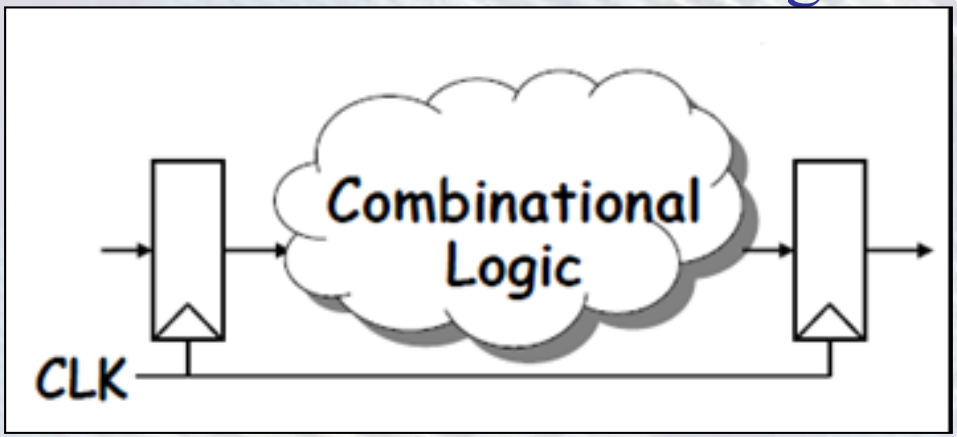
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Digitális technika
VIMIAA01

Fehér Béla
BME MIT

Sorrendi logikák alap építőeleme

- **REGISZTER**
- **Adott számú DFF párhuzamos működtetése, közös, globális, időben és térben egyidejű CLK órajellel**
- **A szinkron digitális tervezői paradigma biztosítása**
  - Két órajel él között stabil kimeneti értékek biztosítása
  - Két órajel között elegendő idő a feladatok elvégzésére
  - Minden művelet a kombinációs logikában történik

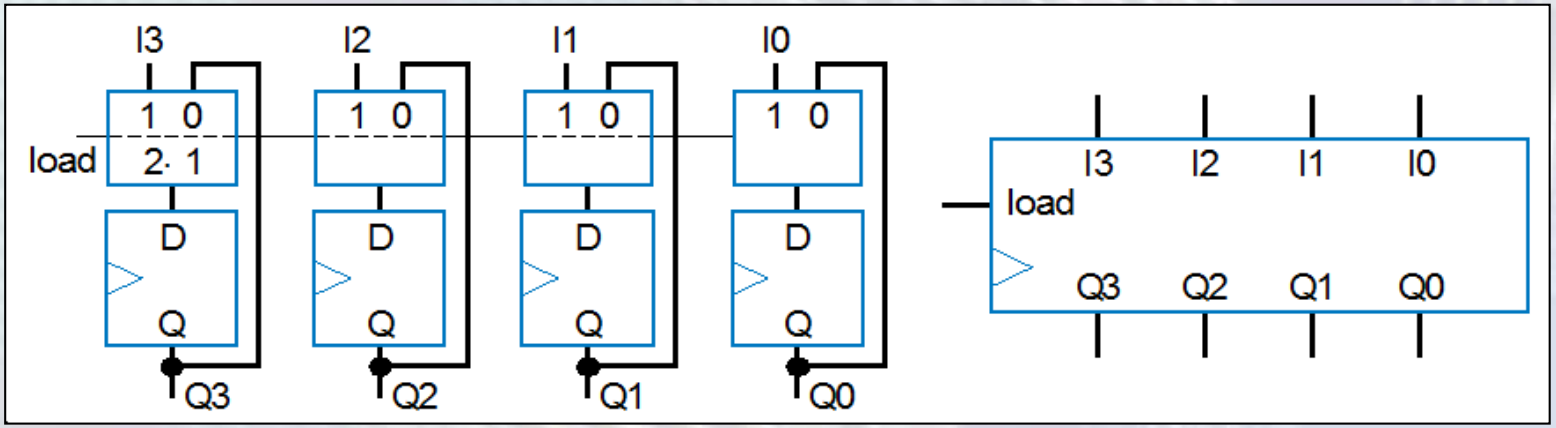


Sorrendi logikák alap építőeleme

- **A regiszterek funkciói:**
  - Alaphelyzet beállítás: lehet tetszőleges induló érték
  - Töltés: Az adatbemeneti vonalak mintavételezése adott kijelölt órajel ciklusokban (akár mindben)
  - Tartás: A regiszter tartalom fenntartása a következő más parancsig
  - Funkció vezérlés:
    - Ha az elemi DFF rendelkezik a szükséges jelekkel, akkor használhatjuk azokat is (RESET, SET, CE)
    - Ha nem, akkor felépítjük a többfunkciós regisztert

