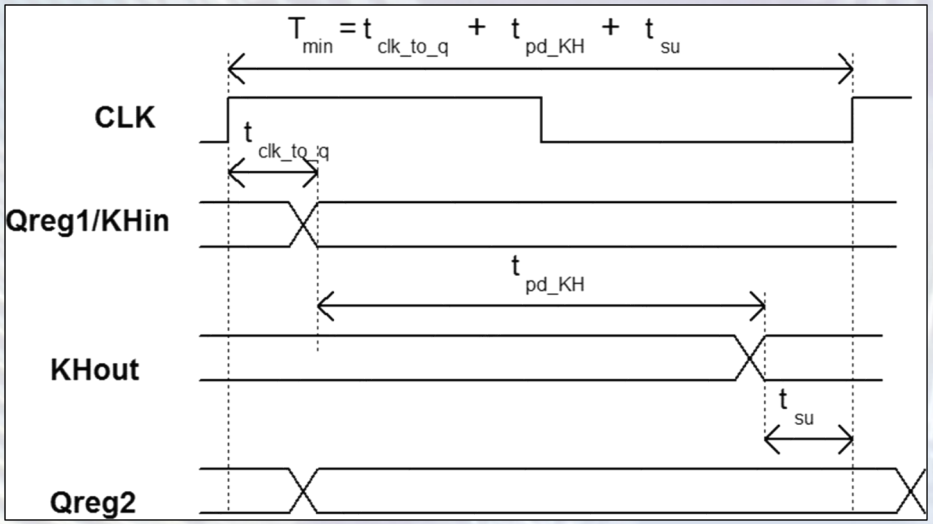
- **A műveleteket (számításokat) a kombinációs logika végzi.**

Szinkron sorrendi logikák működési feltételei

A szinkron digitális tervezői paradigma biztosítása

- Két órajel között elegendő idő kell a feladatok elvégzésére. (A kombinációs hálózat műveletvégzésére és a DFF-ok beírására.)



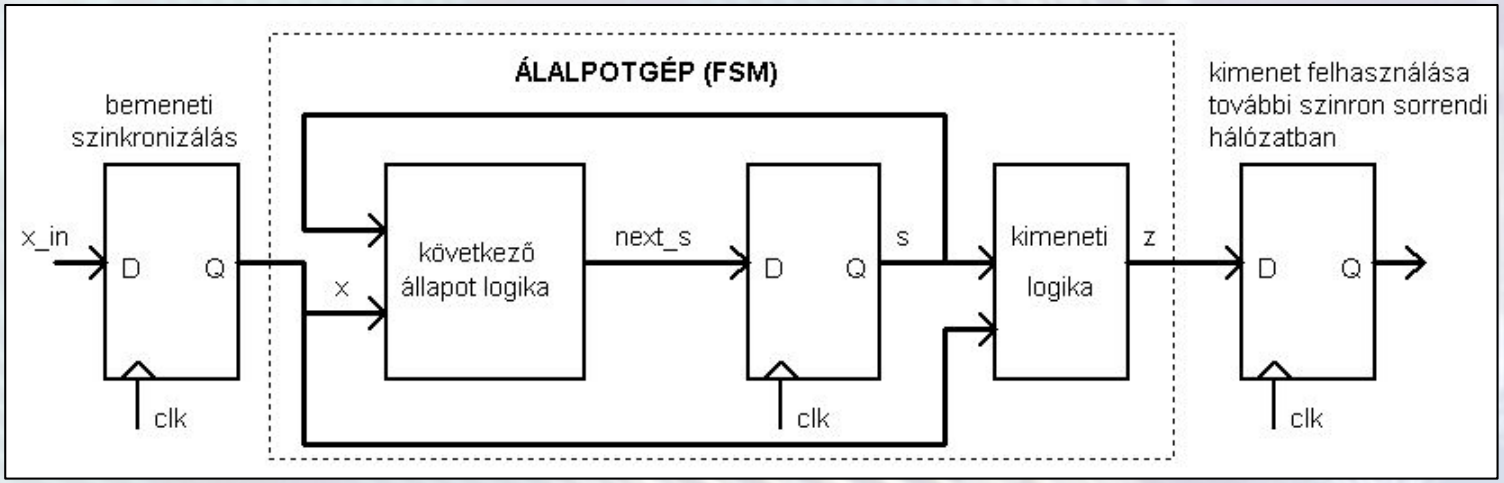


Szinkron sorrendi logikák működési feltételei

- **Az idődiagram magyarázata**
  - Az órajel aktív éle után kis késéssel ($t_{clk\_to\_q}$) megváltozik az 1. regiszter kimenete, vagyis a kombinációs hálózat bemenete.
  - A kombinációs hálózaton keresztül is véges idő alatt terjed a jel (t_{pd_KH}). A jelterjedési idő logikai áramkörök sebességétől és a kombinációs hálózat megvalósításától (pl. 2 szintű vagy többszintű hálózat) is függ. Itt a leghosszabb jelterejedési idő érdekes.
  - Ha megfelelően nagy az órajel periódusideje (T_{clk} >= Tmin), akkor a következő aktív éle előtt előírt idővel (t_{su}, adat előkészítési idő) a 2. regiszter bemenetén már stabil a kombinációs hálózat kimenetéről érkező adat és ez az él hatására hibátlanul beíródik a regiszterbe (ha még megfelelő ideg stabil marad). Ennél jobban nem részletezzük.
- **Két órajel él között stabil kimeneti értékek biztosítása.**
  - A regiszter biztosítja, ha az előbbi feltételek teljesülnek. Ha a bemenetén a beírandó adat az aktív órajel él előtt megfelelő idővel már stabil (és utána is megfelelő ideig még stabil). Egyébként hibás értéket tárolhat. (Fontos az időzítési előírások betartása.)

Szinkron sorrendi logikák működési feltételei

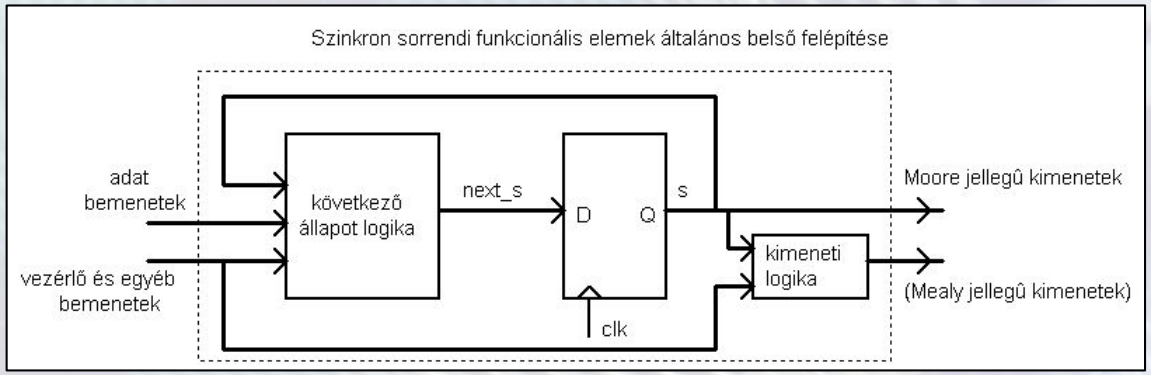
- A ***szinkron álapotgép*** (FSM) esetében ***a kombinációs hálózat kimenete ugyanazon regiszter bemenetére (is) kapcsolódik, mint amely a bemenetét meghajtja***:



- A bemenetét az órajelhez kell szinkronizálni, így tarthatók be az időzítési követelmények

Szinkron sorrendi funkcionális egységek

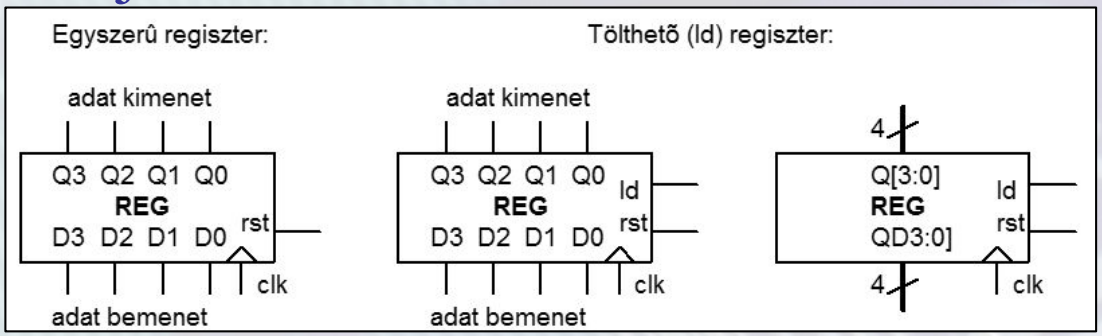
- A szinkron funkcionális egységek a kombinációs funkcionális elemekhez hasonlóan ***gyakran használt többnyire komplex sorrendi hálózatot igénylő funkciót valósítanak meg***.
- Ezek tipikus, ***a megvalósított funkcióhoz illeszkedő be és kimenetekkel rendelkeznek***
- Általános belső felépítésük:



