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

Digitális technika
VIMIAA01

Fehér Béla
BME MIT

Véges állapotú vezérlők tervezése

- **A tervezés menete:**
  - Specifikáció analizálása, teljes megértése (gondolat kísérlet a működésre)
  - A megértési folyamat során az állapot diagram vagy az előzetes állapottábla felvétele
    → lényegében itt dől el a terv helyessége
  - A továbbiak is fontos technológiai lépések, de a megoldás minőségét már nem annyira befolyásolják
    - Állapotminimalizálás
    - Állapotkódolás
    - Állapotátmeneti logika specifikálása
    - Kimeneti logika specifikálása

Véges állapotú vezérlők tervezése

- **Állapotminimalizálás:**
  - Sok esetben a specifikáció alapján felrajzolt állapot diagram/előzetes állapottábla tartalmaz(hat) redundáns állapotokat.
  - Ez természetes, hiszen elsősorban a feladat pontos megértésére törekedtünk, nem pedig azonnal a legkedvezőbb verzió megtalálására
  - Az állapotminimalizálás alapja az állapotok közötti tulajdonságok kiértékelése az állapotgép típusa szerint
    - TSH: Teljesen specifikált hálózat
    - NTSH: Nem teljesen specifikált hálózat

Véges állapotú vezérlők tervezése

- **Állapotminimalizálás:**
  - TSH: Teljesen Specifikált Hálózat
    - Minden állapotban a következő állapot/kimenet előírt érték
    - Tulajdonság: EKVIVALENS ÁLLAPOTOK
  - NTSH: Nem Teljesen Specifikált Hálózatok
    - Az állapotátmenetek/kimenetek nem teljesen specifikáltak, néhány esetben lehet közömbös bejegyzés (don’t care) (pl. egy reakcióidő mérő kimenete a START-STOP között lehet kikapcsolt 0, vagy tetszőleges futó idő számláló)
    - Tulajdonság: KOMPATIBILIS ÁLLAPOTOK
  - A továbbiakban csak a TSH esettel foglalkozunk

Véges állapotú vezérlők tervezése

- **Állapotminimalizálás TSH-ban:**
  - EKVIVALENS állapotpárok (jele ≡)
    - Az adott állapotokban azonos bemenetre azonos a kimenet
    - Az adott állapotokból azonos bemenetre ekvivalens állapotokba lépnek tovább
  - Ekvivalencia tulajdonság:
    - Reflexív: $A \equiv A$
    - Szimmetrikus: $A \equiv B \rightarrow B \equiv A$
    - Tranzitív: $A \equiv B$ és $A \equiv C \rightarrow B \equiv C$
  - Diszjunktív ekvivalencia osztályok
    → Maximális Ekvivalencia Osztályok (MEO)

Véges állapotú vezérlők tervezése

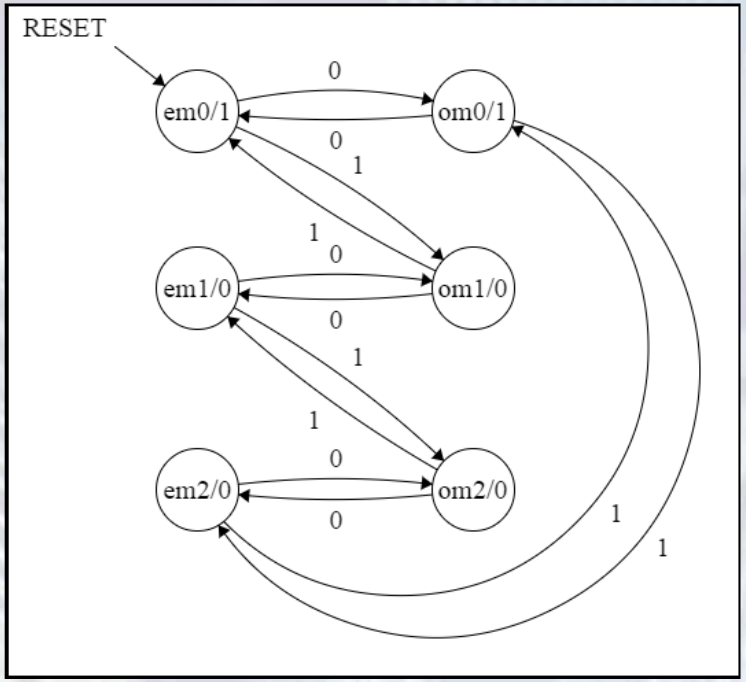
- **Állapotminimalizálás TSH-ban, MEO generálása**
  - Partíció finomítás
    1. Az összes állapot képez egy közös kiinduló partíciót
    2. Eltérő kimenetek alapján szétválogatjuk → új partíciók
    3. Felírjuk a következő állapotok partícióit minden állapothoz
    4. Ellenőrizzük a továbblépési előírás zártságát a következő állapotok partíciói lapján
    5. Teljesül mindenhol az EKV állapotba továbblépés? → KÉSZ
    6. Ha nem, újabb szétválogatás (partíció finomítás) és ismétlés a 3. ponttól

Véges állapotú vezérlők tervezése

- **Példa állapotminimalizálásra MEO generálásával**
- **Feladat: Tetszőleges méretű (bitszámú) bináris számról eldönteni, hogy osztható-e 3-mal?**
- **Előzetes megjegyzések:**
  - 0 osztató, tehát egy ilyen állapotból indulunk
  - 11 osztható, 110, 1100, 11000, 1100000…, tetszőleges számú 0 után osztható 3, 6, 12, 24…
  - 11 osztható, 1111, 111111, .. Azaz páros számú 1 után osztható (3, 15, 63…
  - 1001 osztható, 10010, 1001000… tetszőleges számú 0 után osztható 9, 18, 36, 72…
  - 10101 osztható, tehát 21, 42, 84….

Véges állapotú vezérlők tervezése

- **Tehát a bejövő bitek számolása segíthet**
- **Működés: MSb…..2^5, 2^4, 2^3, 2^2, 2^1, 2^0 LSb-vel kezdve**
- **Az állapotok jelölése:**
  - e páros, o páratlan sorszámú bit érkezett (első bit (2^0) páratlan!)
  - m0, m1, m2, a maradék abban az állapotban 0, 1, 2
  - /0, /1, a kimenet értéke, m0-nál 1, m1, m2-nél 0
  - Pl. om2/0, a páratlan sorszámú bit beérkezése után az aktuális maradék értéke 2, ezért a kimenet 0.



Véges állapotú vezérlők tervezése

- **A teljes állapot halmaz: (6 állapot)**
  - em0/1, om0/1, em1/0, om1/0, em2/0, om2/0
  - Kiindulás: Kimenet alapján 2 partíció
    P01 (em0/1, om0/1), P02(em1/0, om1/0, em2/0, om2/0)

	P01		P02			
INPUT	em0/1	om0/1	em1/0	om1/0	em2/0	om2/0
0	P01	P01	P02	P02	P02	P02
1	P02	P02	P02	P01	P01	P02

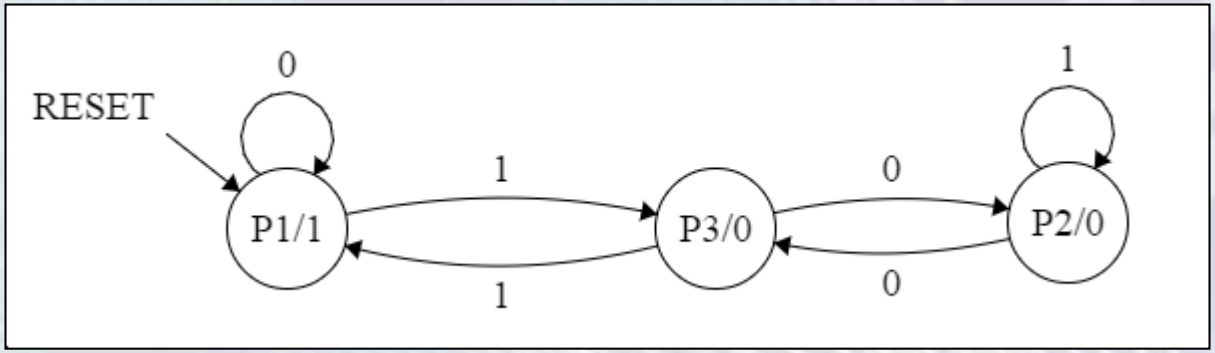
  - Ellenőrzés: A P02 partíció nem zárt a sárgával jelölt állapotátmenetek miatt → További partíció finomítás

