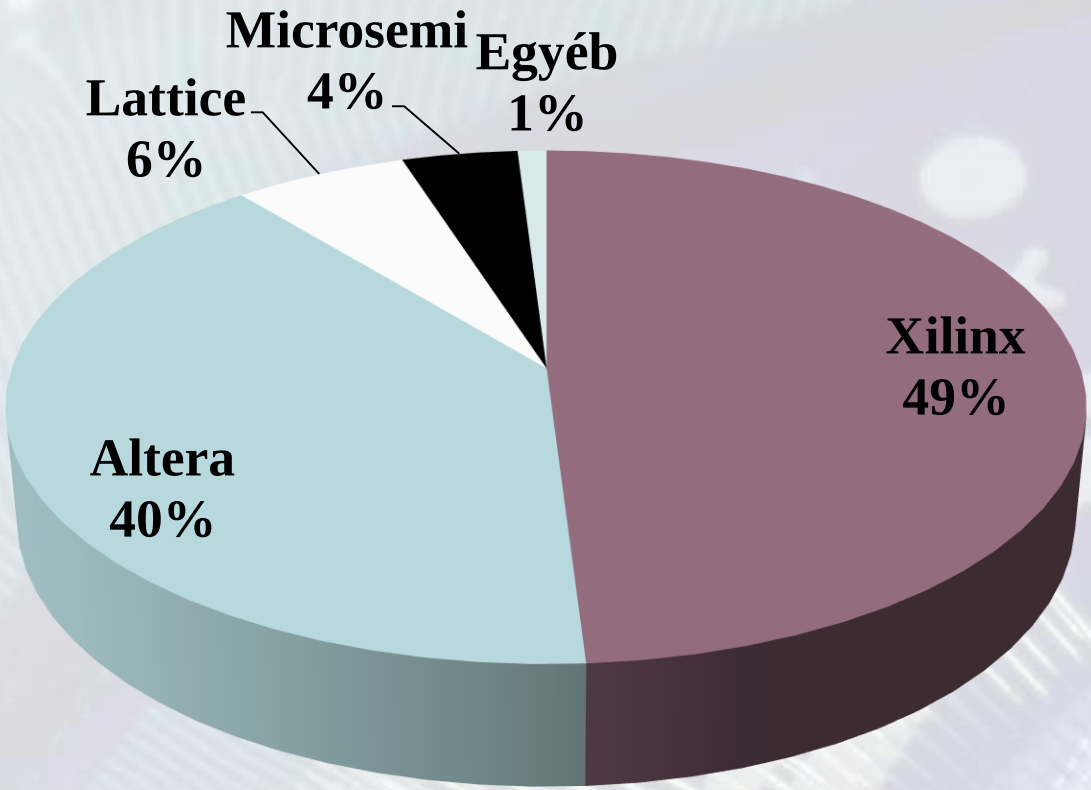
Xilinx 7-es sorozatú FPGA-k architektúrája

FPGA gyártók részesedése



- **Altera: Intel felvásárolta**
- **Microsemi: Actel felvásárlás**
- **Lattice: saját + SiliconBlue felvásárlás**
- **Megemlítendő még: Achronix**

Xilinx FPGA architektúra

- **Az alap építőelemek minden Xilinx FPGA-ban megegyeznek**
  - Logikai erőforrások
    - Slice (CLB-be (configurable logic block) szervezve)
    - Kombinációs logikát és tárolót tartalmaz
  - Memória
  - Szorzók (DSP blokkok)
  - Programozható összeköttetés
  - IO cellák
    - Interfész az FPGA belső erőforrásai és a külvilág között
  - Órajel bufferek
  - JTAG (Boundary scan) logika

