Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Automatizálási és Kommunikáció-Technológiai Tanszék

2019/2020. tanév
I. félév

Programozható logikák c. tantárgy
Előadásának és gyakorlatainak ütemterve
BSC szintű villamosmérnök hallgatók részére.

Tárgynév:	**Programozható logikák**					
Rövid név:	Prog_Log			*Kód*	**GEVAU 518B**	
Angol név:	Programmable Logic					
Tanszék:	Villamosmérnöki Intézet, Automatizálási és Kommunikáció-technológiai Tanszék					
Tárgyfelelős:	Dr. Vásárhelyi József egyetemi docens, tel: (46) 565 111 /1753 vajo@mazsola.iit.uni-miskolc.hu					
Előtanulmányok:	nincs			*Kódja:*	GEVAU505B	
Kredit:			*Követelmény:*	kollokvium		
Heti óraszámok	*Előadás:*	2	*Gyakorlat:*		*Labor:*	2
Oktatási cél:	A villamosmérnöki ismeretek elsajátítása a szakirányú képzéseben					
Tárgy tartalom:	Digitális áramköri technológiák. Programozható logikák – a felhasználó által specifikált programozható eszközök csoportosítása. Egyszerű programozható logikai áramkörök. Egyszerű PLD áramkörök típusai: PAL, PLÁ, PLS, GAL áramkörök. Konfigurálható makró cellás PLD-k. CPLD eszközök ismertetése. FPGA áramköri architektúrák ismertetése, különös tekintettel az Altera és Xilinx típusokra. Programozható logikák tervezési környezetének ismertetése és megismerése. FPGA családok. Tervezési szempontok (C)PLD, FPGA áramköröknél. Programozható logikai áramkörök fejlesztőrendszerei. Hardver leíró nyelvek. VHDL nyelv modellezés és szimuláció, PLD tesztelés. Tervezési példák. Hierarchikus tervezési módszerek.					
Irodalom:	Gál Tibor: Programozható logikák, Műegyetemi Kiadó, MOKKAZ0004928232, 2002, pp.					
Ajánlott Irodalom	1. Scott Hauck, Andree Dehon ed. *Reconfigurable Computing The Theory and Practice of FPGA-Based Computation*, Elsvier, ISBN 978-0-12-370522-8, 2008, pp. 945 2. S. Brown, J. Rose: *Architecture of FPGAs and CPLDs: A Tutorial, http://www.freebookcentre.net/electronics-ebooks-download/Architecture-of-FPGAs-and-CPLDs-A-Tutorial-%28PDF-41p%29.html, pp. 41* 3. C. “Max” Maxfield**:** *The Design Warrior’s Guide to FPGAs, Elsvier,* ISBN: 0-7506-7604-3, http://www.google.hu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&ved=0CHIQFjAH&url=http%3A%2F%2Fprofs.basu.ac.ir%2Fabdoli%2Fupload_file%2F722.file_ref.2202.2686.pdf&ei=ClUEVJLeI6Ob0QWY8YB4&usg=AFQjCNF-Fa3vKZfNQE41oswi5b-0GfwBl9g, 2004, pp. 560					

Mintatantervi elhelyezkedés szakok szerint				
Szak	*Szakirány/sáv*	*Tantervi modul-tantervi kód*	*Mintatantervi félév*	*Választhatóság*
Villamosmérnöki Szak	minden	BV	1	BVE kötelező;