Sorrendi logikák alap építőeleme

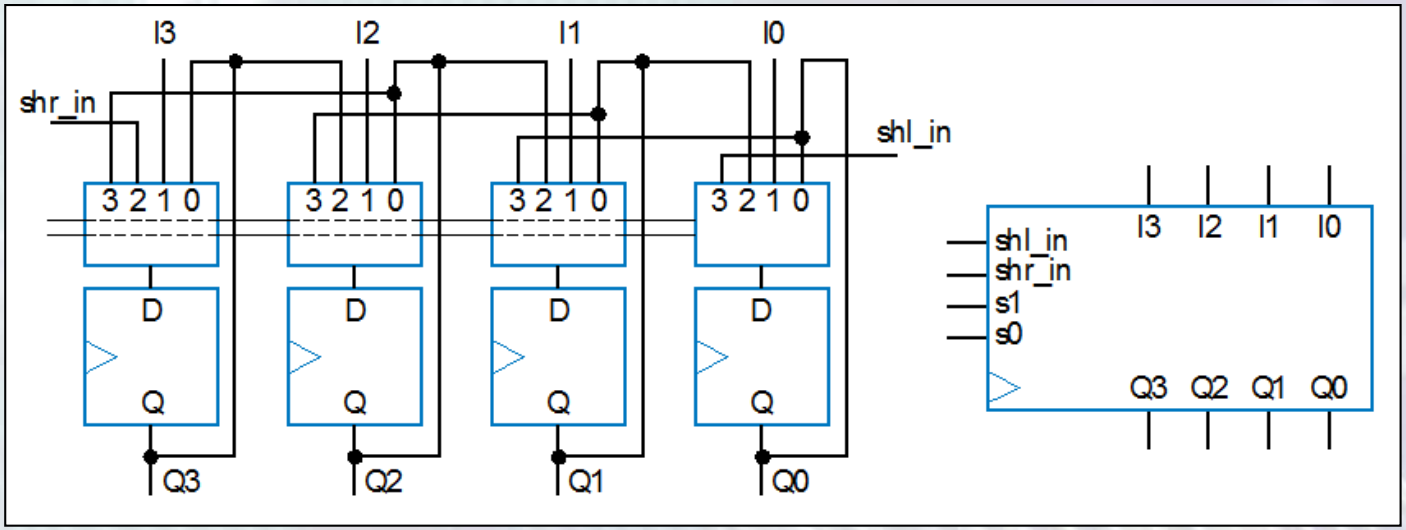
- **Multifunkciós regiszter:**
  - Általános séma: A regiszter bemenetét MUX vezérli
  - A MUX bemeneteire kapcsoljuk az adott vezérléshez szükséges adatot, ami a regiszter következő állapotát meghatározza Töltés:Bemenetről, Tartás:Kimenetről
  - Vezérlés a LOAD jellel: 1 → Tölt, 0 → Tart



  - Több funkció → több bemenetű MUX + vezérlő logika

Funkcionális egységek

- **Shiftregiszter (egyszerű verziója szerepelt már)**
  - Bővítés: Tölt / Tart / Léptet Jobbra/ Léptet Balra

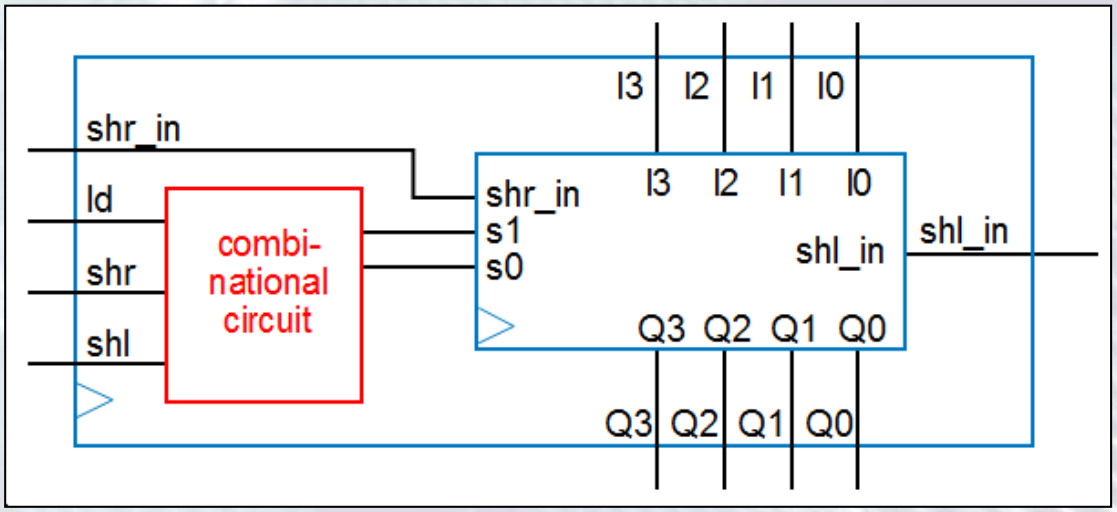


S1	S0	FUNK
0	0	TART
0	1	TÖLT
1	0	SHR
1	1	SHL

  - Több funkció → több bemenetű MUX + vezérlő logika
  - Minden művelet a MUX bemenetek huzalozásával
  - Kódolt vezérlés: S[1:0] bináris funkciókódok
  - Az shr_in, shl_in a soros beléptető adatbemenetek

Funkcionális egységek

- **Shiftregiszter funkciók dekódolt vezérlőjelekkel:**
  - Gyakran a használat szempontjából kedvezőbb



CONTROL INP			MUX INP		FUNK
ld	shrl	shl	s1	s0	
0	0	0	0	0	TART
0	0	1	1	1	SHL
0	1	0	1	0	SHR
0	1	1	1	0	SHR
1	0	0	0	1	TÖLT
1	0	1	0	1	TÖLT
1	1	0	0	1	TÖLT
1	1	1	0	1	TÖLT

  - A dekódolt jelek között prioritás van → ld > shr > shl
  - Tömörített vezérlési tábla
    - Igen, ez egy rejtett prioritás enkóder, 3 aktív bemeneti vonallal

CONTROL INP			MUX INP		FUNK
ld	shr	shl	s1	s0	
0	0	0	0	0	TART
0	0	1	1	1	SHL
0	1	X	1	0	SHR
1	X	X	0	1	TÖLT

Funkcionális egységek

- **Shiftregiszter Verilog HDL kódolása**
- **Kódolt vezérlőjelekkel,**
  - Erre a formára a **case** szerkezet a legmegfelelőbb
  - Nincs prioritás, egyenrangú parancsok
    - Ha nincs speciális vezérlés, tartja az addigi értéket

```
reg [3:0] sr;                    // 4 bit shiftregiszter
wire [1:0] fun;                  // 2 bit kódolt vezérlés


always @ (posedge clk)
   case (fun)
      2'b01 :  sr <= par_inp;              // Tölt
      2'b10 :  sr <= {shr_in, sr[3:1]};    // SHR
      2'b11 :  sr <= {sr[2:0], shl_in};    // SHL
   endcase
```