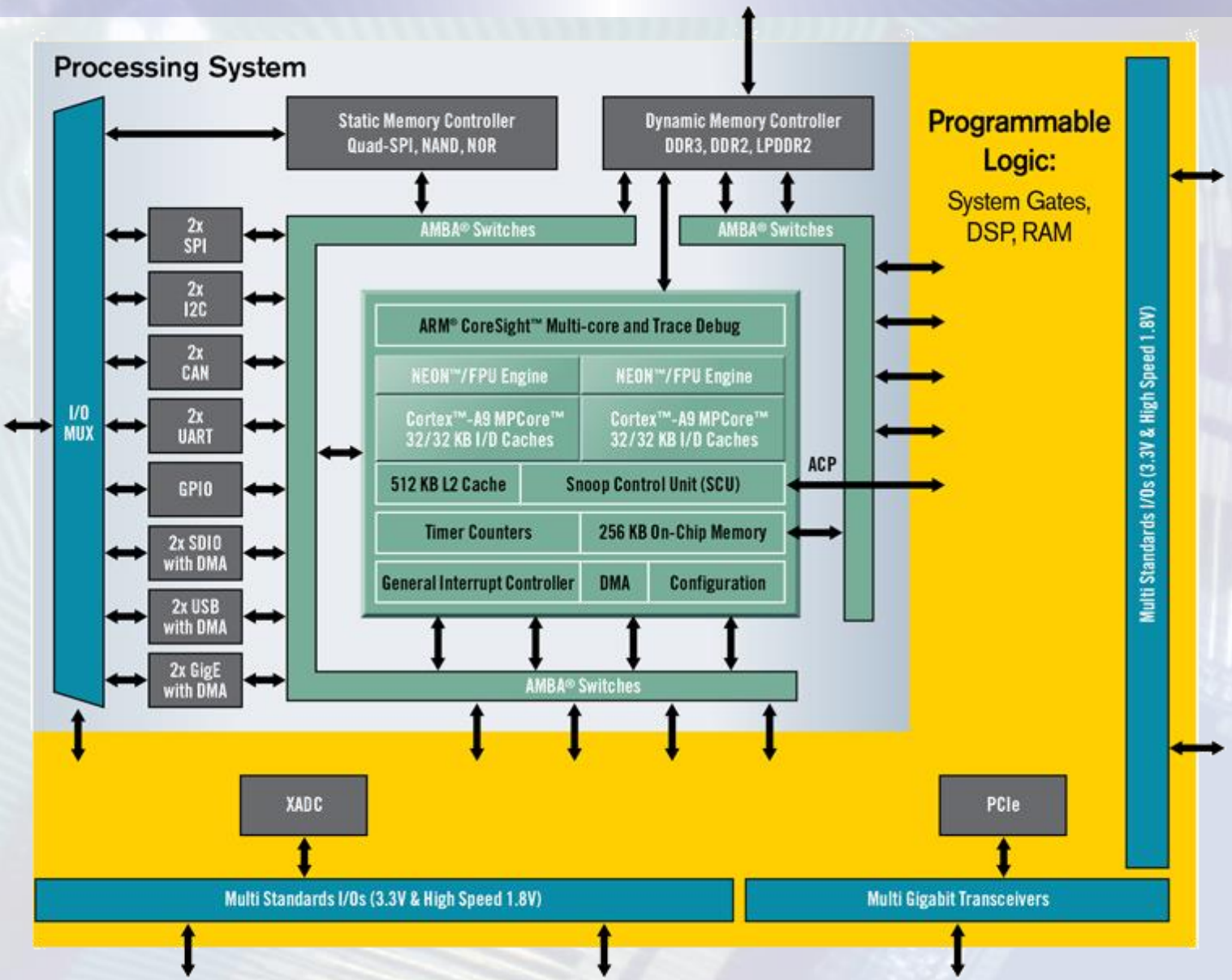
Xilinx FPGA családok (1.)

	Spartan-3	Spartan-6	Virtex-6	Artix-7	Kintex-7	Virtex-7
Logikai cella	33k	150k	760k	215k	480k	2000k
BRAM	0,65 Mb	4,8 Mb	38 Mb	13 Mb	34 Mb	68 Mb
DSP	36	180	2016	740	1920	3600
GMAC/s*	7,2	140	2419	929	2845	5335
Transciever	-	8	72	16	32	96
Tr. Bit rate	-	3,2 Gb/s	11,18 Gb/s	6.6 Gb/s	12.5 Gb/s	28 Gb/s
Serdes	-	1080 Mb/s	1400 Mb/s	1250 Mb/s	1600 Mb/s	1600 Mb/s
PCIe	-	1.0 x1	2.0 x8	2.0 x4	2.0 x8	3.0 x8
Mem. vezérlő	166 MT/s	800 MT/s	1066 MT/s	1066 MT/s	1866 MT/s	1866 MT/s
Ár (USD)	~70	~240	~14.000	380	2.000-3.000	~24.000