Jellemző oktatási módok				
Oktatási nyelv:	Magyar, angol			
Előadás:	Tábla + számítógépes vetítés			
Gyakorlat:				
Labor:	laboratórium gyakorlat + egyéni feladatokkal			
Évközi feladatok, zárthelyik:	1			
Lezárási feltételek:	A Tanulmányi és Vizsgaszabályzat szerint. Az Előadások legalább 60%-ának látogatása, a gyakorlatok legalább 75%-ának teljesítése. Gyakorlatokon aktív részvétel; az előírt feladatok teljesítése; a két évközi zárthelyi dolgozat eredményes megírása (legalább elégséges); az évközi (házi) feladatok elfogadható szintű elkészítése. A lezáráshoz írásbeli- és szóbeli vizsgát kell tenni. Az évközi teljesítmény 40%-a és az aláírás 60% összege a tárgyat lezáró jegy.			
Ütemterv				
36.	EA: Digitális áramköri technológiák. Programozható logikák – a felhasználó által specifikált programozható eszközök csoportosítása. Gyak: VHDL alapismeretek			
37.	EA:PLD (Programozható logikák) fejlesztő eszközei. Hardver leíró nyelvek. Gyak: A laboratóriumban használt fejlesztő rendszer megismerése.			
38.	EA:VHDL hardver leíró nyelv ismeretek. Modellezés és szimuláció. Gyak: VHDL ismeretek gyakorlása.			
39.	EA:Egyszerű programozható logikai áramkörök (SPLD). SPLD áramkörök általános architektúrája, típusok, Makrócellás PLD-k. Gyak: VHDL példák			
40.	Tervezési szempontok PLD áramköröknél. Időzítési modell. Állapotkódolás, termek számának csökkentése, tervezési szempontok PLD-s vezérlők esetén. Gyak: Egyéni feladat. beadási határidő: 45. hét			
41.	EA: FPGA áramkörök általános ismertetése. FPGA áramköri architektúrák, Logikai cellák, ki/bemenetei cellák, huzalozási erőforrások. Gyak: Egyéni feladat. beadási határidő: 45. hét			
42.	EA: Xilinx FPGA eszközök, Xilinx Spartan/Virtex családok ismertetése Gyak: Egyéni feladat. beadási határidő: 45. hét			
43.	EA: Altera FPGA áramkörök, Egyéb FPGA eszközök; Gyak: Egyéni feladat. beadási határidő: 45. hét			
44.	EA: Xilinx fejlesztői környezet: Vivado és ISE, Gyak: Egyéni feladat. beadási határidő: 45. hét			
45.	EA: Áramköri tesztelés, Hardver hurkos tesztelés (Hardver in the loop), JTAG, Virtuális szkóp használat Gyak: Egyéni feladat. beadási határidő: 49. hét			
46.	EA: Zárthelyi dolgozat Gyak: Egyéni feladat. beadási határidő: 49. hét			
47.	EA: Lágy magos és kemény magos processzorok (Szoft processzor, hard processzor) Gyak: Egyéni feladat. beadási határidő: 49. hét			
48.	EA: Rendszer a lapkán (System-on-chip -SOC) SOC technológiák Xilinx FPGA családok, Hálózatok a lapkán rendszerek (Network-on-Chip - NOC) Gyak: Egyéni feladat. beadási határidő: 49. hét			

49.	Többmagos rendszerek - Epiphany Gyak: Egyéni feladat. beadási határidő: 49. hét
50.	EA: ZH pótlás, Gyak: gyakorlatok pótlása