- **A shift művelete a { } operátorral, egzakt módon kijelölve a következő értékeket, bitről-bitre**

Funkcionális egységek

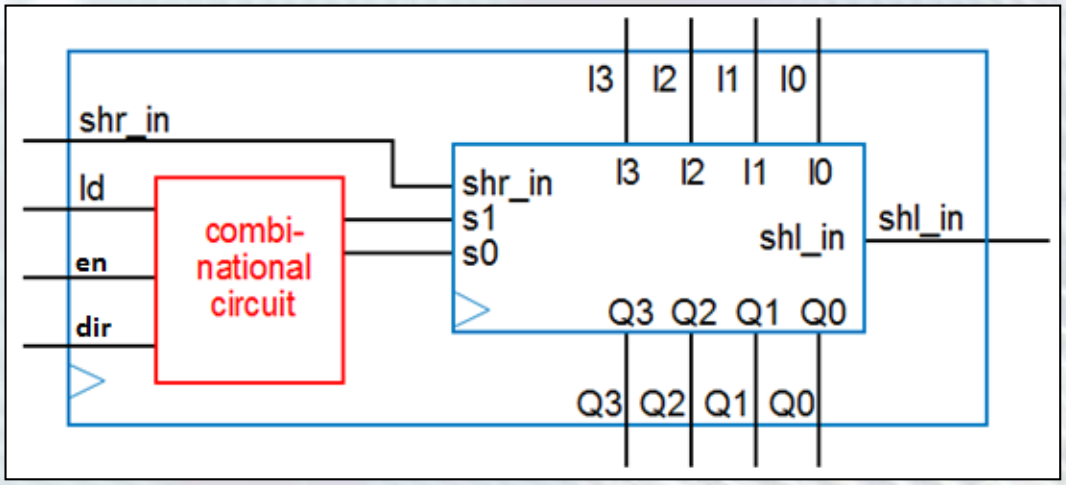
- **Shiftregiszter Verilog HDL kódolása**
- **Szétválasztott vezérlőjelekkel**
  - Prioritás a kiértékelési sorrend szerint

```
reg [3:0] sr;                    // 4 bit shiftregiszter
wire ld, shr, shl;               // Egyedi vezérlőjelek

always @ (posedge clk)
   if (ld)             sr <= par_inp;             // Tölt
   else if (shr)       sr <= {shr_in, sr[3:1]};   // SHR
        else if (shl)  sr <= {sr[2:0], shl_in};   //SHL
```

- **A shift műveletet itt is a { } operátorral írjuk elő**
  - Így használhatjuk a shr_in és shl_in soros adatbeléptető bemeneteket, ahol tetszőlegesen 0 vagy 1, vagy más, valódi adatforrást léptethetünk be (A Verilog >>,<< shift operátorok 0-t léptetnek be.)

Funkcionális egységek

- **Shiftregiszter Verilog HDL kódolása**
- **Szétválasztott vezérlőjelekkel, de más stílusban**
  - Töltés, léptetés engedélyezés, irányvezérlés

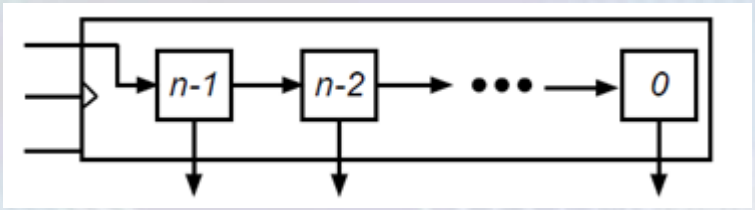


CONTROL INP			MUX INP		FUNK
ld	en	dir	s1	s0	
0	0	X	0	0	TART
0	1	0	1	1	SHL
0	1	1	1	0	SHR
1	X	X	0	1	TÖLT

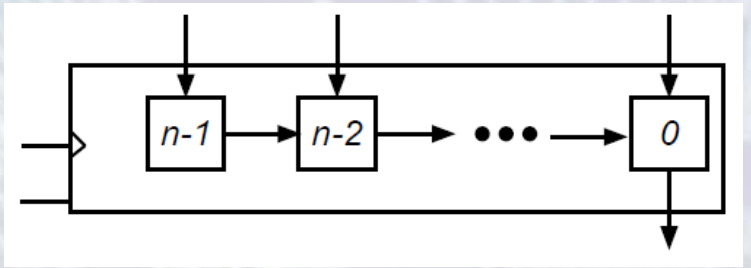
```
reg [3:0] sr;                    // 4 bit shiftregiszter
wire ld,en,dir;                  // Egyedi vezérlőjelek

always @ (posedge clk)
   if (ld)              sr <= par_inp;              // Tölt
   else if (en)
           if (dir)     sr <= {shr_in, sr[3:1]};    // SHR
           else         sr <= {sr[2:0], shl_in};    //SHL
```

Funkcionális egységek

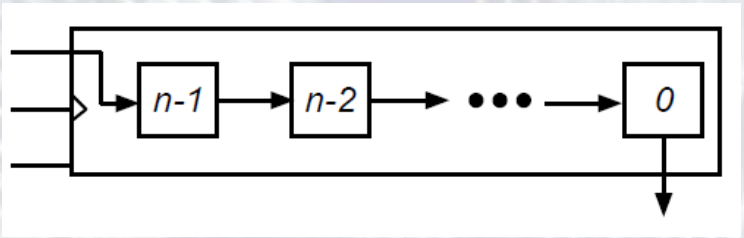
- **Shiftregiszter alkalmazási területek**
  - Soros-párhuzamos átalakítás



