Digitális rendszerek

I. rész

Dr. Turóczi Antal
turoczi.antal@nik.uni-obuda.hu

Bevezető

- **A tárgy célja**
  - Digitális rendszertervezés alapjai
    - Programozható logikai áramkörök
    - Hardverleíró nyelvek (VHDL)
  - A digitális technika és az elektronika kapcsolata
    - Digitális elektronika alapjai
    - A digitális rendszerek, logikai hálózatok alapelemeinek megvalósítása
  - Digitális rendszerek működésének megértéséhez
  - Számítástechnikai alapismeretek
  - Mérnöki szemlélet kialakításához

Bevezető

- **Tananyag**
  - Digitális rendszertervezés
    - Programozható logikai áramkörök
    - Hardver leíró nyelvek, VHDL alapismeretek
    - Digitális rendszertervezés VHDL-el
  - Digitális Elektronika
  - Történelmi áttekintés
  - A bipoláris tranzisztor mint kapcsoló
    - TTL áramkörök statikus és dinamikus tulajdonságai
  - A térvezérlésű tranzisztor (FET) mint kapcsoló
    - CMOS áramkörök statikus és dinamikus tulajdonságai
  - CMOS alapáramkörök
    - Kombinációs és sorrendi hálózatok
    - Memóriák, SRAM, DRAM, EPROM, EEPROM

Bevezető

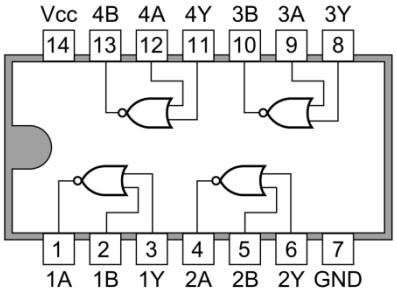
- **Követelmények**
  - Heti óraszámok: 3 óra előadás
  - Számonkérés módja: félév közben: 2 zh,
    - 7. hét - 1.zh (on-line zh, tesztkérdések) max. 100%
    - 13. hét - 2.zh (on-line zh, tesztkérdések) max. 80%
      további 20% a beadott házi feladat értékeléséből
      Házi feladat leadás (Vasárnap 24:00-ig)
    - 14. hét - pót zh-k (on-line zh, tesztkérdések)
  - Aláírás
    - Az aláírás megszerzéséhez mindkét ZH-n a maximális pontszám legalább 51%-át el kell érni.
    - A vizsgaidőszakban egy alkalommal lehetőség van az aláírás pótlására.
  - Vizsga
    - On-line tesztkérdések és papíros feladatmegoldás

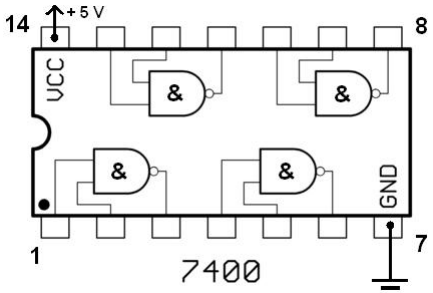
Bevezető

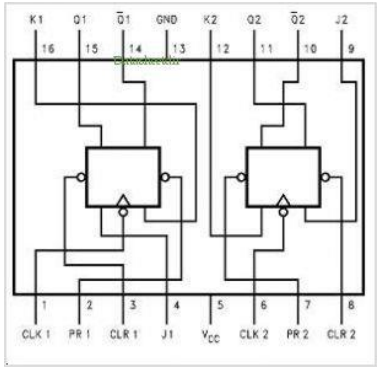
- **Ajánlott irodalom**
  - Zsom Gyula: Digitális technika I-II, Műszaki Könyvkiadó, Budapest, 2000, (KVK 49-273/I).
  - Arató Péter: Logikai rendszerek. Tankönyvkiadó, Bp. 1985.
  - U. Tietze, Ch. Schenk: Analóg és digitális áramkörök, Műszaki Könyvkiadó, Budapest, 1993
  - További segédletek
    - http://nik.uni-obuda.hu/vill/Digit_rendsz_I/

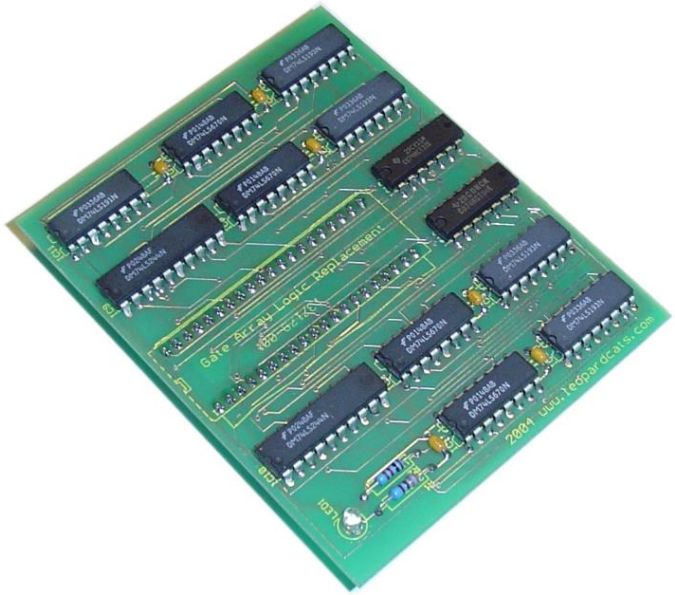
Digitális rendszer-tervezés

- Digitális áramkörök tervezése
  - Egyszerűbb logikai áramkörök
    - Hagyományos módszerekkel
    - Diszkrét kapuáramkörök, funkcionális elemek felhasználásával
    - Olcsó, gyors
    - Nagy fizikai méret mellett csak egyszerű funkciók megvalósítása lehetséges







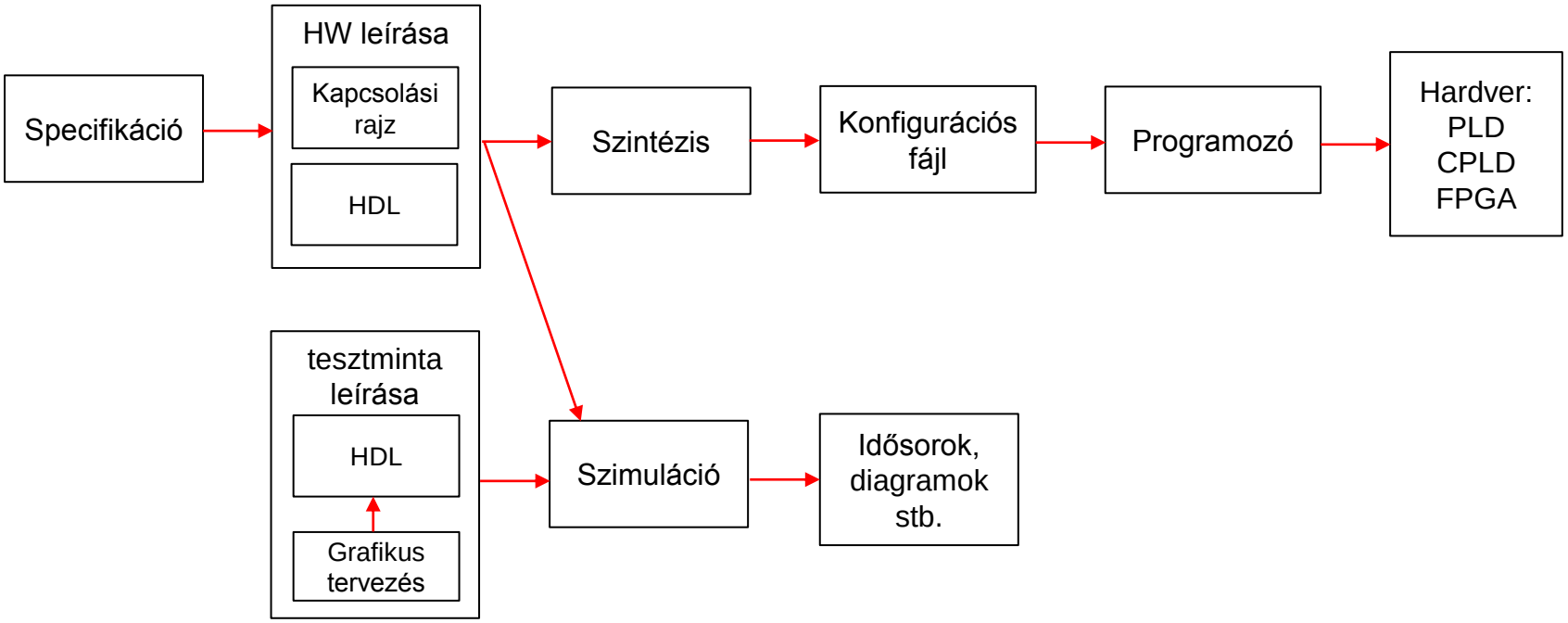


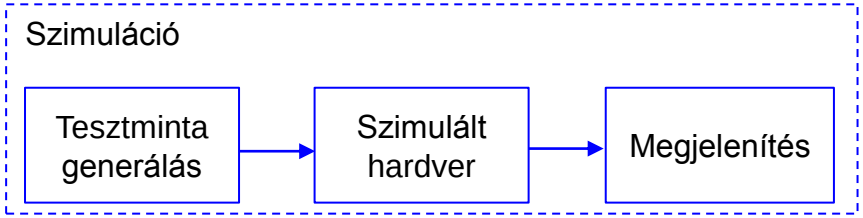
Digitális rendszer-tervezés

- Digitális áramkörök tervezése
  - Bonyolult logikai áramkörök
    - Felhasználói IC
      - Általános felhasználású digitális áramkörök
      - Mikroprocesszorok
    - Alkalmazás specifikus áramkör (application specific integrated circuit, ASIC)
      - SoC (system-on-chip)
    - Programozható logikai áramkör (PLD, CPLD, FPGA)
  - A hagyományos tervezési módszerek nem adnak hatékony megoldást
    - Számítógéppel segített tervezés (Computer Aided Design, CAD)
    - Komplex tervező rendszerek
      - Tervezés, ellenőrzés, szintetizálás és automatikus hardvergenerálás (berendezés programozás) egyben
      - Hardverleíró nyelvek (hardware description language, HDL) és az ezeken alapuló tervezési metódusok
        - » VHDL, Verilog

Digitális rendszer-tervezés

- Digitális áramkörök tervezése
  - A tervezés folyamata (programozható logikai áramkörök)



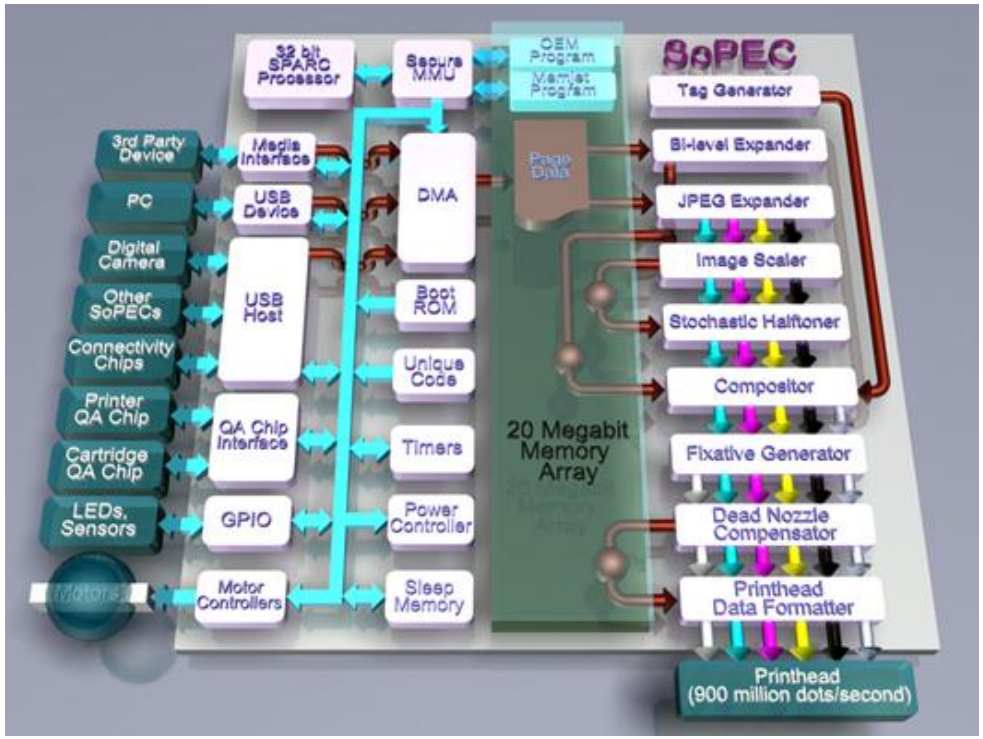


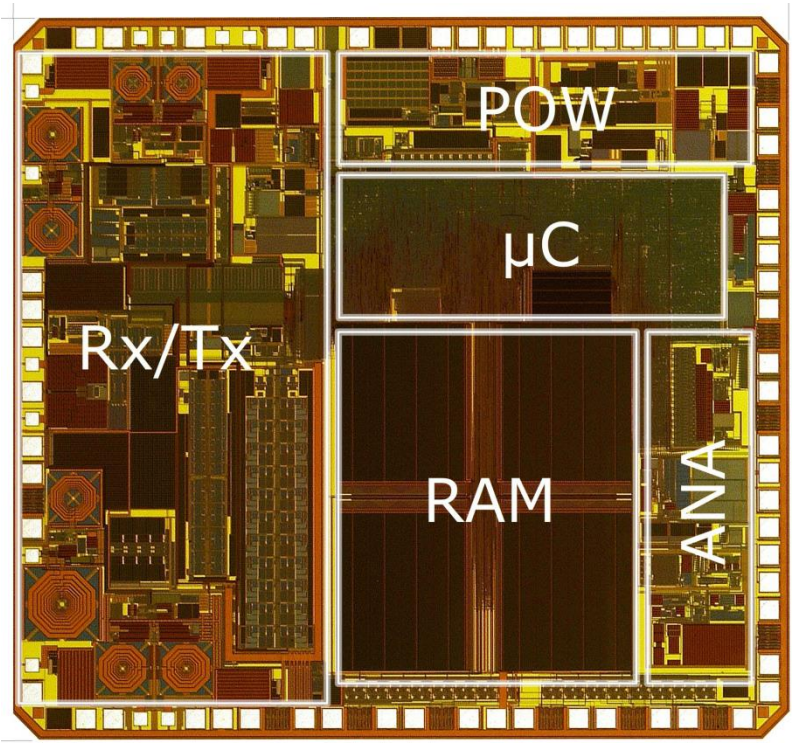
Digitális rendszer-tervezés

- Digitális áramkörök tervezése
  - A tervezés folyamata
    - Specifikáció
      - Különböző szinteken megadjuk mit kell „csinálnia” az áramkörnek
      - Hogyan épül fel
      - Belső részegységek, modulok specifikációja
    - HW leírás
      - Tervezői forrásfájl (HDL)
      - A specifikációban megadott viselkedés leírása HDL szintaxissal
      - Modul példányok, busz és vezeték hozzárendelések
      - Formális ellenőrzés (ált. beépített szintaktikai ellenőrzés)
        - » Ami szintaktikailag helyes nem biztos, hogy szintetizálható
        - » Digitális elektronika fogja megvalósítani a leírást
    - Szintézis
      - A HW leírás, a terv konkrét hardver elemekre történő „lefordítása”
      - A tervben leírt funkciót milyen konkrét digitális elektronika valósítja meg
      - Végeredménye egy összekötés lista