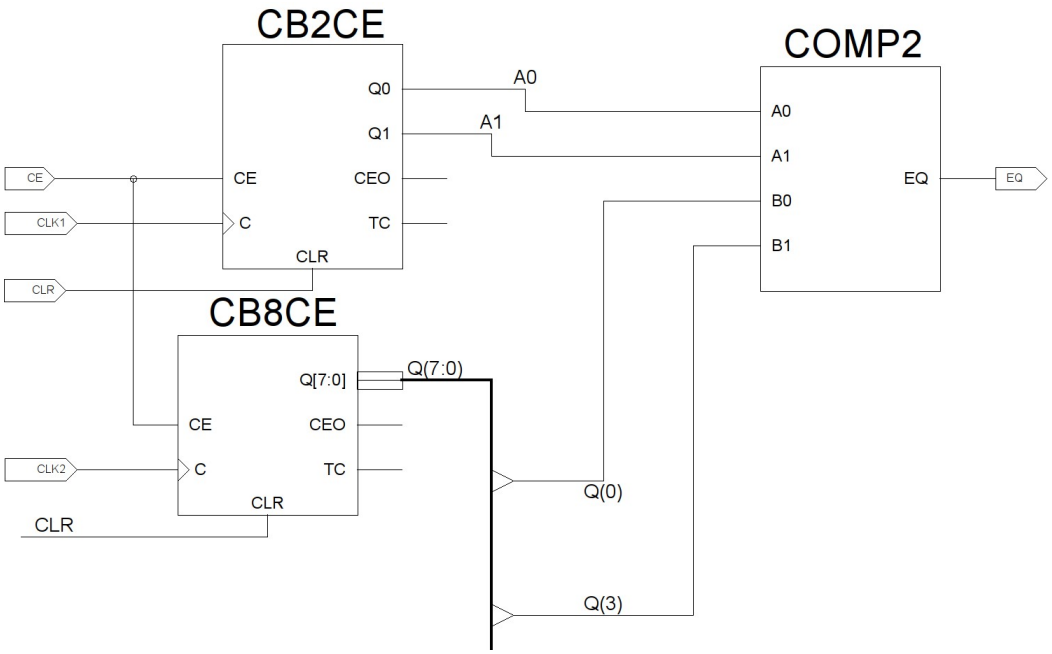
Miskolc, 2019. szeptember. 1.

Dr. Trohák Attila
tanszékvezető egyetemi docens

Dr. Vásárhelyi József
egyetemi docens

Programozható logikák GEVAU 518B
ZH Minta

Elemezze az ábrán látható kapcsolási rajzot. Valósítsa meg VHDL nyelvű programmal és végezze el az egyes elemek és az áramkör szimulációját, ha CLK1=10ns; CLK2=16ns;



Megoldás:
A Xilinx fejlesztőrendszerben megnézzük az egyes alkatrészek igazságtábláját és annak alapján, megírjuk a VHDL programot. Majd a szimulátorban vizsgáljuk a programot.
A ZH-t Xilinx fejlesztő rendszer segítségével oldjuk meg.

Főprogram:

```
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all; -- unsigned arithmetic
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ZH is
    Port ( CE : in  STD_LOGIC; -- KIMENETEK
           CLK1 : in  STD_LOGIC;
           CLK2 : in  STD_LOGIC;
           CLR : in  STD_LOGIC;
           EQ : out  STD_LOGIC);
end ZH;
```

```
architecture Behavioral of ZH is
-- cb2ce szamlalo alkatresz
component cb2ce
	port (clk : in std_logic;
				ce: in std_logic;
				clr: in std_logic;
				q: inout std_logic_vector ( 1 downto 0)
				);
end component;
-- cb8ce 8bites szamlalo alkatresz
component cb8ce
	port (clk : in std_logic;
				ce: in std_logic;
				clr: in std_logic;
				q: inout std_logic_vector ( 7 downto 0) -- 8bites
				);
end component;

-- komparator
component comp2
	port ( a: in std_logic_vector (1 downto 0);
				b: in std_logic_vector (1 downto 0);
				eq: out std_logic
				);
end component;
-- jelek:
signal q: std_logic_vector (7 downto 0);
signal a, b: std_logic_vector (1 downto 0);
signal op: std_logic_vector (1 downto 0);

begin

-- alkatrészek bekötése
sz2ce: cb2ce
	port map (
					clk => clk1,
					clr => clr,
					ce => ce,
					q  => a);
sz8ce: cb8ce
	port map (
					clk => clk2,
					clr => clr,
					ce => ce,
					q  => q);
komp: comp2
  port map(
				a => a,
				b(0) => q(0),
				b(1) => q(3),
				eq => eq);
```

```
end Behavioral;
```

2 bites számláló:

```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all; -- unsigned arithmetic

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity cb2ce is
	port (clk : in std_logic;
				ce: in std_logic;
				clr: in std_logic;
				q: inout std_logic_vector ( 1 downto 0)
				);
end cb2ce;

architecture Behavioral of cb2ce is

begin
	process (clk, clr)
begin
  if clr='1' then
    q <= (others => '0');
  elsif clk='1' and clk'event then
    if ce='1' then
      q <= q + 1;
    end if;
  end if;
end process;
end Behavioral;
```