  - Párhuzamos soros átalakítás



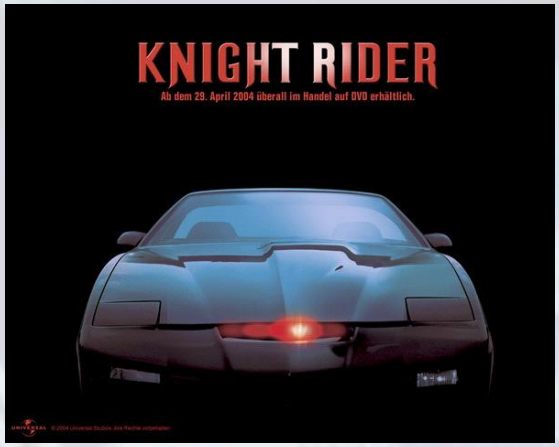
  - Soros adat késleltetés



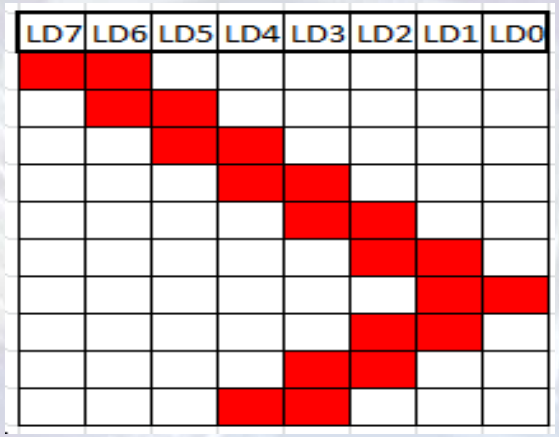
  - Szorzás (osztás) 2 hatvánnyal
    - $* 2^n \rightarrow$ Töltés + n db Léptetés balra
    - Összesen (n+1) órajel

Funkcionális egységek

- **Shiftregiszter alkalmazási területek**
- **Tetszőleges alkalmazás**
  - Knight-Rider LED játék



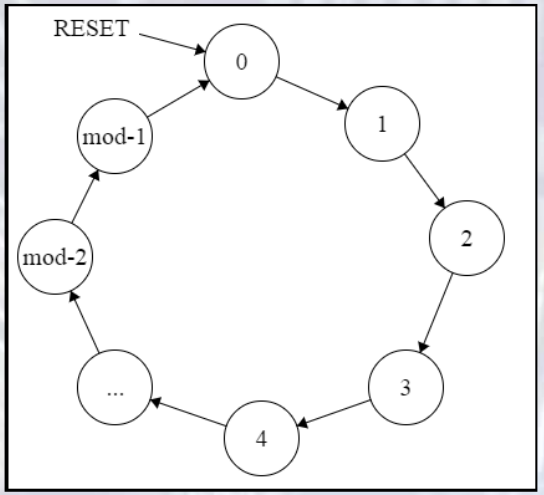

```
reg [7:0] sr;                  // 8 bit shiftregiszter
wire ld, en, dir;              // Egyedi vezérlőjelek

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
```



  - Vezérlő logika?
    Kezdetben egy LD = RST pulzus
    és utána periodikus DIR vezérlés
    Ezt generálhatjuk a LED-ek állapota alapján, egy kétállapotú Mealy vagy Moore vezérlővel (Mi a lényeges különbség?)

Funkcionális egységek

- **Számlálók (Csak szinkron számlálókkal foglalkozunk)**
  - A legfontosabb komponensek
  - Sok fajta verzió, felépítés, szolgáltatások
  - Közös jellemzőjük: Állapotdiagramjuk ciklikus, önmagában záródó
  - Állapotok száma: MODULUS
  - Utolsó állapotátmenetek: (mod-2) → (mod-1) → 0 → 1
  - Pl. 3 biten 0,1,2,3,4,5,6,7,0,1,2,3,4,..
  - A diagram egy egyszerű, folyamatosan egy irányba számoló számlálót mutat (az állapot sorszáma nem feltétlenül numerikus érték, lehet más kód is)



Funkcionális egységek

- **Számlálók**
  - Tetszőlegesen ciklikus állapotgépek
    - (pl. a HF1 is 3 db számláló jellegű egység)
  - Shiftregiszter alapú számlálók
    - Gyűrűs, Johnson, LFSR
  - Bináris számlálók
    - Bináris, Gray, BCD (0..9)

Típus	Állapotszám n bitre	Adatforma
gyűrűs számláló	n	k-az-n-ből
Johnson számláló	2n	k db 0, (n-k) db 1
LFSR (lineárisan visszacsatolt SHR)	maximum 2^n-1	bináris
bináris számláló	2^n	bináris
dekadikus számláló	$10^{n/4}$, ha 4|n	BCD

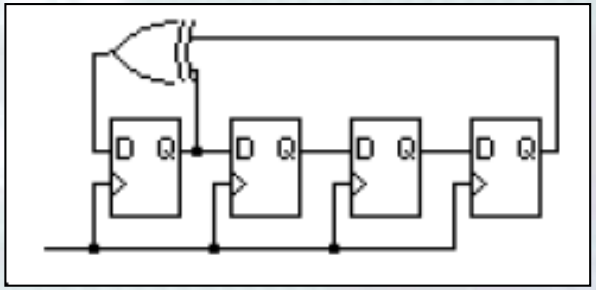
Funkcionális egységek

- **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 $\mathbf{2^n - 1}$,
    - 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

