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

Digitális technika (VIMIAA03)
4. gyakorlat és laboratórium

Raikovich Tamás
BME MIT

Kombinációs funkcionális elemek

- **1. feladat: multiplexer**
  - 2/1 és 4/1 multiplexerek megvalósítása
- **2. feladat: engedélyezhető dekóder / demultiplexer**
  - 3/8 engedélyezhető dekóder megvalósítása
- **3. feladat: hétszegmenses dekóder**
- **A megoldásokat ellenőrizzük szimulációval is**
- **A feladatokhoz szükséges új Verilog ismeretek**
  - Feltételes operátor
  - ***reg*** típus
  - ***always*** blokk (kombinációs hálózatok leírásához)
  - ***if*** és ***case*** utasítások

Verilog HDL ismeretek

Feltételes operátor, reg típus

- **Feltételes operátor**

```
(feltétel) ? [kifejezés1] : [kifejezés2]
```

  - Ha a megadott feltétel igaz (értéke nem 0), akkor az operátor eredménye a ***kifejezés1*** értéke lesz
  - Ha a megadott feltétel hamis (értéke 0), akkor az operátor eredménye a ***kifejezés2*** értéke lesz

- ***reg* típus**
  - Leírástól függően nem csak vezetéket (kombinációs hálózat), hanem tároló funkciót (sorrendi hálózat) is megvalósíthat
  - Portok esetén csak a kimeneti irányú lehet **reg** típusú
  - Csak speciális blokkokban (***initial***, ***always***) adható ilyen típusú jeleknek érték
  - Hardver leírása esetén az értékadáshoz mindig használjuk a **<=** (nem blokkoló) értékadás operátort
    - Ezek egymással párhuzamosan értékelődnek ki (ez áll közelebb a hardveres szemlélethez és kevesebb értelmezési hibát okozhat)

Verilog HDL ismeretek

always blokk

Az ***always*** blokk szintaxisa a következő

```
always [@(érzékenységi lista)]
begin
   utasítás(ok);
end
```

- Magasszintű, viselkedési leírást biztosít
- Az **always** blokkban lévő utasítások akkor értékelődnek ki, ha bekövetkezik az érzékenységi listában megadott esemény
  - Ha nincs megadva érzékenységi lista, akkor az utasítások folyamatosan végrehajtódnak (***csak tesztkörnyezetben!***)
- Csak ***reg*** típusú jelnek adható benne érték
  - Hardver leírása esetén egy jel csak egy blokkban kaphat értéket
- A leírási módtól függően képződhet belőle kombinációs és sorrendi hálózat is
- Használható benne az ***if*** és a ***case*** utasítás

Verilog HDL ismeretek

if utasítás

Az *if* utasítás szintaxisa a következő

```
if (kifejezés) utasítás1; [else utasítás2;]
```

- **Először a megadott *kifejezés* kerül kiértékelésre**
  - Ha értéke nem 0: az ***utasítás1*** hajtódik végre
  - Ha értéke 0: az ***utasítás2*** hajtódik végre
- **Az *else* ág opcionális, elhagyható**
- **Több utasítás esetén azokat a *begin* és az *end* kulcsszavak közé kell csoportosítani**
- **Az egymásba ágyazott *if* utasítások hierarchikus, sorrendi kiértékelést jelentenek (*prioritás!*)**
  - Ez a tipikus megvalósítása a funkcionális egységek vezérlő jeleinek
- **Csak speciális blokkokban (pl. *initial*, *always*) szerepelhet**

Verilog HDL ismeretek

case utasítás

A *case* utasítás szintaxisa a következő

```
case (kifejezés)
   alternatíva1: utasítás1;
   alternatíva2: utasítás2;
        ⋮                ⋮
   default      : default_utasítás;
endcase
```

- A ***kifejezés*** értéke összehasonlításra kerül az ***alternatívákkal*** a megadásuk sorrendjében (***a sorrendjük prioritást jelenthet!***)
- A legelső, a kifejezés értékével egyező alternatívához tartozó utasítás kerül végrehajtásra
- Ha nincs egyező alternatíva, akkor a ***default*** kulcsszó után lévő ***default_utasítás*** kerül végrehajtásra (opcionális)
- Egy alternatívához tartozó több utasítás esetén: ***begin…end***
- A ***casex*** utasítás esetén az alternatívák tartalmazhatnak ***x*** (don't care) értékű bitet is, ez néha egyszerűbb leírást tesz lehetővé
- Csak speciális blokkokban (pl. ***initial***, ***always***) szerepelhet