8 bites számláló:

```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all; -- unsigned arithmetic

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
```

```
--use UNISIM.VComponents.all;

entity cb8ce is
        port (clk : in std_logic;
                                ce: in std_logic;
                                clr: in std_logic;
                                q: inout std_logic_vector ( 7 downto 0)
                                );
end cb8ce;

architecture Behavioral of cb8ce is

begin
        process (clk, clr)
begin
  if clr='1' then
    q <= (others => '0');
  elsif clk='1' and clk'event then
    if ce='1' then
      q <= q + 1;
    end if;
  end if;
end process;

end Behavioral;
```

Komparátor:

```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity comp2 is
   Port ( a : in  STD_LOGIC_VECTOR (1 downto 0);
          b : in  STD_LOGIC_VECTOR (1 downto 0);
          eq : out  STD_LOGIC);
end comp2;

architecture Behavioral of comp2 is

begin
                process(a, b)
begin
```

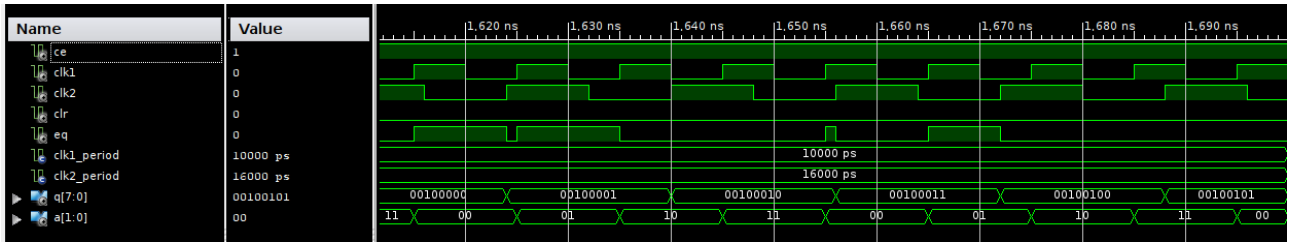
```
    if ( a = b) then
       eq <= '1';
    else
      eq <= '0';
    end if;

end process;



end Behavioral;
```

Szimulációs állomány/test bench:

```
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY zh_bench IS
END zh_bench;

ARCHITECTURE behavior OF zh_bench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT ZH
   PORT(
        CE : IN  std_logic;
        CLK1 : IN  std_logic;
        CLK2 : IN  std_logic;
        CLR : IN  std_logic;
        EQ : OUT  std_logic
       );
   END COMPONENT;
  --Inputs
  signal CE : std_logic := '0';
  signal CLK1 : std_logic := '0';
  signal CLK2 : std_logic := '0';
  signal CLR : std_logic := '0';
         --Outputs
  signal EQ : std_logic;

  -- Clock period definitions
  constant CLK1_period : time := 10 ns;  – orajel periodus az feladat adatai szerint
  constant CLK2_period : time := 16 ns;
BEGIN
         -- Instantiate the Unit Under Test (UUT)
```

```
  uut: ZH PORT MAP (
        CE => CE,
        CLK1 => CLK1,
        CLK2 => CLK2,
        CLR => CLR,
        EQ => EQ
      );
  -- Clock process definitions
  CLK1_process :process
  begin
                CLK1 <= '0';
                wait for CLK1_period/2;
                CLK1 <= '1';
                wait for CLK1_period/2;
  end process;
  CLK2_process :process
  begin
                CLK2 <= '0';
                wait for CLK2_period/2;
                CLK2 <= '1';
                wait for CLK2_period/2;
  end process;
  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    wait for CLK1_period*10;
    -- insert stimulus here
    wait;
  end process;
END;
```

Szimulációs eredmény:

