



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

Digitális technika

VIMIAA02

3. EA

Fehér Béla, Benesóczky Zoltán
BME MIT

Funkcionális egységek

- Az egyedi „kapuszentű” logikai függvények tervezésénél sokszor egyszerűbb/célravezetőbb szabványos modulokból építkezni és ezekből felépíteni a rendszert.
- A szabványos modulok (funkcionális egységek) célja, hogy a leggyakoribb, *tipikus feladatokra* (funkciókra) *biztosítsanak egyszerűen használható, megbízhatóan működő, skálázható (könnyen módosítható méretű) megoldási lehetőséget.*
- A *logikai alapelemek* (kapuk stb.) *helyett* azoknál akár sokkal komplexebb *funkcionális elemekből* (modulokból) *építkezünk.*

Funkcionális egységek

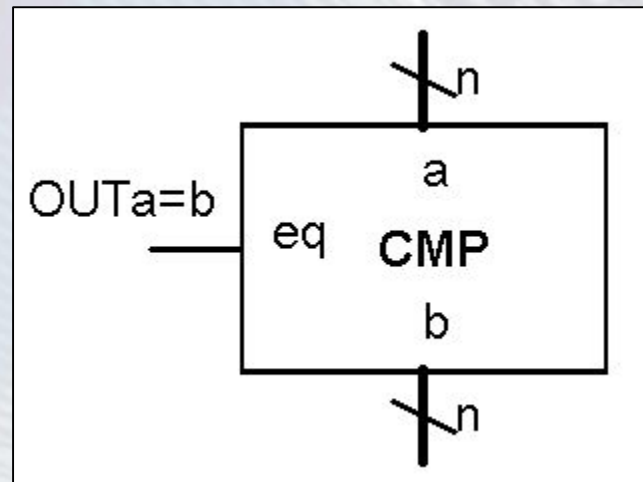
- **Általános célú kombinációs logikai funkciók**
 - **Aritmetikai funkciók:**
összeadó (**ADD**), kivonó (**SUB**), (komparátor (**CMP**)
 - **Logikai funkciók:**
dekóder (**DEC**), demultiplexer (**DEMUX**),
multiplexer (**MUX**), enkóder (**ENC**), prioritás enkóder (**PRI**),
 - Egyéb, esetleg szükséges funkciók

Funkcionális egységek

- Hagyományos technológiában (TTL MSI IC-k) *minden funkcióra önálló elem/alkatrész létezett*, különböző méretekben, tokozásban, lábszámban.
- FPGA-ban, könyvtári alapú, modulszintű építkezésnél paraméterezhető alapfunkciók használhatók, az aktuális terv igényei szerint beállítva.
- **HDL alapú tervezésnél a tanult *tervezési mintákat használunk***. A továbbiakban ilyeneket mutatunk be.

Funkcionális egységek komparátor

- **Komparátor (CMP)**
 - Feladata értékek, adatok összehasonlítása
 - Két azonos bitszámú számot hasonlít össze
- **Egyenlőség komparátor**
 - Akkor ad 1-et a kimenete, ha a két szám egyenlő, vagyis a két szám azonos sorszámú bitjei azonosak



Funkcionális egységek komparátor

- Egybites számok esetén az XNOR (ekvivalencia kapu) művelet azonos értékű bitek esetén jelez

$$f = ab + \neg a \neg b = a \odot b$$

a	b	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

- n bites számok esetén akkor kell jelezni, ha az egyenlőség minden bitre teljesül:

$$\text{equ} = (a_0 \odot b_0)(a_1 \odot b_1) \dots (a_{n-1} \odot b_{n-1})$$

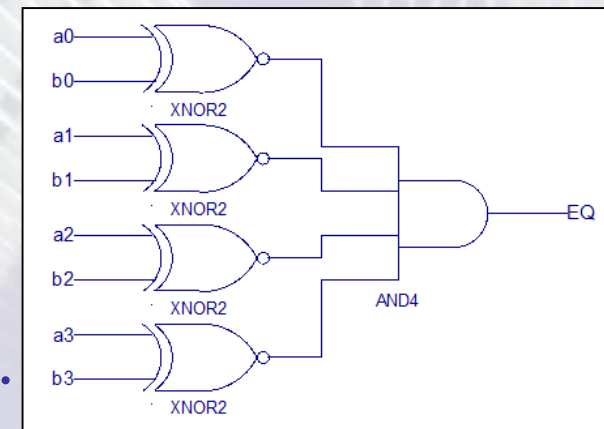
- Verilog leírása 4 bitre

```
wire[3:0] a,b;
```

```
wire equ;
```

```
assign equ = (a == b);
```

Tetszőleges kódolásra működik.



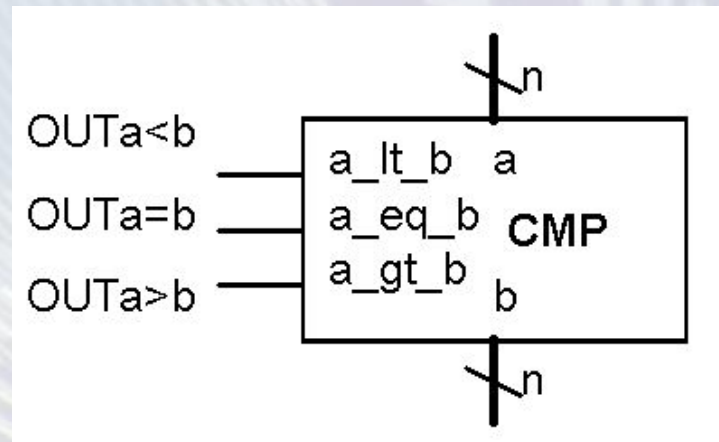
Funkcionális egységek

komparátor

Teljes funkciójú (magnitudo vagy nagyság) komparátor

- A teljes funkciójú komparátor a két adat *egyenlősége* (*a_eq_b*) mellett egy-egy kimenetén azt is jelzi, hogy melyik a *nagyobb* (*a_gt_b*) ill. *kisebb* (*a_lt_b*).
- Verilog leírás, 4 bites példa:

```
wire [3:0] a, b;  
wire a_lt_b, a_eq_b, a_gt_b;  
assign a_lt_b = (a < b);  
assign a_eq_b = (a == b);  
assign a_gt_b = (a > b);
```
- Kivonó is használható a nagysági viszony eldöntésére (lásd később).