	P11		P12		P13	
INPUT	em0/1	om0/1	em1/0	om2/0	om1/0	em2/0
0	P11	P11	P13	P13	P12	P12
1	P13	P13	P12	P12	P11	P11

  - em0 ≡ om0; em1 ≡ om2; em2 ≡ om1;

Véges állapotú vezérlők tervezése

- **3 MEO: P11, P12, P13, mindegyik 2-2 állapottal**
  - em0 ≡ om0; em1 ≡ om2; em2 ≡ om1;
- **Ezek összevonhatók, pl. P1, P2, P3 néven**
- **Az új állapotdiagram:**



  - A 3 állapotra redukált működés egyszerűbben realizálható, a rendszer áttekinthetőbb
  - Mindig vizsgáljuk meg az állapot minimalizálás lehetőséget (a fejlesztőrendszerek egy része képes rá)

Véges állapotú vezérlők tervezése

- **Állapotkódolás: M állapotot kell kódolni**
- **Minimum n = $\lceil \log_2 M \rceil$ bit kell ehhez**

$$\mathbf{N} = \frac{\binom{2^n}{M} M!}{n! 2^n}$$

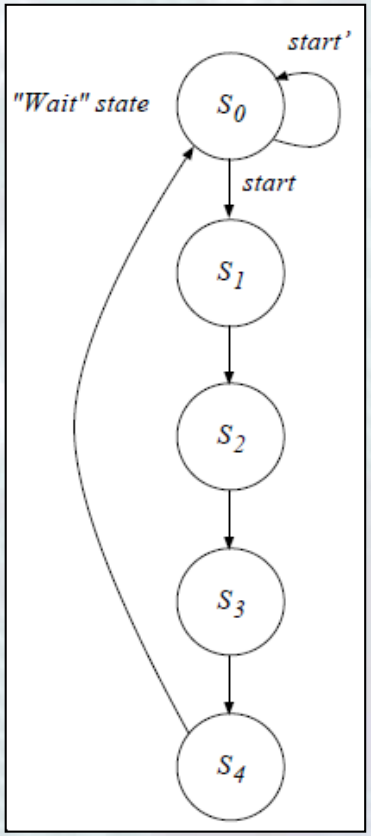
  - (n=2, M=4, N=3), (n=3, M=5, N=140), (n=3, M=8, N=840)
  - Nagyon nagy számú kódolási variációs lehetőség
  - Minimális bitszám: (Bináris, Gray vagy más kiosztás)
    - Kevés FF, de esetleg bonyolult függvények (Áll. át. + kim)
  - Több bit használata (1-az-N-ből, 2-az-N-ből, kimeneti)
    - Több FF, de összességében gyakran egyszerűbb logika

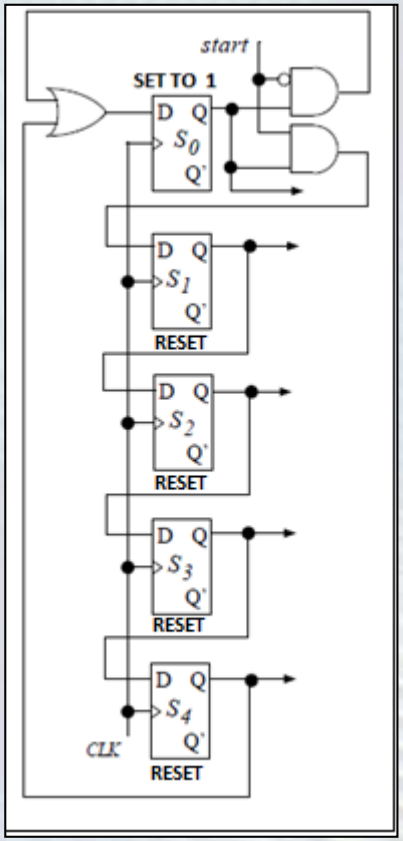
Véges állapotú vezérlők tervezése

- **Állapotkódolás**
- **Nem vizsgáljuk meg az összes lehetséges verziót**
  - Manapság nem szempont a **speciális egyedi** előnyök kihasználása fáradságos analízissel
  - Az állapotkódok kijelölését gyakran a fejlesztőrendszerre hagyjuk (Synt. Opt.: Auto), az eredményt értékeljük: sebesség, erőforrásigény
  - Ha nem megfelelő, akkor előírásokat tehetünk:
    - Kompakt, bináris, a szokásos sorrendű kijelöléssel
    - Gray, ahol a hosszabb állapotsorozatokat így jelöli ki
    - 1-az-N-ből kódolás, (One-Hot) sok FF, egyszerű logika
    - 2-az-N-ből, hasonló, mint az előző, de kevesebb FF kell
    - Kimenet szerinti kódolás, azaz a kimenőjeleket közvetlenül az állapotbitekkel realizáljuk

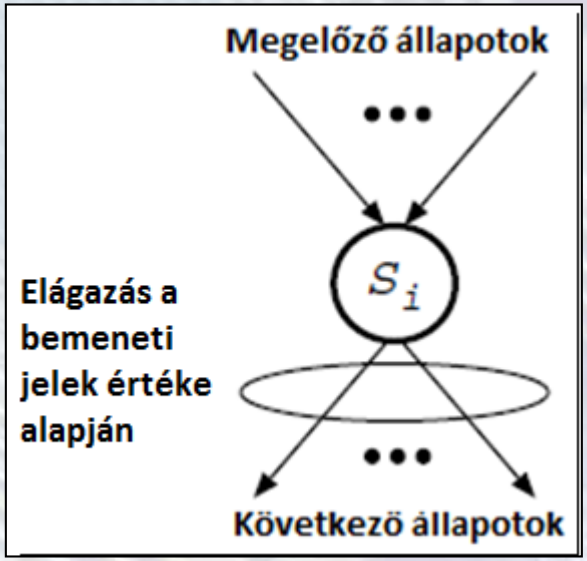
Véges állapotú vezérlők tervezése

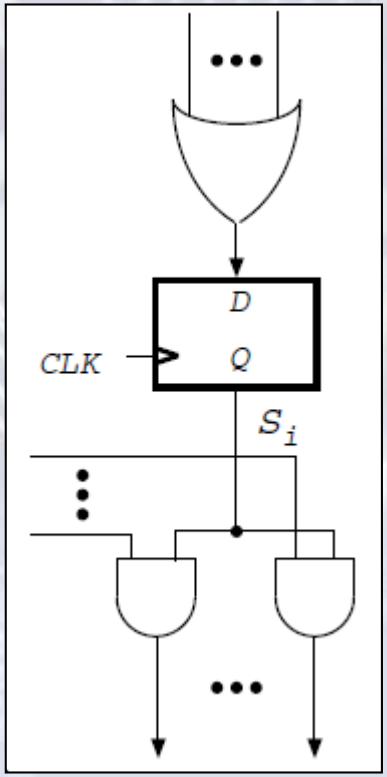
- **1-az-N-ből kódolás előnyei**
  - Közvetlen kapcsolat az állapot és a realizáció között