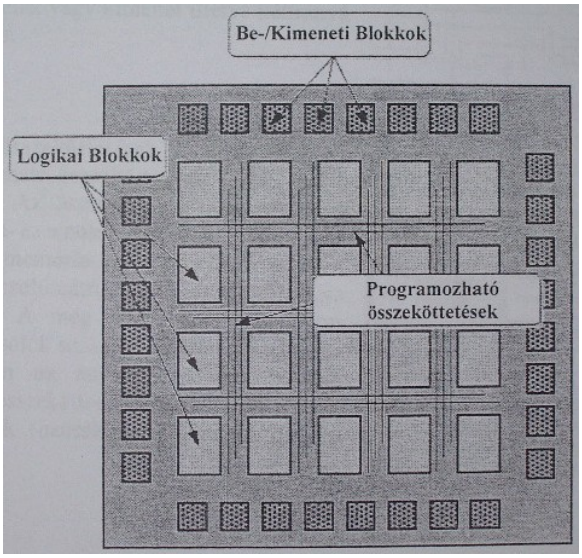


8. Szimmetrikus csatornájú FPGA eszközök jellemzése, Xilinx FPGA-k felépítése, programozása

8.1. A szimmetrikus FPGA eszközök általános felépítése

Az első FPGA eszközt, amely egyben szimmetrikus csatorna típusú eszköz volt, a Xilinx cég fejlesztette ki 1985-ben. A Xilinx FPGA-k elnevezése logikai cella tömb (Logic Cell Array = LCA). Három generációjuk, ennek megfelelően három családjuk van. Ezek az XC2000, XC3000 és az XC4000 család, melyek teljesítőképessége a nagyobb sorozatszámok irányába nő.

A szimmetrikus csatorna architektúrát használják, logikai blokkjaik igen összetettek, programozási technológiájuk a statikus RAM-ra épül. A chipen több, azonos felépítésű logikai struktúra, cella foglal helyet. A felhasználónak a kész logikai hálózat kialakításához mind a logikai cellákat, mind a cellák közti összeköttetéseket konfigurálni kell. A 8.1. ábrán a szimmetrikus FPGA-k egy leegyszerűsített, általános felépítéstét láthatjuk.



8.1. ábra

A szimmetrikus FPGA-kat alapvetően három fő egység építi fel:

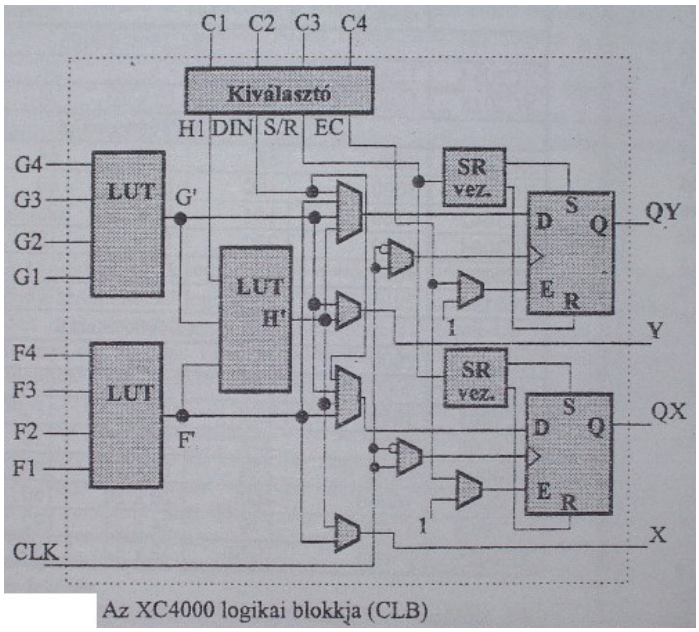
- konfigurálható logikai blokkok (CLB-k);
- be- és kimeneti blokkok (IOB-k);
- és programozható összeköttetések.

A Xilinx XC4000 FPGA család legfelsőbb szintű architektúrája megegyezik az FPGA-kra megadott általános modell architektúrájával, így csak az alsóbb szintekkel kell foglalkoznunk.