Funkcionális egységek: Összeadó

- Az összeadás szabályai alapján készítünk egy 1 bites teljes összeadót. A teljes összeadó bemenetei: átvitel bemenet (carry in, **ci**), összeadandók (**a**, **b**) átvitel kimenet (carry output, **co**)

- Most kivételesen megtervezzük, az igazságtáblával kezdve. Az összeadási szabályok 3 bitre

- $0+0+0=0$
 $1+0+0=1$ sorrend függetlenül
 $1+1+0=0$ $co=1$ -''-
 $1+1+1=1$ $co=1$ -''-

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

Funkcionális egységek: Összeadó

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

- A sum függvény akkor ad 1-et, ha a 3 bemenet közül 1 vagy 3 db 1 értékű. Pontosan ilyen tulajdonságó az XOR függvény!

$$s = a \oplus b \oplus ci$$

A co-t 1-eseit felírjuk szorzatok összegeként DNF alakban:

$$co = \overset{1.}{a*b*ci} + \overset{2.}{a*/b*ci} + \overset{3.}{a*b*/ci} + \overset{4.}{a*b*ci} \quad (\overset{4.}{+ a*b*ci} + \overset{4.}{+ a*b*ci})$$

$$1.+4.: a*b*ci + a*b*ci = b*ci \quad 2.+4.: a*/b*ci + a*b*ci = a*ci$$

$$3.+4.: a*b*/ci + a*b*ci = a*b$$

Az egyszerűsítések után: $co = a*b + a*ci + b*ci$

Funkcionális egységek: Összeadó

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

$$\text{sum} = a \oplus b \oplus ci$$

$$\text{co} = a*b + a*ci + b*ci$$

Verilogban:

```
wire a, b, ci, s;
```

```
assign s = a ^ b ^ ci;
```

```
assign co = a&b | a&ci | b&ci;
```

Ugyanez a + operátorral:

```
assign {co, s} = a+b+ci;
```

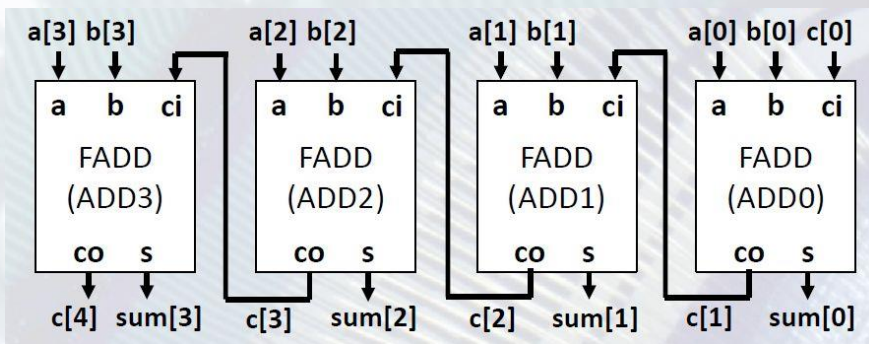
A {} jelek (konkatenálás) közé vesszővel felsorolt 1 vagy több bites változókból

egytelen annyi bites változó lesz, amennyi a bitszámok összege. A bitek sorrendje is a felsorolás szerinti lesz.

Tehát, ha $a+b+ci$ összeg 2 vagy 3, akkor a 2 bites {co, sum}-ban co értéke 1 lesz, egyébként 0.

Funkcionális egységek: Összeadó

Egy bites összeadókából készíthetünk több bitest, ahogy az alábbi ábrán látható. ($c[0] = 0$) Azonban nem érdemes.

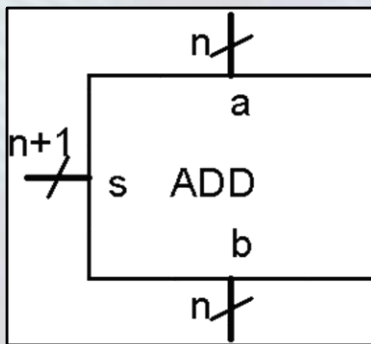


Optimálisabb megoldás készíthet a CAD rendszer, ha rögtön a kívánt bitszámú összeadót készítjük el.

Funkcionális egységek: Összeadó

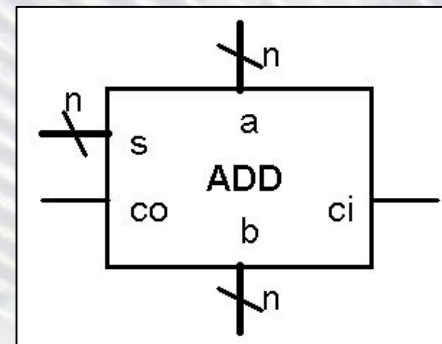
Ha az eredmény és az operandusok azonos bitszámúak, akkor az eredmény nem mindig férne el a kimenet bitjein pl. $1000_2 + 1000_2 = 10000_2$
Ha az eredmény bitszáma 1-e nagyobb, mint az operandusoké, akkor nincs probléma.

Pl: **wire [3:0] a,b;**
wire [4:0] s;
assign s = a + b;



Ha több bites kaszkádosítható megoldás kell akkor azt így lehet megadni:

wire [3:0] a,b, s;
wire co, ci;
assign {co, s} = a + b + ci;



Funkcionális egységek: Kivonó

Kivonás (előjel nélküli számok esetén)

- Szabályok: $0-0=0$, $1-1=0$, $1-0=1$, $0-1=1$ átvitel **1**
- Magyarázat az átvitelhez: (0-hoz hogy 1 legyen kell 1, átvitel **1**)
- Az eredmény előjel nélküli számok esetén *csak akkor helyes*, ha a kisebbítendő nagyobb vagy egyenlő mint a kivonandó.

Példa:

14	1110
- 11	- 1011
<hr/>	<hr/>
3	0011

Magyarázat:

1-hez, hogy 0 legyen kell 1, **marad 1**

$1+1=0$, **marad 1**, 0-hoz, hogy 1 legyen, kell 1

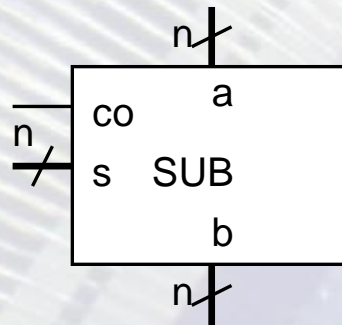
$1+0=1$, 1-hez, hogy 1 legyen kell 0, maradék 0

1-hez, hogy 1 legyen kell 0, maradék 0

Verilogban:

```
wire [3:0] a, b, c;
```

```
assign s = a - b;
```

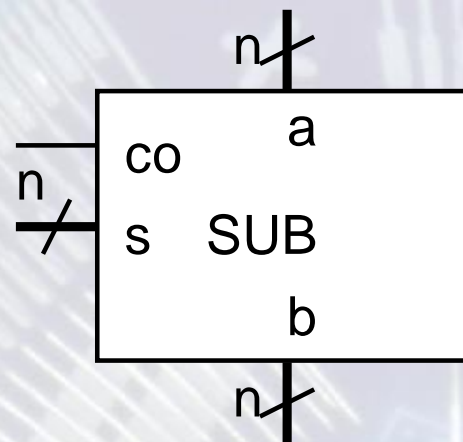


Funkcionális egységek: Kivonó

Ha előjel nélküli számok kivonásánál a kivonadó nagyobb a kisebbítendőnél akkor az utolsó átvitel értéke $co = 1$ és az eredmény hibás.