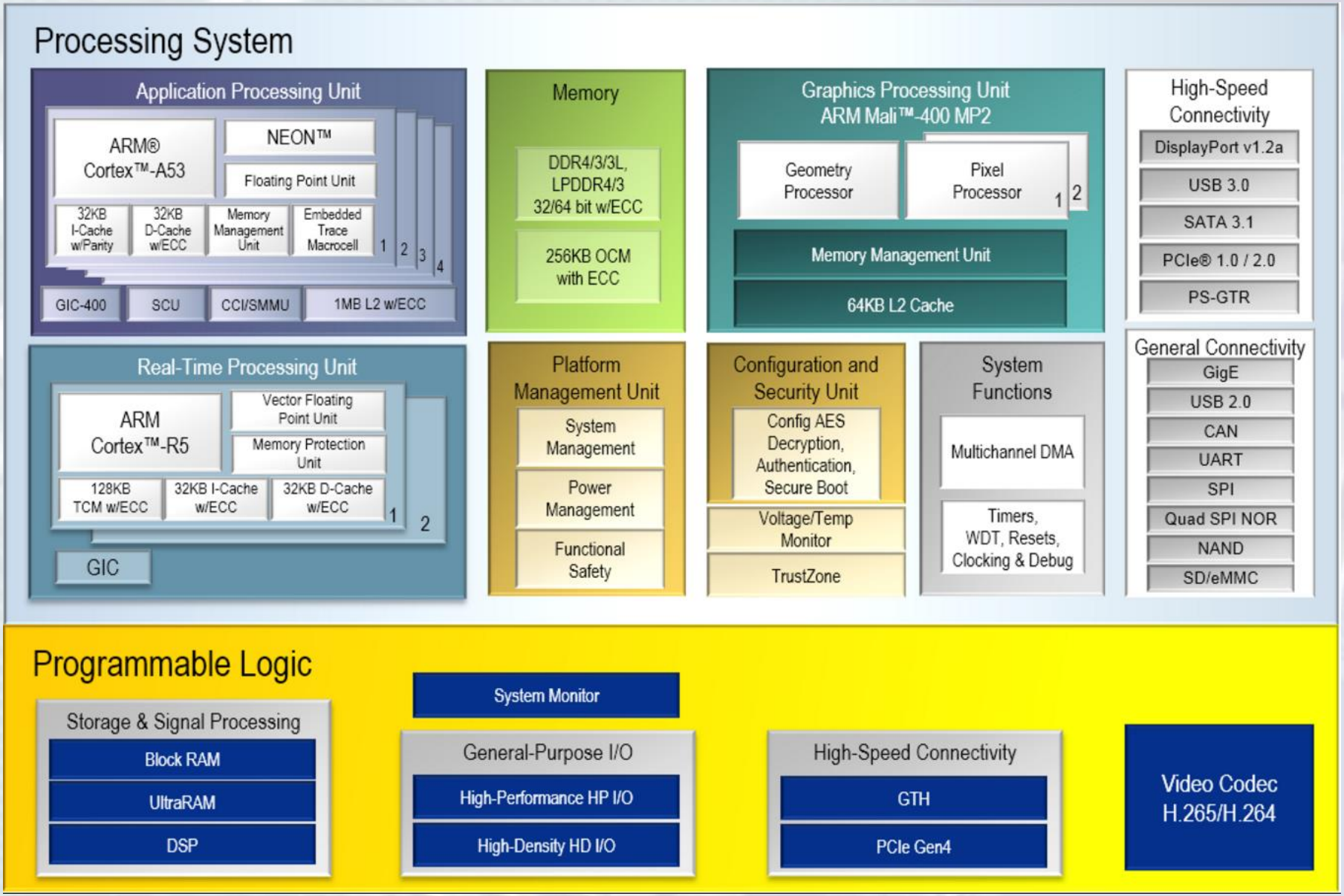
Xilinx FPGA családok (2.)

	Kintex UltraScale	Virtex UltraScale	Kintex UltraScale+	Virtex UltraScale+	Virtex UltraScale+
Logikai cella	1,4M	5,5M	1,14M	3,78M	2,85M
BRAM	75,9 Mb	88,6Mb	34,6Mb	94,5+360	70+270
DSP	5520	2880	1968 (3528)	12.288	9024
GMAC/s*	4000	2100	1750/3140	10.936	8000
Transciever	64	60 60	44 32	128	96
Tr. Bit rate	16,3 Gb/s	16,3 Gb/s 30,5 Gb/s	16,3 Gb/s 32,75 Gb/s	32,75 Gb/s	32,75 Gb/s
Serdes	1600 Mb/s	1600 Mb/s	1600 Mb/s	1600 Mb/s	1600 Mb/s
PCIe	6 Gen3 x8	6 Gen3 x8	5 Gen3 x16 Gen4 x8	4 Gen3 x16 Gen4 x8	6 Gen3 x16 Gen4 x8
Mem. Vezérlő	2400 MT/s	2400 MT/s	2666 MT/s	2666 MT/s	8 GB HBM2
Ár (USD)	~9000	~55.000	~6000	~40.000	~50.000