- 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

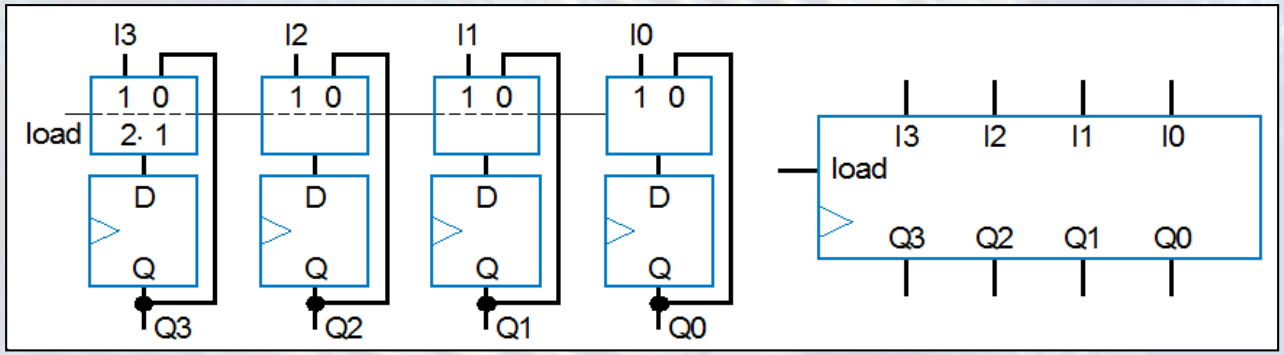
- **REGISZTER** (már ismerjük)
- Kapcsolási rajz szimbólumai:



- **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 (CE = 0)

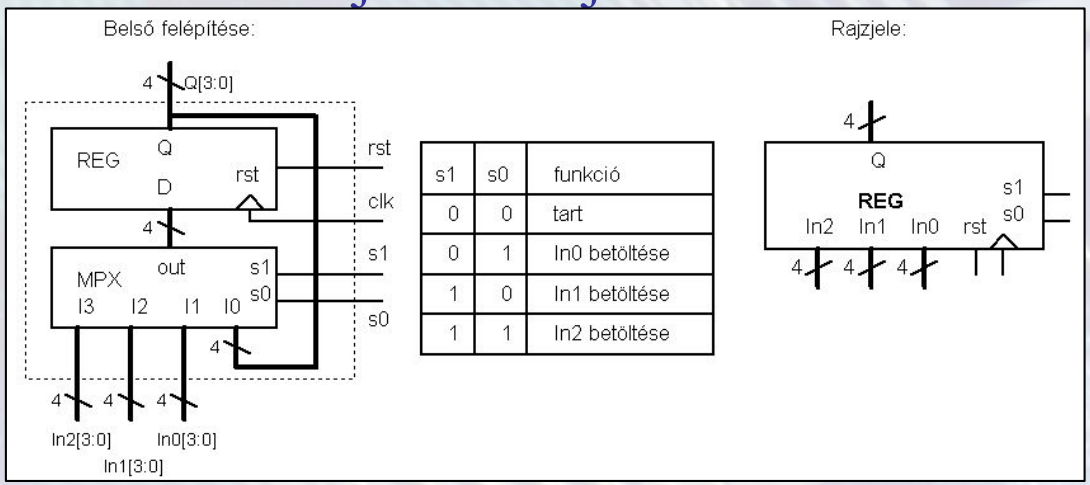
Szinkron sorrendi funkcionális egységek

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



Szinkron sorrendi funkcionális egységek

- **Multifunkciós regiszter**
  - ***A multifunkciós regiszter választhatóan több forrásból képes adat betöltésére*** (a tartás funkció mellett). Ehhez ***busz multiplexer*** kell (be és kimenetei több bitesek).
    Kódolt funkció választású verzió esetén a MPX kiválasztó jeleinek kombinációja választja a funkciót.



  - 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

Verilog kód kódolt funkció választású multifunkciós regiszterhez

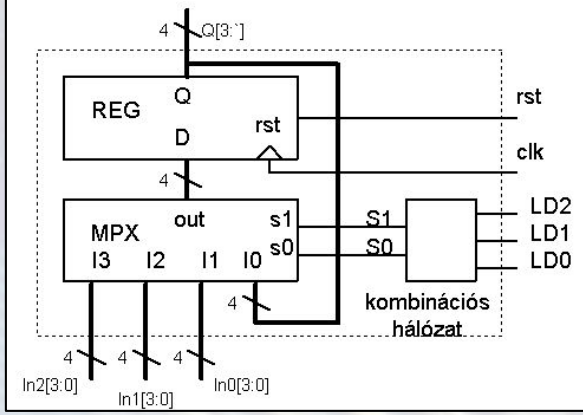
```
module MFREG3(input clk, rst, input [1:0]s,
                    input [3:0] in0, in1, in2, output reg [3:0] q);
always@(posedge clk)
if(rst) q <= 4'h0;
else
    case(s)
    2'b01: q <= in0;
    2'b10: q <= in1;
    2'b11: q <= in2;
    endcase          // egyéb s[1:0] bemenetnél tartás, mert nem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

Szinkron sorrendi funkcionális egységek

- Sokszor egyszerűsíti a tervezést, ha ***dekódolt betöltő jelek*** állanak rendelkezésre, vagyis az egyes csatornák betöltését egyetlen jel vezérli. (Pl. LD0 =1 hatására In0 töltődik be.) Ehhez a betöltő jelek (LD0, LD1, LD2) és a multiplexer kiválasztó bemenetei (S1, S0) közé egy kombinációs hálózatot kell tenni. Itt 3 betöltő jel (LD0, LD1, LD2) esetén rajzoltuk fel a belső felépítést és igazságtáblát. Az igazságtáblában x-el a bit közömbös (tetszőleges) értékét jelöltük.

Multifunkciós regiszter dekódolt vezérlőjelekkel

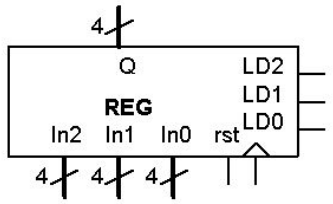
Belső felépítése



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

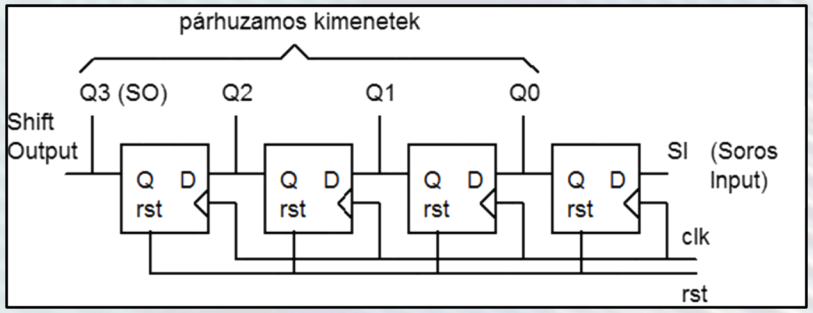
Verilog kód dekódolt funkció választó jelekkel multifunkciós regiszterhez

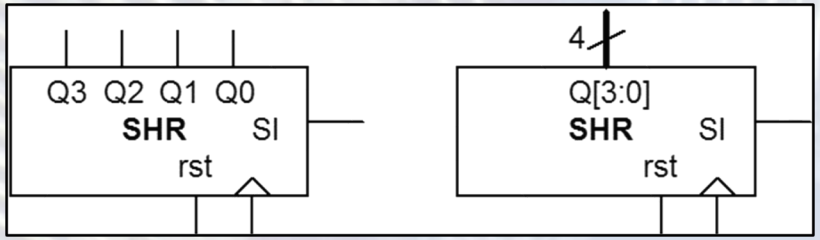
```
module MFREG3(input clk, rst, ld0, ld1, ld2,
                        input [3:0] in0, in1, in2, output reg [3:0] q);
always@(posedge clk)
    begin
    if(rst) q <= 4'h0;     // rst a a legnagyobb prioritású jel
    else if(ld0)  q <= in0;          // ld0 a 2. legnagyobb prioritású
          else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

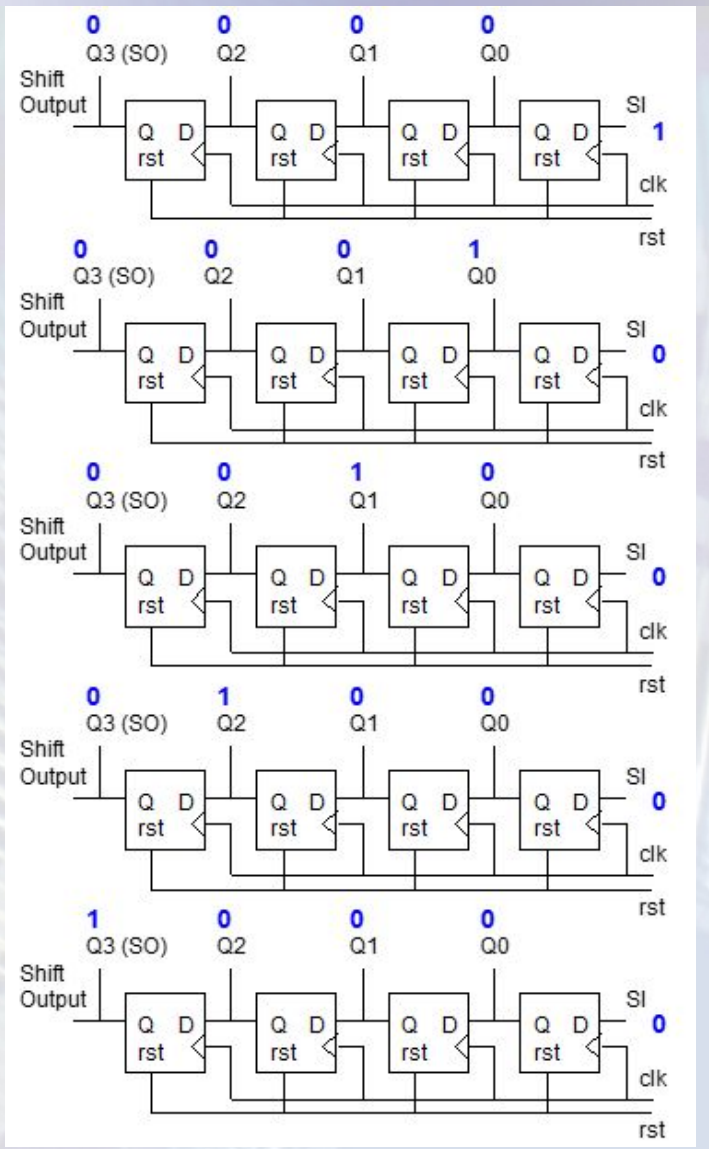




Szinkron sorrendi funkcionális egységek, shiftregiszter

Példa: Ha az eredetileg törölt SHR shift inputjára az 10000 sorozatot adjuk az órajellel szinkronban, akkor az 1-es az első órajelre beíródik, majd minden órajel után eggyel jobbra shiftelődik. (Az SI bemeneten a további bitek mind 0-ák, ezért csak 1 db 1-es lép be.) ***Verilogban a shiftelést konkatenálással adjuk meg.*** (A Verilog shift operátora 0-át léptet be, ezért nem megfelelő.)