Verilog HDL ismeretek

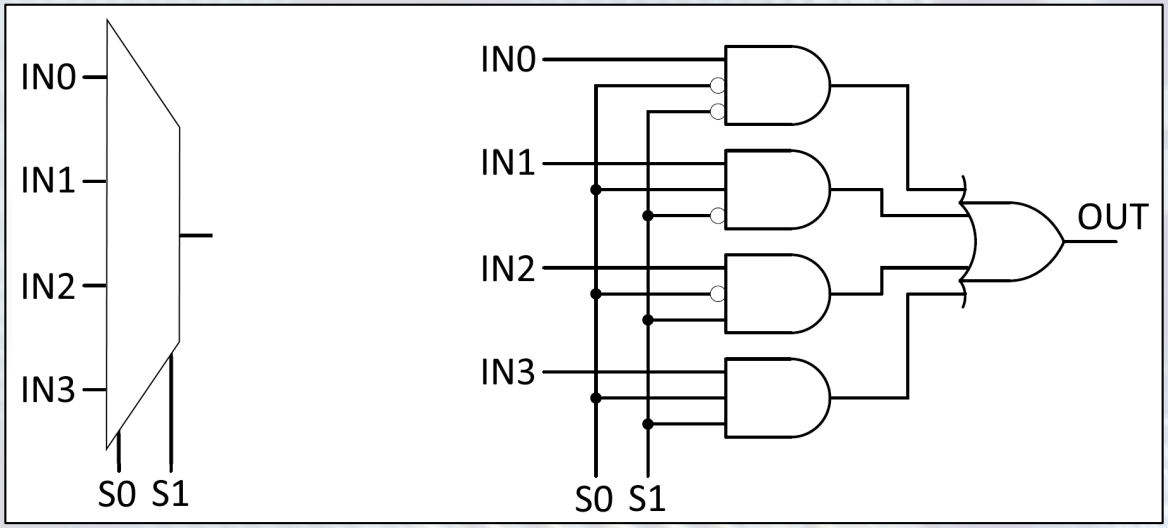
Kombinációs hálózatok leírása

- **A kombinációs hálózatok *kimeneteinek értékét minden időpillanatban meghatározza a bemeneteinek értéke, nem rendelkeznek memória (tároló) tulajdonsággal***
- **Kombinációs hálózatok leírása *wire* típusú jellel**
  - Ebből csak kombinációs hálózat képződhet
- **Kombinációs hálózat leírása *reg* típusú jellel**
  - Az ***always*** blokk érzékenységi listájában fel kell sorolni az összes bemenetet vagy használjuk a * rövidítést (az összes bemeneti jelet jelöli)
  - A vizsgált feltételek minden kombinációja esetén kell, hogy legyen kimeneti értékadás
    - ***if*** utasítás: kötelező az ***else*** ág megadása
    - ***case*** utasítás: minden alternatívát fel kell sorolni vagy használni kell a ***default*** alternatívát

1. feladat: multiplexerek

Hogy írjuk le Verilogban a működését?

- **A logikai függvény közvetlen megadásával?**
  - Nem jól olvasható és értelmezhető leírási mód!

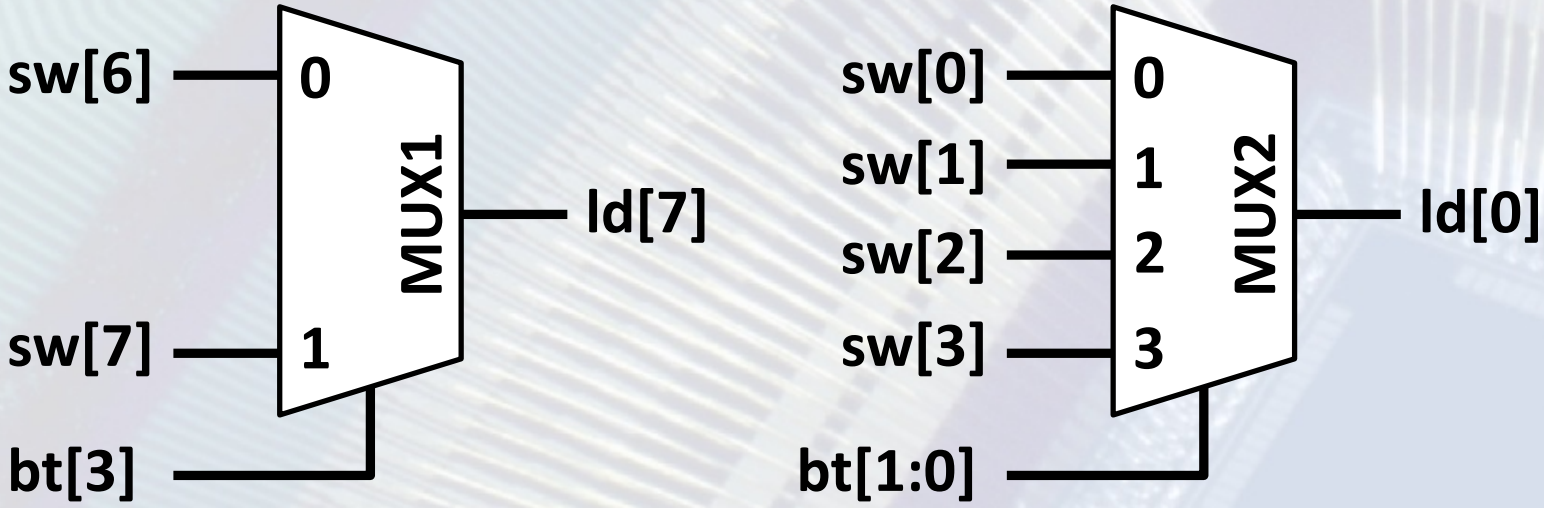


- **Milyen nyelvi eleme(ke)t használjunk a 2/1 MUX-hoz?**
- **Milyen nyelvi eleme(ke)t használjunk a 4/1 MUX-hoz?**