Xilinx Zynq SOC



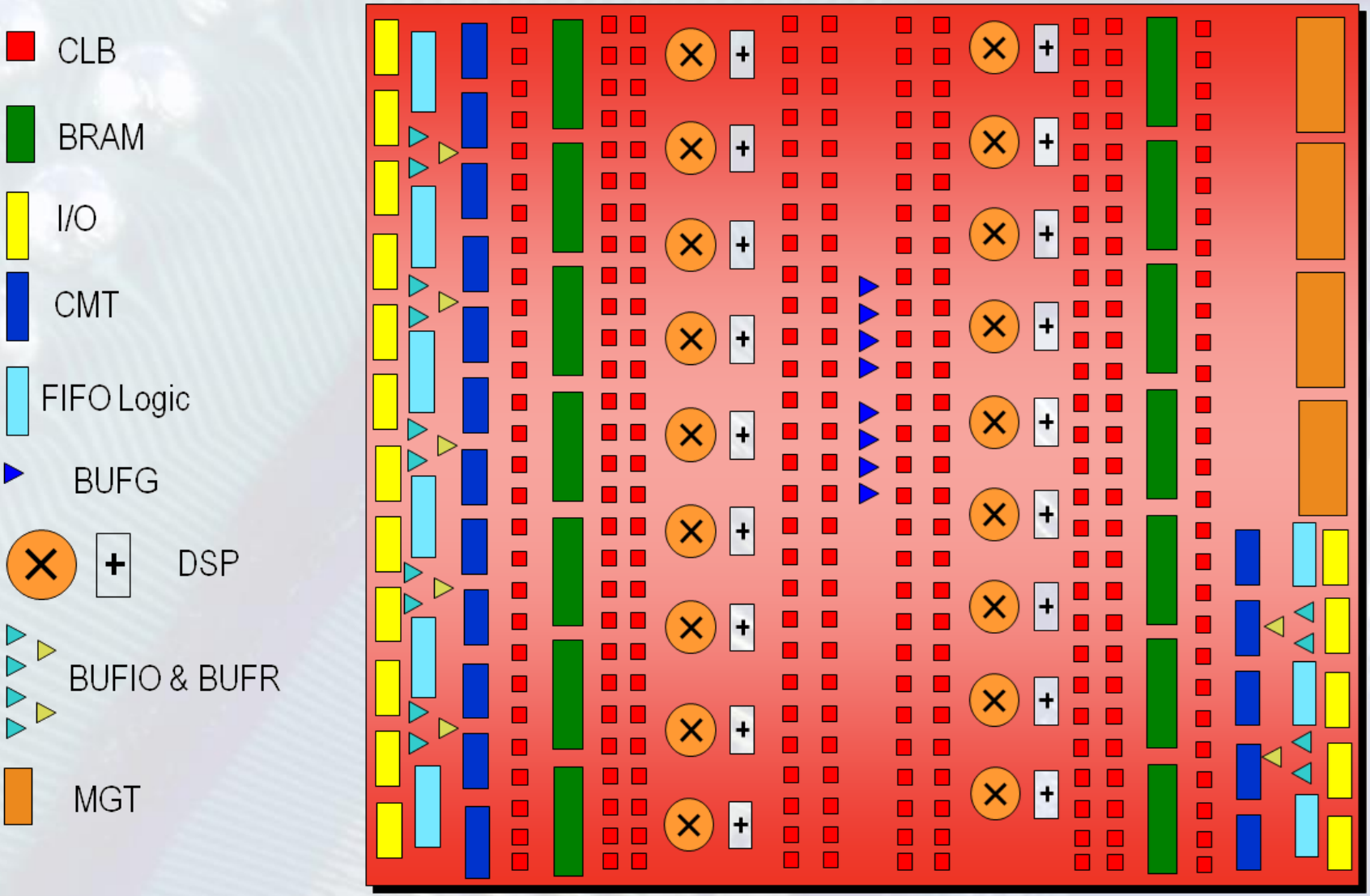
Xilinx UltraScale+ MPSOC



Xilinx 7-es család

	Artix-7	Kintex-7	Virtex-7	Zynq
Logikai cella (k)	33 - 215	66 – 478	583 – 1139	28 – 444
BRAM (Mb)	2 – 12	4 – 34	28 – 68	2 – 27
DSP	90 – 740	240 – 1920	1260 – 3360	80 – 2020
GMAC/s	929	2.845	5.335	2.622
Transceiver (max)	16	32	88	16
Tr. Gb/s	6,6	12,5	12,5; 13,1; 28	6,6; 12,5
Külső memória Mb/s	1066	1866	1866	1333
I/O lábak	106 – 500	285 - 500	350 – 1.100	54 – 400
I/O feszültségek	<=3,3V	<=3,3V <=1,8V	<=3,3V <=1,8V	<=3,3V <=1,8V