Digitális rendszer-tervezés

- Digitális áramkörök tervezése
  - A tervezés folyamata



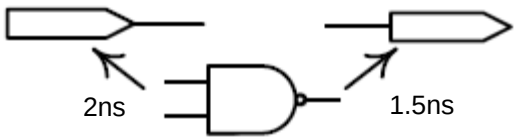


Digitális rendszer-tervezés

- Digitális áramkörök tervezése
  - A tervezés folyamata
    - Specifikáció
      - Különböző szinteken megadjuk mit kell „csinálnia” az áramkörnek
      - Hogyan épül fel
      - Belső részegységek, modulok specifikációja
    - HW leírás
      - Tervezői forrásfájl (HDL)
      - A specifikációban megadott viselkedést leírása HDL szintaxissal
      - Modul példányok, busz és vezeték hozzárendelések
      - Formális ellenőrzés (ált. beépített szintaktikai ellenőrzés)
        - » Ami szintaktikailag helyes nem biztos, hogy szintetizálható
        - » Digitális elektronika fogja megvalósítani a leírást
    - Szintézis
      - A HW leírás, a terv konkrét hardver elemekre történő „lefordítása”
      - A tervben leírt funkciót milyen konkrét digitális elektronika valósítja meg
      - Végeredménye egy összekötés lista
        - » IC, ASIC: Tranzisztorszintű, kapuszintű IC terv
        - » Prog. logika: Részegységek közötti összeköttetések listája, kezdőértékek (konfigurációs fájl, program)

Digitális rendszer-tervezés

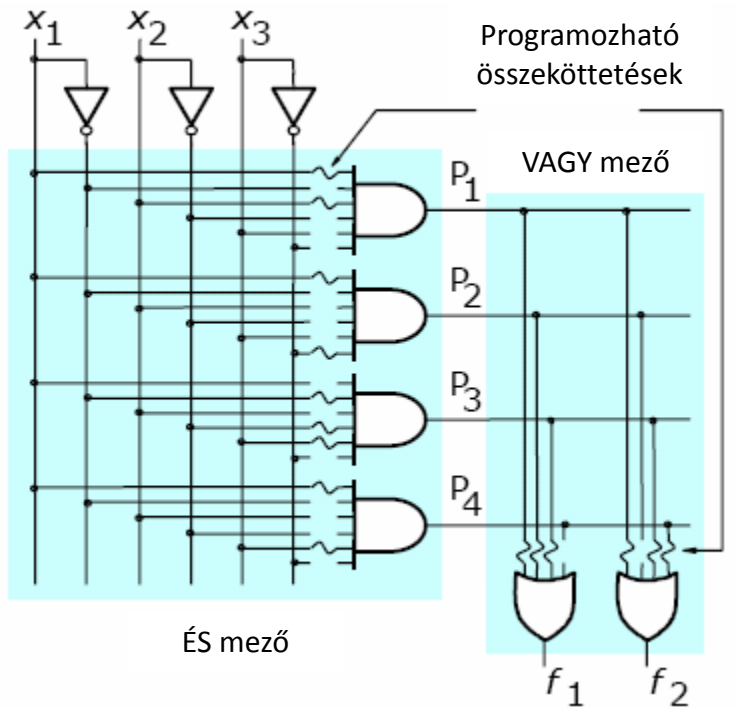
- Digitális áramkörök tervezése
  - A tervezés folyamata
    - Szimuláció
      - Viselkedés szintű szimuláció (Behavioral Simulation)
        - » Ellenőrizzük, hogy a terv a specifikációknak megfelelően működik-e
        - » Bemeneti tesztsorozatot állítunk elő, vizsgáljuk az ennek hatására kialakuló kimeneti jelet (Testbench)
        - » Nem veszi figyelembe a valós áramkör tulajdonságait (időzítési problémák, késleltetés stb.)
      - Szintézis utáni szimuláció (Post-synthesis Simulation)
        - » A valós áramkör fizikai modelljét felhasználva a vezetékezés és a kapuáramkörök késleltetési idejét is figyelembe veszi
        - » Timing analysis (setup-time, hold-time)
        - » Az előállított tesztsorozattal ellenőrizzük, hogy a terv a specifikációknak megfelelően működik-e
        - » A szintézis utáni szimulációnak is igazolni kell a helyes működést



Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - PLA (Programmable Logic Array)
    - A felhasználó által programozható logikai elrendezés
    - Különféle logikai elemeket tartalmaz, melyek közötti összekötetés többféleképen is kialakítható
      - ÉS, VAGY ill. INVERTER kapuk
    - Logikai elemek (kapuk) közötti összeköttetéseket „programozható” kapcsolók biztosítják

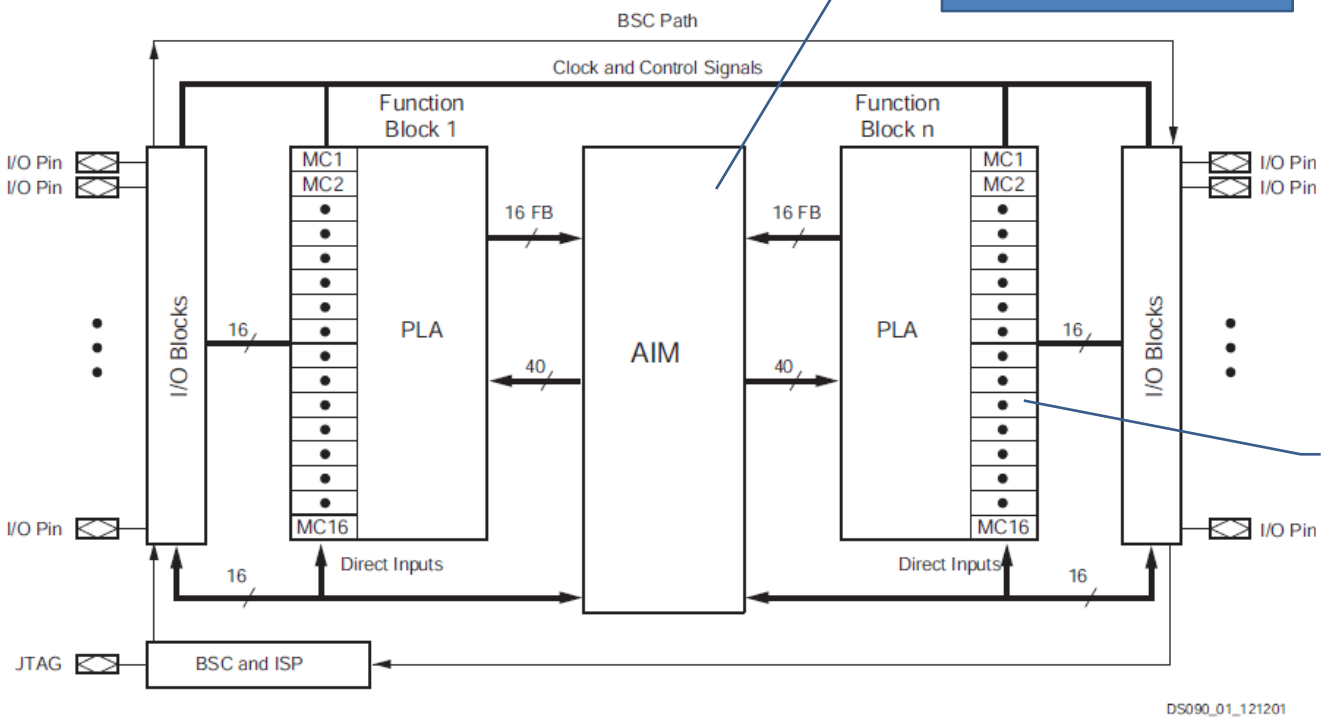
  - Közvetlenül a logikai függvények diszjunktív alakját valósítják meg
  - Az ÉS kapcsolatok programozhatók a VAGY kapcsolatok általában rögzítettek
  - Az adott számú be- és kimenet, valamint az ÉS kapuk száma korlátozza a megvalósítható függvény bonyolultságát



Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - CPLD (*Complex programmable logic device*)
    - Több, egymással programozható összeköttetésben lévő PLA-t tartalmazó eszközök
    - Egyéb funkcionális elemeket is tartalmaznak
      - Órajel kezelő áramkör
      - Tároló elemek, multiplexerek

PLA-k közötti programozható összeköttetések



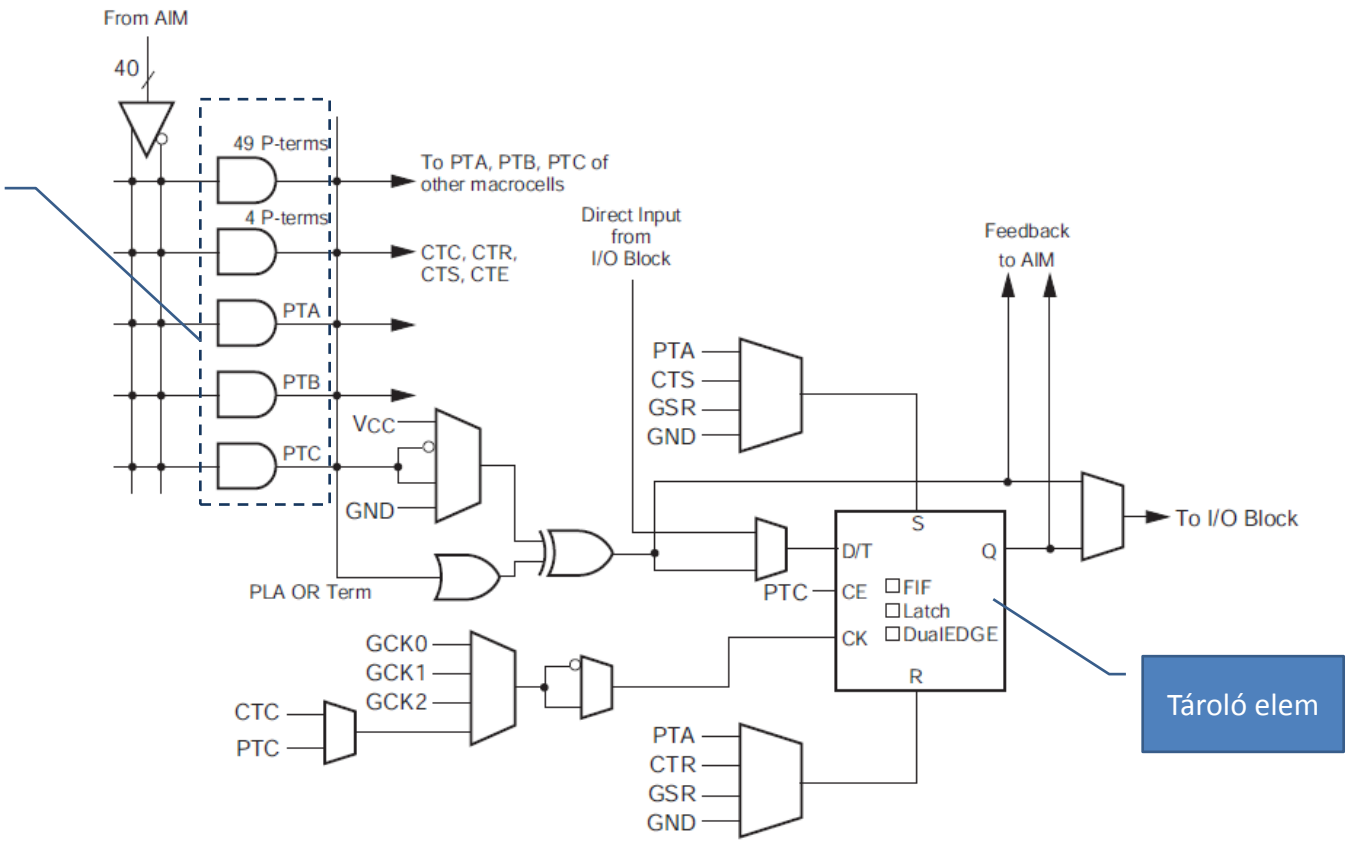
Speciális logikai elemek (Macrocells)

Figure 1: **CoolRunner-II CPLD Architecture**

Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - CPLD (*Complex programmable logic device*)
    - Tipikusan néhány ezer kapu

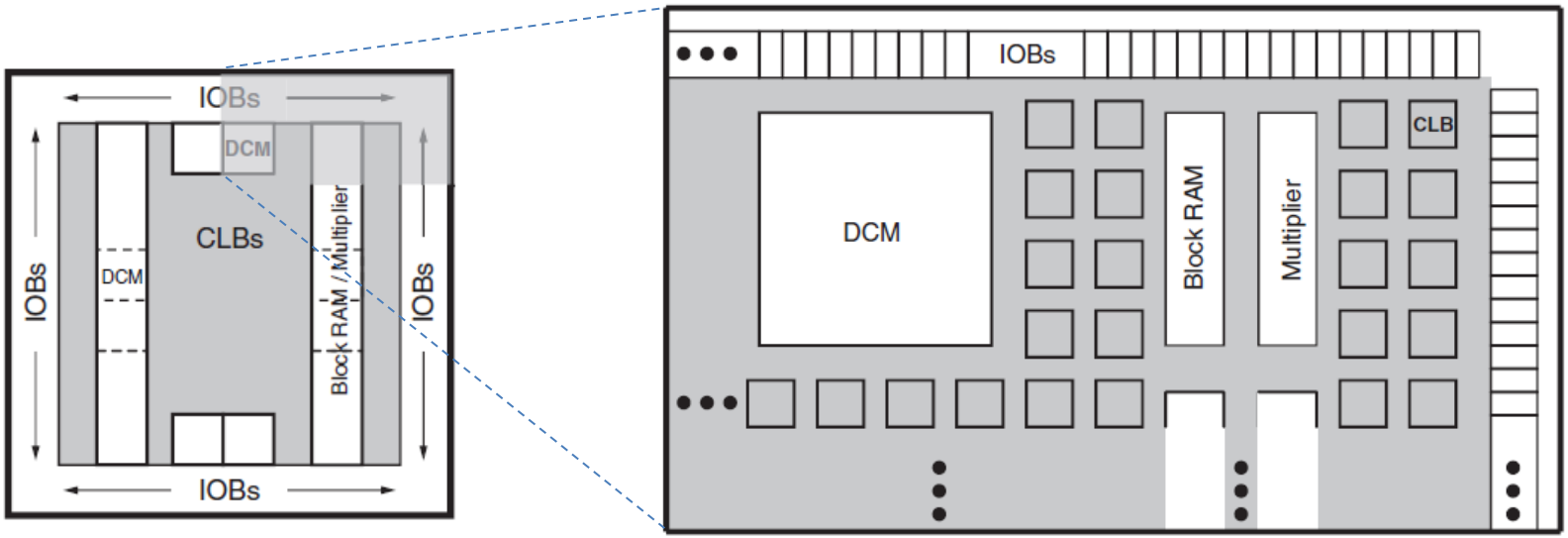
Diszjunktív függvény előállítása (szorzatok összege) Max. 56 term



Tároló elem

Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Field-Programmable Gate Array*)
    - **Konfigurálható logikai blokkok** (CLB-k): logikai függvények és tároló funkció megvalósítására alkalmas elemek
    - **Input/Output blokkok** (IOB-k): A be- és kimenetek (a külvilág) valamint a belső logika elemek közötti adatáramlást valósítják meg. Lehetővé teszik a kétirányú és háromállapotú (3-state) interfészek valamint különböző szabványú és feszültségszintű digitális jelek illesztését.
    - **Blokk RAM**: Adattárolásra alkalmas memória elemek.
    - **Szorzó blokk** (Multiplier): Két 18-bites bináris szám gyors összeszorzására alkalmas egységek.
    - **Digitális órajel menedzser blokk** (DCM): Az órajelek kezelésére szolgáló programozható egység. Szolgáltatásai: késleltetés, frekvencia szorzás-osztás, fázistolás.