Ez a CLB-k és IOB-k felépítésének, az összekötési erőforrások megvalósításának és a programozás módjának bemutatását jelenti.

8.2. A CLB-k felépítése

Ez a konfigurálható logikai egység az áramkör központi alapeleme, melynek segítségével megvalósíthatjuk a kívánt logikai hálózatot. A CLB-k tartalmaznak FF-okat, független négy bemenetű funkciógenerátorokat és multiplexereket. A fejlesztői rendszer képes külön-külön használni a funkciógenerátorokat, így javul a CLB kihasználhatósága. A CLB használható latch-ként és élvezérelt tárolóként is. Az XC4000 család CLB moduljának felépítését a 8.2. ábrán láthatjuk.



8.2. ábra

Ez a CLB is LUT-al (függvénygenerátorral) valósítja meg a logikai funkciókat. Összesen három LUT-ot tartalmaz. A két elsődleges LUT (G és F) egy-egy négyváltozós függvény előállítására alkalmas. Ezek kimenete egy másodlagos, három bemenetű LUT-ba (H) csatlakozik. Ezáltal egy igen rugalmas függvénygenerátort kapunk. A CLB felépítése a ki- és bemenetek szempontjából teljesen szimmetrikus. Az ábrán trapéz alakú, konfigurálható multiplexerek jelölik ki az egyes elemek közötti kapcsolatokat.

Az ábra felső részében látható kiválasztó egység a négy bemenőjelből a következő jeleket állítja elő:

- EC (órajel engedélyezés), S/R (aszinkron beírás/törlés), DIN (adat bemenet) és H1 (a harmadik LUT járulékos bemenőjele), ha a memória funkció tiltva van;

- EC (órajel engedélyezés), WE (írás engedélyezés), D0-D1 (a memória adatbitjei), ha a memória funkció engedélyezve van.

A CLB két D típusú FF-ot tartalmaz. A két LUT és a két FF kimeneti jele egymástól függetlenül is megjelenhet a CLB kimenetén, ezért ez a CLB két kombinációs és két sorrendi funkció egyidejű megvalósítására is alkalmas.

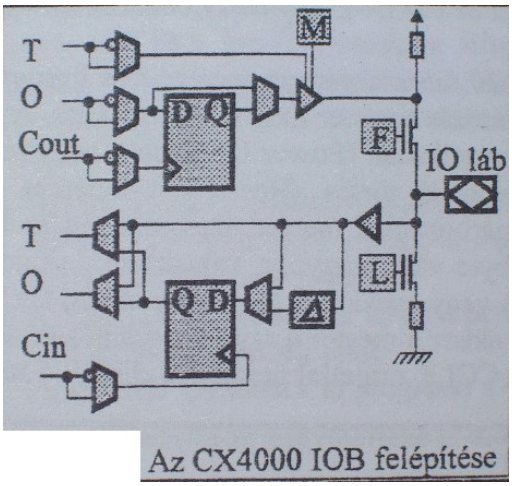
Az S/R vezérlő elemeken keresztül a két FF vagy aszinkron beállításra vagy aszinkron törlésre konfigurálható. Ezzel azt is meghatározhatjuk, hogy konfigurálás után az adott FF milyen kezdeti állapotot vegyen fel.

Arra is lehetőség van, hogy a LUT-okat a felhasználó aktuális (konfigurált) rendszere mint memóriákat írhassa, azaz azokat mint RAM memóriákat használhassa.

8.3. Az IOB-k felépítése

A chip szélein elhelyezkedő be- és kimeneti blokkok teremtik meg a kapcsolatot a tok kivezetései é a belső logika között. Mindegyik IOB konfigurálható bemenet vagy kimenet ill. kétirányú csatlakozásként is.

Az XC4000 család IOB-je sokkal összetettebb mint az egyszerűbb FPGA eszközöknél fellelhető IOB-k. Az üzemi funkciók mellett azokat az elemeket is tartalmazza, melyek a peremfigyeléses tesztelés elvégzéséhez szükségesek (8.3. ábra).