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

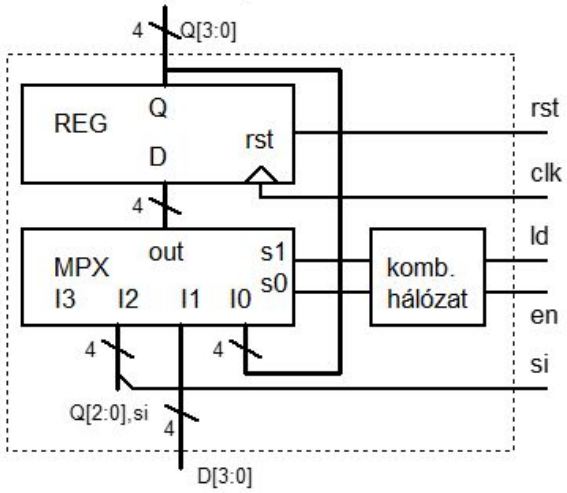


Szinkron sorrendi funkcionális egységek, shiftregiszter

- 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***.
- Belső felépítése multifunkciós regiszterrel megvalósítva:

Tölthető, engedélyezhető shiftregiszter

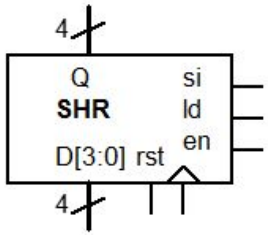
Belső felépítése:



A multiplexer select jeleit előállító kombinációs hálózat igazságtáblája

dekódolt funkcióválasztás 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)

Rajzjele:



Szinkron sorrendi funkcionális egységek, shiftregiszter

Verilog kód egyszerű törölhető, tölthető, engedélyezhető SHR-hez

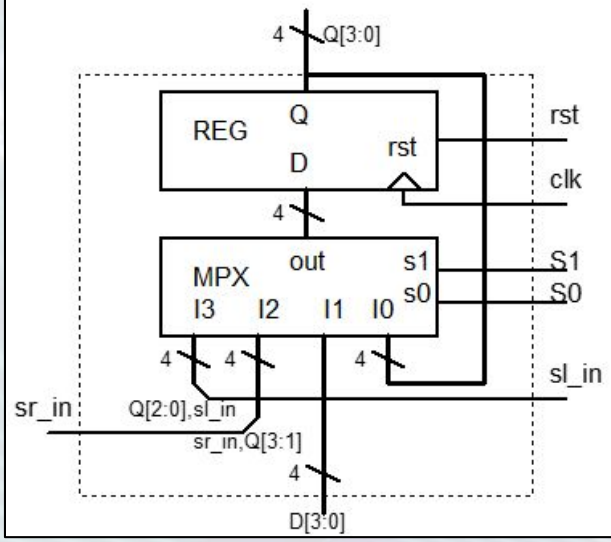
```
module SHR4EN(input clk, rst, ld, en, si,
                        input [3:0] d, output reg [3:0] q);
    always@(posedge clk)
    begin
      if (rst) q <= 4'h0;       // rst a legnagyobb prioritású jel
      else if (ld) q <= d;      // utána a betölés
           else if (en) q <= {q[2:0], si};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

Szinkron sorrendi funkcionális egységek, shiftregiszter

- A shiftregiszter kétirányú is lehet. Először a ***kódolt vezérlő jeles*** megvalósítást mutatjuk: ***S[1:0] bináris funkciókódok***
  - tart / tölt / jobbra shiftel / balra shiftel

Tölthető, engedélyezhető kétirányú shiftregiszter kódolt vezérlőjelekkel

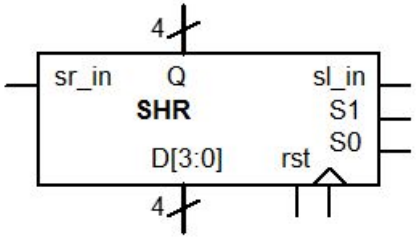
Belső felépítése:



A multiplexer select jeleit előállító kombinációs hálózat igazságtáblája

S1	S0	funkció
0	0	tart (Q[3:0] betöltése)
0	1	tölt (D[3:0] betöltése)
1	0	jobbra shift (sr_in,Q[3:1]) betöltése
1	1	balrra shift (Q[2:0],sl_in) betöltése

Rajzjele:



- Az shr_in és shl_in a jobbra, ill. balra shifteléshez tartozó adatbemenetek.

Szinkron sorrendi funkcionális egységek

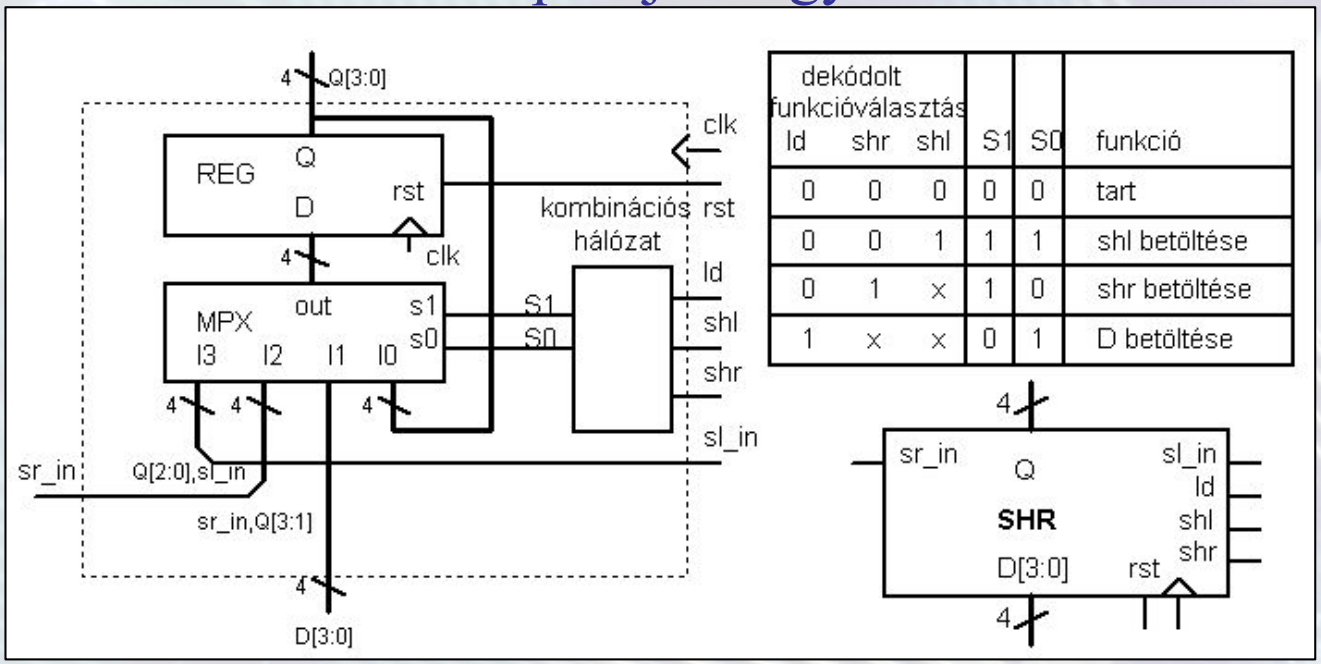
Verilog kód kétirányú kódolt vezérlésű SHR-hez