Általános struktúra



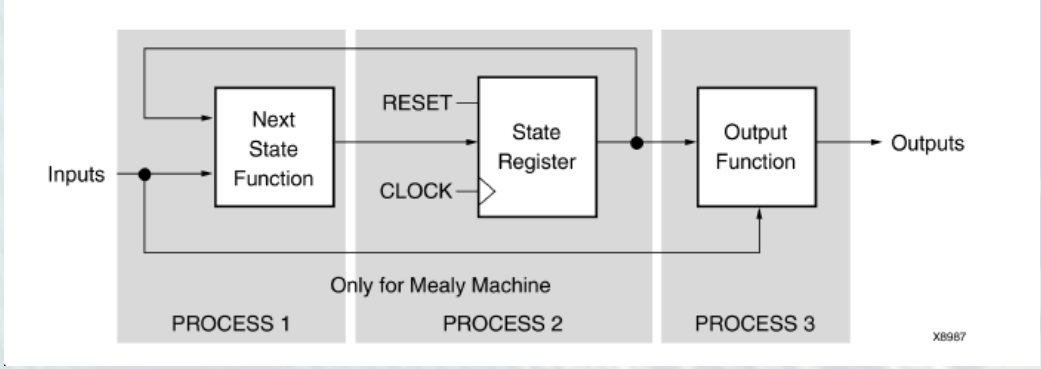


Véges állapotú vezérlők tervezése

- **Az állapotkódolás melléktermék**
  - Gyakran maradnak nem használt állapotkódok, ezeket a működés szempontjából tekinthetjük illegális állapotoknak.
  - Elméletileg ezeket soha nem veszi fel a rendszer
    - Elméletileg…(pl. BCD számláló töltése 12-vel 1100, mi legyen a következő állapot?)
    - Bármilyen zavar (táp, külső sugárzás, nagy energiájú alfa részecske), →”soft error”, a FF állapotot vált, mi történik?
  - Mi a jó stratégia?
    - Szükséges foglalkozni vele?
    - Mikor kell kezelni?
    - Mi a biztonsági kérdés?

Véges állapotú vezérlők tervezése

- **Javasolt Verilog HDL kódolási technika**
  - 3 független **always** blokk



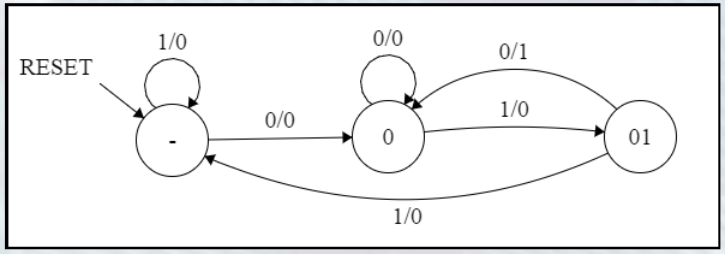

```
always @(*)
   begin
      case (state)
        A:   nextstate <= B;
        B:   nextstate <= C;
        C:   nextstate <= A;
        default:
             nextstate <= A;
      endcase
  end
```

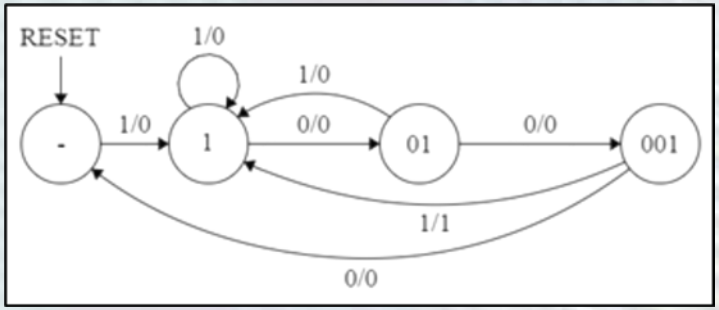
```
parameter A = 2'b00;
parameter B = 2'b01;
parameter C = 2'b10;

reg [1:0] state, nextstate;

always @ (posedge clk)
   begin
      if (rst) state <= A;
      else     state <= nextstate;
   end
```

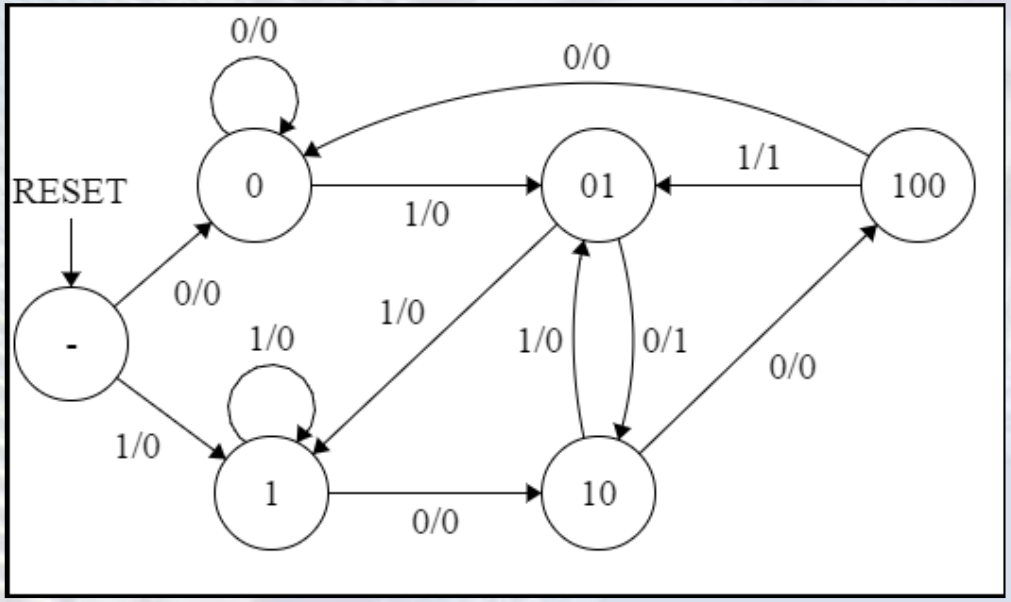
```
always @(*)
   case (state)
      A:       out <= 1'b0;
      B:       out <= 1'b1;
      C:       out <= 1'b1;
      default: out <= 1'b0;
   endcase
```

Véges állapotú vezérlők tervezése

- **Speciális esetek**
- **Mintafelismerő több mintára**
  - Pl. az 1001 és a 010 minták egyidejű figyelése
  - Triviális verzió: Független mintafelismerők párhuzamos működtetése
  - Jobb verzió: Felismerés együttes állapotdiagramban







Véges állapotú vezérlők tervezése

- **Speciális esetek**
- **Hierarchikus vezérlőegységek**
  - Alkalmazástól függően esetleg az állapot sorozat túl sok állapotból állhat egyszerű egyszintű realizációnál
  - Pl. kerékpár hátsó világítás
  - Funkciók:
    - Kikapcsolt
    - Futófény
    - Villog
    - Világít

Véges állapotú vezérlők tervezése

- **Két verzió:**
  - Közvetlen egyszintű állapotgép (kb. 30 állapot)
    - Szintézis opció: FSM kódolás Auto
    - Szintézis opció:FSM kódolás User
  - Hierarchikus vezérlési struktúra
    - Felső szint:Üzemmód beállítás (1-az-N-ből kódolás)
    - Alsó szint: Villogás, futófény (egyszerű bináris számláló)