8.3. ábra

A kimenet lehet kombinációs és regiszteres is. A kimenőjelek már a kimeneti engedélyezés előtt beírhatók a regiszterbe, majd ettől függetlenül a kimeneti puffer egy másik jellel engedélyezhető. Ezzel csökkenthető az órajel és a kimenet közötti késleltetés káros hatása. A kimeneti meghajtó gyors vagy lassú felfutásúra is programozható. A bemenet szintén

kombinációs vagy regiszteres is lehet és regiszteres változatnál akár szint- akár élvezérelt üzemmódra konfigurálható.

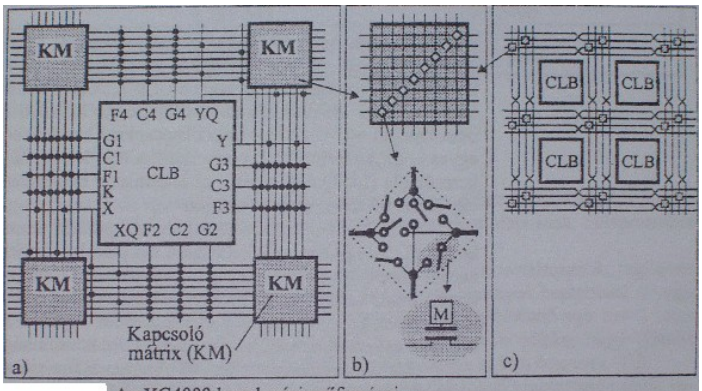
8.4. A huzalozási erőforrások

A programozási összeköttetések teszik lehetővé az egyes CLB-k és IOB-k megfelelő összekapcsolását. Az összeköttetéseknek relatív hosszuk szerint három típusa létezik, az egyszerű-, a dupla- és a hosszú vonalak. A függőleges és vízszintes vonalak kereszteződésénél a konfigurációs memória (SRAM) által vezérelt tranzisztoros kapcsolómátrix végzi el a szükséges összeköttetéseket. A még fel nem programozott eszközben a kapcsolók szakadás állapotban vannak. A tervezés során az automatikus huzalozónak köszönhetően az összeköttetések kialakításával nem kell foglalkoznunk.

A késleltetések csökkentése és az útválasztás lehetőségeinek növelése céljából az összeköttetések hierarchikus rendszerét valósították meg:

- általános célú, egy vagy két blokknyi távolságot áthidaló vezetékszakaszok, melyeket egyszeres és kétszeres hosszúságú szegmenseknek nevezünk (a szomszédos blokkok közötti gyors összeköttetéseket valósítják meg);
- a hosszú vonalak a chip teljes szélességét és hosszúságát áthidalják egyetlen kapcsoló közbeiktatása nélkül (a jelek nagy távolságokra való szétosztására használhatók);
- a globális hálózatok az órajelek, az időzítések szempontjából kritikus jelek és a sok bemenettel terhelt jelek számára lettek kialakítva.

A CLB-k és az egyszeres hosszúságú szegmensek kapcsolatát a 8.4. ábra szemlélteti.



Az XC4000 huzalozási erőforrásai
a) Az egyszeres szegmensek és a CLB kapcsolata
b) A kapcsoló mátrix szerkezete
c) A kétszeres szegmensek

8.4. ábra

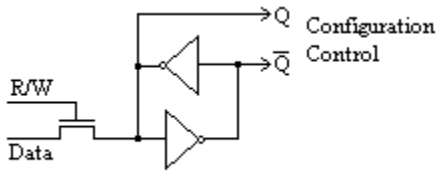
A kapcsolómátrixok programozható *n*-csatornás áteresztő tranzisztorokból épülnek fel. Két vezeték metszéspontjában hat áteresztő tranzisztort helyeznek el. Az alkalmazott kapcsolómátrix által megvalósítható összeköttetések száma kisebb, mint a korábbi generációs FPGA-kban. Ez előnyös abból a szempontból, hogy a kapacitív hatások kisebbek lesznek, s így a késleltetések is csökkenni fognak.