```
module SHR4_RLC(input clk, rst, sl_in, sr_in, input [1:0] s,
                                input [3:0] d, output reg [3:0] q);
    always@(posedge clk)
    if(rst) q <= 4'h0;
    else
        case(s)
          2'b01: q <= d;                    //betöltés
          2'b10: q <= {sr_in, q[3:1]};      // jobbra shift
          2'b11: q <= {q[2:0], sl_in};      // balra shift
        endcase                             // egyéb s[1:0] bemenetnél tartás
endmodule                                   // (2'b00) mert q nem változik
```

Nincs prioritás, egyenragúak a parancsok.

Szinkron sorrendi funkcionális egységek

- **Kétirányú shiftregiszter funkciók dekódolt vezérlőjelekkel**
- ld: tölt, shr: jobbra shift, shl: balra shift
  - A használat szempontjából gyakran kedvezőbb



dekódolt funkcióválasztás					
ld	shr	shl	S1	S0	funkció
0	0	0	0	0	tart
0	0	1	1	1	shl betöltése
0	1	x	1	0	shr betöltése
1	x	x	0	1	D betöltése

A dekódolt jelek között prioritás van → ld > shr > shl

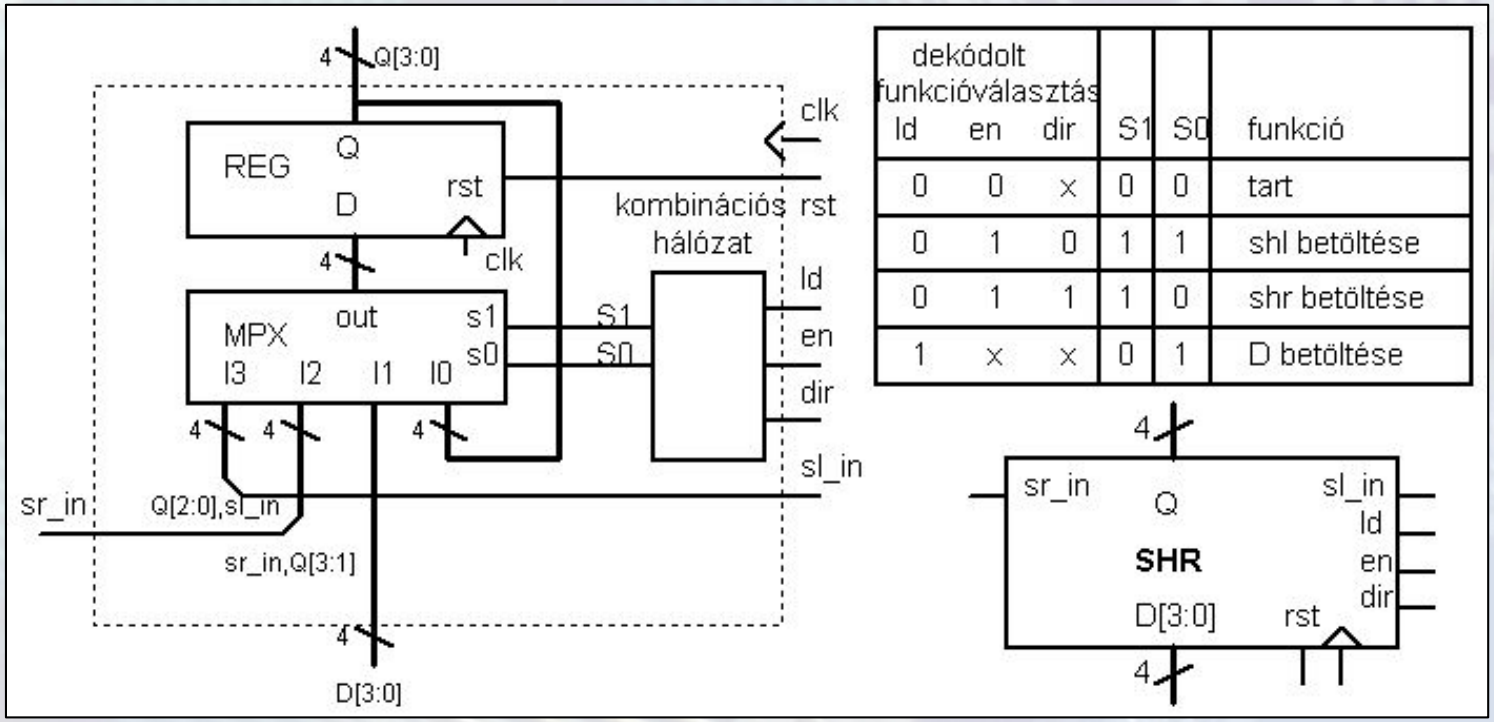
Szinkron sorrendi funkcionális egységek

Veilog kód dekódolt vezérlésű (ld, shr, shl) **kétirányú shiftregiszterhez**

```
module SHR4LRD(input clk, rst, ld, shl, shr, sl_in, sr_in,
                        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;       // ld a 2. legnagyobb prioritású
            else if (shr) q <= {sr_in, q[3:1]};    // jobbra shift eng.
                else if(shl) q <= {q[2:0], sl_in}; // balra shift eng.
    end                                            // egyéb esetben tartás
endmodule
```

Szinkron sorrendi funkcionális egységek

- **Shiftregiszter Verilog HDL kódolása**
- **Szétválasztott vezérlőjelekkel, de más stílusban**
  - Töltés (ld), léptetés engedélyezés (en), irányvezérlés (dir) (en =1 és dir = 0 balra shift, en =1 és dir = 1 jobbra shift)



dekódolt funkcióválasztás					
ld	en	dir	S1	S0	funkció
0	0	x	0	0	tart
0	1	0	1	1	shl betöltése
0	1	1	1	0	shr betöltése
1	x	x	0	1	D betöltése

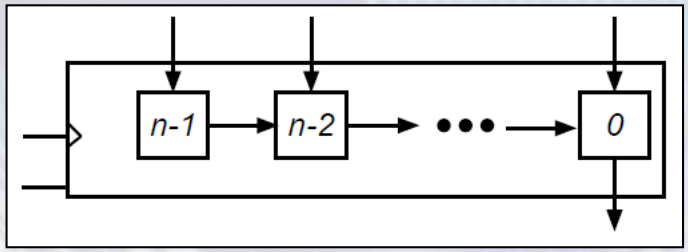
Szinkron sorrendi funkcionális egységek

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

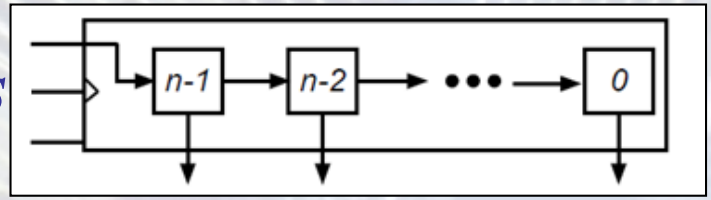
```
module SHR4_RLC(input clk, rst, en, dir, sl_in, sr_in,
                        input [3:0] d, output reg [3:0] q);
    always@(posedge clk)
    if(rst) q <= 4'h0;
    else if(ld) q <= d;    //betöltés
        else if(en)
            if (dir) q <= {sr_in, q[3:1]};  // jobbra shift
            else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

Szinkron sorrendi funkcionális egységek

- **Shiftregiszter 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 elölről



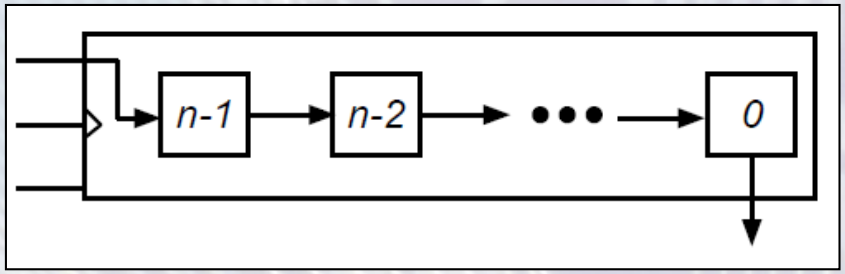
  - ***Soros-párhuzamos átalakítás***
    - A shift inputon időben egymás után jövő biteket beléptetjük az SHR SIN bemenetén
    - Ha az összes bit bejött, átírjuk egy másik regiszterbe, ahol ezután az összes bit egyszerre rendelkezésre áll



Szinkron sorrendi funkcionális egységek

- **Shiftregiszter alkalmazási területek**
- ***Soros 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).

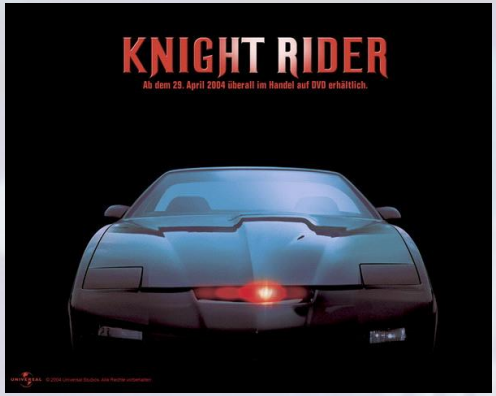


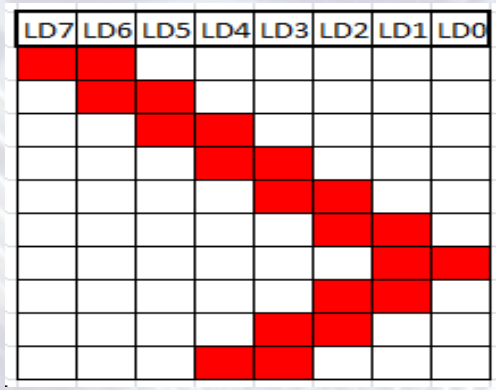
- ***Szorzás (osztás) 2 hatvánnyal***

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

Szinkron sorrendi funkcionális egységek

- **Shiftregiszter alkalmazási területek**
- **Tetszőleges alkalmazás**
  - Knight-Rider LED játék
  - Vezérlő logika kell. Kezdetben egy LD = RST pulzus és utána periodikus DIR vezérlés. Ezt generálhatjuk a LED-ek állapota alapján egy kétállapotú Mealy vagy Moore vezérlővel. (A Mealy vezérlő kimenetének már a következő órajelnél lesz hatása, a Moore vezérlőnek eggyel később.)