Pl.

		11	
3		0011	
-5		-0101	
<hr/>			
-2		1110	$= 14_{10}!$



Verilogban:



```
wire [3:0] a, b, s;
```

```
wire co;
```

```
assign {co, s} = a - b; // Ha co = 1 akkor a < b
```


Funkcionális egységek: Összeadó

- **Kettes komplement kódú számok összeadásására is alkalmas az összeadó.** Kivonni is tudunk vele, ha előtte a kivonandó 2-es komplementjét képezzük.
- Figyelni kell a *számtartományra*! 2-es komplement *túlcsordulás* lehet!
Ha az összeadandó számok előjele különbözik, akkor biztosan nem lesz túlcsordulás, viszont azonos előjelűek esetén előfordul.

4 bites 2-es komplement számok tartománya: +7...-8					
(+7)	0111	(-7)	1001	(+2)	0010
+ (-1)	+1111	+ (-2)	+1110	+ (+7)	+0111
<hr/>		<hr/>		<hr/>	
+6		-9		+9	
0110		0111		1001	
					
		+7		-7	
		hibás!		hibás!	

- Akkor keletkezik 2-es komplement túlcsordulás, ha az összeadandók előjele azonos, de az eredmény előjele ettől eltérő.

4 bites számok esetén:

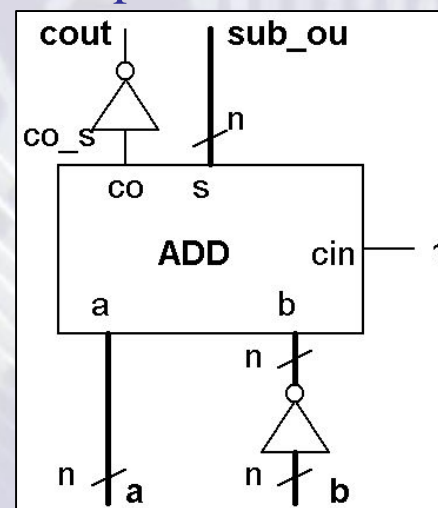
$$OVF = a[3]*b[3]*s[3] + /a[3]*/b[3]*s[3]$$

- **Probléma elkerülése:** használjunk annyi bites számábrázolást, amiben az eredmények elférnek.

Funkcionális egységek: Kivonó

- **Kivonó – kettes komplement hozzáadásával**
 - *Az összeadó jól működik előjel nélküli (pozitív) és kettes komplement számokra is!* (Ezért terjedt el a 2-es komplement számábrázolás.)
 - Az **a-b** művelethez képezzük **-b**-t a 2-es komplement képzéssel:
$$\mathbf{a-b = a+(-b) = a + /b + 1}$$

A (-b) előállításához 2-es komplementet képzünk (b minden bitjét invertáljuk és hozzáadunk 1-et)
 - Az 1- hozzáadásához az összeadó cin bemenetére 1-et kapcsolunk ($S = a+b+cin$)
 - *Az összeadó co kimenete itt fordítva működik, mint a hagyományos kivonónál, ha zavaró, meginvertáljuk.*



Funkcionális egységek: Komparátor kivonóval

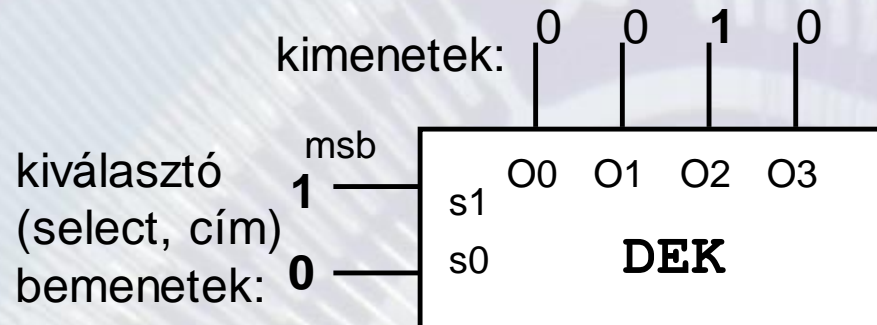
Nagyság komparátor megvalósítása kivonóval

- Előjel nélküli számbábrázolás esetén: átvitel (Co) keletkezik, ha a kisebbítendő (a) kisebb, mint a kivonandó (b)
 - Kivonó – írásbeli kivonásra alapozva:
 - **co = 0:** ha $a \geq b$ (azaz $a - b \geq 0$) vagyis, ha a különbség pozitív
 - **co = 1:** ha $a < b$ (azaz $a - b < 0$) vagyis, ha a különbség negatív volna azonban ilyenkor az eredmény hibás!
 - Kivonó – kettes komplement hozzáadásával:
 - b kettes komplemente: $2^N - b = /b + 1$ (N: bitszám) Pl. 4 bites b = 1 esetén 10000-0001=1111=-1
 - $a - b = a + (-b) = a + (2^N - b) = 2^N + (a - b)$
 - A 2^N hozzáadása invertálja a kimenő átvitel bitet! (Az átvitel bit is az N-edik biten keletkezik.)
 - **co_s = 1, ha $a \geq b$** (azaz $a - b \geq 0$) vagyis, ha a különbség pozitív
 - **co_s = 0, ha $a < b$** (azaz $a - b < 0$) vagyis, ha a különbség negatív egy invertálással megkaphatjuk a megszokottat: $cout = /co_s$
- Kettes komplement számbábrázolásnál az előjeles számoknál a kivonás után (a kivonandó 2-es komplementének hozzáadása) az előjel bitet kell nézni:
 - **N = 0:** ha $a \geq b$
 - **N = 1:** ha $a < b$
 - (Kivonásnál is 2-es komplement túlsordulás lehet, mint az összeadásnál! Nem részletezzük.)
- Szoftverben kivonásra képződik le a komparálás!
- A processzorokban a státusz bitek jelzik az eredménnyel kapcsolatos információkat:
 - Z (zero, eredmény==0) NOR kapcsolat az eredmény bitekre
 - C (carry, átvitel),
 - N (negative, előjel bit 1), Az eredmény legnagyobb bitje, csak 2-es komplement számok esetén használható
 - V (overflow, 2-es komplement túlsordulás, az eredmény nem fér bele a számtartományba)

Funkcionális egységek: Dekóder

- A dekóder funkciója, hogy a bemenetére érkező n bites bináris számnak megfelelő sorszámú kimenetét aktivizálja a maximálisan 2^n darab kimente közül. Másképp fogalmazva *dekódolja* a select (kiválasztó) bementén lévő számot. (A rajzon $s=10_2 = 2$, csak $O2=1$)

s1	s0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Elnevezése: DEK bemenet szám/kimenet szám.