Közvetlen egyszintű állapotgép

- **Működés minden fázisa egy-egy állapotsorozat**
- **Kikapcsolt→ Futófény→Villog→Világít→Kikapcs**
  - Összesen kb. 30 állapot, egy-egy lokális ciklussal
  - PULSE lépteti tovább a következő ciklusba
  - Egyetlen **case** szerkezet
  - Működik…

```
reg [4:0] state;
always @ (posedge clk)
if (rst) state <= 5'h00;
else
   case (state)
   5'h00: if (pulse) state <= 5'h01; else state <= 5'h00;   // Az első 12 állapotban
   5'h01: if (pulse) state <= 5'h10; else state <= 5'h02;   // gyors pásztázó villogás,
   5'h02: if (pulse) state <= 5'h10; else state <= 5'h03;
   5'h03: if (pulse) state <= 5'h10; else state <= 5'h04;
   5'h04: if (pulse) state <= 5'h10; else state <= 5'h05;
   5'h05: if (pulse) state <= 5'h10; else state <= 5'h06;
   5'h06: if (pulse) state <= 5'h10; else state <= 5'h07;
   5'h07: if (pulse) state <= 5'h10; else state <= 5'h08;
   5'h08: if (pulse) state <= 5'h10; else state <= 5'h09;
   5'h09: if (pulse) state <= 5'h10; else state <= 5'h0a;
   5'h0a: if (pulse) state <= 5'h10; else state <= 5'h0b;
   5'h0b: if (pulse) state <= 5'h10; else state <= 5'h0c;
   5'h0c: if (pulse) state <= 5'h10; else state <= 5'h01;

   5'h0d: if (pulse) state <= 5'h10; else state <= 5'h00;   // Nem használt állapotok
   5'h0e: if (pulse) state <= 5'h10; else state <= 5'h00;
   5'h0f: if (pulse) state <= 5'h10; else state <= 5'h00;

   5'h10: if (pulse) state <= 5'h1f; else state <= 5'h11;   // A második 16 állapotban
   5'h11: if (pulse) state <= 5'h1f; else state <= 5'h12;   // 4 órajel bekapcsolás,
   5'h12: if (pulse) state <= 5'h1f; else state <= 5'h13;   // éa 4 órajel kikapcsolás
   5'h13: if (pulse) state <= 5'h1f; else state <= 5'h14;
   5'h14: if (pulse) state <= 5'h1f; else state <= 5'h15;
   5'h15: if (pulse) state <= 5'h1f; else state <= 5'h16;
```

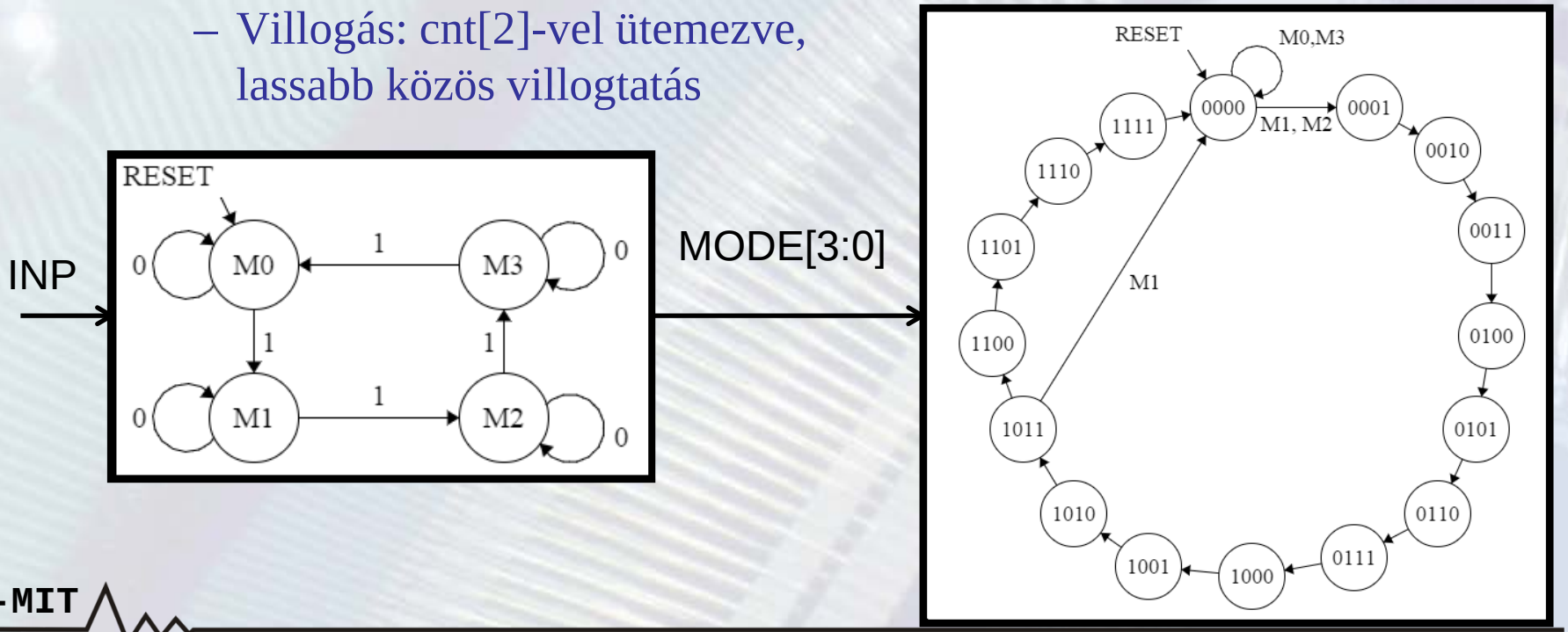
Hierarchikus verzió

- **Hierarchikus vezérlés**
  - Felső szint: Módvezérlő állapotgép (1-a-4-ből kódolás)
  - Alsó: Villogás ütemezés állapotgép (bináris számláló)
  - **Mode0**: LED OFF
  - **Mode1**: Működés cnt[0] bit ütemében
  - **Mode2**: Működése cnt[2] bit ütemében (1/4 frekvencia)
  - **Mode3**: LED ON

```
// Üzemmód vezérlő egység
// A működésnek megfelelően a 4 üzemmód között válas
// A nyomógomb hatására vált üzemmódot,
// a mode változó értékei 0001, 0010, 0100, 1000

reg [3:0] mode;

always @ (posedge clk)
if (rst) mode <= 4'h1;
else
   if (pulse) mode <= {mode[2:0],mode[3]};    // Nem

// Villogás ütemezés az órajel leosztásával
// 1. üzemmódban a cnt[0], 2. üzemmódban a cnt[2] üte
// 1. üzemmódban a modulus csökkentve 12-re, a J_J_K

reg [3:0] cnt;
always @ (posedge clk)
if (rst) cnt <= 4'h0;
   else  if (mode[1] & (cnt == 4'hb))  cnt <= 4'h0;
         else cnt <= cnt +1;
```

Hierarchikus verzió

- **Hierarchikus vezérlési struktúra**
  - Felső szint: Üzem módbeállítás (1-az-N-ből kódolás)
  - Alsó szint: Villogás, futófény (egyszerű bináris számláló)
  - Kimenet:
    - Futófény: cnt[0]-val ütemezve, gyors léptetés 3 LED-en, B B K J J K B B K J J K, 12 ütem alatt
    - Villogás: cnt[2]-vel ütemezve, lassabb közös villogtatás



Eredmények elemzése

- **Az XST szintézis eszköz riportja**
- **1a. verzió: Nem adtunk meg állapotkódokat, csak leírtuk a működést (felsoroltuk „state[4:0]” értékeit) → Felismeri, hogy ez lehet egy FSM**

```
Found finite state machine <FSM_0> for signal <state>.
-----------------------------------------------------------------------
| States             | 29
| Transitions        | 58
| Inputs             | 1
| Outputs            | 15
| Clock              | clk                           (rising_edge)
| Reset              | rst                           (positive)
| Reset type         | synchronous
| Reset State        | 00000
| Encoding           | automatic
| Implementation     | LUT
-----------------------------------------------------------------------
Summary:
    inferred   1 Finite State Machine(s).
```

```
Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <state/FSM> on signal <st
-----------------------------------------
 State | Encoding
-----------------------------------------
 00000 | 00000000000000000000000000000001
 00001 | 00000000000000000000000000000010
 00010 | 00000000000000000000000000001000
 00011 | 00000000000000000000000000010000
 00100 | 00000000000000000000000000100000
 00101 | 00000000000000000000000001000000
 00110 | 00000000000000000000000010000000
 00111 | 00000000000000000000000100000000
 01000 | 00000000000000000000001000000000
 01001 | 00000000000000000000010000000000
 01010 | 00000000000000000000100000000000
 01011 | 00000000000000000001000000000000
 01100 | 00000000000000000010000000000000
 01101 | unreached
 01110 | unreached
 01111 | unreached
 10000 | 00000000000000000000000000000100
 10001 | 00000000000000100000000000000000
 10010 | 00000000000001000000000000000000
 10011 | 00000000000010000000000000000000
 10100 | 00000000000100000000000000000000
 10101 | 00000000001000000000000000000000
```