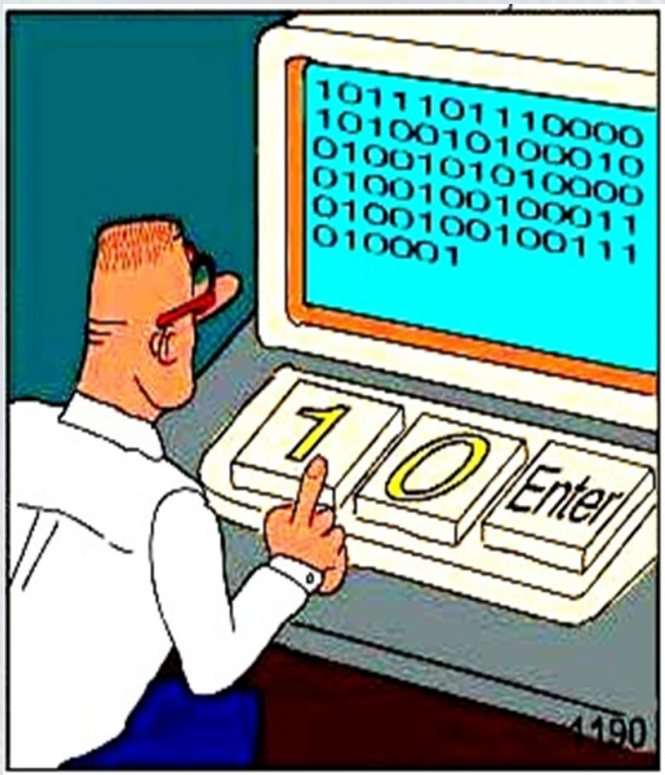



```
reg [7:0] sr;                   // 8 bit shiftregiszter
wire ld, en, dir;               // Egyedi vezérlőjelek

always @ (posedge clk)
   if (ld)              sr <= 8'b11000000;        // Kezdőállapot
   else if (en)                                   // LÉptetés ütemezés
           if (dir)     sr <= {1'b0, sr[7:1]};    // 6 lépés jobbra
           else         sr <= {sr[6:0], 1'b0};    // 6 lépés balre
```

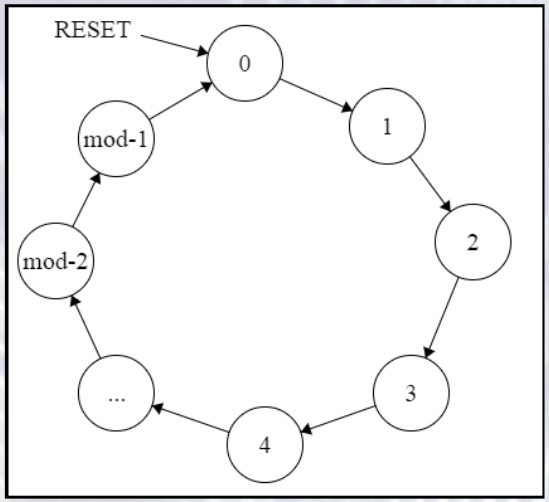
Kérdések, észrevételek

- **A kérdéseket a benes@mit.bme.hu címre várjuk de csak ... @edu.bme.hu címről, tárgy: DIGIT**



Szinkron sorrendi funkcionális egységek

- **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,….



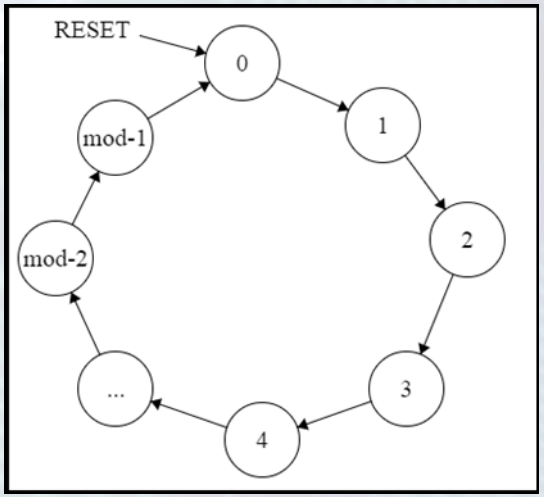
  - ***Hagyományos lefele számlálókra ugyanez fordított sorrendben.*** 7,6,5,4,3,2,1,0,7,6,5,4,....
  - Általános esetben egy N modulusú számláló állapot kódolása tetszőleges N darab azonos bitszámú, különböző érték lehet.

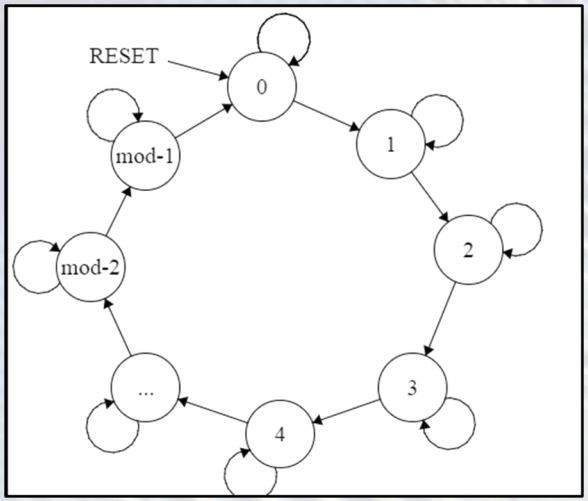
Szinkron sorrendi funkcionális egységek

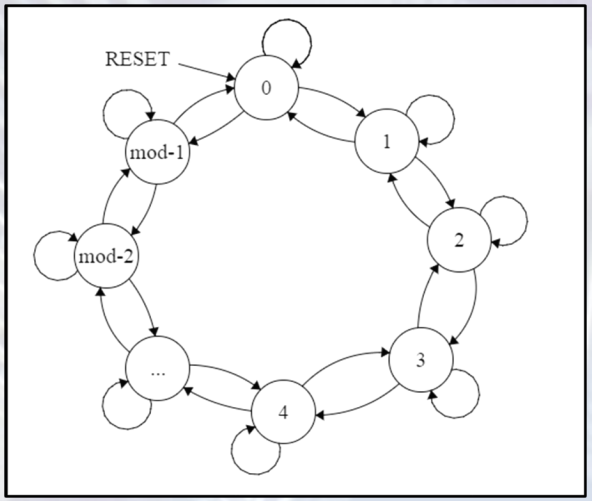
- **Számlálók**
  - Tetszőleges ciklikus állapotgépek
  - (nem tananyag: Shiftregiszter alapú számlálók)
    - (Gyűrűs, Johnson, LFSR)
  - Bináris modulusú számlálók
    - Bináris kódú, (nem tananyag: Gray kódú)
  - Nem bináris modulusú
    - (BCD (0..9), egyéb)

Szinkron sorrendi funkcionális egységek

- **Bináris számlálók** (modulus = 2^n)
- **A legfontosabb számláló típus**
- **n bit 2^n állapot: 0 → $(2^n - 1)$**
- **Állapotdiagramok a működési opciókra:**
- **Egyszerű** **Engedélyezhető** **Kétirányú, eng.**



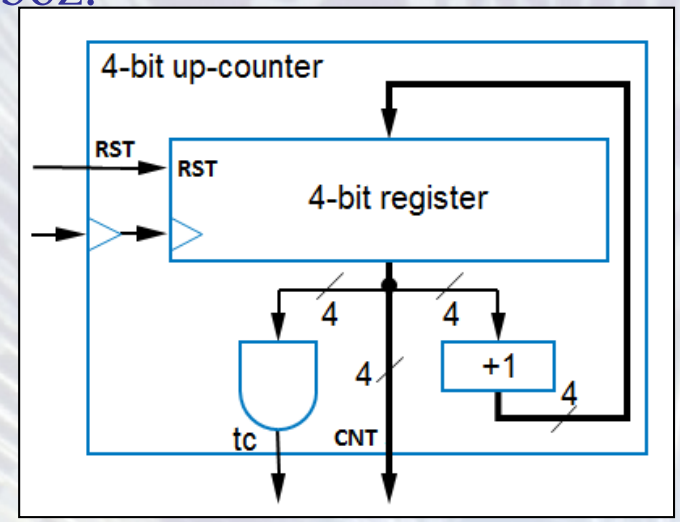




- A tölthetőséget nem rajzoljuk fel, túl sok állapot átmenet áttekinthetetlen lenne.

Szinkron sorrendi funkcionális egységek

- **Bináris számlálók**
- **Általános modell**
  - Egyszerű bináris felfele számláló: REG + ***INCREMENTER***
  - ***Az INCREMENTER az n bites bemenetére adott számhoz hozzáad 1-et.*** A rajzon +1-el jelölt doboz.
  - Összeadóból is készíthető, az egyik operandusa 0, és a carry bemenetére (Ci) 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 a végérték jelzés tc = 1, ha a számláló elérte a végértéket.*** Bináris számlálónál minden bitje 1 (itt tc = 1, ha CNT = 1111)
  - A bináris számláló a csupa 1 kimeneti érték után átfordul és újra indul 0-ról.