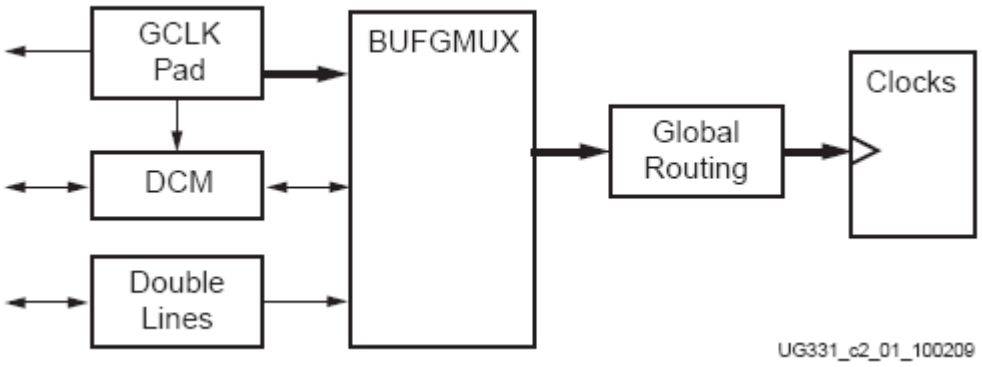


Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA
    - Tipikusan több százezer kapu
    - Az FPGA-n belül sűrű vezetékhálózat biztosítja az egyes elemek közötti kapcsolatot
    - A funkcionális blokkok programozható összeköttetéseken (kapcsoló mátrix) keresztül kapcsolódnak a vezetékhálózathoz
    - Az FPGA programja (konfigurációja) a funkcionális blokkok vezérlőjeleit valamint a kapcsolómátrixok állapotát határozza meg
      - Mely egységek kerüljenek egymással összeköttetésbe
    - A programot az FPGA-n belül statikus konfigurációs memória (SRAM) tárolja
      - A tápfeszültség rákapcsolása után valamilyen külső forrásból fel kell tölteni
      - EEPROM
      - Mikroprocesszor

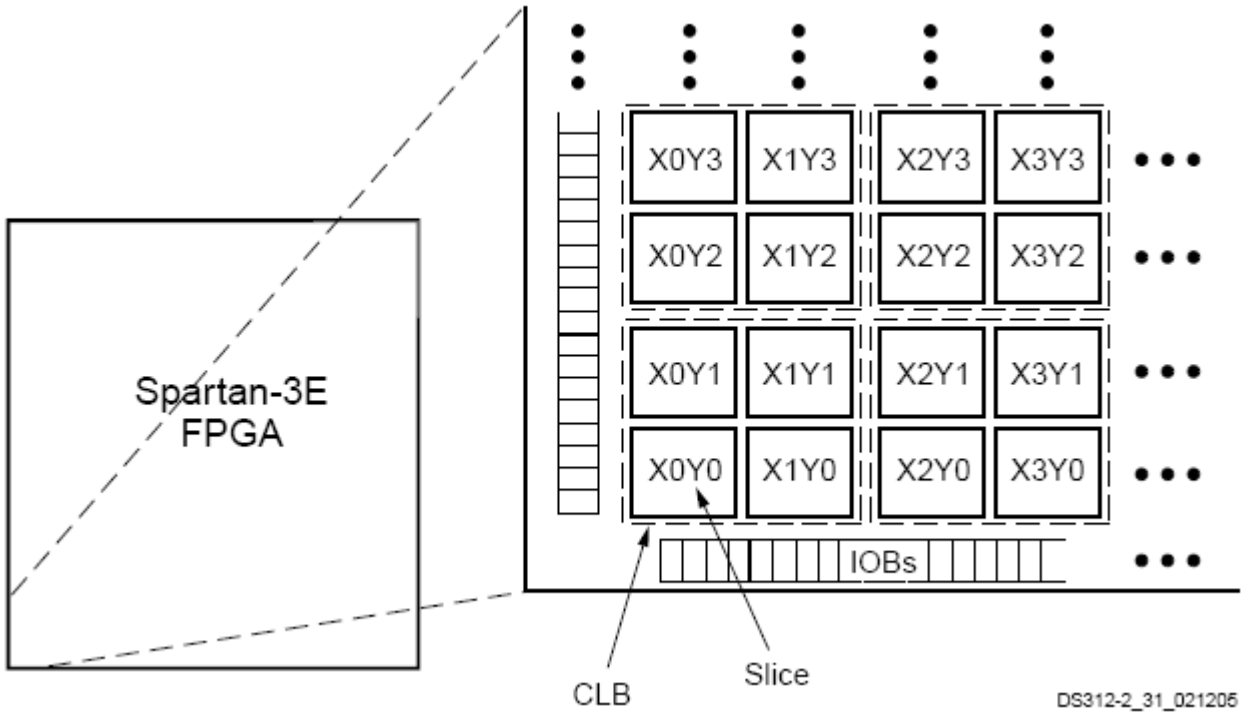
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - Az órajelek FPGA-n belüli elosztásáért speciális belső vezetékhálózat felelős
      - Az órajel hálózathoz speciális bemeneti blokk (GCLK) és meghajtó-multiplexer (BUFGMUX) tartozik
      - Az órajel hálózatra kerülő jelet is multiplexer választja ki a GCLK bementről vagy a DCM valamelyik kimenetéről
    - Általános felhasználású bemenetről érkező jel vagy belső jel is lehet órajel, a nagysebességű szinkron működés biztosítása érdekében azonban ez nem javasolt



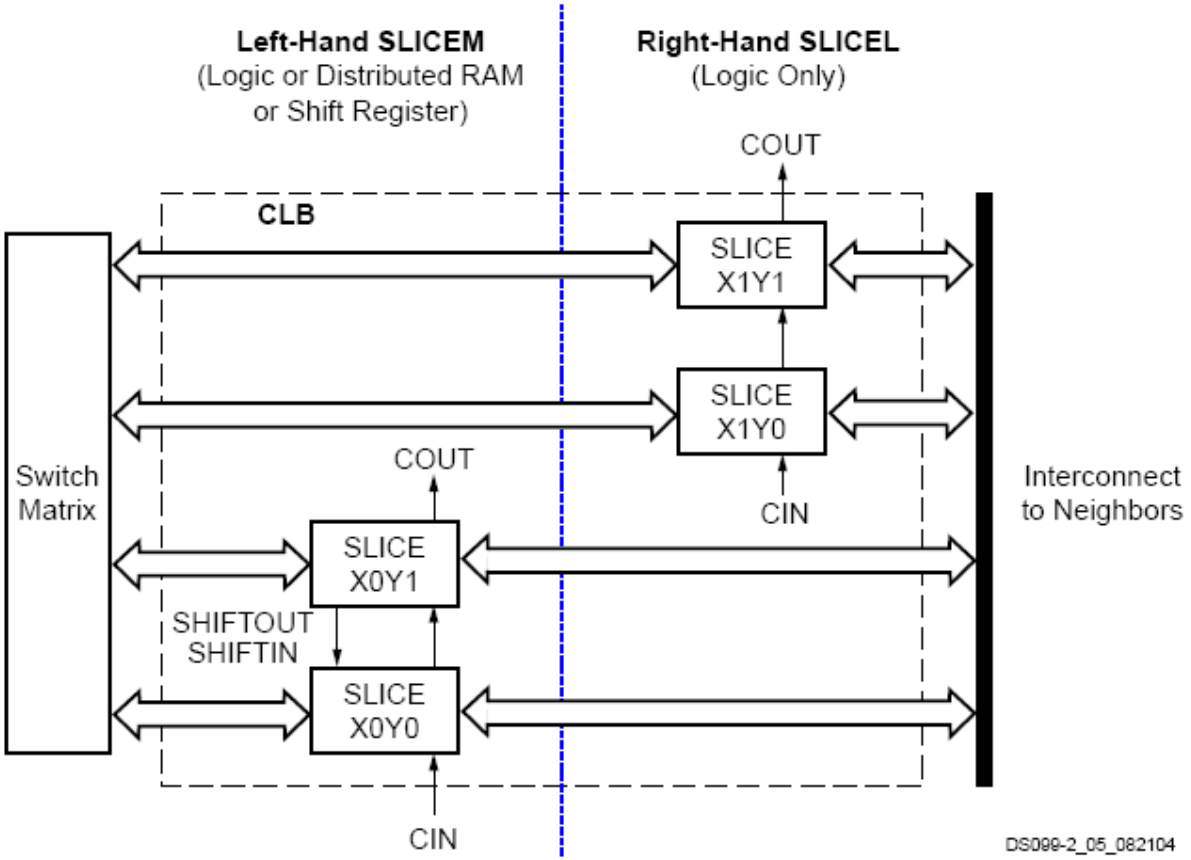
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - A CLB-k az elsődleges építő elemei az FPGA-ban felépített logikai hálózatoknak



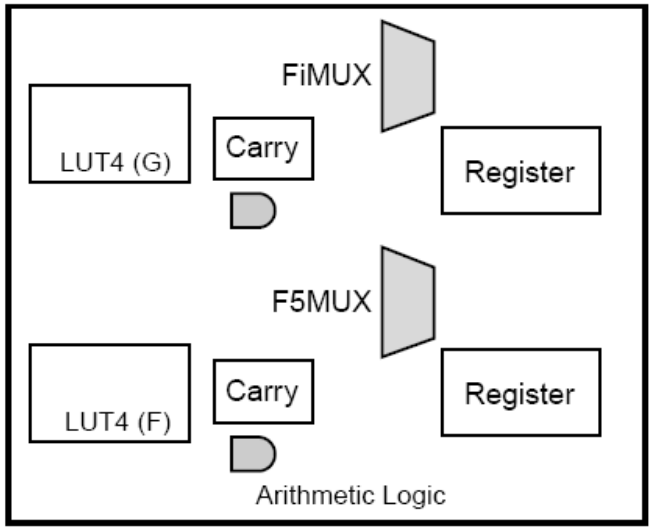
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - A CLB-k az elsődleges építő elemei az FPGA-ban felépített logikai hálózatoknak
    - A CLB-k egymással összeköttetésben lévő szeletekből (SLICE) épülnek fel

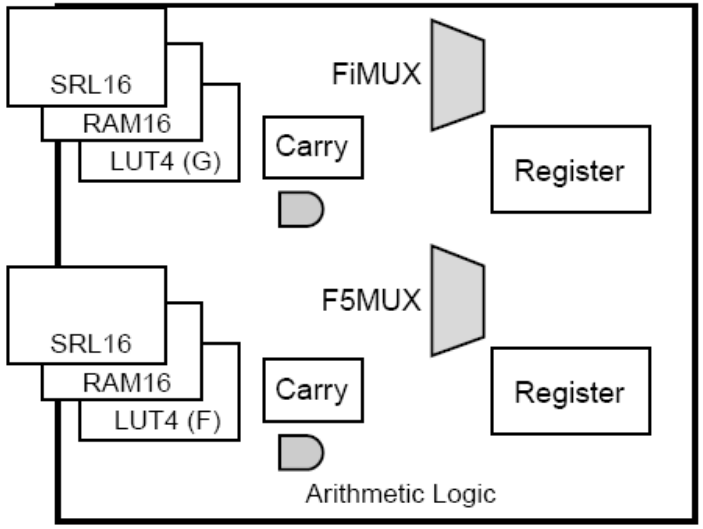


Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - Két szelet-típustól
      - SLICEM: Logikai és memória funkció
      - SLICEL : Csak logikai funkció
    - Mindkét szelet tartalmazza
      - Két 4-bemenetű LUT (Look-Up-Table)
      - Két tároló elem
      - Két multiplexer
      - Carry és aritmetikai logika
    - A SLICEM szeletek további összetevői
      - Két 16x1 bit RAM blokk
      - két 16-bites shift-regiszter



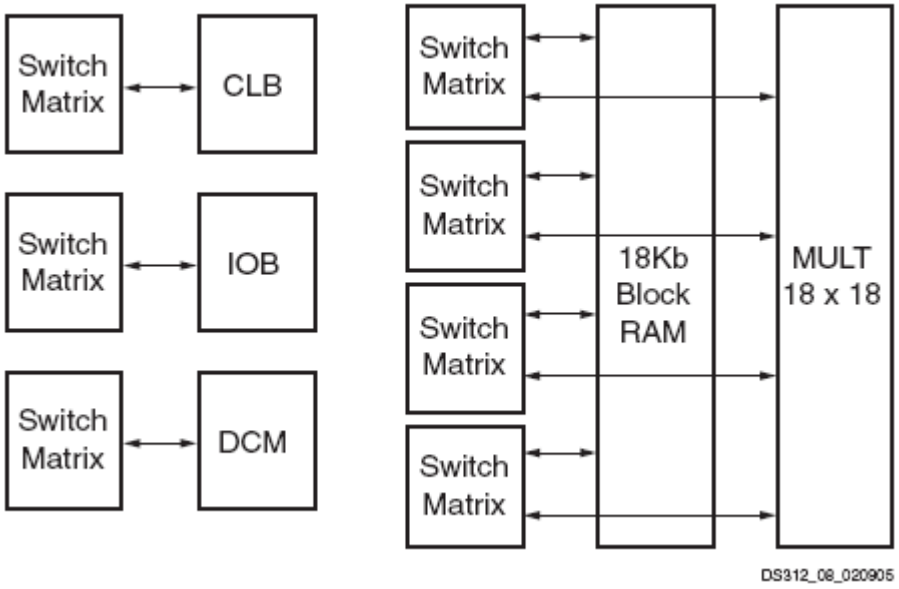
SLICEL DS312-2_13_020905



SLICEM

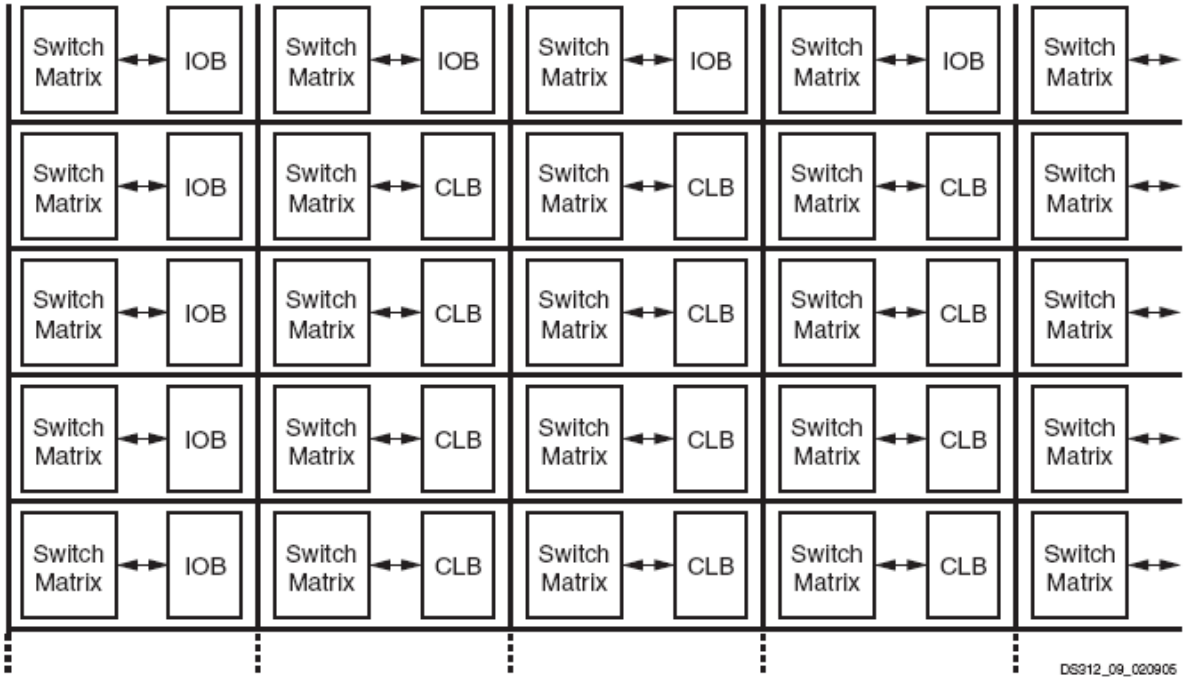
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - Az FPGA erőforrásai közötti kapcsolatot a kapcsoló mátrixok biztosítják
      - » CLB, IOB, DCM, RAM, szorzó
    - A kapcsoló mátrixok a belső vezetékhálózatra kapcsolódnak, ami horizontálisan és vertikálisan az egész FPGA-t lefedi
      - » bizonyos megkötésekkel bármely elem bármelyik másikkal összeköttetésbe hozható