- **Az automatikus állapotkódolás megengedi, hogy az 1-az-N-ből kódolásra áttérjen**

Eredmények elemzése

- **Az XST szintézis eszköz riportja**
- **1b. verzió: Nem adtunk meg állapotkódokat, csak leírtuk a működést (felsoroltuk „state[4:0]” értékeit) → Felismeri, hogy ez lehet egy FSM**
- **De nincs megengedve, hogy optimalizáljon, ezért az előírt feltételekkel dolgozik**

```
Found finite state machine <FSM_0> for
-----------------------------------
| States             | 29
| Transitions        | 58
| Inputs             | 1
| Outputs            | 15
| Clock              | clk
| Reset              | rst
| Reset type         | synchronous
| Reset State        | 00000
| Encoding           | user
| Implementation     | LUT
-----------------------------------
```

```
Optimizing FSM <state/FSM> on signal <state[1:5]> with user encoding.
-------------------
 State | Encoding
-------------------
 00000 | 00000
 00001 | 00001
 00010 | 00010
 00011 | 00011
 00100 | 00100
 00101 | 00101
 00110 | 00110
 00111 | 00111
 01000 | 01000
 01001 | 01001
 01010 | 01010
 01011 | 01011
 01100 | 01100
 01101 | unreached
 01110 | unreached
 01111 | unreached
 10000 | 10000
 10001 | 10001
```

Eredmények elemzése

- **Az XST szintézis eszköz riportja**
- **2. verzió: Nem írtunk elő állapotkódokat, csak leírtuk a működést →**
  Nem ismer fel FSM-et, csak egyedi logikai egységeket

```
Synthesizing Unit <bicycle_rear>.
    Related source file is "../bicycle_rear.v".
    Found 4-bit up counter for signal <cnt>.
    Found 4-bit register for signal <mode>.
    Summary:
        inferred   1 Counter(s).
        inferred   4 D-type flip-flop(s).
Unit <bicycle_rear> synthesized.
```

- **Nem alkalmazza azokat a módszereket, amik az FSM optimalizációra szolgálnak**

Kimeneti LED-ek vezérlése

- **Mindkét esetben az állapotértékek alapján**
  - Moore modell, az egyszintű vezérlőre mutatva
    - FUTÓFÉNY: 3 LED-en (JOBB – KÖZÉP – BAL)
    - Minden más üzemmódban COMMON, azaz mind a 3 LED együttesen működik

```
// A LED vezérlés az állapotok alapján kiválogatva
assign common = (state == 5'h10) | (state == 5'h11) | (state == 5'h12) | (state == 5'h13) |
                (state == 5'h18) | (state == 5'h19) | (state == 5'h1a) | (state == 5'h1b) |
                (state == 5'h1f) ;

// Jobb oldali LED-ek
assign led[7:6] = {2{(state == 5'h01) | (state == 5'h03) | common}};

// Középső LED-ek
assign led[5:4] = {2{(state == 5'h05) | (state == 5'h0b) | common}};

// Bal oldali LED-ek
assign led[3:2] = {2{(state == 5'h07) | (state == 5'h09) | common}};
```

Minőségi paraméterek

- **Erőforrásigény, működési sebesség**
- **1.a verzió, FSM encoding: Auto (One Hot)**

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	24	2448	0%
Number of Slice Flip Flops	31	4896	0%
Number of 4 input LUTs	42	4896	0%
Number of bonded IOBs	11	108	10%
Number of GCLKs	1	24	4%

```
Minimum period: 4.314ns (Maximum Frequency: 231.786MHz)
Minimum input arrival time before clock: 5.097ns
Maximum output required time after clock: 7.439ns
Maximum combinational path delay: No path found
```

- **1.b verzió, FSM encoding: User (Bináris)**

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	16	2448	0%
Number of Slice Flip Flops	7	4896	0%
Number of 4 input LUTs	31	4896	0%
Number of bonded IOBs	11	108	10%
Number of GCLKs	1	24	4%

```
Minimum period: 6.070ns (Maximum Frequency: 164.745MHz)
Minimum input arrival time before clock: 6.791ns
Maximum output required time after clock: 6.722ns
Maximum combinational path delay: No path found
```

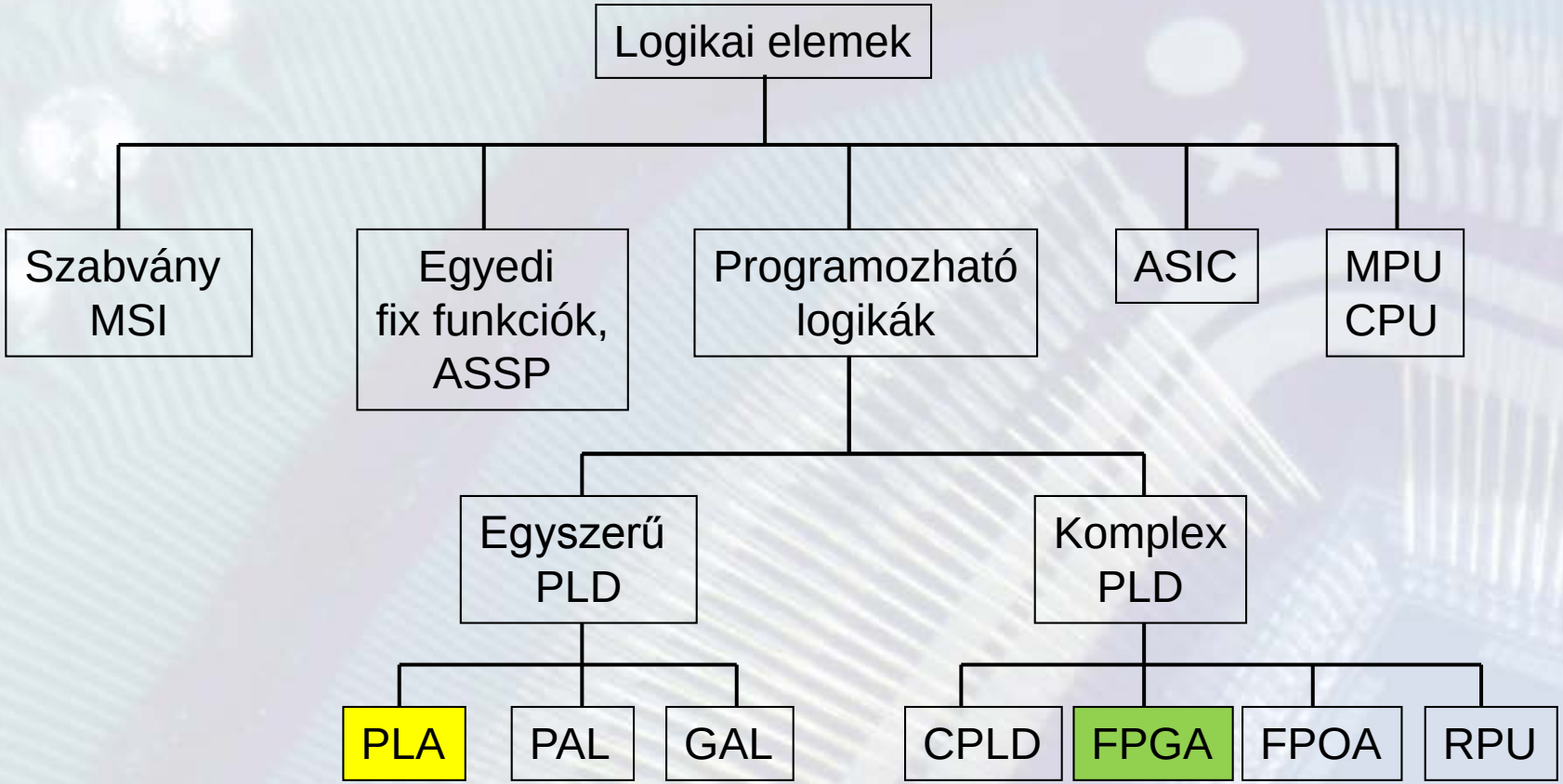
- **2. verzió Hierarchikus terv (SHR + CNT)**