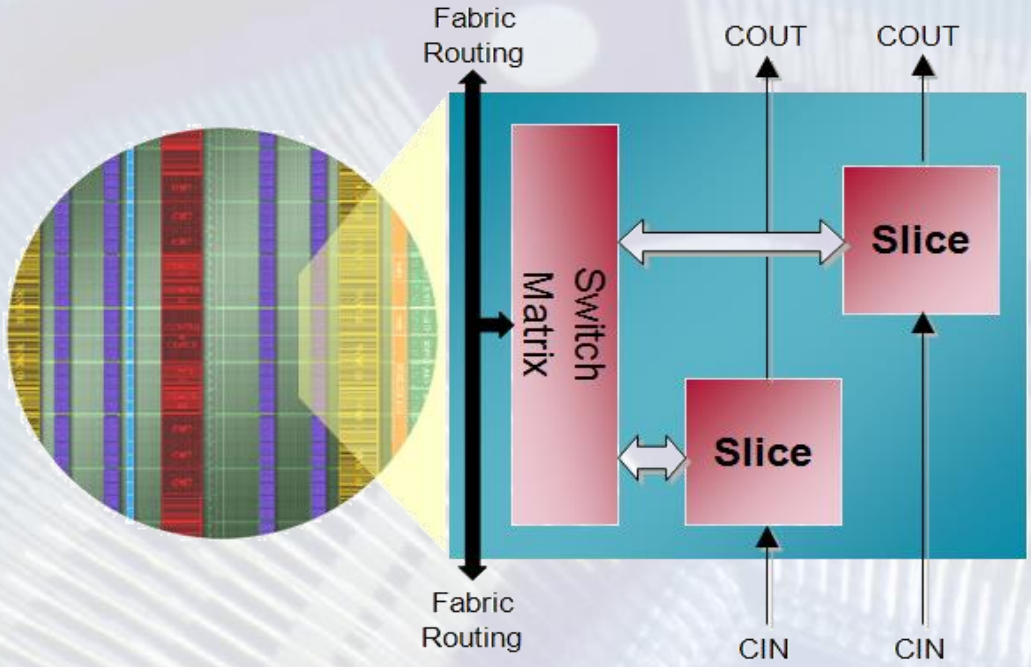
Architektúra



CLB

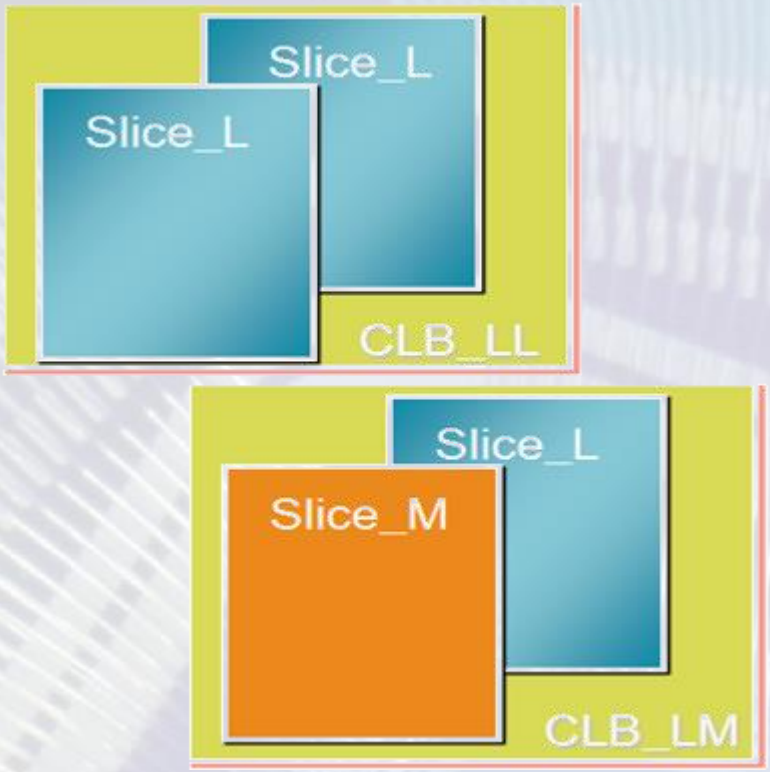
Xilinx CLB

- **Az FPGA elsődleges erőforrása**
  - Kombinációs logika
  - Tárolók
- **Minden CLB két SLICE-t tartalmaz**
- **A switch matrix-hoz kapcsolódik, amin keresztül összeköthető az FPGA többi erőforrásával**
  - Dedikált vertikális huzalozás a carry logikának