A példa dekódere: DEK2/4

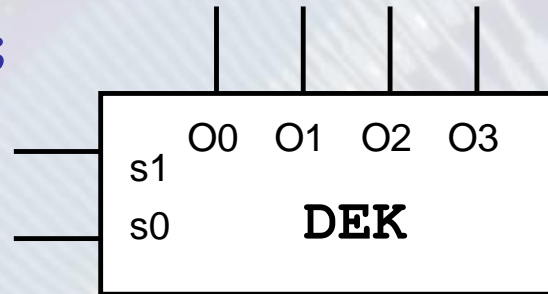
- A dekóder a binárisan értelmezett bemeneti jelekből az n bementű általános *logikai függvény összes mintermjét előállítja, minden kimenetén egyet:*

BME-MIT  $O0 = \neg s1 \neg s0 \quad O1 = \neg s1 s0 \quad O2 = s1 \neg s0 \quad O3 = s1 s0$

• Funkcionális egységek: Dekóder

- Mivel a dekóder O_i kimenete akkor 1, ha az s bementén levő számra $s = i$ ezért az egyes kimeneti függvényeket **konstans komparátoroknak** (konstans összehasonlítóknak) is tekinthetjük.
- DEK2/4 kimeneteinek szorzat alakú leírása **Verilogban**:

```
wire s0,s1,O0,O1,O2,O3;  
assign O0 = ~s1&~s0;  
assign O1 = ~s1&s0;  
assign O2 = s1&~s0;  
assign O3 = s1&s0;
```



Ugyanez az `==` operátorral és vektor változókkal:

```
wire [1:0] s;  
wire [3:0] O;  
assign O[0] = (s == 2'b00);  
assign O[1] = (s == 2'b01);  
assign O[2] = (s == 2'b10);  
assign O[3] = (s == 2'b11);
```

• Funkcionális egységek: Dekóder

- A dekódernek lehet **engedélyező (en, e) bemenete** is. Ha nincs engedélyezve, akkor az összes kimenete 0. Ha engedélyezve van, akkor a megszokott módon működik.

wire e;

wire [1:0] s;

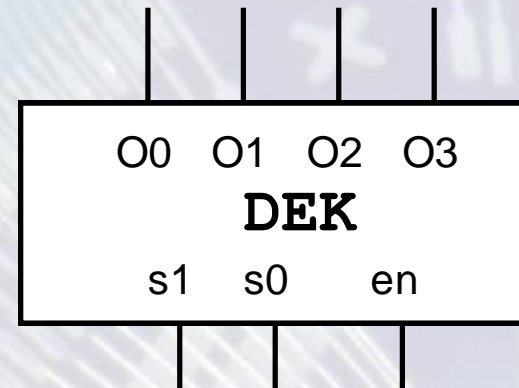
wire [3:0] O;

assign O[0] = en & (s == 2'b00);

assign O[1] = en & (s == 2'b01);

assign O[2] = en & (s == 2'b10);

assign O[3] = en & (s == 2'b11);



Kombinációs funkcionális egységek

Verilog viselkedési leírása

- A Verilog-ban viselkedési leírással is megadhatók a logikák.
- A Verilog nyelvi elemeknél csak a Digitális technika tárgyan használt lehetőségeket részletezzük, már a szintaxis megadásánál is. Pontos (általános) leírás a Verilog bevezetőben található. Azonban csak az előadás és gyakorlat-labor anyagban található részt kérjük vissza.
- A viselkedési leírás szintaxisa: **always@**(érzékenységi lista)
feltételes vagy
feltétel nélküli értékadások
- Az **always** blokkban csak **reg** típusú változónak szabad értéket adni
- Az **érzékenységi listában** fel kell sorolni a **bemeneti jeleket**.
- Az értékadások akkor értékelődnek ki, ha az érzékenységi lista bármely jele megváltozik. (Olyan logikát generál a CAD rendszer, amely mindig az aktuális kiértékelődésnek megfelelően viselkedik.)
- **Kombinációs hálózatok** viselkedési leírásában az **érzékenységi lista helyett elég egy *-ot írni:**
always@(*)
- Ha az **always** blokk belső leírása nem egyértelmű kezdettel és véggel rendelkező nyelvi konstrukció, akkor **begin** és **end** közé kell tenni. (Egyébként sem árt.)

Kombinációs funkcionális egységek

Verilog viselkedési leírása

Az **always** blokkban használható **feltételes szerkezet**
if else szerkezet

```
if(kifejezés)
    értékadás1;
else
    értékadás2;
```

Ha a kifejezés nem 0,
akkor az **if else** közötti
rész hajtódik végre,
egyébként az **else** utáni.
Az **else** ág **elhagyható**.

Több utasítás esetén az utasításokat
begin end közé kell tenni:

```
if(kifejezés)
    begin
        értékadás1;
        értékadás2;
    end
else
    begin
        értékadás3;
        értékadás4;
    end
```

Kombinációs funkcionális egységek

Verilog viselkedési leírása

Az always blokkban használható többutas elágazás
case szerkezet

case (kifejezés)

konstans1: értékadás(ok)1;

konstans2: értékadás(ok)2;

...

default: default_értékadás(ok);

endcase

Minden konstansnak különbözőnek kell lenni. Ha a kifejezés megegyezik valamely konstanssal, akkor az ahhoz a részhez tartozó értékadások hajtódnak végre. Ha több értékadás tartozik egy konstanshoz, azokat **begin end** közé kell tenni.

Ha a case kifejezése **egyik konstanssal sem egyezik**, akkor a

default ág él.