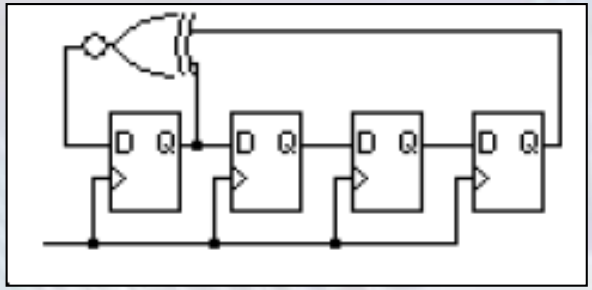
- **LFSR számlálók 4 bitre, maximális hosszúságra**

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

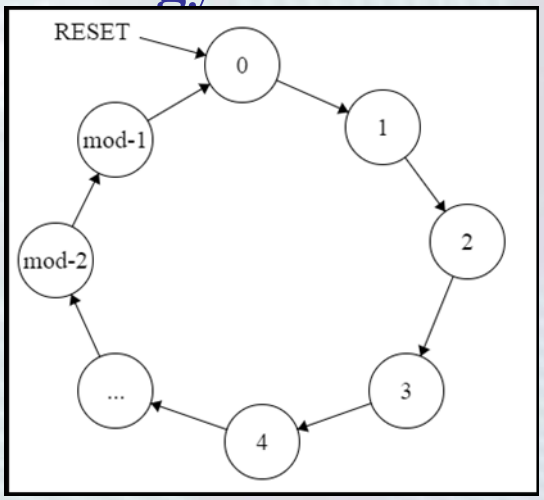
XNOR

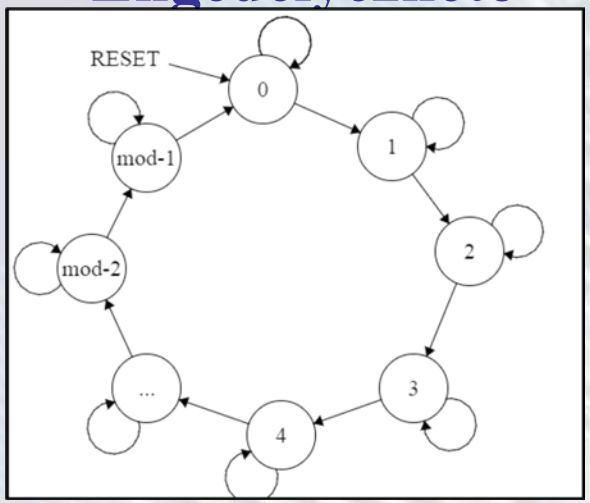


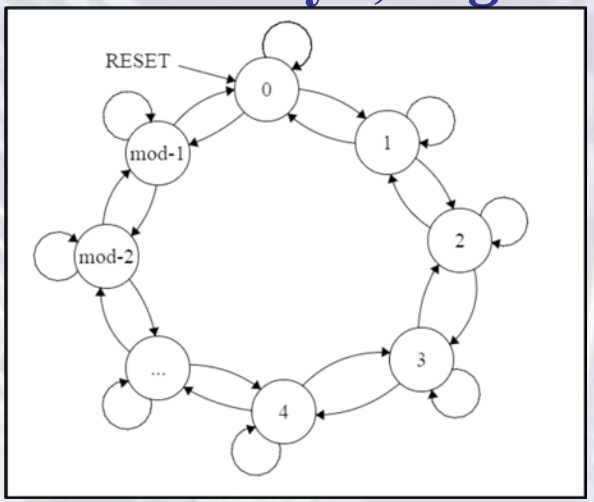
0	0	0	0	0	0
1	8	1	0	0	0
2	4	0	1	0	0
3	A	1	0	1	0
4	5	0	1	0	1
5	2	0	0	1	0
6	9	1	0	0	1
7	C	1	1	0	0
8	6	0	1	1	0
9	B	1	0	1	1
10	D	1	1	0	1
11	E	1	1	1	0
12	7	0	1	1	1
13	3	0	0	1	1
14	1	0	0	0	1

Funkcionális egységek

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



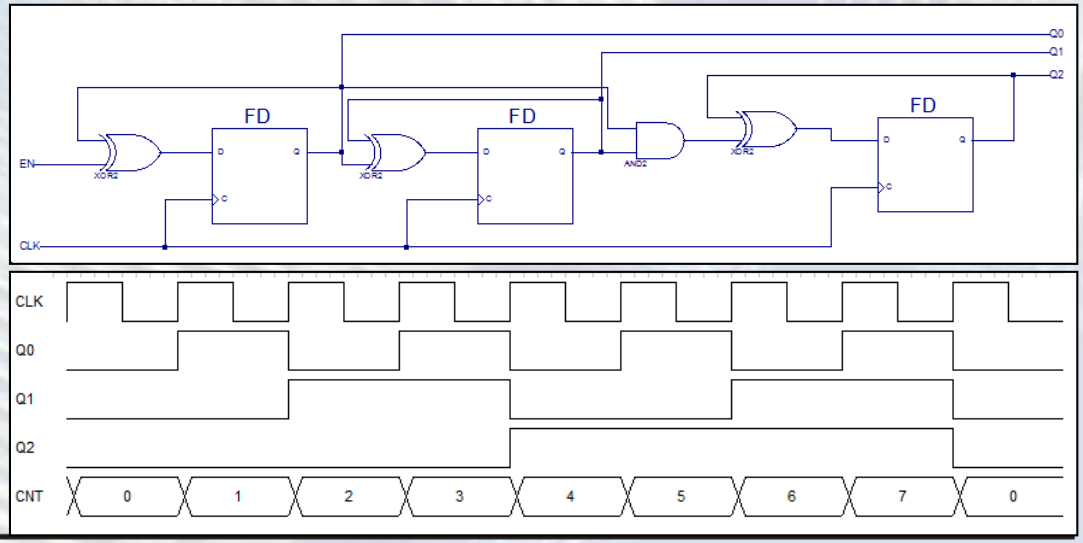




- **A tölthetőséget nem rajzoljuk fel, túl sok állapot átmenet, áttekinthetetlen lenne**