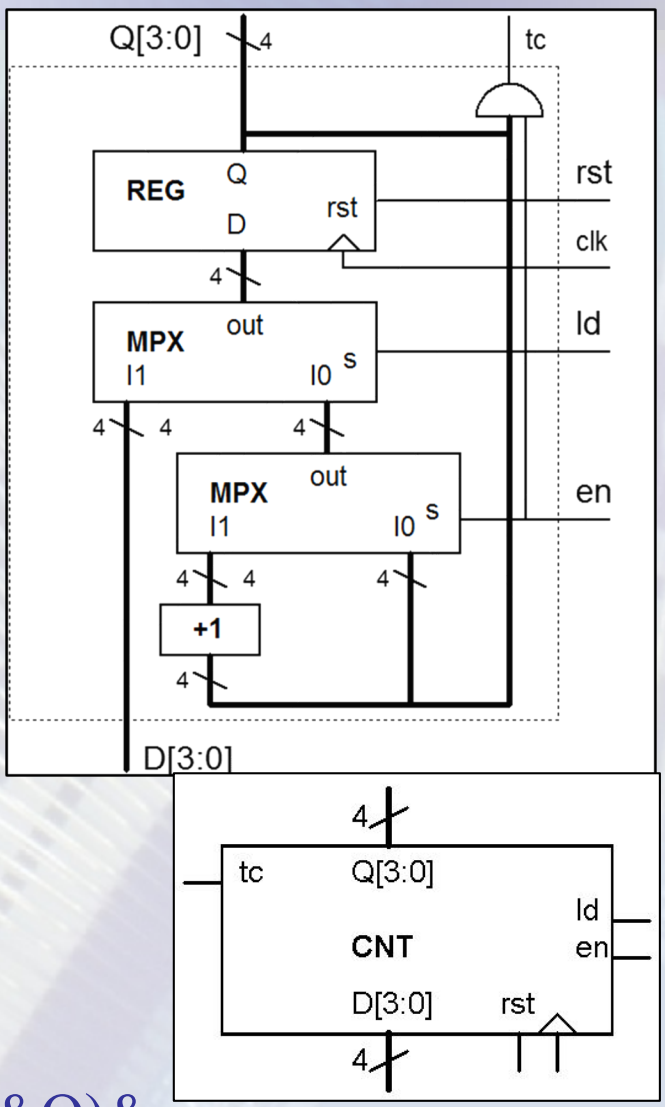


Szinkron sorrendi funkcionális egységek

Bináris számlálók különböző verziói

- **Felfele számláló, törölhető, tölthető**
- A prioritást itt a MPX-ex sorba kötése biztosítja.
- Az rst törli a regisztert.
- Ha ld aktív, D kerül a regiszter bemenetére (betöltés).
- Ha ld = 0 és en = 0, akkor Q jut a regiszter bemenetére (tartás).
- Ha ld = 0 és en = 1, akkor Q+1 kapcsolódik a regiszter bemenetére, (számlálás üzemmód).
- ***Engedélyezhető számlálók esetén végérték jelzés csak en=1 esetén van***:

  tc = Q[3]&Q[2]&Q[1]&Q[0]&en = (&Q)&en



Szinkron sorrendi funkcionális egységek

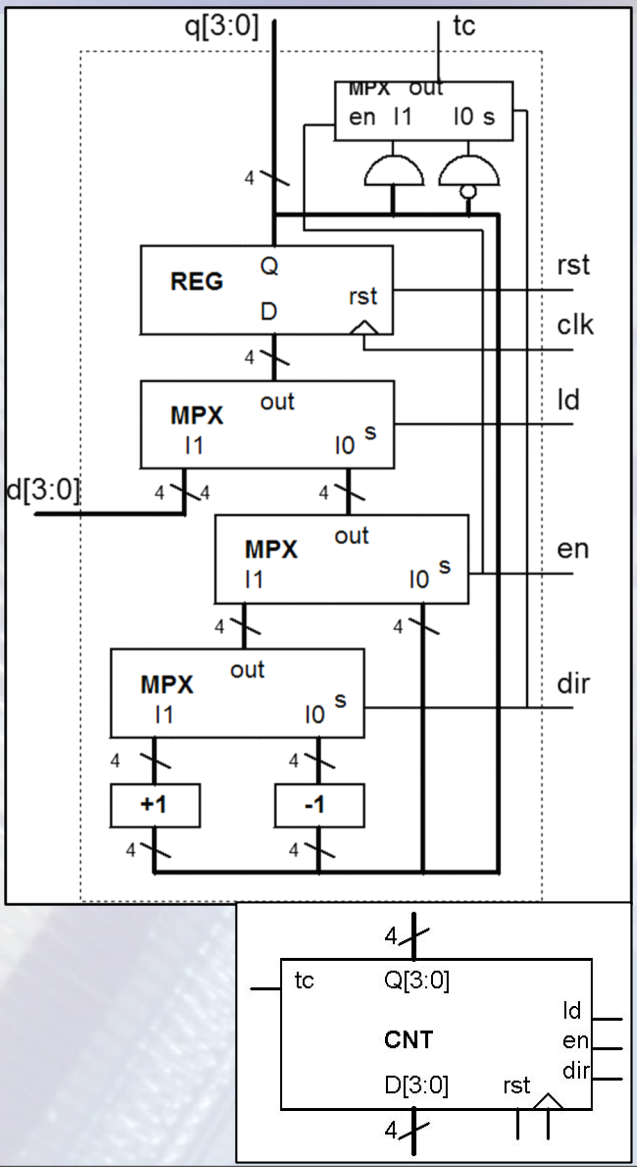
Veilog kód (törölhető, tölthető bináris felfele számláló)

```
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

- A számlálási irányt a ***dir*** vezérli. Ha dir = 1 fel, egyébként le számlálás, ha az en jel engedélyezi. (A dir fordítva is lehetne.)
- ***A DECREMENTER az n bites bemenetére adott számból kivon 1-et.***
- ***A tc most irányfüggő.***
- Bináris számlálóknál felfele számlálás esetén tc = 1, ha Q csupa 1 és en = 1, lefele számlálás esetén, ha Q csupa 0 és en = 1.



Szinkron sorrendi funkcionális egységek

- **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

Bináris számlálók modulusának módosítása

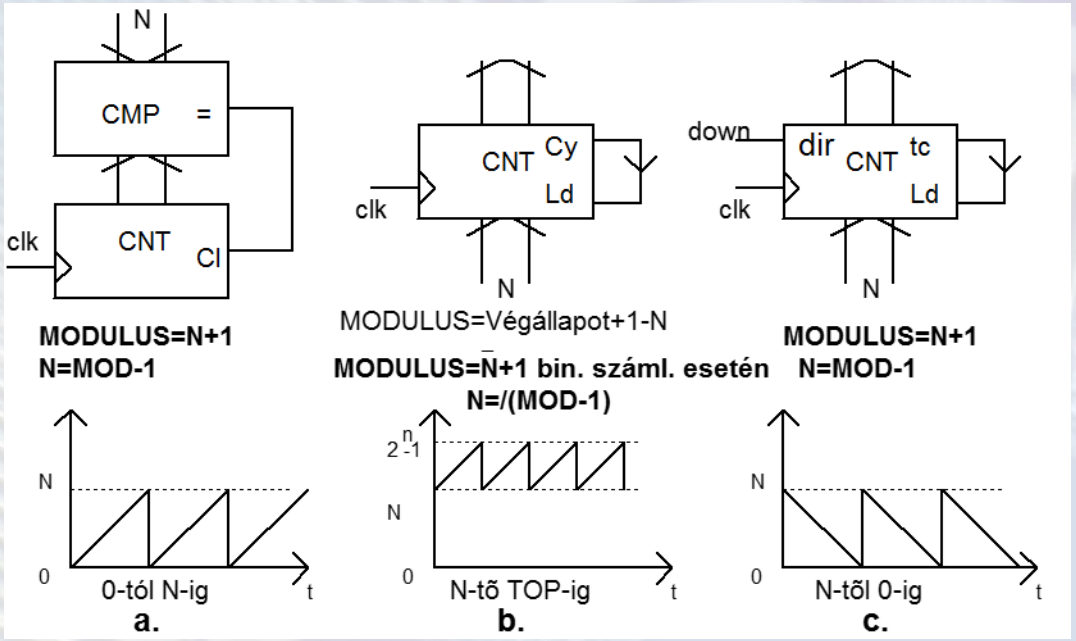
- Modulus csökkentés N értékre a 2^n – ről

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

b. Felfele számláló, ha eléri a csupa 1-et (2^n-1)-et, betöltünk N=/(MOD-1)-et. (Ezt ritkán használják.)

c. Lefele számláló, ha eléri a 0-át, betöltünk N= MOD-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.



Szinkron sorrendi funkcionális egységek

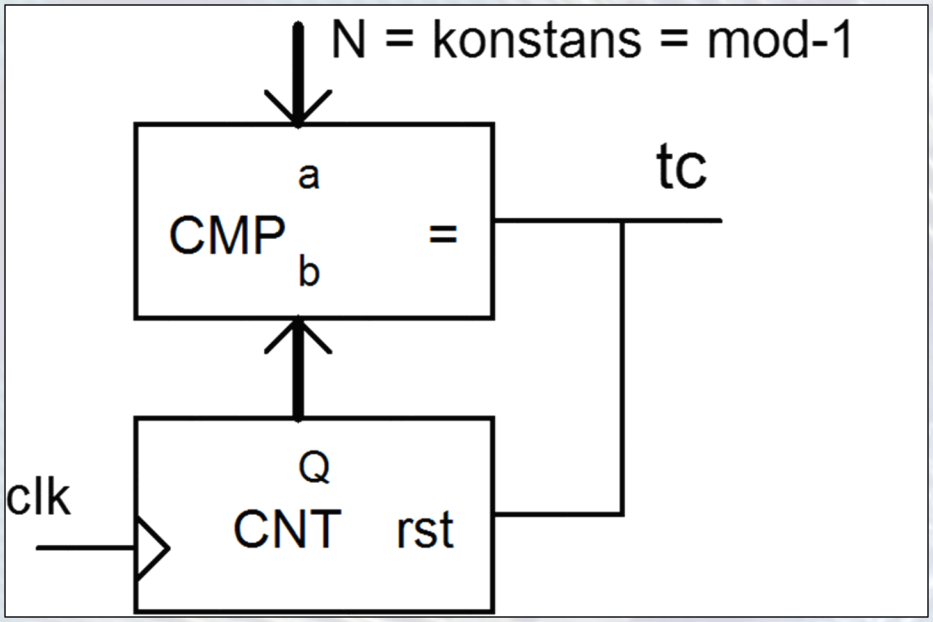
- **Modulus növelés**
  - Verilog HDL realizációnál egyszerűen msb megfelelő megválasztásával **reg [msb:0] cnt;**
  - A kaszkádosítás (méretbővítés kisebb egységek sorbakötésével) kerülendő, az eredmény minden tekintetben rosszabb lesz. Csak nem bináris számláló egységek sorbakötése esetén használják (pl. BCD számlálók).

Szinkron sorrendi funkcionális egységek

- **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

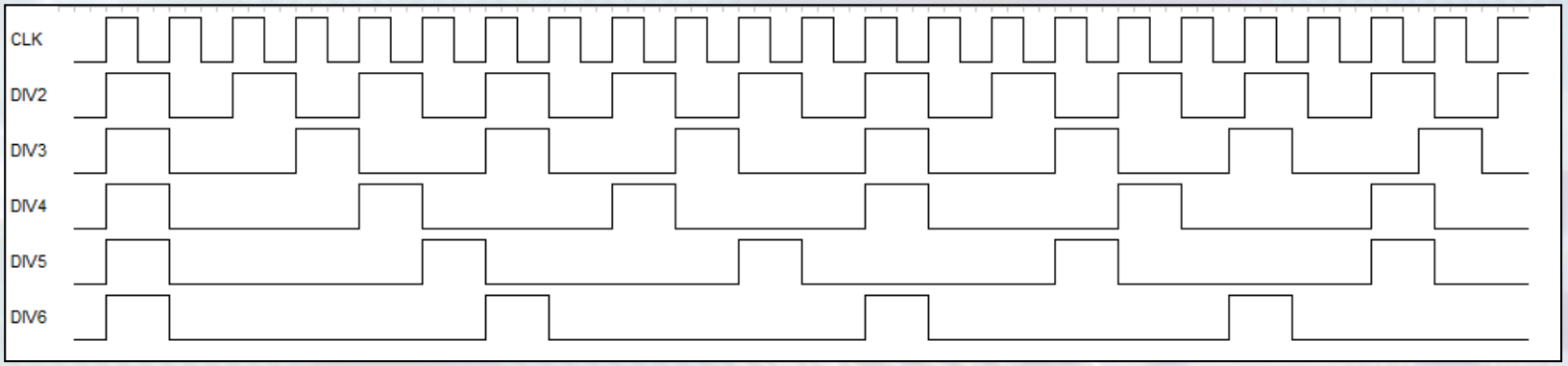
Szinkron sorrendi funkcionális egységek

- **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



Szinkron sorrendi funkcionális egységek

- Tipikus hullámformák kis osztásviszonyra



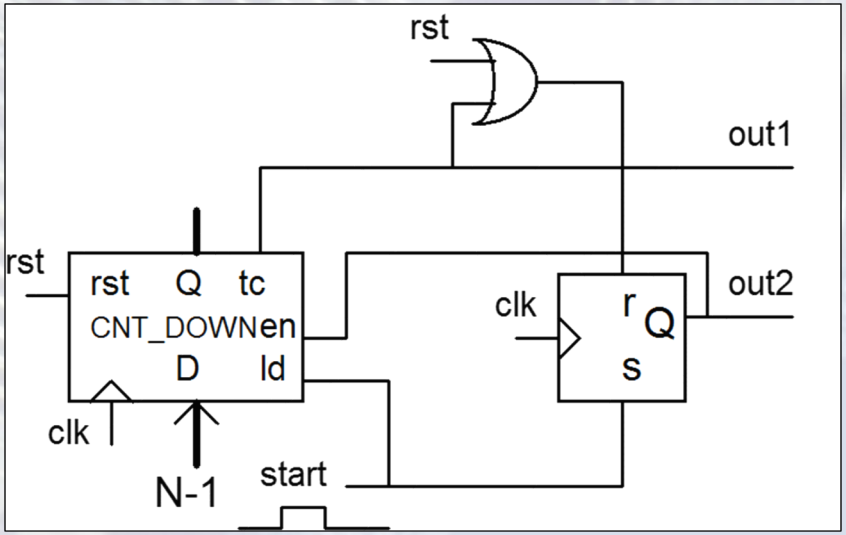
- Verilog kód a következő oldalon mod = 10 esetére. A számláló így számol (q[3:0]): 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 0000, ...

Szinkron sorrendi funkcionális egységek