Funkcionális egységek: Dekóder

Engedélyezhető DEK2/4 viselkedési leírás always blokkal:

reg [3:0] out; // aminek értéket adunk az always blokkban az **csak reg típusú lehet!**

wire [1:0] s;

always @ (*)

begin

if (en)

// ha en =1

case (s)

2'b01: out = 4'b0010;

// ha sel = 01, akkor az 1. kimenet lesz 1

2'b10: out = 4'b0100;

2'b11: out = 4'b1000;

// mindem lehetőséget felsoroltunk,

default: out = 4'b0001;

// ha sel = 00, akkor a 0. kimenet lesz 1

endcase

// a default helyett 2'b00-t is írhatnánk

else

// mert úgy is minden lehetőséget lefedünk

out = 4'b0000;

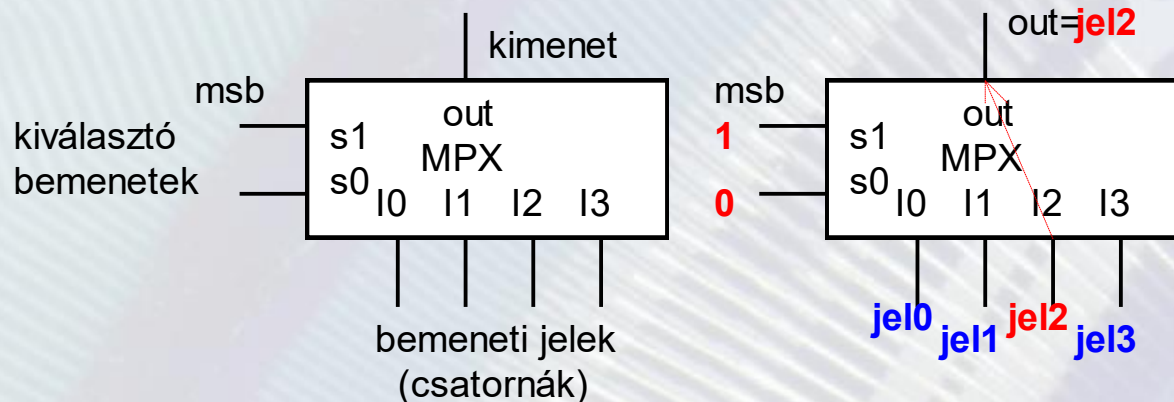
// ha en = 0, minden kimenete 0.

end

BME-MIT

Funkcionális egységek: Multiplexer

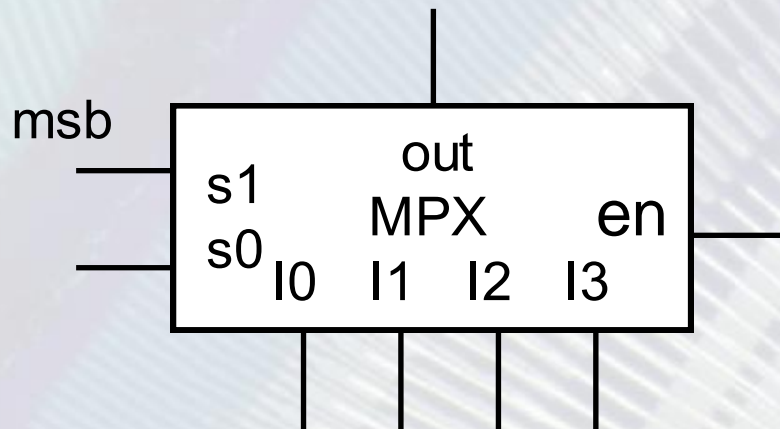
- A multiplexer feladata a több bemeneti adata (csatornája) közül a kiválasztó (select, s) bementére adott bináris számnak megfelelő sorszámúnak a kimenetre (out, o) kapcsolása
- Ezt *adatút választásnak* is szokás nevezni
(Pl. a rajzon $s=10_2 = 2$, csak $out = I_2$, jel2-öt választotta ki.)



- Elnevezése MPX(vagy MUX) bemeneti csatorna szám/1
A példában MPX4/1
- Jellemző méretek: 2/1, 4/1, 8/1, 16/1

Funkcionális egységek: Multiplexer

- A multiplexer lehet engedélyezhető (en, e).
- Ha nincs engedélyezve, a kimenete 0 értékű. Ha engedélyezve van, a megszokott módon viselkedik.



Funkcionális egységek: Multiplexer

Engedélyezhető MUX4/1 viselkedési leírás always blokkal:

reg out; // aminek értéket adunk az always blokkban az **csak reg típusú lehet!**

wire [3:0] I;

wire [1:0] s;

always @ (*)

begin

if (en) // ha en =1

case (s)

2'b01: out = I[1]; // ha sel = 01, akkor az I1 bemenetet választja ki

2'b10: out = I[2]; //

2'b11: out = I[3]; // mindem lehetőséget felsoroltunk,

default: out = I[0]; // ha sel = 00, akkor az I0 bemenetet választja ki

endcase // minden lehetséges s értéket lefedtünk

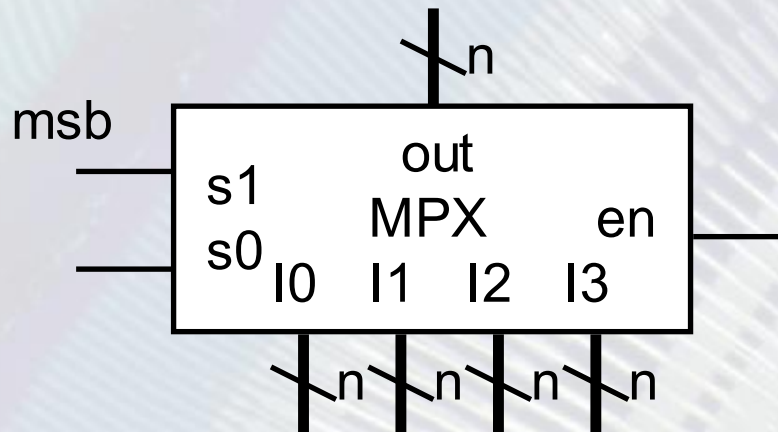
else

out = 1'b0; // ha en = 0, kimenete 0.

end

Funkcionális egységek: Multiplexer

- A multiplexer adat bemenetei és kimenete lehet több bit széles.
- A több bitből álló jelcsoport neve **busz**
- Az ilyen multiplexer neve **busz multiplexer**



Funkcionális egységek: Multiplexer

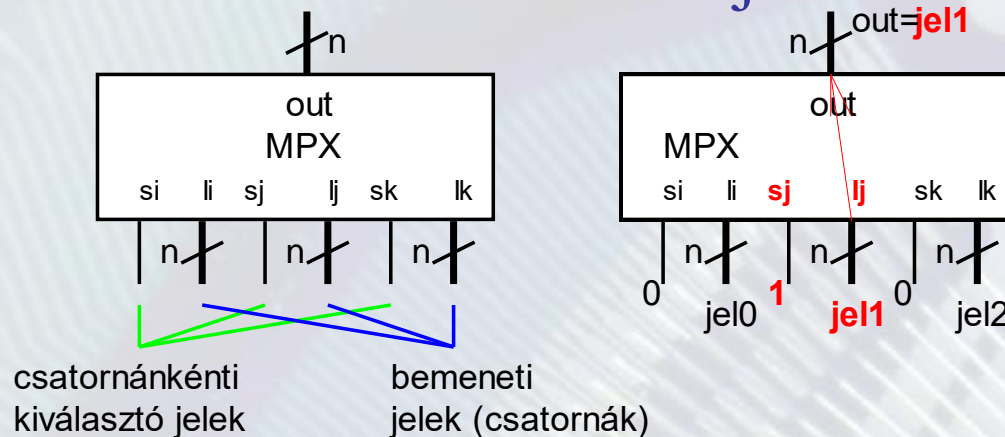
Engedélyezhető MUX8_4/1 viselkedési leírás always blokkal:

```
reg [7:0]out; // aminek értéket adunk az always blokkban az csak reg típusú lehet!
wire [7:0] I0,I1,I2,I3; // az out és az in azonos méretű (most 8 bites) bitvektorok
wire [1:0] s;
wire en;
always @ (*)
begin
    if (en) // ha en =1
        case (s)
            2'b01: out <= I1; // ha sel = 01, akkor az I1 bemenetet választja ki
            2'b10: out <= I2; //
            2'b11: out <= I3;
            default: out <= I0 // mindem lehetőséget lefedtünk
        endcase
    else
        out = 8'b0; // ha en = 0, kimenete 0.
end
```