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	8	2448	0%
Number of Slice Flip Flops	10	4896	0%
Number of 4 input LUTs	14	4896	0%
Number of bonded IOBs	11	108	10%
Number of GCLKs	1	24	4%

```
Minimum period: 4.445ns (Maximum Frequency: 224.972MHz)
Minimum input arrival time before clock: 4.309ns
Maximum output required time after clock: 7.136ns
Maximum combinational path delay: No path found
```

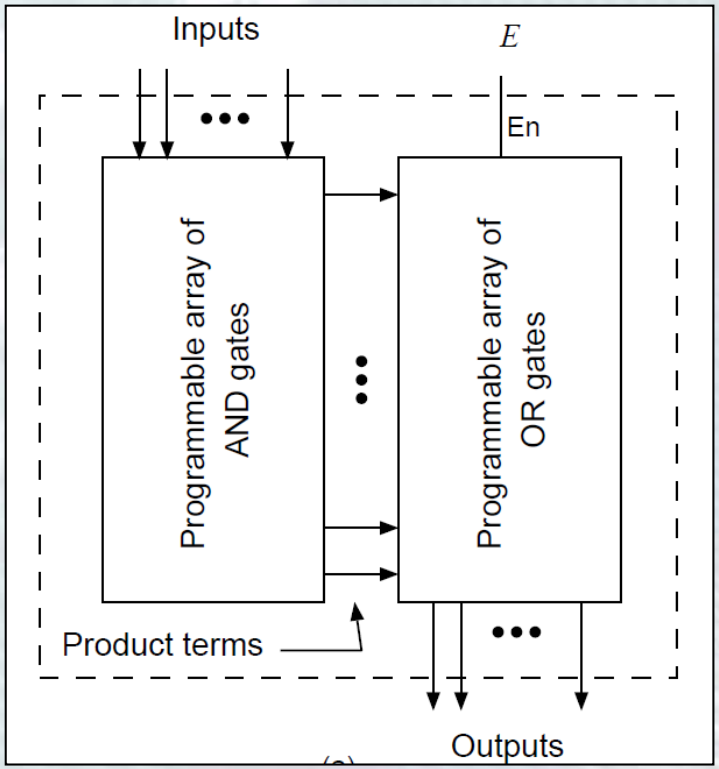
Programozható logika: Az FPGA

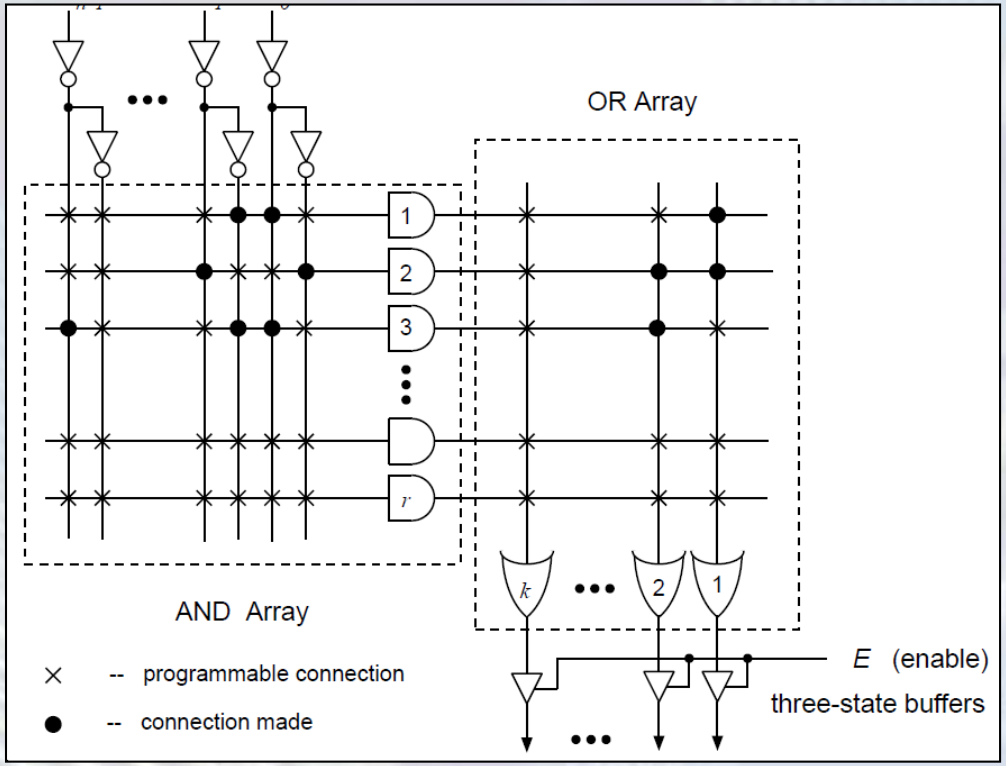
- **Digitális eszközök típusai**



Programozható logika: FPGA

- **Korai programozható logikák:**
  - **PLA: Programmable Logic Array, kétszintű AND-OR**