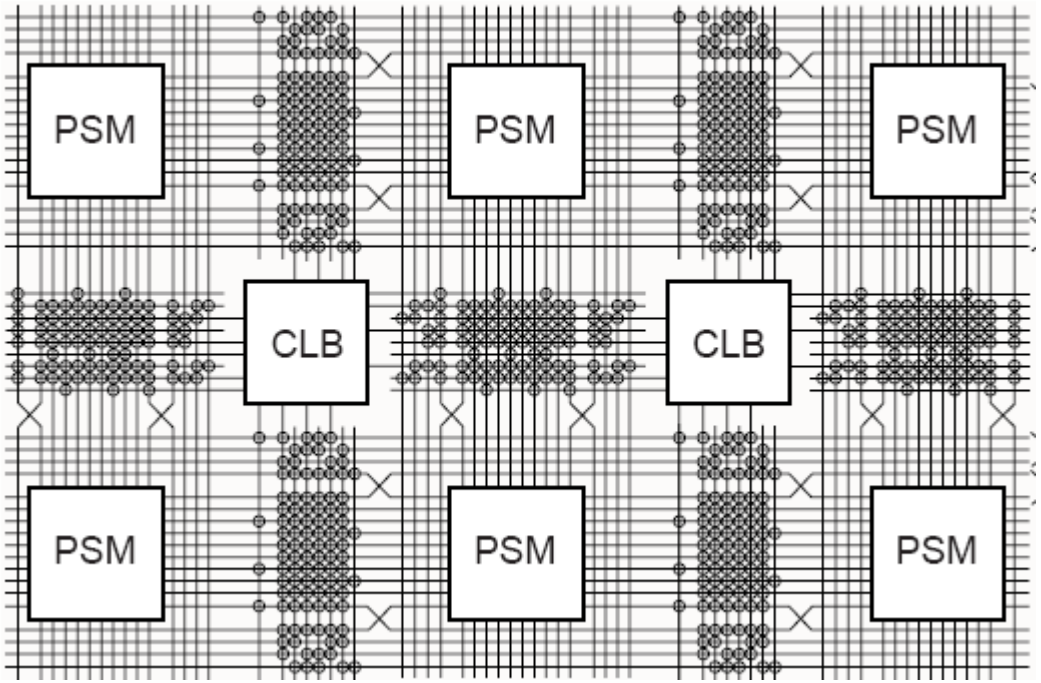
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - Az FPGA erőforrásai közötti kapcsolatot a kapcsoló mátrixok biztosítják
      - » CLB, IOB, DCM, RAM, szorzó
    - A kapcsoló mátrixok a belső vezetékhálózatra kapcsolódnak, ami horizontálisan és vertikálisan az egész FPGA-t lefedi
      - » bizonyos megkötésekkel bármely elem bármelyik másikkal összeköttetésbe hozható



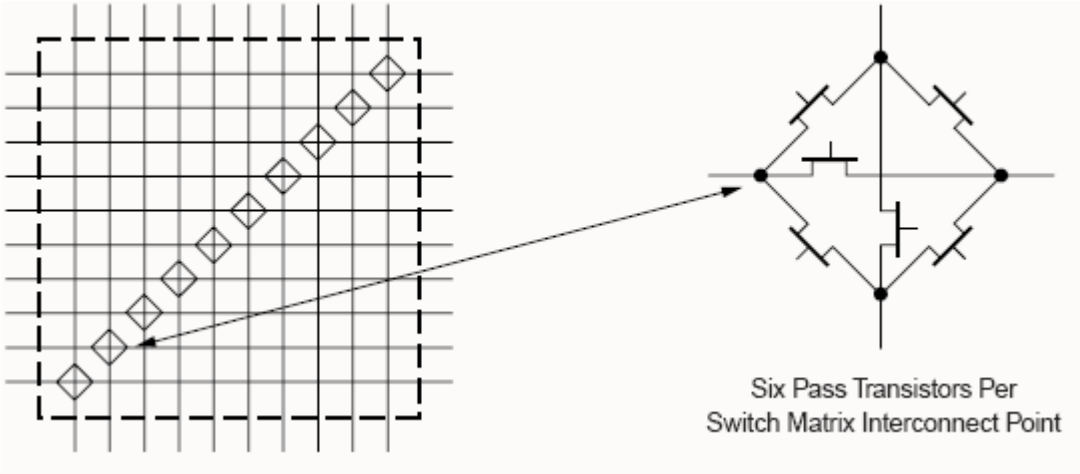
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - Az FPGA erőforrásai közötti kapcsolatot a kapcsoló mátrixok biztosítják
      - » CLB, IOB, DCM, RAM, szorzó
    - A kapcsoló mátrixok a belső vezetékhálózatra kapcsolódnak, ami horizontálisan és vertikálisan az egész FPGA-t lefedi
      - » bizonyos megkötésekkel bármely elem bármelyik másikkal összeköttetésbe hozható



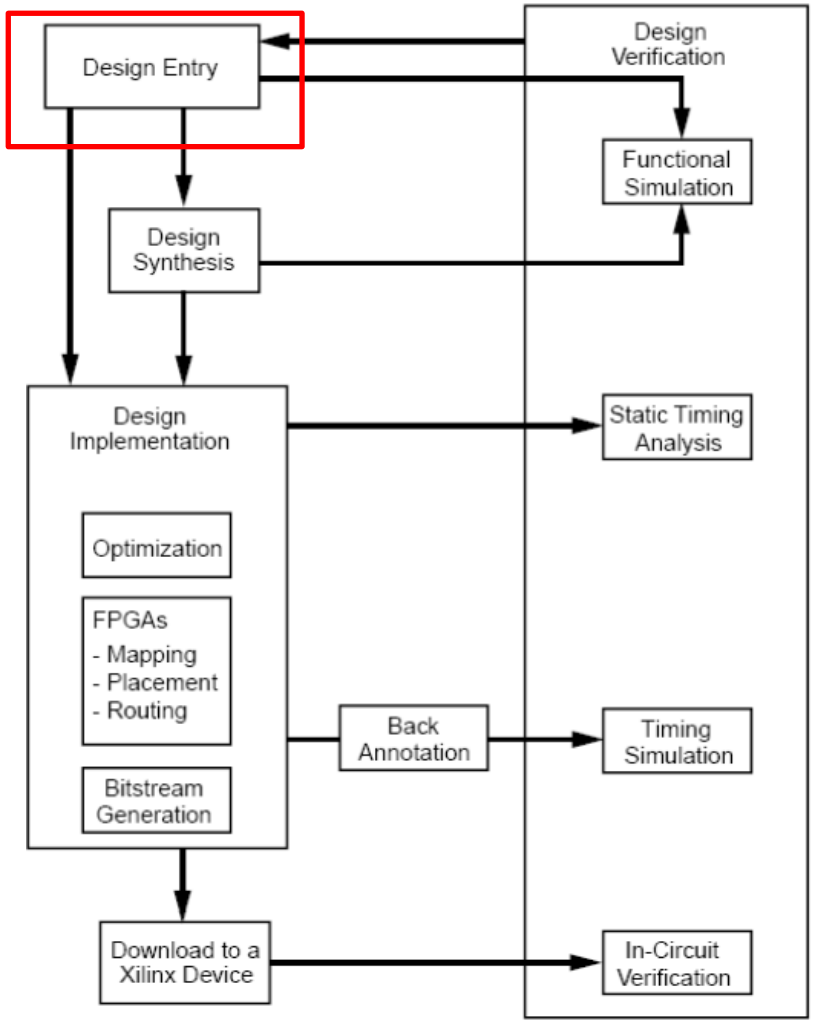
Digitális rendszer-tervezés

- Programozható logikai áramkörök
  - FPGA (*Xilinx Spartan 3E*)
    - Az FPGA erőforrásai közötti kapcsolatot a kapcsoló mátrixok biztosítják
      - » CLB, IOB, DCM, RAM, szorzó
    - A kapcsoló mátrixok a belső vezetékhálózatra kapcsolódnak, ami horizontálisan és vertikálisan az egész FPGA-t lefedi
      - » bizonyos megkötésekkel bármely elem bármelyik másikkal összeköttetésbe hozható
    - Az FPGA programja (konfigurációja) a funkcionális blokkok vezérlőjeleit valamint a kapcsolómátrixok állapotát határozza meg



Digitális rendszer-tervezés

- A tervezés folyamata
  - Xilinx ISE



Xilinx ISE Design Entry/Synthesis:

- Terv létrehozása
- Kapcsolási rajz alapon
- HDL alapon (hardverleíró nyelv)
- Egyéb forrásból (FSM,…)
- Kitételek/korlátozások megadása

Design Implementation:

- A terv (logikai leírás) konvertálása fizikai információvá (konfiguráló bitfolyammá)
- Mapping(MAP): a terv adaptálása az adott eszközben, kitételek feldolgozása, tervezési szabályok ellenőrzése
- Placement/Routing(PAR): elemek elhelyezése, összekötések megvalósítása, optimalizálás

Design Verification:

- Az elkészült áramkör funkcionális és minőségi vizsgálata (szimuláció/in-circuit ellenőrzés)

Digitális rendszer-tervezés

- VHDL
  - VHDL: VHSIC HDL (Very High Speed Integrated Circuit Hardware Description Language)
  - 1981: Amerikai Védelmi Minisztérium (US DoD) kezdeményezése
  - Céljai:
    - Rugalmas leírást nyújtson
    - Minden szimulátorral ugyanazt az eredményt adja
    - Technológia függetlenség
  - 1983-85: fejlesztés: Intermetrics, IBM és TI
  - 1987: első szabvány (IEEE Std. 1076-1987)
  - Módosítások:
    - IEEE Std. 1076-1993
    - IEEE Std. 1076-2000
    - IEEE Std. 1076-2002
    - IEEE Std. 1076-2008

Digitális rendszer-tervezés

- VHDL
  - 2007: VHPI (VHDL Procedural Interface): ezen interfészen keresztül pl. C-ben írt programmal hozzáférhetünk a VHDL modellhez
    - Alkalmazás: például processzor utasításkészletének szimulációja
  - Kiterjesztések:
    - VHDL-200X: HDL & HVL (Hardware Verification Language)
    - VHDL-AMS: Analog & Mixed-Signal
  - 1990: Verilog
    - IEEE Std. 1364-1995; IEEE Std. 1364-2001
  - A szabványok fenntartója
    - Accellera (Open Verilog International (OVI) +VHDL International)
  - Verilog kiterjesztések:
    - SystemVerilog (IEEE Std. 1800-2005): HDL & HVL
    - Verilog-AMS (draft): Analog & Mixed-Signal

Digitális rendszer-tervezés

- VHDL
  - Rendszertervezésre és szimulációra egyaránt alkalmas
  - Eszközfüggetlen tervezés
    - Többféle gyártó programozható eszközeihez is alkalmazható
    - Könnyű átjárás különböző technológiák között
      - FPGA -> ASIC
      - Különböző gyártástechnológiák
  - Alaptulajdonságok
    - Konkurens utasítások (kombinációs logika)
      - Párhuzamosan végrehajtandó utasítások megadása (mint a HW-ben)
    - Szekvenciális utasítások (sorrendi logika)
      - Egymás után végrehajtandó utasítások megadása
    - Erősen típusos
      - Minden elemnek van egy típusa, és csak ilyen típusú értéket vehet fel
    - Hierarchikus leírás
    - Gyártói könyvtárak támogatása

Digitális rendszer-tervezés

- VHDL
  - Lexikai elemek
    - Azonosítók:
      - Betűk (`A-Z, a-z`), számok (`0-9`), aláhúzás (_)
      - Betűvel kell kezdődnie, pl.: `sig; Sig; SIG; S_sig34`
      - A VHDL jelölésmódja nem tesz különbséget a nagybetűk és a kisbetűk között
      - Kivétel az egyes idézőjelek (' ') vagy a kettős idézőjelek (" ") közötti karakterek esetén
    - Megjegyzések: -- (2 db egymást követő kötőjel)

      ```
      -- comment
      ```

    - Karakterek: ' '

      ```
      'c'; 'C'; '56'
      ```

    - Stringek: " "