Funkcionális egységek: Multiplexer

Multiplexer csatornához rendelt kiválasztó jelekkel

- Időnként hasznos, ha a multiplexer csatorna kiválasztása a csatornához tartozó kiválasztó jellel történik.



- Az i -edik bemenet (X_i , i -edik csatorna) E_{ni} (S_{eli}) jele engedélyezi, hogy X_i bemenet megjelenjen a kimeneten.
- Az s jelek közül csak 1 db lehet aktív!** Ezt az s jeleket előállító logikával biztosítjuk. A dekóder pontosan ilyen tulajdonságú kimenetekkel rendelkezik.

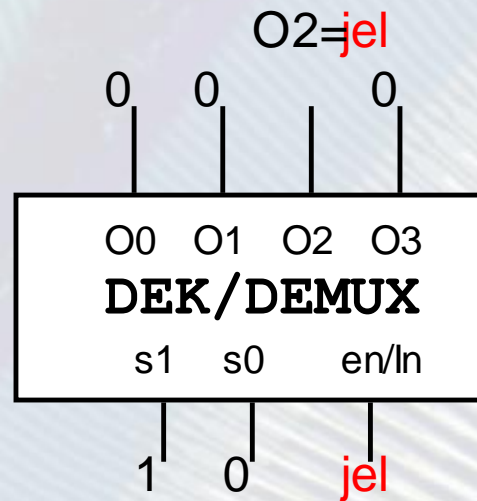
Funkcionális egységek: Multiplexer

Csatornánkénti kiválasztású buszmultiplexer viselkedési leírása always blokkal:

```
reg [7:0]out; // aminek értéket adunk az always blokkban az csak reg típusú lehet!
wire [3:0] s;
wire [7:0] I0,I1,I2,I3; // az out és az in azonos méretű bitvektorok
wire en;
always @ (*)
begin
    if (en)                // ha en =1
        case (s)
            4'b0001: out = I0;        // ha sel = 0001, akkor az I0 bemenetet választja ki
            4'b0010: out = I1;        // ha sel = 0010, akkor az I1 bemenetet választja ki
            4'b0100: out = I2;        //
            4'b1000: out = I3;        //
            default: out = 8'h00;      // Ha nem 1 a 4-ből kódú az s akkor a kimenet 0!
        endcase                // most 12 ilyen eset van!
    else
        out = 8'b0;            // ha en = 0, kimenete 0.
end
```

Funkcionális egységek: DEMUX

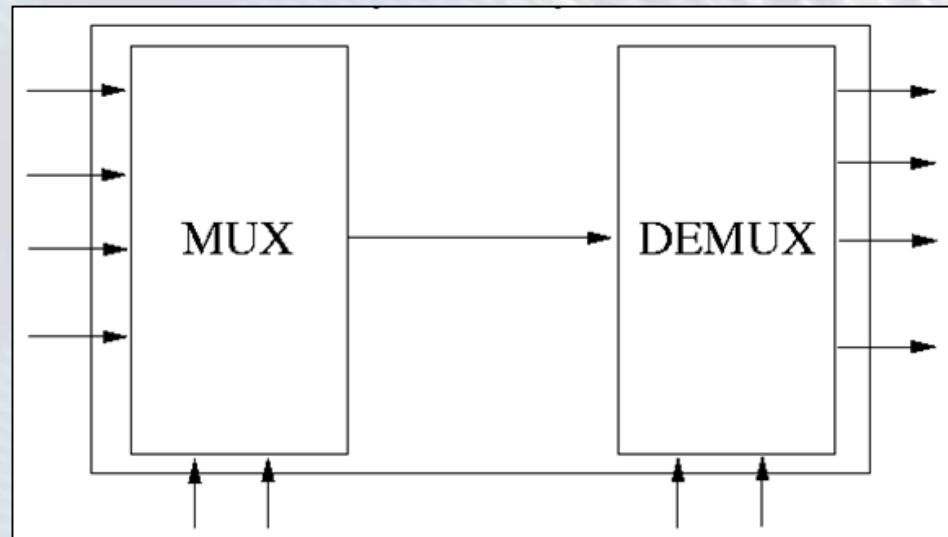
- A **DEMUX** demultiplexer a multiplexer funkció inverze
- Az egyetlen adat bemenet értéke a kiválasztó jelekkel megadott sorszámú kimeneten jelenik meg, a többi kimenet 0. Lényegében egy *engedélyezhető dekóderrel azonos felépítésű, csak az en bemenet szerepe itt adat bemenet* (In).



- Szokásos méretei 1/2, 1/4, 1/8

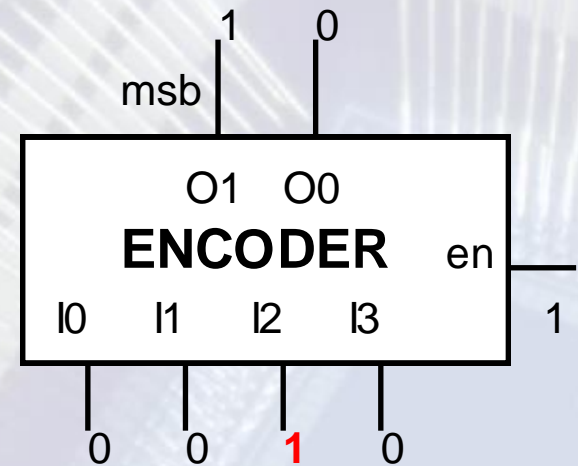
Funkcionális egységek: DEMUX

- **A MUX-DEMUX egységek használata**
 - Több adat átvitele egyetlen vonalon
 - Forrásoldalon MUX, vételi oldalon DEMUX
 - Az átviteli vonal csak egyetlen vezeték, de sokkal nagyobb sávszélességű
 - Időosztásos adatátvitel, a két oldalon azonosan ütemezett kiválasztó jelekkel



Funkcionális egységek: Enkóder

- Az ENC enkóder N bites **bemenetére 1-az-N-ből kódolású kód kapcsolható** (N bitből 1 db 1-es). A kimenet **bináris számként kiadja, azon bemenet sorszámát, ahol az 1-es van.**
- **Nem 1-az-N-ből kódolású szám esetén hibás kimenetet ad.** Ezért **engedélyező (en) bemenete** is van.
- Akkor kell engedélyezni, amikor érvényes értéket tesznek a bemenetére.