1. feladat: multiplexerek

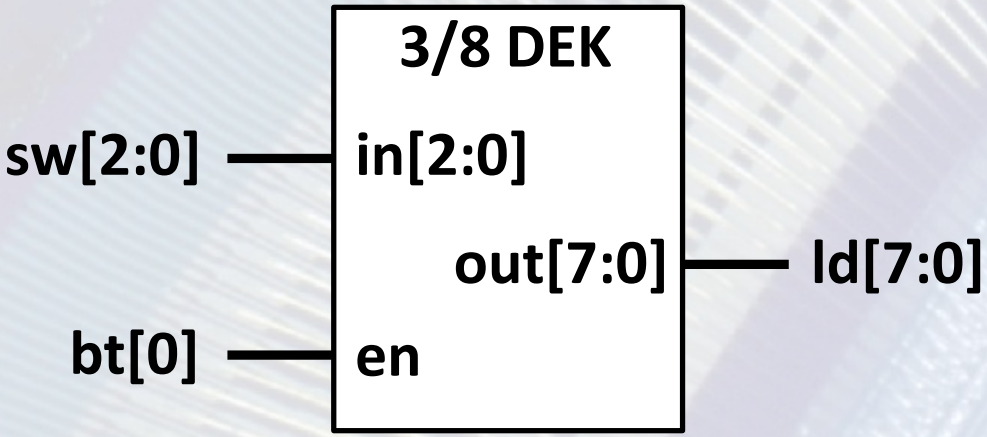
A két multiplexert egy modulban valósítsuk meg

- A 2/1 multiplexer (MUX1) leírásához használjuk a feltételes operátort
- A 4/1 multiplexer (MUX2) leírásához használjuk az always blokkot és a case utasítást



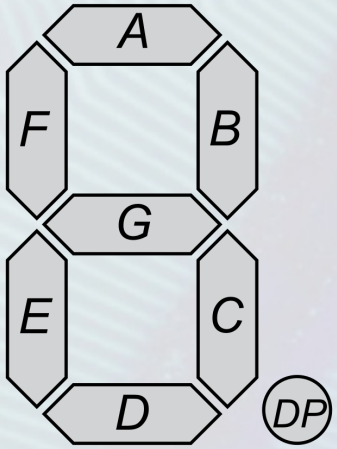
2. feladat: 3/8 engedélyezhető dekóder

- **A bináris dekóder kimenete 1-az-N-ből kódolású, az aktív kimenet sorszámát a bemenetére kell adni**
  - Tiltott (azaz en=0): minden kimenet inaktív (azaz 0)
  - Engedélyezett (azaz en=1): normál működés
- **Mi a fő különbség a multiplexer leírásához képest?**
- **Milyen nyelvi elemmel valósítható meg az engedélyezés?**
- **Hogy használható ez a funkcionális elem demultiplexerként?**



3. feladat: hétszegmenses dekóder

- **A dekóder feladata általánosságban véve a kódátalakítás**
- **A hétszegmenses dekóder a bemeneti 4 bites bináris értéket alakítja át a kijelzőn megjeleníthető képpé**
  - 0-9 és A-F, azaz a hexadecimális számjegyek képévé
  - Írjuk fel az igazságtábláját! Hogy lehet ezt Verilog nyelven áttekinthetően megadni?
- **Valósítsuk meg a hétszegmenses dekódert a projekt vázban**



Segments (✓ = ON)							Display	Segments (✓ = ON)							Display
a	b	c	d	e	f	g		a	b	c	d	e	f	g	
✓	✓	✓	✓	✓	✓		0	✓	✓	✓	✓	✓	✓	✓	8
	✓	✓					1	✓	✓	✓	✓		✓	✓	9
✓	✓		✓	✓		✓	2	✓	✓	✓		✓	✓	✓	A
✓	✓	✓	✓			✓	3			✓	✓	✓	✓	✓	b
	✓	✓			✓	✓	4	✓			✓	✓	✓		C
✓		✓	✓		✓	✓	5		✓	✓	✓	✓		✓	d
✓		✓	✓	✓	✓	✓	6	✓			✓	✓	✓	✓	E
✓	✓	✓					7	✓				✓	✓	✓	F

BCD Signal				Display	BCD Signal				Display
D	C	B	A		D	C	B	A	
0	0	0	0	0	0	1	0	1	5
0	0	0	1	1	0	1	1	0	6
0	0	1	0	2	0	1	1	1	7
0	0	1	1	3	1	0	0	0	8
0	1	0	0	4	1	0	0	1	9

0 = logic low 1 = logic high