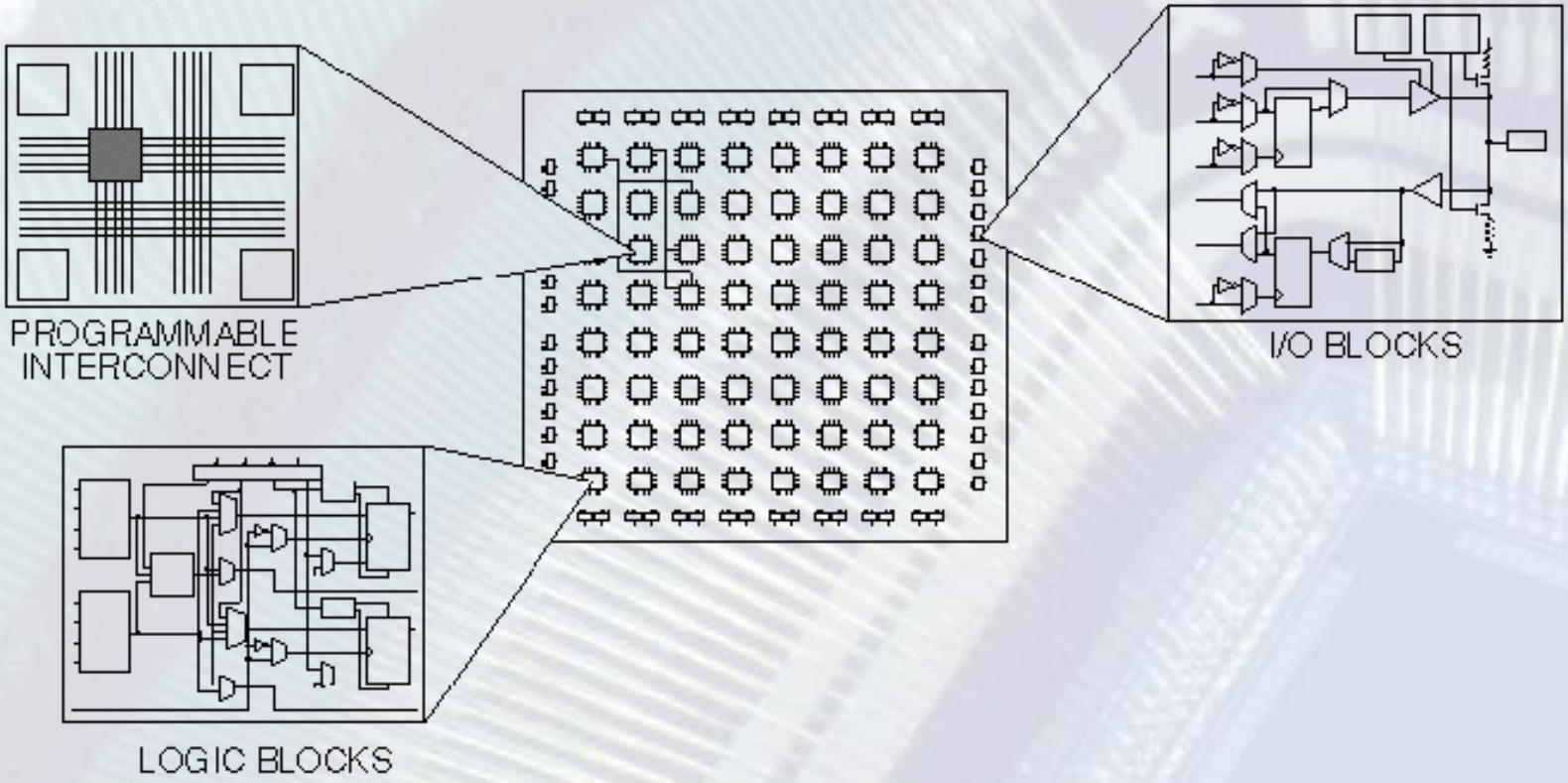




- **Közvetlen kétszintű realizáció, teljes változó használat, bemenet/kimenet leképezés**

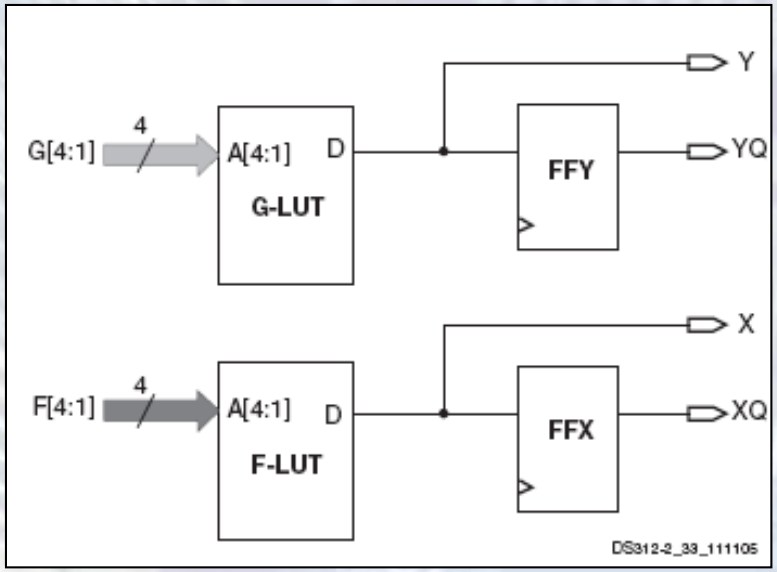
FPGA felépítés

- **Általános felépítés**
  - Logikai blokkok, huzalozás, I/O blokkok



A logikai blokk

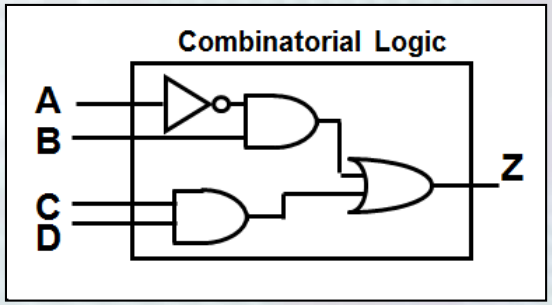
- **A logikai blokk felépítése**
- **Elemi kombinációs és elemei szekvenciális alapelemek**
- **Az alap erőforrás a Logic Cell LC = 1 LUT + 1 FF**
- **LUT4**
  - tetszőleges 4 változós fgv.
  - 1 változóra hazárdmentes
  - Végrehajtási idő bemeneti jel és logikai komplexitás invariáns
- **DFF**
  - Élvezérelt, ↑↓, órajel eng.
  - Szink/Aszink. SET/RESET



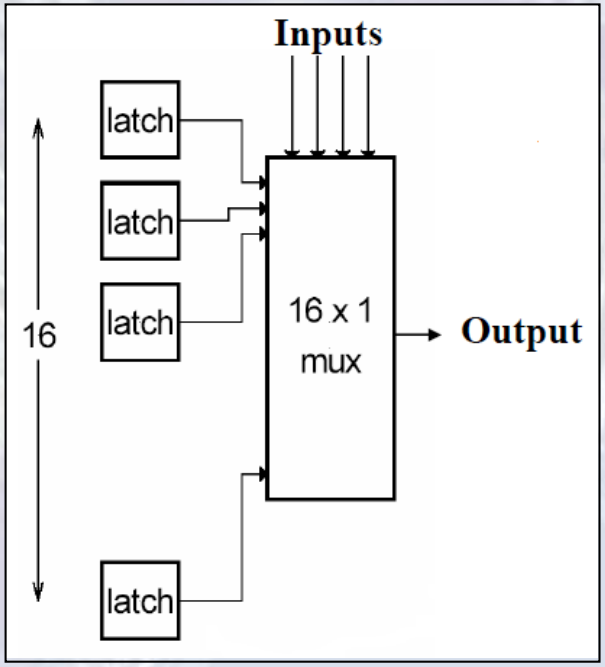
- **Független kombinációs és regiszteres kimenet, vagy 2:1 MUX választ, hogy kombinációs vagy szekvenciális kimenet**

A logikai blokk

- **A LUT4 funkcionalitása: Egy táblázatnak tekintjük**
- **Tetszőleges tartalom betölthető**
  - A komplexitás a bemenetek számában korlátos, nem a logikai függvény összetettségében