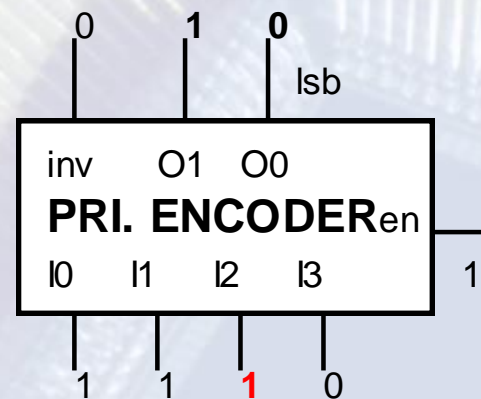
Funkcionális egységek: Enkóder

Enkóder viselkedési leírása always blokkal:

```
reg [1:0]out; // aminek értéket adunk az always blokkban az csak reg típusú lehet!
wire [3:0] I;
wire en;
always @ (*)
begin
    if (en) // ha en =1
        case (I)
            4'b0001: out = 2'b00; // a 0. bemenet 1, ezért out =0
            4'b0010: out = 2'b01; // az 1. bemenet 1, ezért out =1
            4'b0100: out = 2'b10; // a 2. bemenet 1, ezért out =2
            4'b1000: out = 2'b11; // a 3. bemenet 1, ezért out =3
            default: out = 2'b00; // Ha nem 1 a 4-ből kódú az s akkor a kimenet 0!
        endcase // most 12 ilyen eset van!
    else
        out = 2'b0; // ha en = 0, kimenete 0.
end
```


Funkcionális egységek: Prioritás enkóder

- Az enkódernél jobban használható a **prioritás enkóder**. Az N bites **bemenetére kapcsolt kód alapján a kimenetén bináris számként kiadja, azon *legnagyobb sorszámú* bemenetének sorszámát, ahol 1-es van.** (Függetlenül attól, hogy a többi bemeneten milyen érték van.)
- Ha minden bemeneten 0 van, hibás a kimenete. Ezért **invalid (inv, érvénytelen) jelzés kimenete és engedélyező (en) bemenete is van.**
- Az **inv** kimenete akkor 1, ha nincs engedélyezve vagy csupa 0 a bemenete.

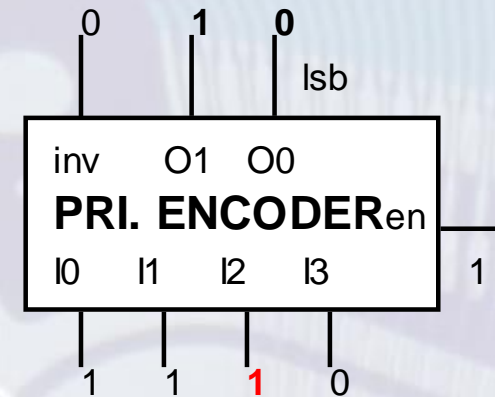


Funkcionális egységek:

Prioritás enkóder

- A prioriás enkóder igazságtáblája.

en	I3	I2	I1	I0	O1	O0	inv
0	x	x	x	x	0	0	1
1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	x	0	1	0
1	0	1	x	x	1	0	0
1	1	x	x	x	1	1	0



- Az x-el a don't care (közömbös, tetszőleges) értéket jelöljük. Ilyen a Verilogban is van.
Pl. Az utolsó sorban az I2I1I0 értéke közömbös. Ez $2^3 = 8$ igazságtábla sort helyettesít.
- A Verilog **casex** szerkezetében a case konstansai x értékeket is tartalmazhatnak.

Funkcionális egységek:

Prioritás enkóder

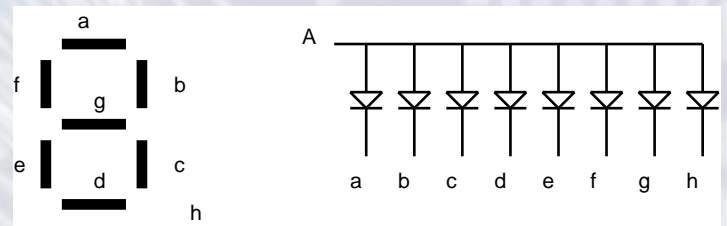
Prioritás enkóder viselkedési leírása always blokkal:

```
reg [1:0] out; // aminek értéket adunk az always blokkban az csak reg típusú lehet!
wire [3:0] I; // bemenetek
wire en, inv; // engedélyezés és invalid kimenet jelzés
always @ (*)
begin
    if (en) // ha en =1
        casex (I)
            4'b0001: out = 2'b00; // a 0. bemenet 1, ezért out =0
            4'b001x: out = 2'b01; // az 1. bemenet 1, előtte 0-ák, out =1
            4'b01xx: out = 2'b10; // a 2. bemenet 1, előtte 0-ák, out =2
            4'b1xxx: out = 2'b11; // a 3. (legnagyobb) bemenet 1, out =3
            default: out = 2'b00; // Ha I=0, akkor a kimenet 0!
        endcase
    else
        out = 2'b0; // ha en = 0, kimenete 0.
end
assign inv = ~en | (I == 4'b0000); // a kimenet nem érvényes, ha en=0 vagy a
// bemenet csupa 0.
```


Funkcionális egységek:

Hex/7szegmenses dekóder

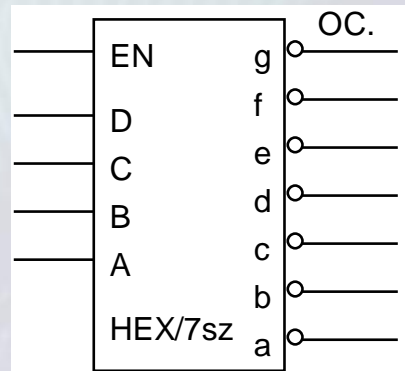
- A 7 szegmenese kijelzőn számokat és korlátozottan betűket lehet megjeleníteni.
- A LED kijelző szegmenseinek elhelyezkedése és bekötésük. A ú.n. közös anódos kijelző LED-jeinek pozitív lába közösítve van, erre kell a tápot adni. Azon LED-ek fog-
nak világítani, amelyek katódja ellenálláson keresztül a 0V-ra kapcsolódik.
- Például az 1 kijelzéséhez a b és c LED-eket kell kigyújtani.
- A HEX/7 szegmenses dekóder olyan kombinációs hálózat, amely a bementére kapcsolt 4 bites bináris számhoz a kimenetén az adott hexadecimális számjegy megjelenítéséhez szükséges 7 bites kódot adja.