```
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
   always @(posedge clk)
   begin
         if (rst | ill) q <= 4'h0;
         else if (ld) q <= d;
                  else if (en)
                           if (tc) q <= 4'h0;
                           else q <= q + 4'h1;
    end
endmodule
```

Szinkron sorrendi funkcionális egységek

- **Időzítés bináris lefele számlálóval**
  - Az időzítés T_{CLK} függvénye, ez a legfinomabb felbontás
  - Időtartam betöltése adatbemeneten (N-1) [$*T_{clk}$]
  - Indításkor a START pulzus 1-be írja az out2 = en jelet és betölti N-1-et. Innen a számláló elkezd leszámolni.
  - N ciklus múlva a számláló eléri a 0-át, a TC = 1 pulzus megjelenik és törli az out2=en jelet. A számláló leáll. Újabb START-ra újraindul.



Szinkron sorrendi funkcionális egységek

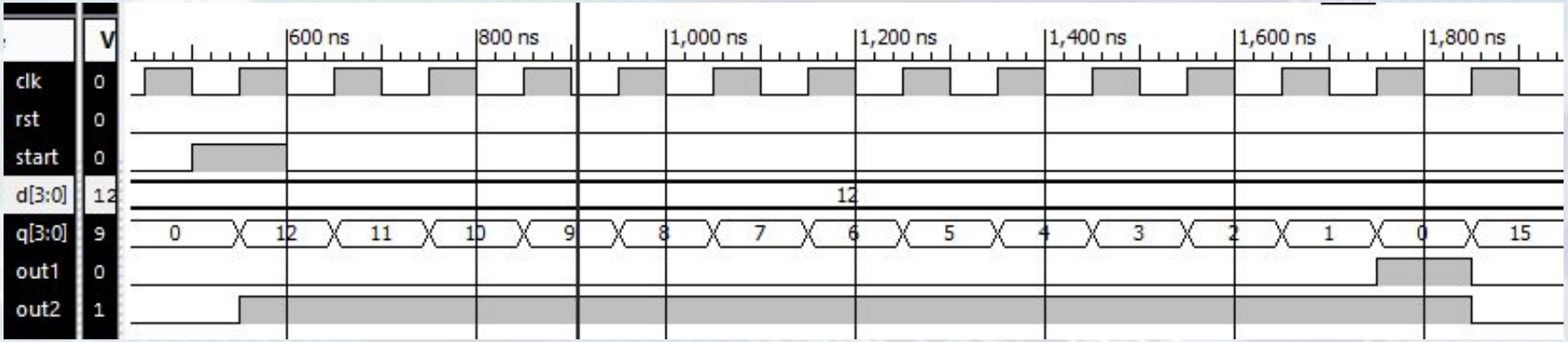
Verilog kód (időzítés bináris lefele számlálóval)

```
module PulseGen( input clk, input rst, input start, input [3:0] d,
                                        output out1, output reg out2 );
reg [3:0] q;                //lefele számláló
assign tc = &(~q)&out2;   // tc = (q == 4'h0);
assign out1 = tc;
always@(posedge clk)              // tölthető lefele számláló
if(rst) q <= 4'h0;
else if(start) q <= d;
       else if(out2) q <= q - 4'h1;