      ```
      "string"; "10011"; X"FFA5"
      ```

    - Lefoglalt szavak nem használhatók

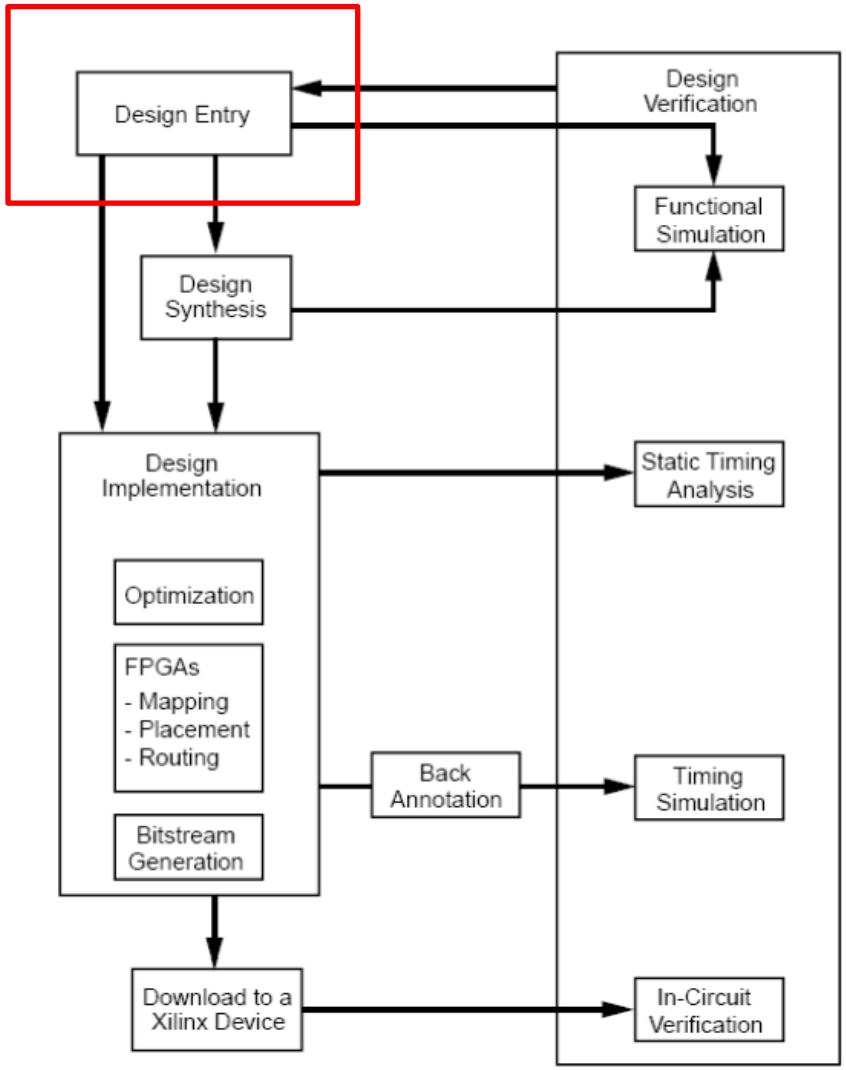
Digitális rendszer-tervezés

- VHDL
  - Lexikai elemek
    - Lefoglalt szavak

abs	access	after	alias	all	and
architecture	array	assert	attribute	begin	block
body	buffer	bus	case	component	
configuration	constant	disconnect	downto	else	elsif
end	entity	exit	file	for	function
generate	generic	group	guarded if	impure	in
inertial	inout	is	label	library	linkage
literal	loop	map	mod	nand	new
next	nor	not	null	of	on
open	or	others	out	package	port
postponed	procedure	process	protected	pure	range
record	register	reject	rem	report	return
rol	ror	select	severity	shared	signal
sla	sll	sra	srl	subtype	then
to	transport	type	unaffected	units	until
use	variable	wait	when	while	with
xnor	xor				

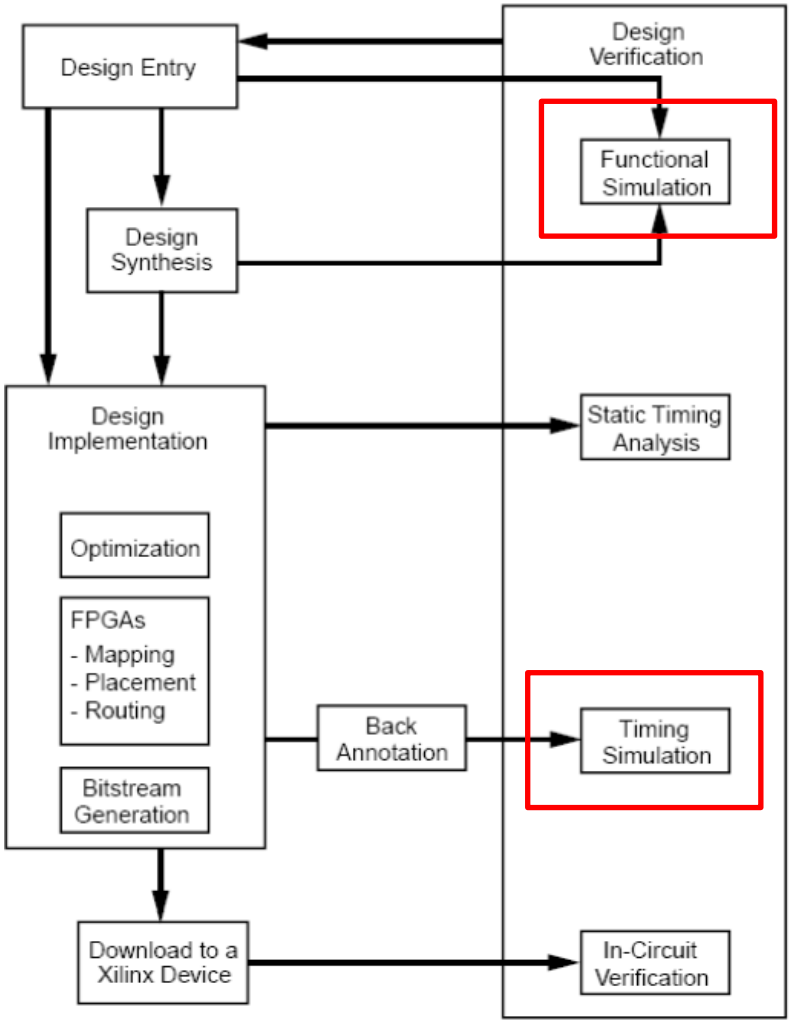
Digitális rendszer-tervezés

- VHDL
  - A terv (Design)



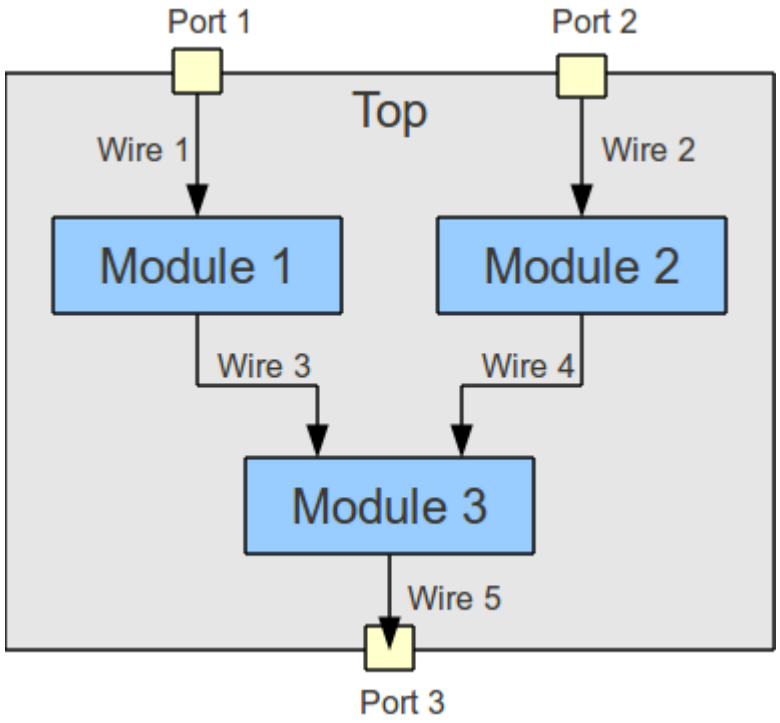
Digitális rendszer-tervezés

- VHDL
  - A terv (Design)
  - Szimuláció



Digitális rendszer-tervezés

- VHDL
  - A terv (Design)
    - Jelek/Vezetékek
    - Modulok (komponensek)
    - Portok



VHDL

- Jelek
  - Feladatuk
    - A modulok közötti információtovábbítás biztosítása (az egységek összekötése)
    - Inicializálás
  - Deklaráció
    - Általános jelek
      - A modulból közvetlenül kivezethető jelek

```
signal azonosító : típus [ := kifejezés ];
```

      - Értékadó operátor <=

    - Változók
      - Nincsenek közvetlenül kivezetve a modulból
      - Átmeneti információ tárolására

```
variable azonosító : típus [ := kifejezés ];
```

      - Értékadó operátor :=

    - Állandó jelek

```
constant azonosító : típus := kifejezés;
```

VHDL

- Adattípusok
  - Egész számok: `integer`
    - Beépített 32 bites, előjeles típus: -2147483648 és +2147483647 között
    - Altípusai
      - `natural`: természetes számok
      - `positive`: pozitív egész számok
    - Ha nem akarjuk a teljes számábrázolási tartományt használni
      - Altípus definiálása:

```
subtype int8 is integer range -128 to 127;
```

    - Az egész számokra értelmezett műveletek:

`:=`	értékadás
`+`	összeadás
`-`	kivonás, vagy előjelváltás
`*`	szorzás
`/`	(egész)osztás
`mod`	maradékképzés, a végeredmény előjele az osztóéval egyezik meg
`rem`	maradékképzés, a végeredmény előjele az osztandóéval egyezik meg
`abs`	abszolút érték
`**`	hatványozás (jobb oldali operandus nem lehet negatív)

VHDL

- Adattípusok
  - Lebegőpontos számok: `real`
    - Beépített VHDL típus
    - Implementáció függő megvalósítás (általában IEEE 64-bites, dupla pontosságú)

    - Az lebegőpontos számokra értelmezett műveletek:
      - `:=` értékadás
      - `+` összeadás
      - `-` kivonás, vagy előjelváltás
      - `*` szorzás
      - `/` osztás
      - `abs` abszolút érték
      - `**` hatványozás (jobb oldali operandus nem lehet negatív)

    - VHDL-ben a lebegőpontos számábrázolást ritkán használják

VHDL

- Adattípusok
  - Időzítési értékek: `time`
    - Beépített VHDL típus
    - Szimulációnál események időzítésére, késleltetések megadására
    - Használt mértékegységek

```
fs  ps  ns  us  ms  sec  min  hr

7 ns        21 us       235.3 ps
```

  - Felsorolás típus

```
type azonosító is (értékek_felsorolása);
```

    - Felsorolásszerűen megmondjuk, hogy a változó milyen értékeket vehet fel

```
type octal_digit is ('0', '1', '2', '3', '4', '5', '6', '7');
```

VHDL

- Adattípusok
  - Logikai típus: `boolean`
  - Logikai műveletek eredményének tárolására
  - Általában vezérlési célokra

```
type boolean is (false, true);
```

  - Logikai típusú eredményt adó operátorok

<	kisebb
<=	kisebb vagy egyenlő
>	nagyobb
>=	nagyobb vagy egyenlő
=	egyenlő
/=	nem egyenlő

VHDL

- Adattípusok
  - Logikai típus: `boolean`
  - A logikai műveletek operandusai logikai típusúak
  - A műveletek eredménye is logikai típusú

```
and     logikai ÉS              a > 12 and b = bits
or      logikai VAGY            a > 12 or b = bits
nand    logikai NEM-ÉS          a > 12 nand b = bits
nor     logikai NEM-VAGY        a > 12 nor b = bits
xor     logikai kizáró-VAGY     a > 12 xor b = bits
xnor    logikai ekvivalencia    a > 12 xnor b = bits
not     logikai negálás         not b = bits
```

VHDL

- Adattípusok
  - Bit típus: `bit`
  - Logikai operátorok a `bit` típussal is használhatók
  - Az operandusok és az eredmény is `bit` típusú lesz

```
type bit is ('0', '1');
```

VHDL

- Adattípusok
  - Standard logikai típus: `std_logic`
  - Logikai operátorok az `std_logic` típussal is használhatók
  - A digitális áramkörök jelvezetékeinek lehetséges állapotai

```
type std_logic is (
                    'U' – uninitialized
                    'X' – strong unknown
                    '0' – strong 0
                    '1' – strong 1
                    'Z' – high impedance
                    'W' – weak unknown
                    'L' – weak 0 (wired-OR)
                    'H' – weak 1 (wired-AND)
                    '-' – don't care
                   );
```

VHDL

- Adattípusok
  - Standard logikai típus: `std_logic`
  - A CPLD-k és FPGA-k jelvezetékeinek lehetséges állapotait az std_logic_1164 csomag tartalmazza.
  - Használat előtt be kell építeni a kódunkba (mint az „include”)

```
library ieee;
use ieee.std_logic_1164.all;
```

  - Egy kimenet értéke nem csak logikai „0” vagy „1” lehet, hanem ismeretlen, nem inicializált, határozott „0”, határozott „1”, nagyimpedanciás, gyenge ismeretlen, gyenge „0”, gyenge „1”, és don't care értéket is felvehet.
  - Definiálhatunk olyan jelvezetékeket is, melyet aktív áramkör hajt meg, vagy amit felhúzó / lehúzó ellenállás állít be valamilyen értékre.

  - Egyéb könyvtárakban számos előre definiált alapelem megtalálható

VHDL

- Adattömbök
  - Azonos típusú adatok halmazának definiálására

```
array ( natural to|downto natural ) of típus;
```

  - A bitvektorokat (vagyis bitek tömbjét) gyakran használjuk, ezért a VHDL nyelv külön típust definiál számukra:

```
type bit_vector is array (natural range <>) of bit;
type std_logic_vector is array ( natural range <> ) of std_logic;
```

  - Értékadás a tömb elemeinek

```
signal bus8bit: std_logic_vector(7 downto 0);
signal bus9bit: std_logic_vector(8 downto 0);
…
bus8bit(0) <= '1';
bus8bit(7 downto 1) <= "1000111";
…
bus9bit <= '1' & bus8bit;   -- eredménye: bus9bit <= "110001111";
```

VHDL

- Jelek
  - Példa jelek deklarálására:

```
signal   a : std_logic;
signal   b : bit := '0';
constant c : integer := 16;
signal   d : std_logic_vector(31 downto 0) := X"0000";
signal   e : std_logic_vector(0 to 15);
signal   f : bit_vector(15 downto 12) := B"0000";
```

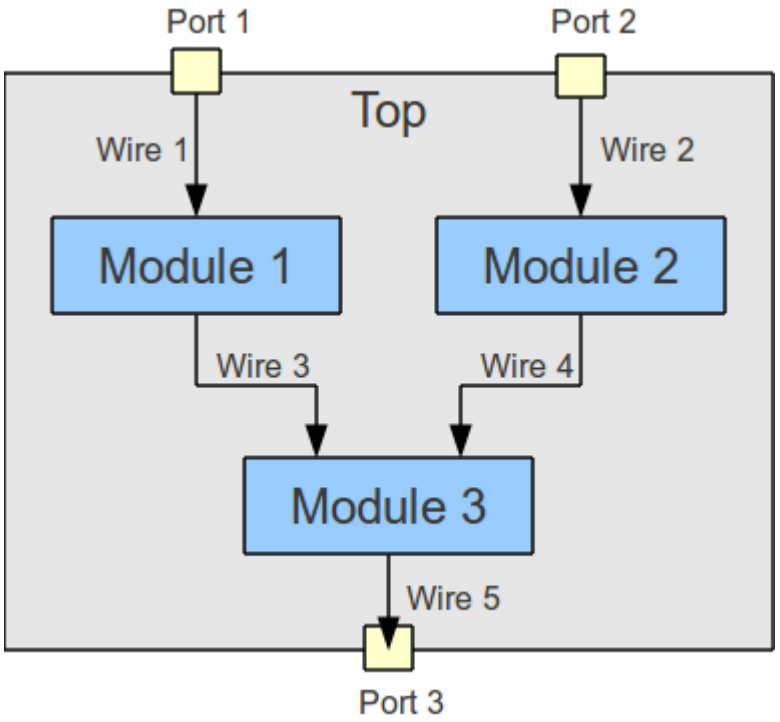
  - Megjegyzés: radix megadása:
    - B – bináris
    - O – oktális
    - X - hexadecimális