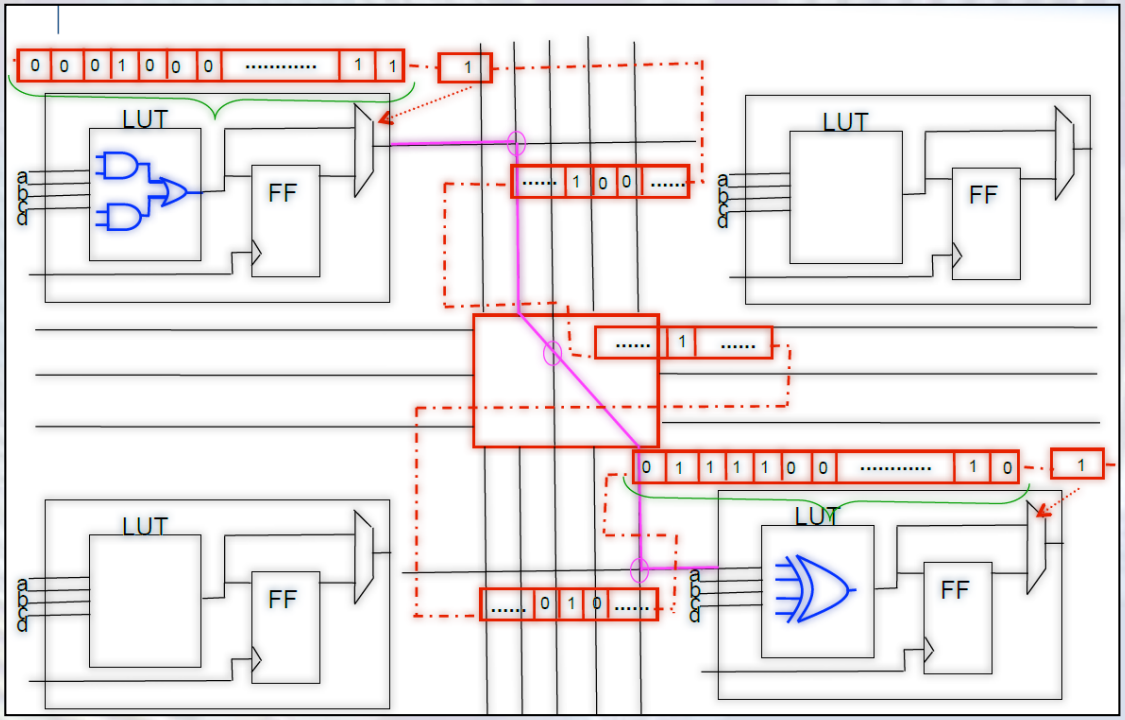


BEMENETEK				KIM
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



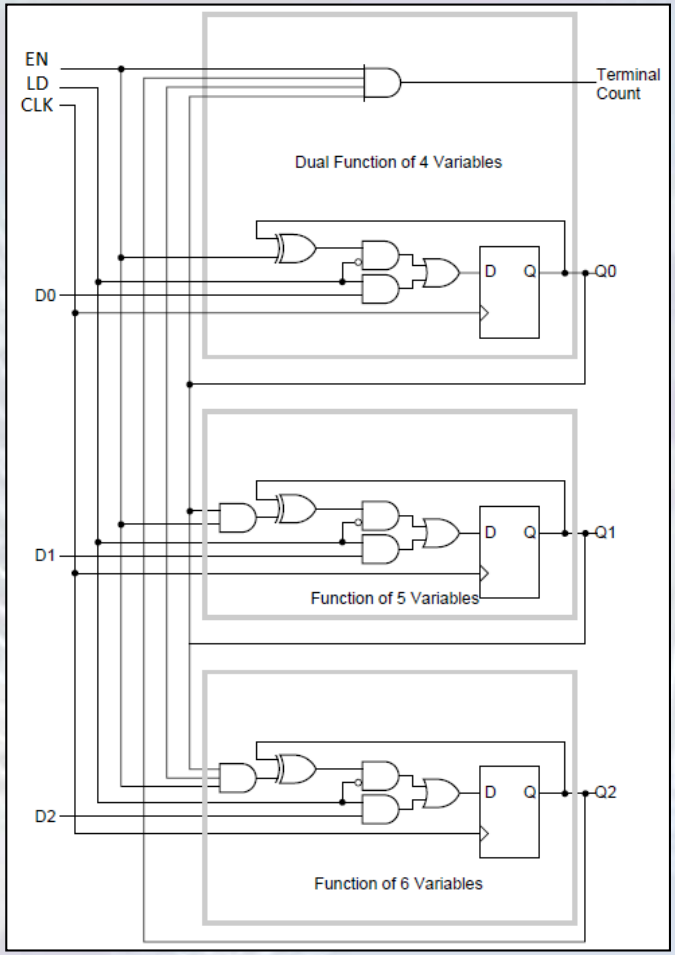
A logikai blokk használata

- **Egy mintaáramkör realizációja**
  - Logika felosztása 4bemenetű, 1 kimenetű funkciókra
  - A „kis” funkciók elhelyezése egy-egy LUT-ban
  - Szükséges kapcsolatok huzalozásának kialakítása
  - Konfigurációs (programozó) bitek generálása és betöltése
  - A kívánt funkcióra felprogramozott eszköz használata



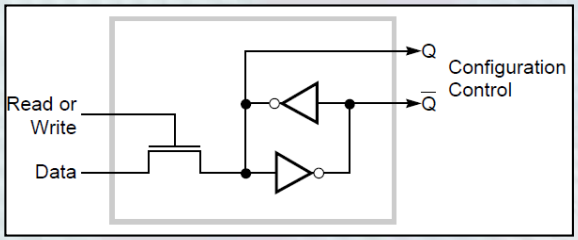
A logikai blokk használata

- **Egy 3 bites bináris számláló**
  - Funkciók: töltés és engedélyezés
  - Minden bitet egy-egy CLB realizál
  - A végértékjelzés egy újabb LUT
  - A huzalozások szomszédosak

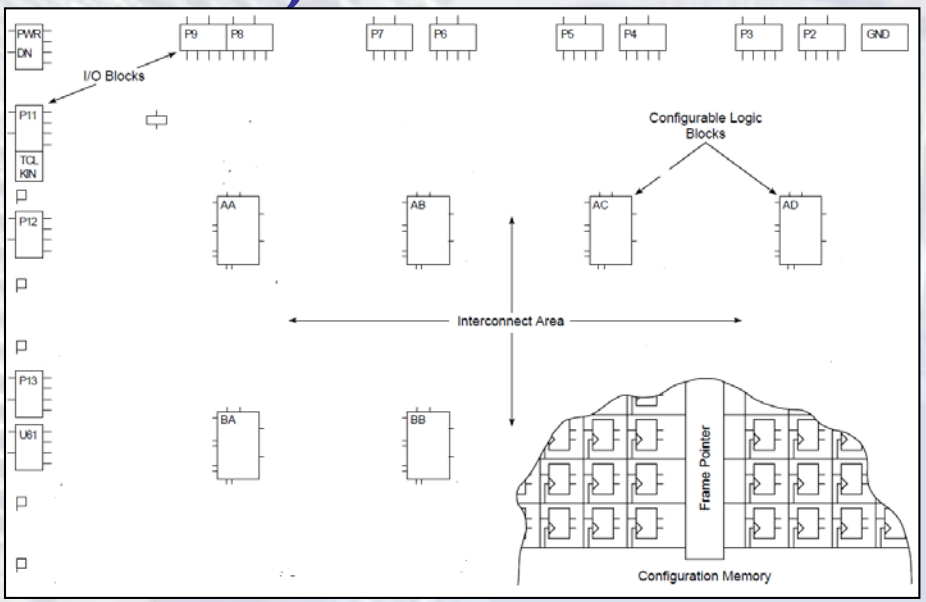


FPGA felépítése

- **A valódi komplexitás részben rejtve van**
- **Két logikai réteg: Konfigurációs + Felhasználói**
  - Mi csak a felhasználóit szeretnénk látni
- **Konfigurációs logika: Shiftregiszter egyszerű SRAM latch tárolókból → A teljes tartalom beléptetése induláskor (CLB, IOB, huzalozás)**

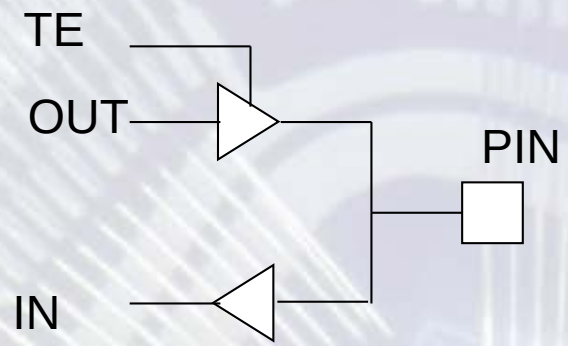


- **Beírás, visszaolvasás, ellenőrzés, indítás**



I/O funciók

- **Alapvetően minden felhasználói láb I/O, tehát lehet kimenet, bemenet**
- **Nem használt lábak fix értéken**
- **Gyakran többfunkciós lábak**
  - Konfiguráció alatt
  - Normál használat alatt



- **A valódi I/O blokkok sokkal bonyolultabbak**
  - Tartalmaznak bemeneti/kimeneti DFF-okat is
  - Különböző kimeneti opciók (Jelszint, sebesség, meghajtás erősség, felhúzó/lehúzó ellenállás)

Digitális technika
7. EA vége