//out2 pulzus kimenet előállítása
always@(posedge clk)               // az out2 kimenet előállítása szinkron set-reset flip-floppal
if(rst | tc) out2 <= 1'b0;             // 0-ba állítás (reset)
     else if(start) out2 <= 1'b1;   // 1-be állítás (set)
endmodule
```

Szinkron sorrendi funkcionális egységek

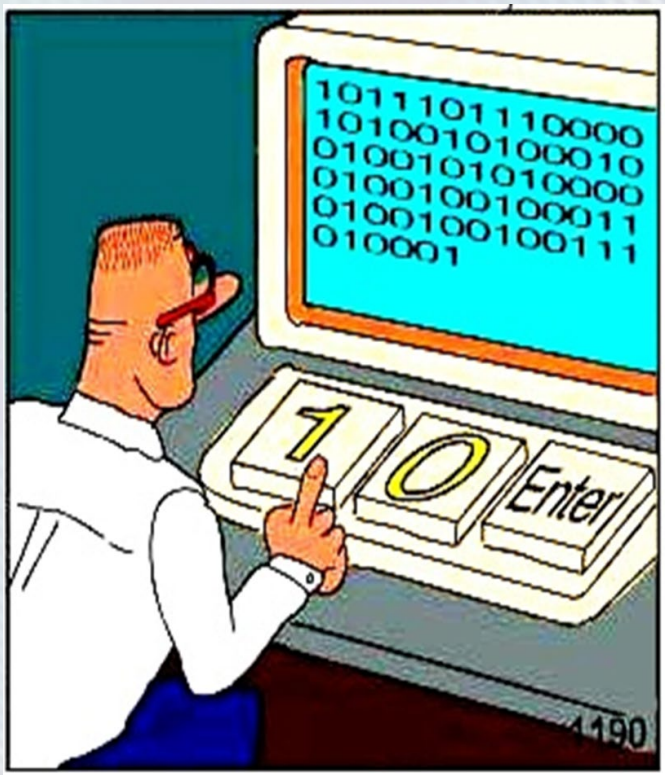
- **A szimuláció időd iagramja:**



- Megjegyzés:
  - A valódi időzítők gyakran többfokozatúak. Az előosztó egy pulzuskimenetű frekvenciaosztó, így az időzítés időtartama eredőben K*N lesz. Pl. N értéke lehet 2^0, 2^4, 2^8, 2^{12}, 2^{16}.
    A legfinomabb időbeli felbontás ekkor $N*T_{clk}$ értékű

Kérdések, észrevételek

- **A kérdéseket a benes@mit.bme.hu címre várjuk de csak ... @edu.bme.hu címről, tárgy: DIGIT**



Digitális technika
5. EA vége
A további diákon érdeklődők számára további funkcionális egységek találhatók, nem vizsgaanyag

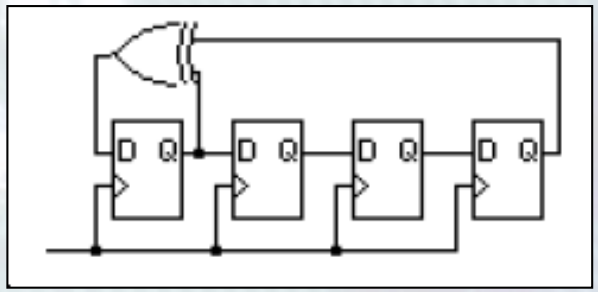
Funkcionális egységek

- **LFSR számláló (Érdekességként, nem vizsgaanyag!)**
  - Lineárisan (XOR vagy XNOR) visszacsatolt shiftregiszter alapú számláló
  - A számolási sorrend látszólag véletlenszerű
    - PRNG Pseudo Random Number Generator
  - A modulus értéke **n** bitre a visszacsatoló függvénytől függ, ami a bitek XOR, vagy XNOR függvénye
  - A maximális esetben az állapotsorozat hossza $2^n - 1$,
    - XOR-nál a csupa 0, XNOR-nál a csupa 1 hiányzik a generált állapotsorozatból (abba beleragad, nem lép ki)
    - Ha kell, esetleg ez beilleszthető (külön logika kell hozzá)
  - Igen sok területen alkalmazott, nagyon hasznos elem

Funkcionális egységek

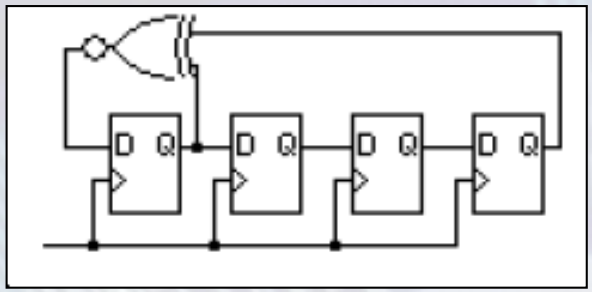
- **LFSR számlálók 4 bitre, maximális hosszúságra**

XOR



0	F	1	1	1	1
1	7	0	1	1	1
2	B	1	0	1	1
3	5	0	1	0	1
4	A	1	0	1	0
5	D	1	1	0	1
6	6	0	1	1	0
7	3	0	0	1	1
8	9	1	0	0	1
9	4	0	1	0	0
10	2	0	0	1	0
11	1	0	0	0	1
12	8	1	0	0	0
13	C	1	1	0	0
14	E	1	1	1	0

XNOR



0	0	0	0	0	0
1	8	1	0	0	0
2	4	0	1	0	0
3	A	1	0	1	0
4	5	0	1	0	1
5	2	0	0	1	0
6	9	1	0	0	1
7	C	1	1	0	0
8	6	0	1	1	0
9	B	1	0	1	1
10	D	1	1	0	1
11	E	1	1	1	0
12	7	0	1	1	1
13	3	0	0	1	1
14	1	0	0	0	1

Funkcionális egységek

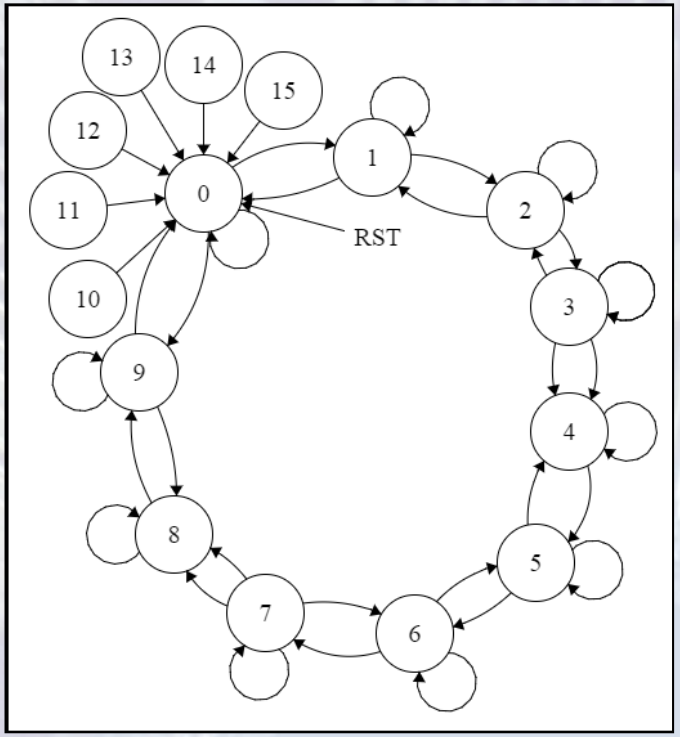
- **BCD számlálók**
- **Jelentőségük: közvetlenül értelmezhető kijelzett érték**
  - A hexadecimális leolvasás nem felhasználóbarát
- **Üzemmódok: Törlés, töltés, engedélyezés, irány**
- **Végértékjelzés (TC) Felfelé: 9-nél, Lefelé: 0-nál**
- **Modulus növelés:**
  - BCD számlálók dekádonként bővíthetők, a 10-100-1000-10000-100000…..kijelzési tartományokhoz
- **Modulus csökkentés:**
  - A szokásos módon törléssel vagy töltéssel

Funkcionális egységek

- **BCD számlálók Verilog HDL kódolása**
- **Illegális állapotok kezelése: A 10, 11, 12, 13, 14, 15 betölthető normál paranccsal, ezért érdemes kezelni**
  - A következő ciklusban 0-ba lépünk

```
////////////////////////////////////////////////////////////////////////////////
// A számláló működését viselkedési stílusban írjuk le
// Sok lehetséges üzemmód, összetett vezérlési logika
// A lehetséges nem BCD állapotot dekódoljuk és egy
// lépésben korrigáljuk, azaz 0-ra töröljük a számlálót
////////////////////////////////////////////////////////////////////////////////
wire ill, q9, q0;
assign ill = (q  > 4'd9);                          // Illegális állapot
assign q9  = (q == 4'd9);                          // Végérték felfelé
assign q0  = (q == 4'd0);                          // Végérték lefelé

always @ (posedge clk)                            // Felfutó élre működik
   begin
      if (rst | ill)          q <= 4'd0;          // Resetnél vagy illegális
         else                                     // állapotnál törlés
            if (load)         q <= d;             // Töltés a d bemenetről
               else                               // Ha nincs RST, ILL vagy LOAD,
                  if (en)                         // de a számolás engedélyezett
                  if (dir)                        // Felfelé számlálásnál
                     if (q9)  q <= 4'd0;          // ha már 9, törlés egyébként
                     else     q <= q + 4'd1;      // inkrementálás
                  else                            // Lefelészámlálásnál
                     if (q0)  q <= 4'd9;          // ha már 0, 9 betöltése
                     else     q <= q - 4'd1;      // egyébként dekrementálás
   end
```



Funkcionális egységek

- **BCD számlálók Verilog HDL kódolása**
- **Kaszkádosítás:**
  - A TC és az engedélyező jelekből felépített átviteli lánc
  - A TC jelzés magába foglalja az engedélyezést is

    TC = EN & (DIR ? (CNT == 9) : (CNT == 0));
  - Az engedélyező lánc kialakítása
    - Egyszerű soros lánc $\mathbf{EN_i \leftarrow TC_{i-1}}$

```
cnt10x egyesek (.clk(clk),.rst(rst),.en(en),    .tc(tc_e), .q(q[ 3: 0]));
cnt10x tizesek (.clk(clk),.rst(rst),.en(tc_e), .tc(tc_t), .q(q[ 7: 4]));
cnt10x szazasok(.clk(clk),.rst(rst),.en(tc_t), .tc(tc_sz),.q(q[11: 8]));
cnt10x ezresek (.clk(clk),.rst(rst),.en(tc_sz),.tc(),     .q(q[15: 12]));
```

    - Az engedélyező lánc késleltetése korlátozza az elérhető f_{max} értékét. Sok érdekes trükk létezik ennek javítására. (Nem tárgyaljuk.)