VHDL

- Modul
  - Funkcionális HW elem viselkedését írja le
    - Képviselheti a tervhierarchia egyik szintjét, vagy egy teljes tervet
  - Szimulációban a tesztminta generátor működését írja le
  - A külvilággal való kapcsolat definiálása

    `entity`
    - Portok
      - Bemeneti jelek
      - Kimeneti jelek
  - A működés leírása

    `architecture`
    - Belső jelek
    - Belső modulok
      - Tetszőleges mélységig



VHDL

- Modul
  - Felépítése
    - entity
    - architecture

```
library IEEE;
use IEEE.std_logic_1164.all;

entity modul_azonosító is
                    … -- az entitás leírása
end modul_azonosító ;

architecture arch_azonosító of modul_azonosító is
                    … -- az architektúra leírása
end arch_azonosító ;
```

VHDL

- Modul
  - Entitás port lista

```
entity module is
    port (
          A : in std_logic;
          B : in std_logic;
          C : out std_logic
         );
end module;
```

- A port lista elemei kizárólag signal-ok (a `signal` kulcsszó el is hagyható)
- Speciális módosítóval rendelkeznek:
  - `in` az összetevő csak olvassa a jelet,
  - `out` az összetevő csak írja a jelet (nem olvassa vissza),
  - `inout` az összetevő olvassa vagy írja a jelet (kétirányú jel),
    - » CPLD-n, FPGA-n belül nincsenek valódi kétirányú jelek
  - `buffer` az összetevő írja és visszaolvassa a jelet (nem kétirányú jel)

VHDL

- Modul
  - Entitás generic lista
    - Elemei bemenő paraméterek

```
entity ram is
    generic (
              width : integer := 32;
              depth : integer := 1024;
              addr_len : integer := 10
             );
    port (
           clk : in std_logic;
           rst : in std_logic;
           rd : in std_logic;
           wr : in std_logic;
           addr : in std_logic_vector(addr_len - 1 downto 0);
           data_in : in std_logic_vector(width - 1 downto 0);
           data_out : out std_logic_vector(width - 1 downto 0)
          );
end ram;
```

VHDL

- Modul
  - A működés leírása: `architecture`
  - Egyidejű, konkurens utasításokat tartalmaz
    - Párhuzamosan végrehajtandó utasítások megadása (mint a HW-ben)

```
architecture azonosító of entity_azonosító is
      … -- deklarációs rész
begin
      … -- konkurens utasítások
end azonosító;
```

VHDL

- Modul
  - Pl.

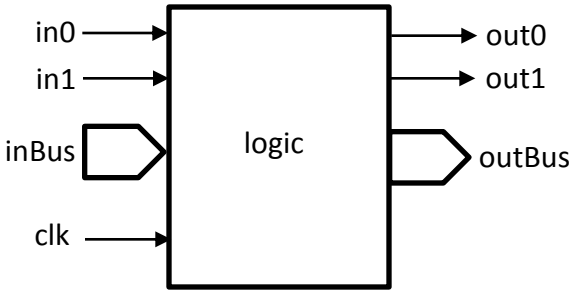
Használt könyvtárak (hasonló: `#include`)

Be- kimenetek definíciója (portok)

Deklarációs rész: Belső jelek, modulok konstansok stb.. definíciója, inicializálása

Működést leíró konkurens utasítások

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5
6  entity logic is
7      Port ( in0    : in  STD_LOGIC;
8             in1    : in  STD_LOGIC;
9             inBus  : in  STD_LOGIC_VECTOR (7 downto 0);
10            out0   : out  STD_LOGIC;
11            out1   : out  STD_LOGIC;
12            outBus : out  STD_LOGIC_VECTOR (7 downto 0);
13            clk    : in  STD_LOGIC);
14 end logic;
15
16 architecture Behavioral of logic is
17
18    signal inner_signal0 : STD_LOGIC := '0';
19    signal inner_signal1 : STD_LOGIC := '0';
20    signal inner_bus0 :                 0) := "00000000";
21    signal inner_bus1 :                 0) := X"00";
22
23 begin
24
25
26
27 end Behavioral;
28
```



VHDL

- Deklarációs rész
  - Jelek, konstansok
  - Komponensek deklarálása
    - Melyeket egy másik VHDL fájlban már megírtunk, vagy könyvtárból beemeltünk
    - Pl.:

```
component comp is
    generic(
              p : integer;
              r : integer
            );
    port(
         a : in std_logic;
         b : out std_logic
        );
end component;
```

    - A deklarált komponensek használatához példányosítani kell őket
      - A deklarációs rész után (**begin** ….. **end** között)
      - Egy komponens akár többször is példányosítható

VHDL

- Utasítások
  - Egyidejű (konkurens) utasítások

    - Összetevő példányosítás
    - `generate` utasítás
    - Egyidejű jel-értékadás, hozzárendelés (<=)
    - `when else` utasítás
    - `with` utasítás
    - `process`: sorrendi (szekvenciális) folyamatok
    - Típusok és állandók definíciója: globális felhasználásra
    - `procedure`, `function`: eljárás és függvény meghatározás
    - Eljárások hívása

VHDL

- Utasítások
  - Sorrendi (szekvenciális) utasítások
    - Process-en, eljáráson és függvényen belül

    - Típusok és állandók definíciója: helyi, lokális felhasználásra
    - Változó deklarálás
    - Változó-értékadás, hozzárendelés (`:=`)
    - Sorrendi jel-értékadás, hozzárendelés (`<=`)
    - `if-then-else`, `case`, `wait` utasítások
    - `loop`, `next`, `exit` utasítások
    - `procedure`, `function`: Eljárás és függvény meghatározás
    - Függvények és eljárások hívása
    - `return` utasítás: visszatérési érték beállítás

VHDL

- Utasítások
  - Összetevő példányosítás: `map`
  - A deklarált komponensből konkrét HW-összetevő (példány)
    - A példánynak egyedi azonosítót kell kapnia (`comp_inst1`)
    - A `generic map` és `port map` lista megadja a példány és a főmodul közötti összeköttetéseket

```
comp_inst1: comp
generic map(
              p => p_top,
              r => r_top
            )
port map(
           a => a_top,
           b => b_top
         );
```

- Bekötetlen kimenetek: `open` kulcsszó
- Bekötetlen bemenetek: `'1'`–hez vagy `'0'`-hoz kell rendelni

VHDL

- Utasítások
  - Hatékony többszöri példányosítás: `generate`
  - Ciklussal

```
comp_16_inst: for i in 0 to 15 generate
    comp_inst1: comp
    generic map(
                p => p_top,
                r => r_top
               )
    port map(
              a => a_top(i),
              b => b_top(i)
             );
end generate comp_16_inst;
```

VHDL

- Utasítások
  - Példányosítás
  - Különböző könyvtárakban számtalan előre definiált alapelem található
    - `library` és `use` kulcsszavak
    - A Xilinx saját (közvetlenül elérhető) alapelemei

    Spartan-3E Libraries

VHDL

- Utasítások
  - Folyamat: `process`
  - Sorrendi (szekvenciális) folyamatok leírása
    - Hasonlít egy sorrendi programhoz
    - Ha több folyamat van egy modulban, akkor azok egyidejűleg hajtódnak végre

```
azonosító: process(signal_1, signal_n)
… -- deklarációs rész
begin
… -- definíciós rész
end process azonosító;
```

    - Érzékenységi listát, várakozás (`wait`) utasítást, vagy eljárás hívást kell tartalmaznia
      - Tisztán kombinációs folyamat: kombinációs logika leírására
      - Órajeles folyamatok: kombinációs és szinkron sorrendi logika leírására
      - Eseményvezérelt
        - » Az érzékenységi jelek valamelyike megváltozik (pl. órajel él-váltás)
        - » Teljesülnek az utasítás végrehajtásához szükséges feltételek
      - A folyamat utolsó utasítása után, a végrehajtás visszatér a folyamat első utasítására, és ez folytatódik

VHDL

- Utasítások
  - Folyamat: `process`
    - Logikai feltételhez kötött végrehajtás: `if-then-else`
    - Csak sorrendi (szekvenciális) részen belül

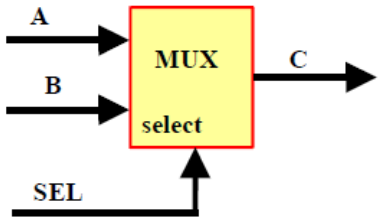
```
if logikai_feltétel_0 then
  …  -- utasítások
{ elsif logikai_feltétel_1 then
  …  -- utasítások }
…
{ elsif logikai_feltétel_n then
  …  -- utasítások }
[ else …
  …  -- utasítások ]
end if;
```

    - Több `elseif` is megengedett
    - De csak egy `else`

VHDL

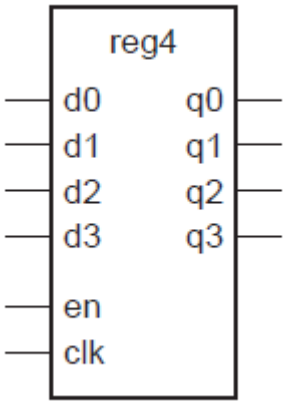
- Utasítások
  - Folyamat: `process`
    - Logikai feltételhez kötött végrehajtás: `if-then-else`

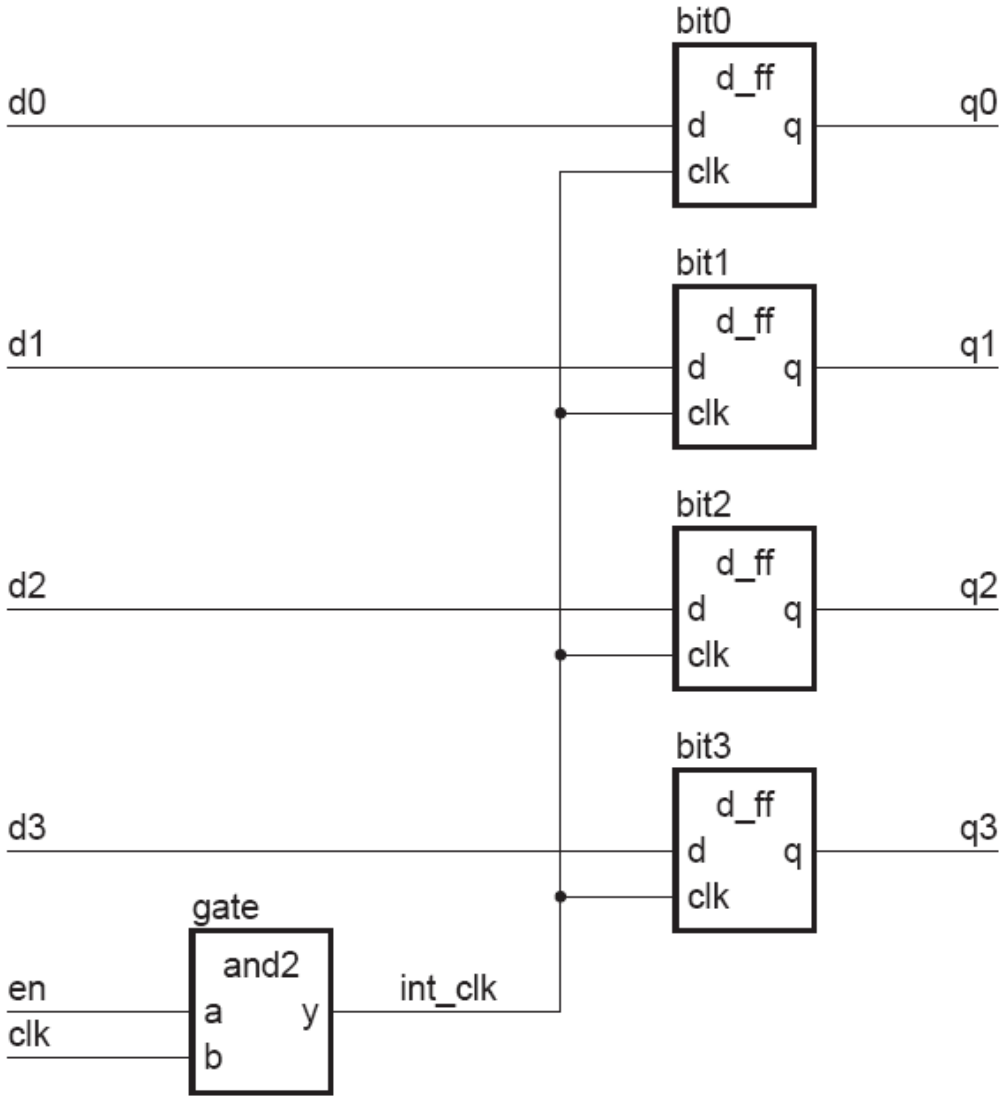
```
entity mux is
  port (a, b, sel :in bit;
        c: out bit);
end;

architecture bhv of mux is
begin
    process (a, b, sel)
    begin
        if sel = '1' then
            c <= b;
        else
            c <= a;
        end if;
    end process;
end bhv;
```



VHDL

- Példa
  - 4-bites regiszter
  - ISE Project





VHDL

- Utasítások
  - Folyamat: `process`
    - Logikai feltételhez kötött végrehajtás: `case-when`

```
case kifejezés is
     when lehetséges_értéke_0 =>
       utasítások_0;
     when lehetséges_értéke_1 =>
       utasítások_1;
                  ...
     when others =>
       utasítások_n;
end case;
```

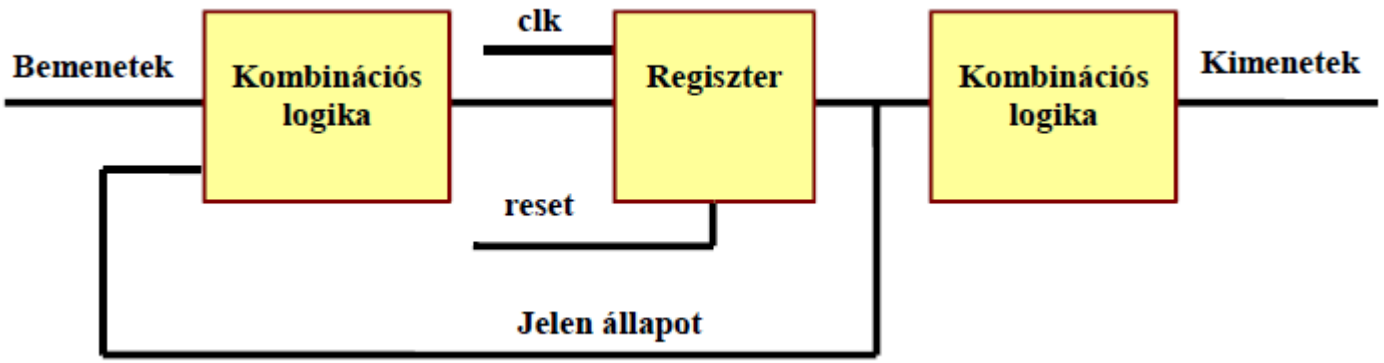
VHDL

- Utasítások
  - Folyamat: `process`
    - Logikai feltételhez kötött végrehajtás: `case-when`