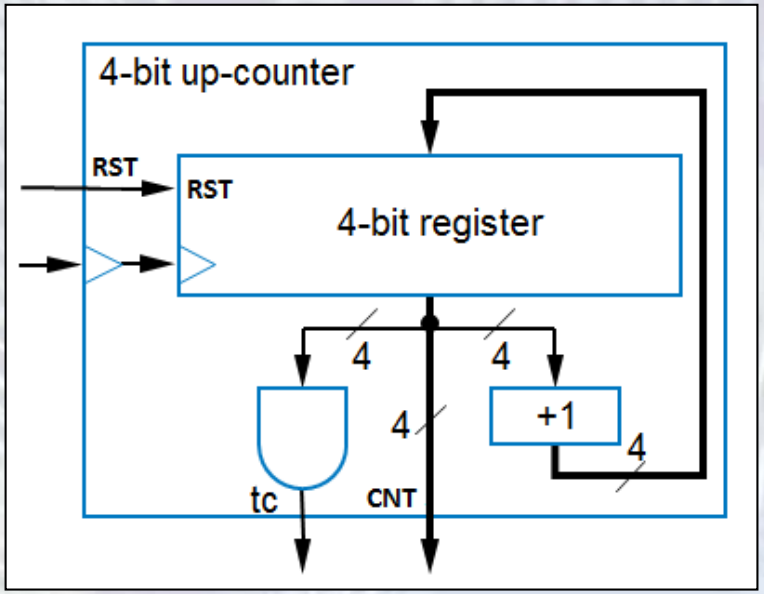
Funkcionális egységek

- **Bináris számlálók**
- **n bit 2^n állapot: 0 → $(2^n - 1)$, kompakt reprezentáció**
- **Sokféle lehetséges realizálási lehetőség**
  - A legegyszerűbb a bitenkénti TFF (XOR + DFF)
  - A TFF vált, ha a bemenete 1, egyébként tart
  - Az ÉS kapu biztosítja a $(2^x - 1) \rightarrow 2^x$ átmenetet, azaz az átvitelt, ha már minden kisebb helyiértékű bit értéke 1.
  - Szimmetrikus négyszögjelek
  - EN bemenet



Funkcionális egységek

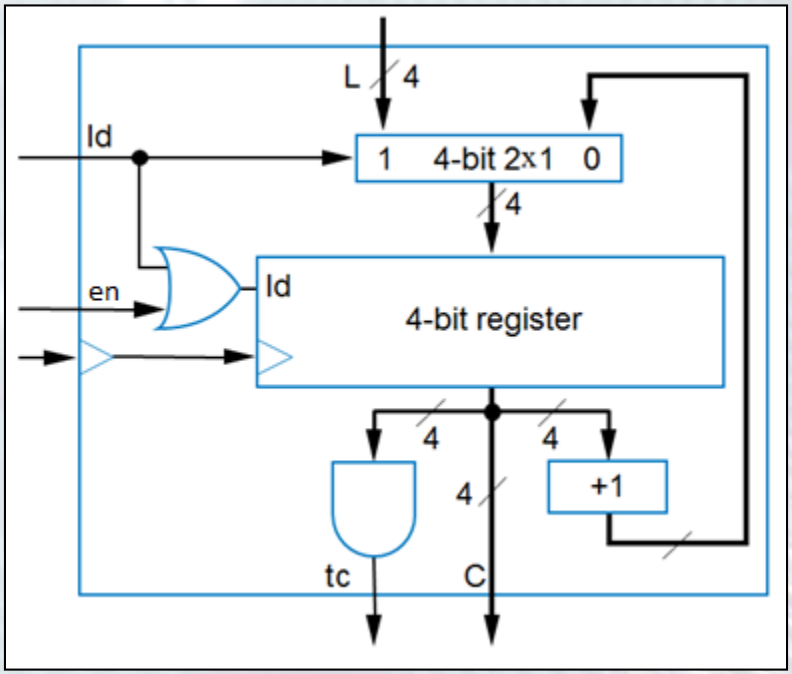
- **Bináris számlálók**
- **Általános modell**
  - Egyszerű BIN számláló: REG + INC
  - INCREMENTER: Mint egy ADDER, de az egyik operandusa 0, és a CIN_0 pedig 1, azaz CNT + 1
  - TC a végérték jelzés TC = 1, ha CNT = 1111
  - A 4 bites regiszter szinkron töltésű, törlésű (közös órajel)
  - A 16-os számláló az 1111 érték után átfordul és újra indul 0-ról