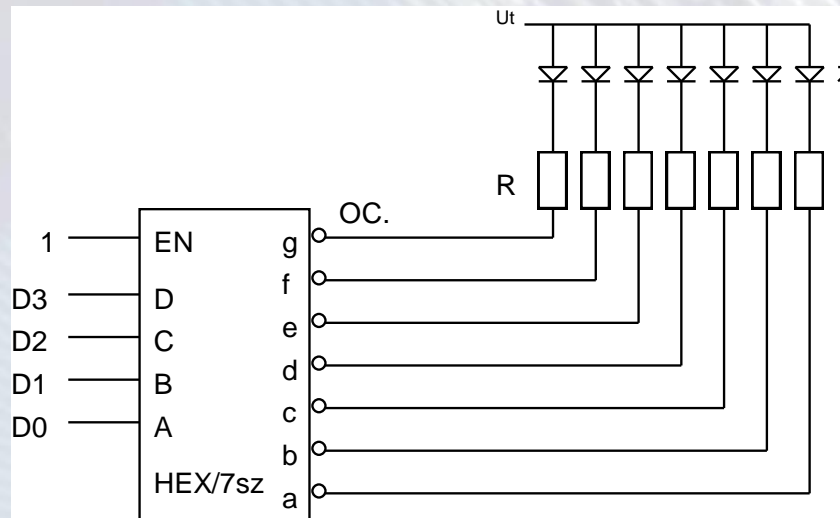
Funkcionális egységek:

Hex/7szegmenses dekóder

- A rajzjele alacsony aktív kimenet esetén:



- Példa a bekötésére:



Funkcionális egységek:

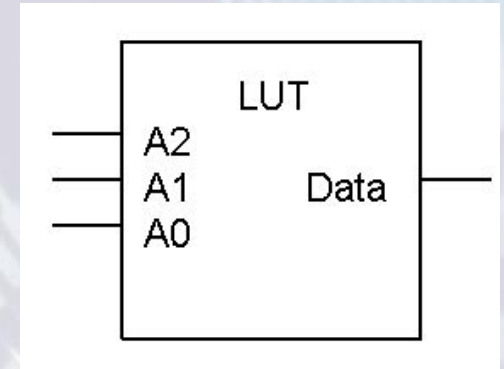
Hex/7szegmenses dekóder

Verilog kód:
Magas aktív kimenet esetén. (Itt azon szegmensek értéke 1, amelyeknek világítani kell. A dec_out kimenet meginvertálásával megkapjuk az alacsony aktív kimenetű verziót.

```
module dec_7seg(  
    input  wire [3:0] dec_in,  
    output reg  [6:0] dec_out  
);  
  
//A hétszegmenses dekóder igazságtábláját könnyen leírhatjuk Verilog nyelven  
//always blokkban case utasítást használva. A kimenet esetén at '1' érték  
//jelentse az aktív (bekapcsolt) szegmenst. A dekóder kimenetének értelmezése:  
//dec_out[0]: A, dec_out[1]: B, dec_out[2]: C, ..., dec_out[6]: F  
always @(*)  
begin  
    case (dec_in)  
        4'h1 : dec_out <= 7'b0000110;  
        4'h2 : dec_out <= 7'b1011011;  
        4'h3 : dec_out <= 7'b1001111;  
        4'h4 : dec_out <= 7'b1100110;  
        4'h5 : dec_out <= 7'b1101101;  
        4'h6 : dec_out <= 7'b1111101;  
        4'h7 : dec_out <= 7'b0000111;  
        4'h8 : dec_out <= 7'b1111111;  
        4'h9 : dec_out <= 7'b1101111;  
        4'ha : dec_out <= 7'b1110111;  
        4'hb : dec_out <= 7'b1111100;  
        4'hc : dec_out <= 7'b0111001;  
        4'hd : dec_out <= 7'b1011110;  
        4'he : dec_out <= 7'b1111001;  
        4'hf : dec_out <= 7'b1110001;  
        default: dec_out <= 7'b0111111;  
    endcase  
end  
endmodule
```


A memória táblázat, mint univerzális logikai elem

- LUT memória táblázat (FPGA logikai elem)
- A címbitekkel ($A[n-1:0]$) megcímzett rekeszének tartalmát adja ki az adat (Data) kimenetenén.
- Pl: 8 rekesssel rendelkező (3 címbit) LUT
- A LUT-ot úgy használhatjuk univerzális kombinációs hálózatként, hogy a változókat a cím bemenetekre kötjük és tartalomként az igazságtáblázatot töltjük bele.
Figyelni kell a bemeneti átlózók sordndjére!



DATA= igazságtábla

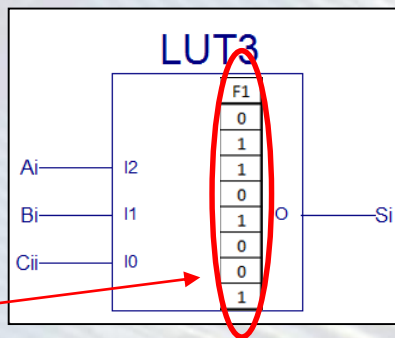
LUT		
		0
		1
a	A2	2
b	A1	3
c	A0	4
		5
		6
		7

0
1
1
0
1
0
0
1

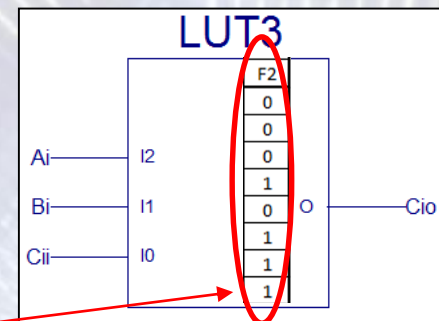
A memória táblázat, mint univerzális logikai elem

- **Függvény megadása:** Az igazságtábla kimeneti jel oszlopának értékeit konstans jelként beírjuk a memória sorindex szerinti címekre
- **A bemeneti változókat az azonos helyiértékű címbitekre kell kötni (A,B,C-> I2,I1,I0)**
 - A korábbi teljes összeadó függvényei
F1: S és F2: Co

BEMENETEK				KIM
INDX	A	B	C	F1
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1



BEMENETEK				KIM
INDX	A	B	C	F2
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Digitális technika 3. EA vége