```
type alu_func is (pass1, pass2, add, subtract);
…
alu_ proc: process (func)
…
begin
     case func is
          when pass1 =>
            result := operand1;
           when pass2 =>
            result := operand2;
          when add =>
            result := operand1 + operand2;
           when subtract =>
            result := operand1 - operand2;
          when others =>
            result := -1;
     end case;
end process alu_ proc;
```

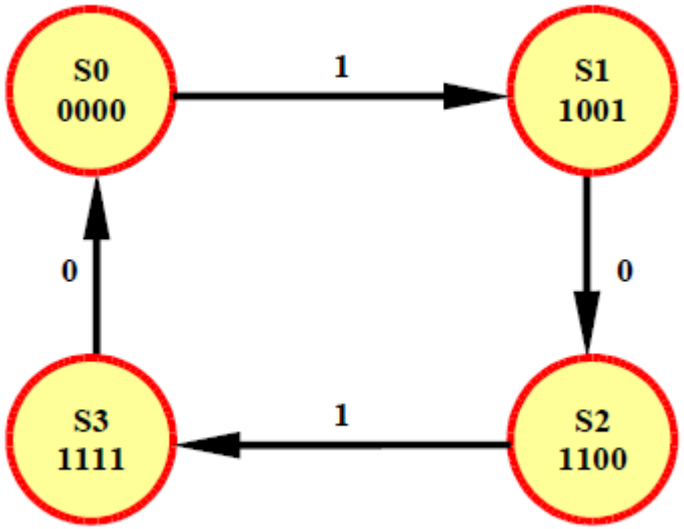
VHDL

- Utasítások
  - Folyamat: `process`
    - Logikai feltételhez kötött végrehajtás: `case-when`
      - Véges állapotú állapotgép (FSM) megvalósítása
      - Moore modell



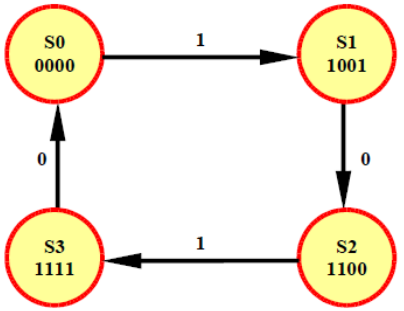
VHDL

- FSM példa
  - Legyen négy állapot
    - S0, S1, S2 és S3
    - A hozzájuk tartozó kimeneti értékek rendre: `"0000", "1001", "1100"` és `"1111",`



VHDL

- FSM példa

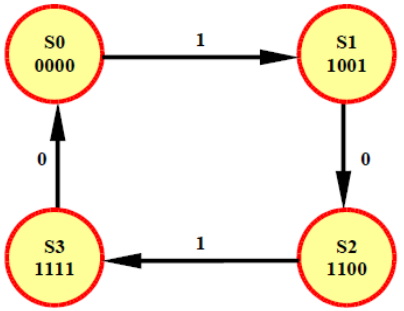



```
entity fsm is port(clk,in1,reset: in std_logic;
                   out1: out std_logic_vector(3 downto 0));
end fsm;

architecture moore of fsm is
  type stateType is (s0,s1,s2,s3); -- Állapot felsorolás
  signal state: stateType;

begin

StateUpdate: process(clk,reset) -- Órajeles folyamat
  begin
    if reset = '1' then state <= s0; -- Állapot törlése
    elsif clk'event and clk = '1' then
      case state is when s0 => if in1 = '1' then state <= s1; end if;
                    when s1 => if in1 = '0' then state <= s2; end if;
                    when s2 => if in1 = '1' then state <= s3; end if;
                    when s3 => if in1 = '0' then state <= s0; end if;
      end case;
    end if;
  end process;
…
```

VHDL

- FSM példa (folyt.)

```
…
  SetOut: process(state) -- Kombinációs folyamat
  begin
    case state is when s0 => out1 <= "0000";
                  when s1 => out1 <= "1001";
                  when s2 => out1 <= "1100";
                  when s3 => out1 <= "1111";
    end case;
  end process;
end moore;
```



VHDL

- Utasítások
  - Értékadás
    - Jeleknél <=
      - Csak konkurens részben lehet jeleket deklarálni
      - Konkurens és sorrendi végrehajtásnál is használják
    - Változóknál :=
      - Csak sorrendi részében lehet változókat deklarálni
      - A változókat folyamatban vagy alprogramban sorrendi végrehajtásnál használják

VHDL

- Utasítások
  - Értékadás
    - A fő különbség egy jel és egy változó között
      - A jelhez az értékét csak egy delta késleltetés után lehet hozzárendelni
      - A változó azonnal megkapja az értékét.

```
-- sum1 és sum2 jelek
P0: process
begin
     wait for 10 ns;
     sum1<=sum1+1;
     sum2<=sum1+1;
end process;
```

Idő	sum1	sum2
0 ns	0	0
10 ns	0	0
10 + Δ ns	1	1
20 ns	1	1
20 + Δ ns	2	2
30 ns	2	2
30 + Δ ns	3	3

```
-- sum1 és sum2 változók
P1: process
variable sum1, sum2: integer;
begin
     wait for 10 ns;
     sum1:=sum1+1;
     sum2:=sum1+1;
end process;
```

Idő	sum1	sum2
0 ns	0	0
10 ns	1	2
10 + Δ ns	1	2
20 ns	2	3
20 + Δ ns	2	3
30 ns	3	4
30 + Δ ns	3	4

VHDL

- Utasítások
  - Feltételes értékadás
    - Feltételes konkurens értékadás: `when-else`
    - Csak konkurens részben

```
cél_azonosító <= kifejezés_0 when feltétel else
                 kifejezés_1 when feltétel else
                   ...
                 kifejezés_n;
```

- Pl.:

```
z <= d0 when sel = "00" and en = '1' else
     d1 when sel = "01" and en = '1' else
     d2 when sel = "10" and en = '1' else
     d3 when sel = "11" and en = '1' else
    '0';
```

VHDL

- Utasítások
  - Feltételes értékadás
    - Feltételes konkurens értékadás: `with-select-when`
    - Csak konkurens részben

```
with kifejezés select
 cél_azonosító <= kifejezés_0 when lehetséges_értéke_0,
                  kifejezés_1 when lehetséges_értéke_1,
                   ...
                  kifejezés_n when others;
```

    - Az összes lehetséges választási értéket fel kell sorolni
    - Az összes maradék választására: `when others` szerkezet
    - Pl.:

```
with data select
 q <= a when "00",
      b when "11",
      c when others;
```

VHDL

- Utasítások
  - Ciklusok
    - Csak szekvenciális részben
    - `loop`

```
loop
   { szekvenciális_utasítások }
   [ cimke : ] exit [ loop_azonosító ] [ when logikai_kifejezés ] ;
end loop;
```

    - A legegyszerűbb ciklus
    - Ciklusból kilépés az `exit` utasítással
    - Csak szimulációban vagy viselkedés leírásban, Pl.:

```
loop
     count_value := (count_value + 1) mod 16;
     count <= count_value;
     exit when reset = '1';
end loop;
```

VHDL

- Utasítások
  - Ciklusok
    - Csak szekvenciális részben
    - `while`

```
while logikai_kifejezés loop
  { szekvenciális_utasítások }
end loop;
```

    - Csak szimulációban vagy viselkedés leírásban
    - Pl.:

```
process
begin
    while error_flag /= '1' loop
        Clock <= not Clock;
        wait for CLK_PERIOD/2;
    end loop;
end process;
```

VHDL

- Utasítások
  - Ciklusok
    - Csak szekvenciális részben
    - `for`

```
for azonosító in tartomány loop
  { szekvenciális_utasítások }
end loop;
```

    - Szimulációban, viselkedés leírásban

```
for i in 0 to 127 loop
    count_out <= count_out + 1;
    wait for 5 ns;
end loop;
```

    - Szintetizálható kódban is használható
      - Szintézis időben a ciklus kifejtésével
      - Flexibilis, jól olvasható VHDL kód

VHDL

- Utasítások
  - Ciklusok
    - `for`
    - Szintetizálható kódban is használható
      - Ha szintetizálni szeretnénk kifejthető kódot kell írni, Pl:

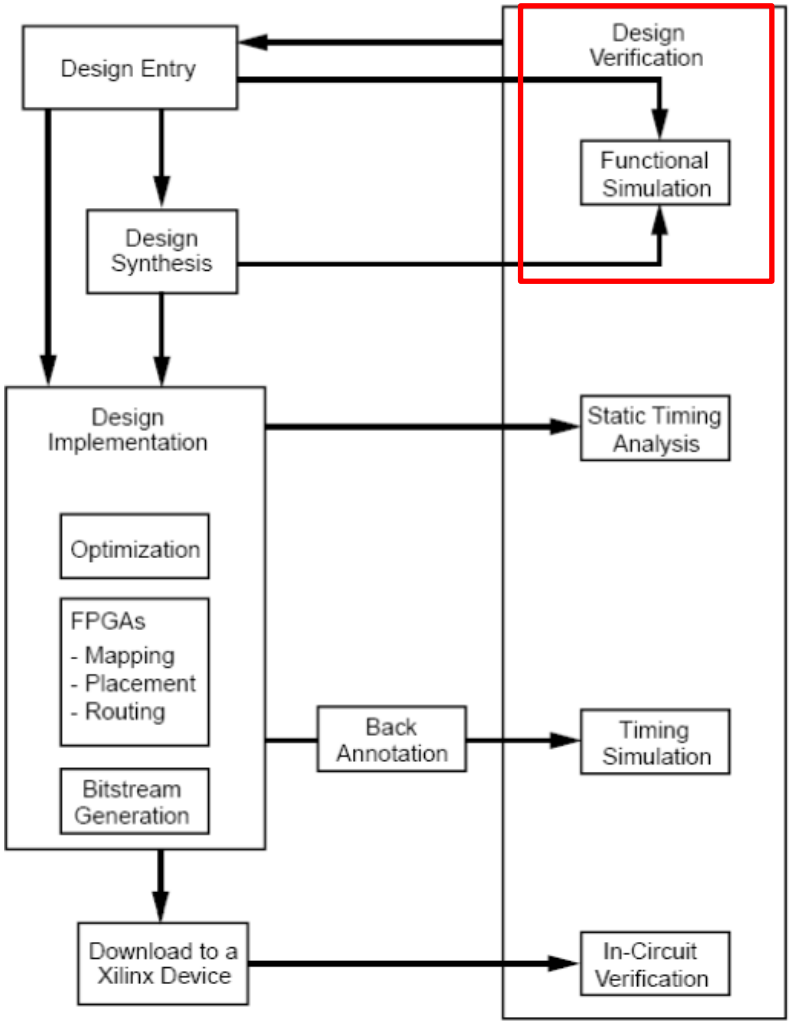
```
for i in 1 to 4 loop
  a(i) <= b(5-i);
end loop;
```

    - A szintézis folyamán a szintézer behelyettesíti a ciklusváltozót és kifejti a négy utasítást