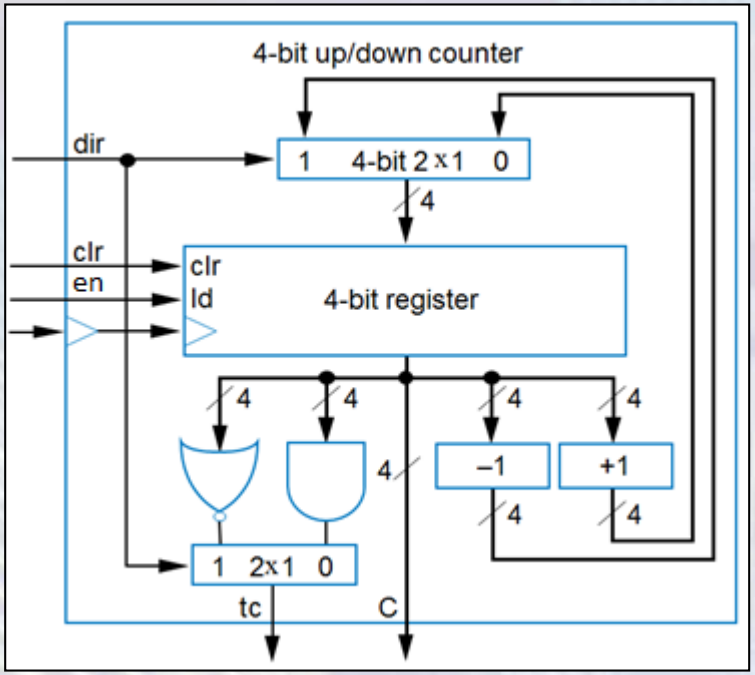
Funkcionális egységek

- **Bináris számlálók különböző verziói**

Tölthető számláló



Törölhető fel/le számláló



- **Látható a vezérlőjelek beavatkozási helye és hatásuk**
  - A Verilog HDL tervezéskor csak a szükséges részek épülnek be a számláló modulba