8.5. A Xilinx FPGA-k programozása

Az FPGA áramkörök esetében a konfiguráció (CLB-k, IOB-k és kötések állapotai) statikus RAM-ban (SRAM) tárolódik. A konfigurációs memória feltöltése történhet sorosan vagy párhuzamosan, ill. Master vagy Slave üzemmódban.

Master üzemmódban az FPGA feltölti magát, azaz beolvassa a hozzákapcsolt PROM tartalmaát. Slave üzemmódban egy külső eszköz ütemjele vezérli a beolvasást. Lehetőség van az eszköz JTAG porton keresztül történő konfigurálására és tesztelésére is. Ebben az esetben a letöltött anyag a tápfeszültség meglétéig marad meg az FPGA-ban.

Az eszköz programozása tulajdonképpen azt jelenti, hogy a megfelelő adatokat (0/1) a CLB-k, IOB-k és a huzalozás statikus RAM celláiba töltjük. Az XC4000 családnál átlagosan 300 bitet igényel egy CLB konfigurálása a hozzá tartózó huzalozással együtt. Az előbb említett CMOS RAM cella felépítése a 8.5. ábrán látható.

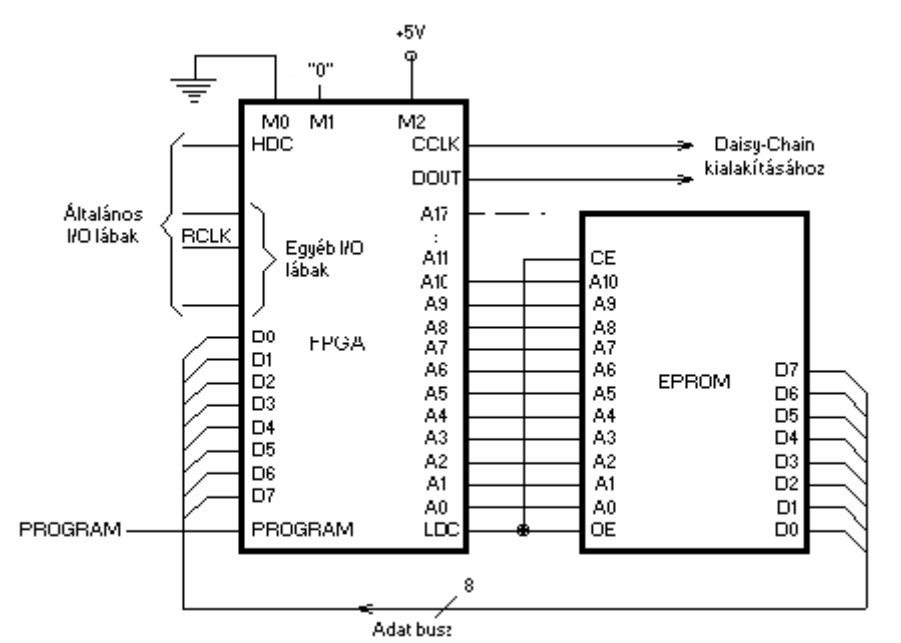


8.5. ábra

A XC4000-es felprogramozása a következő módokon történhet:

- mester soros üzemmód;
- szolga soros üzemmód;
- mester párhuzamos soros üzemmód (bájtos fel illetve le);
- aszinkron periféria soros üzemmód.

A konfigurációs program betöltése történhet az FPGA belső memóriájából, illetve külső memóriából is. A betöltés módját három konfigurációs bemenet segítségével lehet beállítani ezek az M0, M1, illetve az M2 lábak. Egy programozási módot láthatunk a 8.6. ábrán.



8.6. ábra

Összefoglalva: az FPGA-k olyan programozható logikai eszközök, melyek statikus RAM-mal rendelkeznek, így a tápfeszültség elvétele után elveszítik tartalmukat. Indításkor a konfigurációs állományt tartalmazó Boot PROM-ból állnak fel, mely további lehetőségeket kínál az eszköz rugalmasságára nézve.