```
a(1) <= b(5-1);
a(2) <= b(5-2);
a(3) <= b(5-3);
a(4) <= b(5-4);
```

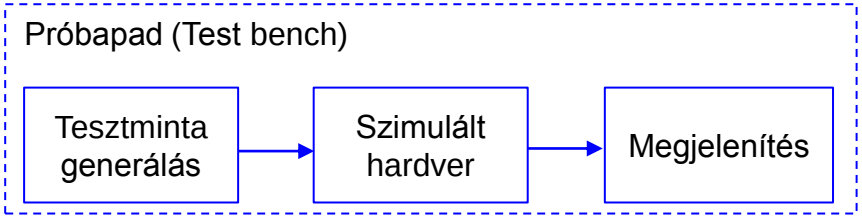
Digitális rendszer-tervezés

- Szimuláció



VHDL

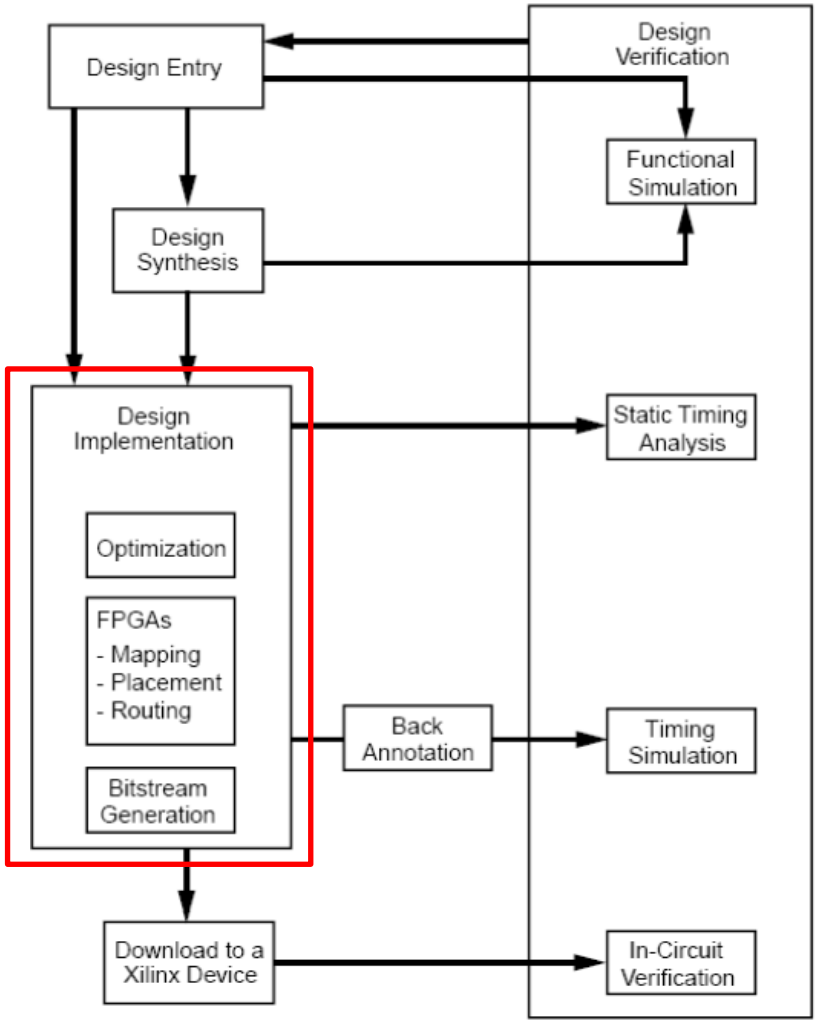
- Viselkedés szintű szimuláció
  - A VHDL modell jóságának igazolása
    - Próbapad (test bench)
      - A vizsgálandó modellből (szimulált HW)
      - És a gerjesztéseket létrehozó modellből (tesztminta generátor) áll



      - A tesztminta generátor kimeneti jeleivel gerjesztjük a vizsgált VHDL modellt
        - » UUT (Uinit Under Test)
        - » Ellenőrizhetjük a szimulált hardver kimeneti jeleit
        - » Hátrány: a próbapad VHDL modelljében is lehetnek hibák

Digitális rendszer-tervezés

- A tervezés folyamata
  - Xilinx ISE



Xilinx ISE Design Entry/Synthesis:

- Terv létrehozása
- Kapcsolási rajz alapon
- HDL alapon (hardverleíró nyelv)
- Egyéb forrásból (FSM,…)
- Kitételek/korlátozások megadása

Design Implementation:

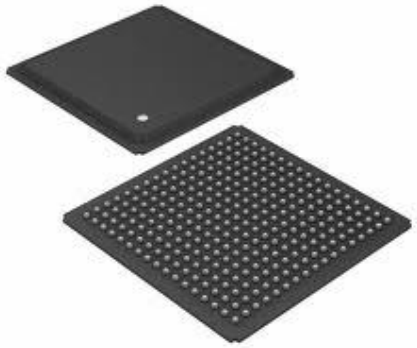
- A terv (logikai leírás) konvertálása fizikai információvá (konfiguráló bitfolyammá)
- Mapping(MAP): a terv adaptálása az adott eszközben, kitételek feldolgozása, tervezési szabályok ellenőrzése
- Placement/Routing(PAR): elemek elhelyezése, összekötések megvalósítása, optimalizálás

Design Verification:

- Az elkészült áramkör funkcionális és minőségi vizsgálata (szimuláció/in-circuit ellenőrzés)

A terv implementálása

- Felhasználói megkötések
  - A VHDL terv fizikai eszközbe történő lefordításának vezérlése
    - A fizikai kivezetések hozzárendelése
    - A fizikai be/kimeneti fokozatok vezérlése
      - Pl. Feszültségszintek, fel/lehúzó ellenállások
    - A időzítéssel, vezetékezéssel kapcsolatos megkötések, elvárások stb.
      - Pl. maximális jelkésleltetés, órajel hálózat használata
  - A tervhez tartozó szöveges UCF forrásfájlban



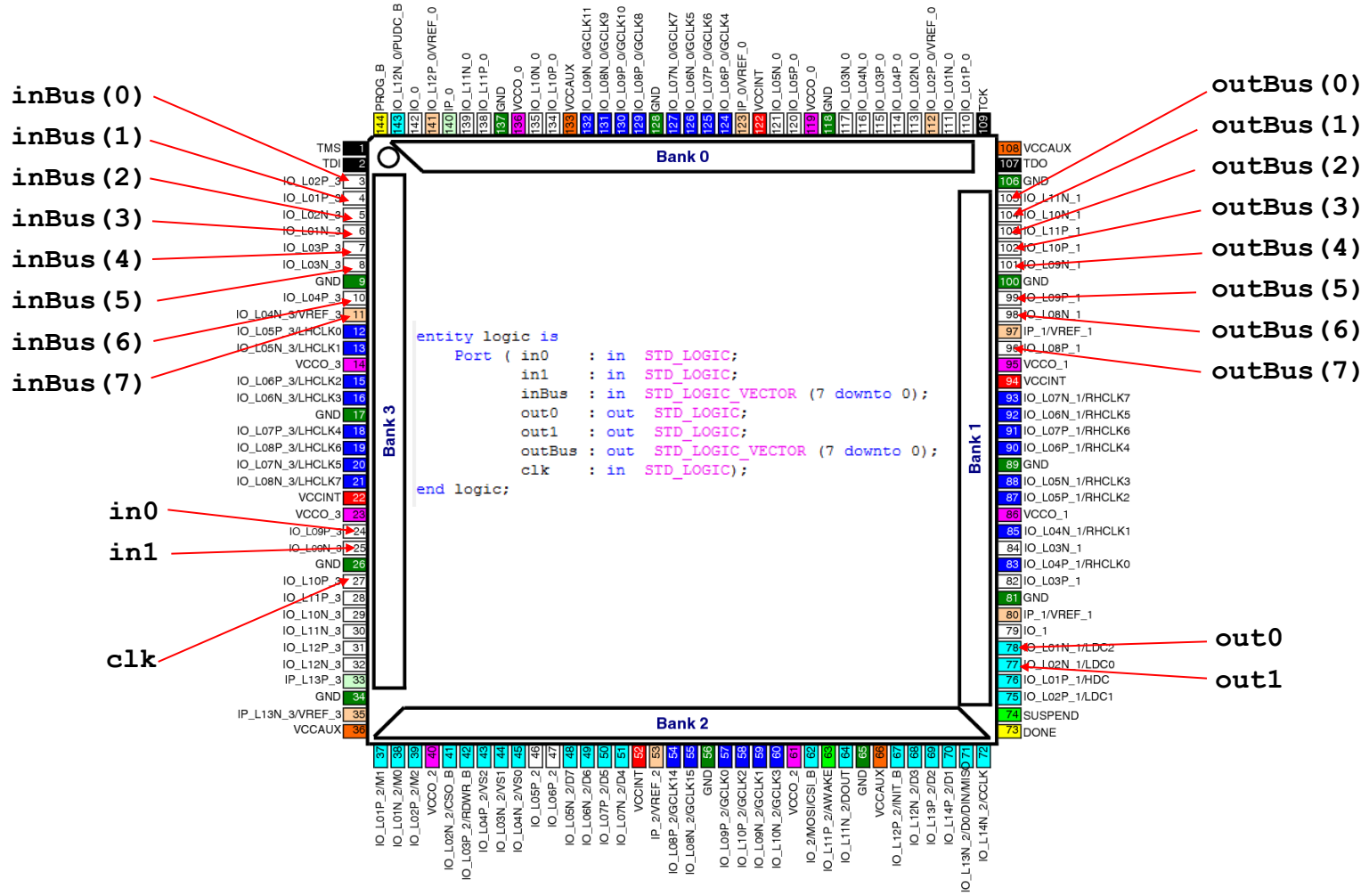
A terv implementálása

- Felhasználói megkötések
  - A fizikai kivezetések hozzárendelése

```
entity logic is
    Port ( in0    : in  STD_LOGIC;
           in1    : in  STD_LOGIC;
           inBus  : in  STD_LOGIC_VECTOR (7 downto 0);
           out0   : out  STD_LOGIC;
           out1   : out  STD_LOGIC;
           outBus : out  STD_LOGIC_VECTOR (7 downto 0);
           clk    : in  STD_LOGIC);
end logic;
```

A terv implementálása

- Felhasználói megkötések
  - A fizikai kivezetések hozzárendelése



A terv implementálása

- Felhasználói megkötések
  - A fizikai kivezetések hozzárendelése

```
# Megjegyzés
NET "in0" LOC = "P24";
NET "in1" LOC = "P25";
NET "inBus<0>" LOC = "P3";
NET "inBus<1>" LOC = "P4";
NET "inBus<2>" LOC = "P5";
NET "inBus<3>" LOC = "P6";
NET "inBus<4>" LOC = "P7";
NET "inBus<5>" LOC = "P8";
...
```

```
entity logic is
    Port ( in0    : in  STD_LOGIC;
           in1    : in  STD_LOGIC;
           inBus  : in  STD_LOGIC_VECTOR (7 downto 0);
           out0   : out  STD_LOGIC;
           out1   : out  STD_LOGIC;
           outBus : out  STD_LOGIC_VECTOR (7 downto 0);
           clk    : in  STD_LOGIC);
end logic;
```

A terv implementálása

- Mapping(MAP)
  - A terv adaptálása az adott eszközben, kitételek feldolgozása, tervezési szabályok ellenőrzése
  - A VHDL-ben definiált logikai feladatok szétosztása a programozható eszköz erőforrásai között
- Placement/Routing(PAR)
  - Elemek elhelyezése, összekötések megvalósítása, optimalizálás
- Konfigurációs bitfolyam előállítása
  - Többféle lehetőség
    - Közvetlenül az eszközbe tölthető bitfolyam
    - Konfigurációs memóriába tölthető bitfolyam
- Eszköz konfigurálás
  - Programozó készülékkel
  - Külső eszközzel (μP, egyéb intelligens eszköz)

A terv implementálása

- Eszköz konfigurálás

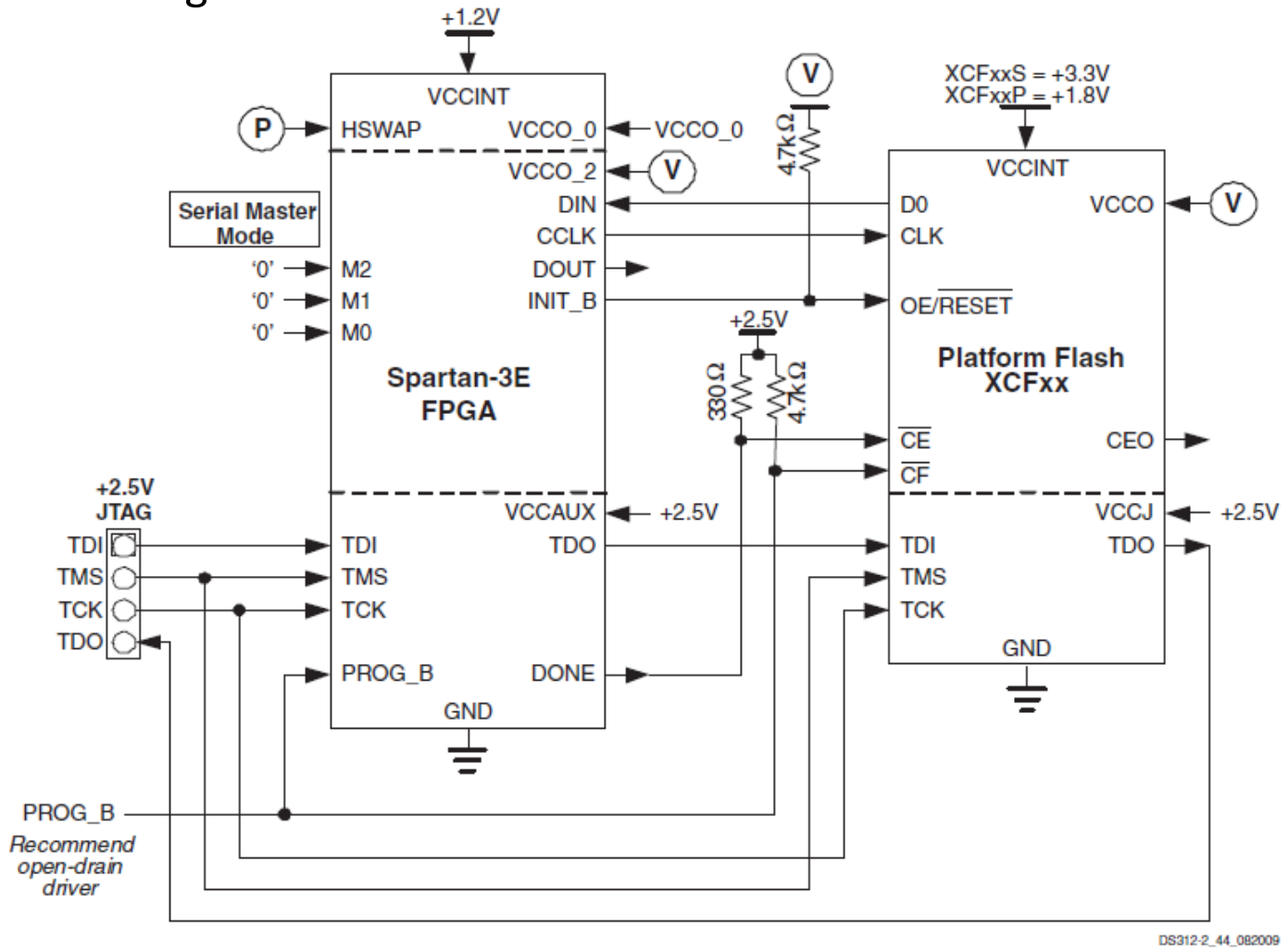


Figure 51: **Master Serial Mode using Platform Flash PROM**

A terv implementálása

- Eszköz konfigurálás

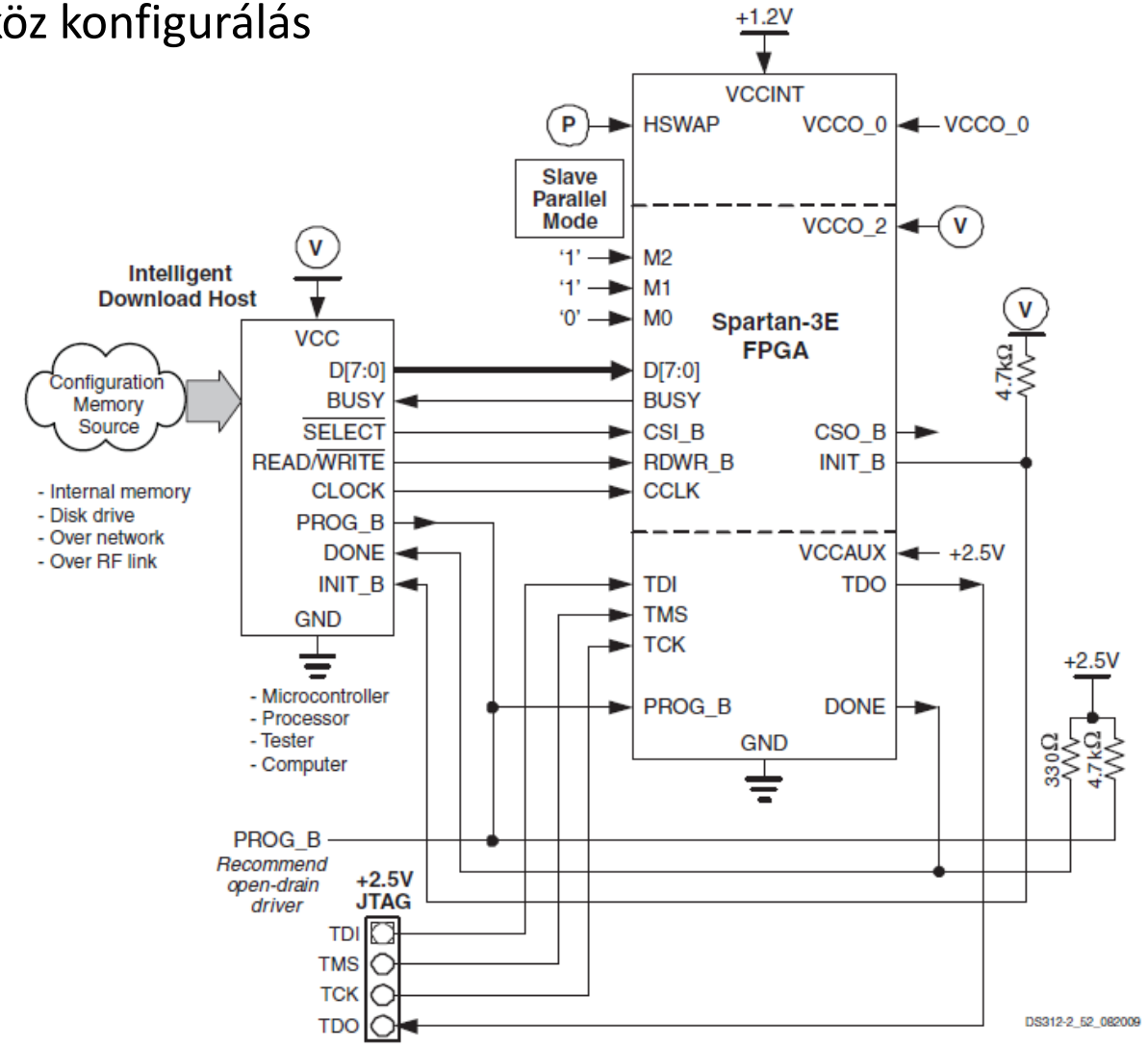


Figure 61: **Slave Parallel Configuration Mode**