Funkcionális egységek

- **Bináris számlálók Verilog HDL tervezése:**
  - Hasonló a multifunkciós regiszterhez
  - Tipikus funkciók: Töröl, tölt, számol (fel/le), tart
  - Ehhez a vezérlőjelek: RST, LOAD, EN, DIR

```
////////////////////////////////////////////////////////////////////////////////
// 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 ? (&CNT) : (~|CNT)

Funkcionális egységek

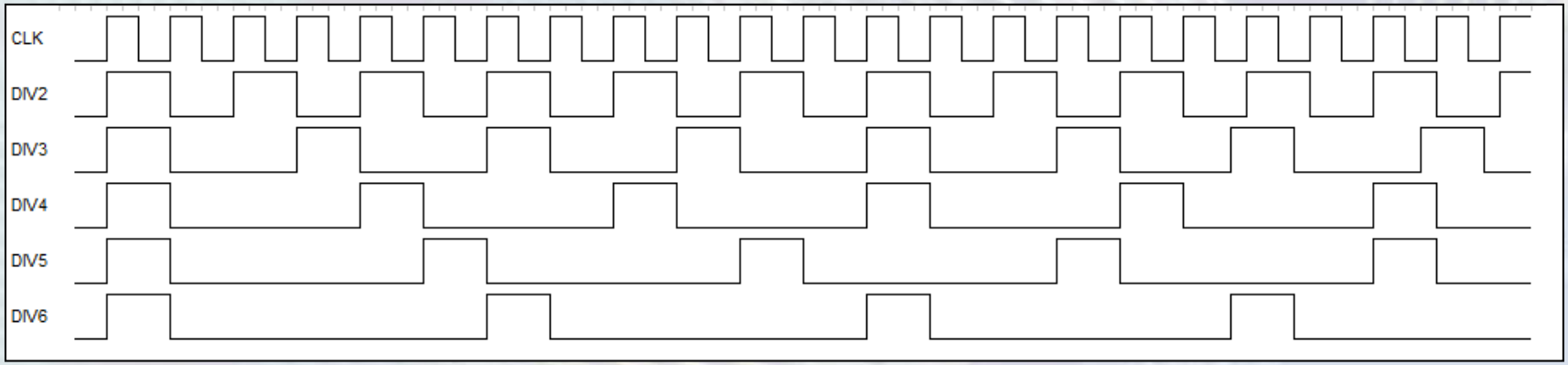
- **Bináris számlálók modulusának módosítása**
- **Modulus csökkentés N értékre**
  - Az elérhető funkciók függvényében
  - Törléssel:
    - Számolás 0 - (N-1), (N-1) dekódolása, majd törlés 0-ra
  - Töltéssel:
    - Számolás 0 - (N-1), majd töltés 0-val← **EZ A TIPIKUS**
    - Számolás M-től (M+N-1)-ig, majd töltés M-nel
    - Számolás (2^n-N) - (2^n -1)-ig, majd töltés (2^n-N)-val
- **Modulus növelés**
  - Verilog HDL realizációnál egyszerűen msb megfelelő megválasztásával **reg [msb:0] cnt;**
  - A kaszkádosítás (méretbővítés) kerülendő, az eredmény minden tekintetben rosszabb lesz

Funkcionális egységek

- **Bináris számlálók alkalmazásai**
  - Frekvenciaosztás
    - Pulzus kimenet kapuzott ütemezett működtetésre
    - Közel szimmetrikus négyszögjel tetszőleges felhasználásra
  - Időzítés
    - Adott késleltetés/időzítés után egy pulzus kiadás
    - Adott időtartamú/szélességű pulzus előállítása
  - Pulzusszámlálás
    - Él detektálással az engedélyező bemeneten
  - Általános vezérlési feladatok
    - n bit, 2^n állapot, törléssel, töltéssel, engedélyezéssel vezérelt állapotátmenetek (RESET,CONT, JUMP)
    - Feltételjelek alkalmazása a megfelelő vezérlő bemenetnél
    - Több utas ugrás megoldása nem gazdaságos

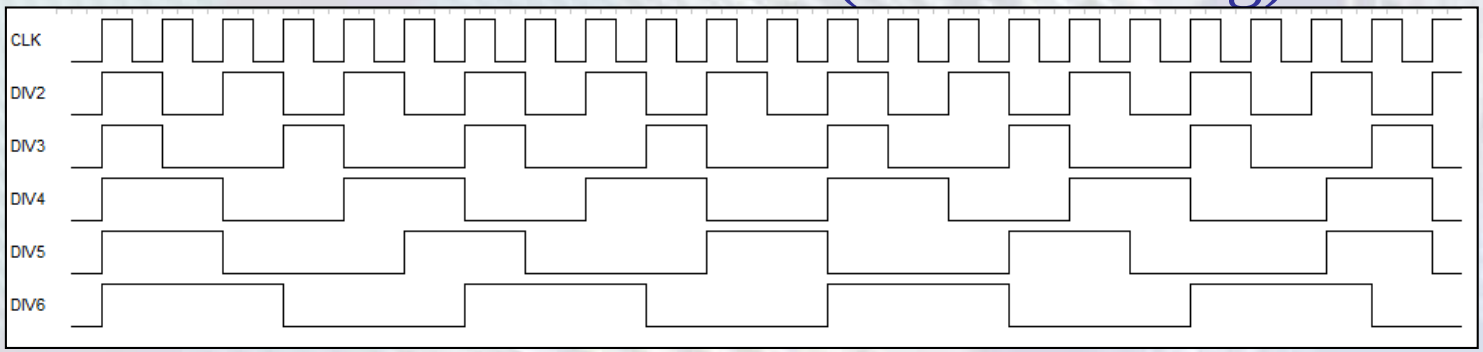
Funkcionális egységek

- **Frekvenciaosztás bináris számlálóval**
  - Pulzus kimenet előállítása ütemezett vezérléshez
  - N modulushoz a végérték dekódolást N-1-re állítjuk
  - Tehát a számláló egyszerűen 0-(N-1)-ig számol
  - TC egy egy órajel széles pulzus lesz
    - Tipikus hullámformák kis osztásviszonyra



Funkcionális egységek

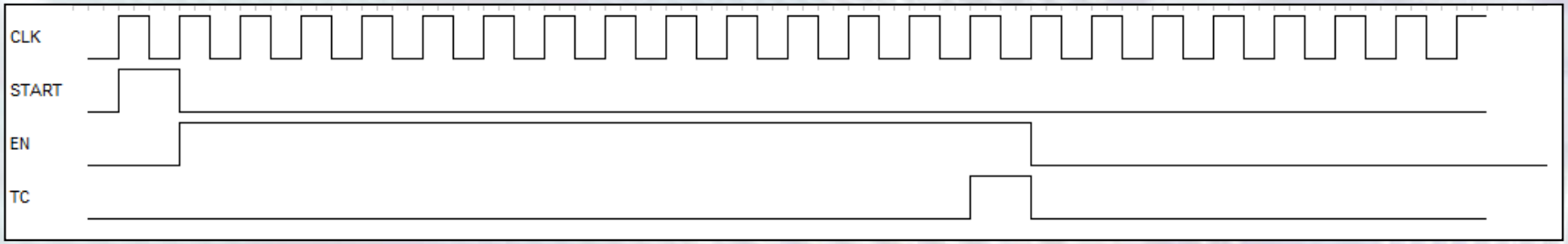
- **Frekvenciaosztás bináris számlálóval**
  - Szimmetrikus kimenet:
  - Páros N-re: A kimenet az MSb bit, ha a kezdőérték $(2^n\text{-}N)/2$, a végértéknél $(2^n\text{-}1) - (2^n\text{-}N)/2$ újra töltve
    - Pl. osztás 6-tal, 3 bites számláló, 1,2,3 – 4,5,6 állapotokkal, ekkor az MSb, a 2. bit 000-111-000-111 módon változik
  - Páratlan N-re a pontos 50% kitöltési tényező közvetlenül nem állítható be (±1 különbség)



    - Hogyan lehet szimmetrikus páratlan osztást tervezni?
      - (A CLK 50%-os és szabad lefutó él vezérelt DFF-ot használni)

Funkcionális egységek

- **Időzítés bináris számlálóval**
  - Az időzítés T_{CLK} függvénye, ez a legfinomabb felbontás
  - Időtartam betöltése adatbemeneten (K-1) [$*T_{clk}$]
  - Indítás START pulzus beállítja EN = 1-et
  - K ciklus múlva TC = 1 pulzus megjelenik a kimeneten és törli az EN jelet, ha csak egyszeres időzítés kell



  - 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 értékével egyenlő.

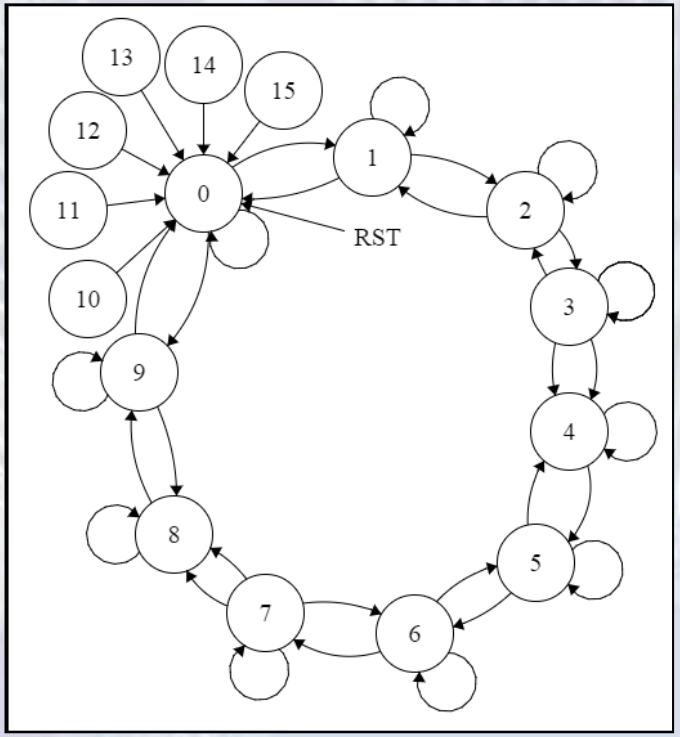
Funkcionális egységek

- **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 kaszkádosítás fontos bővítési lehetőség
- **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.)

Digitális technika
5. EA vége