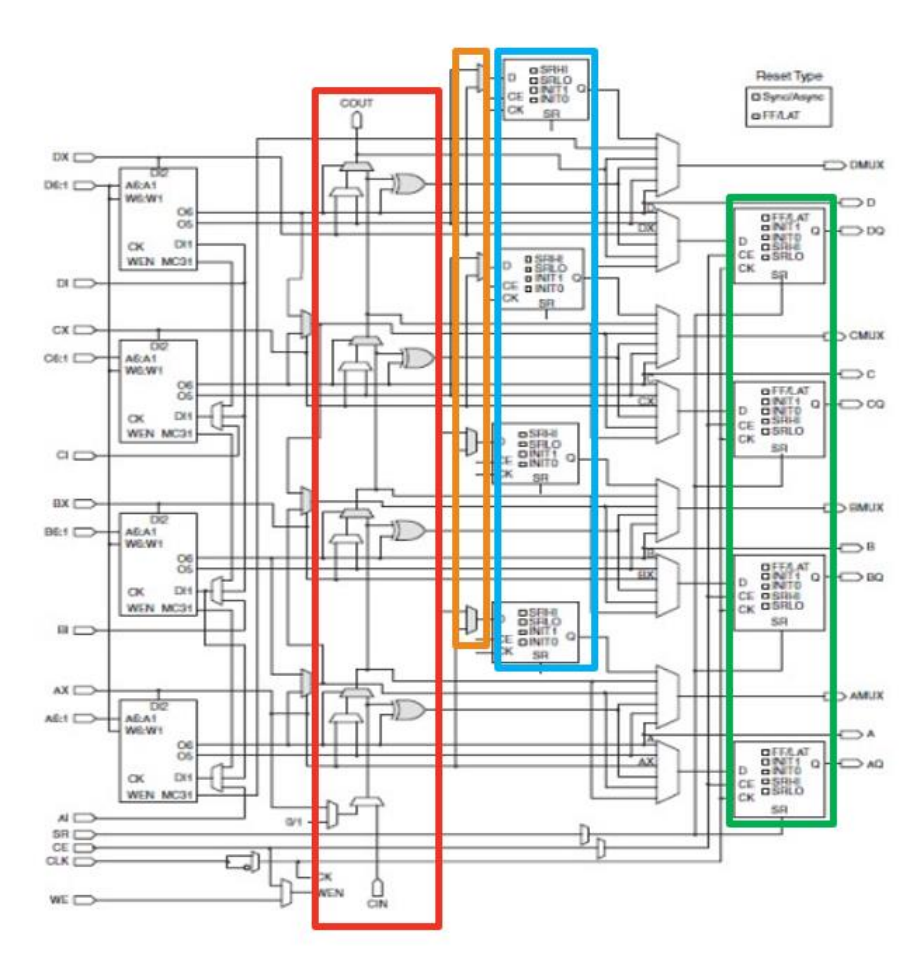
CLB típusok

- **Két féle slice/CLB**
  - SLICEM: Teljes funkcionalitás
    - LUT: logika, shift regiszter, memória
    - Tartalmaz dedikált multiplexert és carry logikát
  - SLICEL: Logic and arithmetic only
    - LUT: CSAK LOGIKA!
    - Tartalmaz dedikált multiplexert és carry logikát



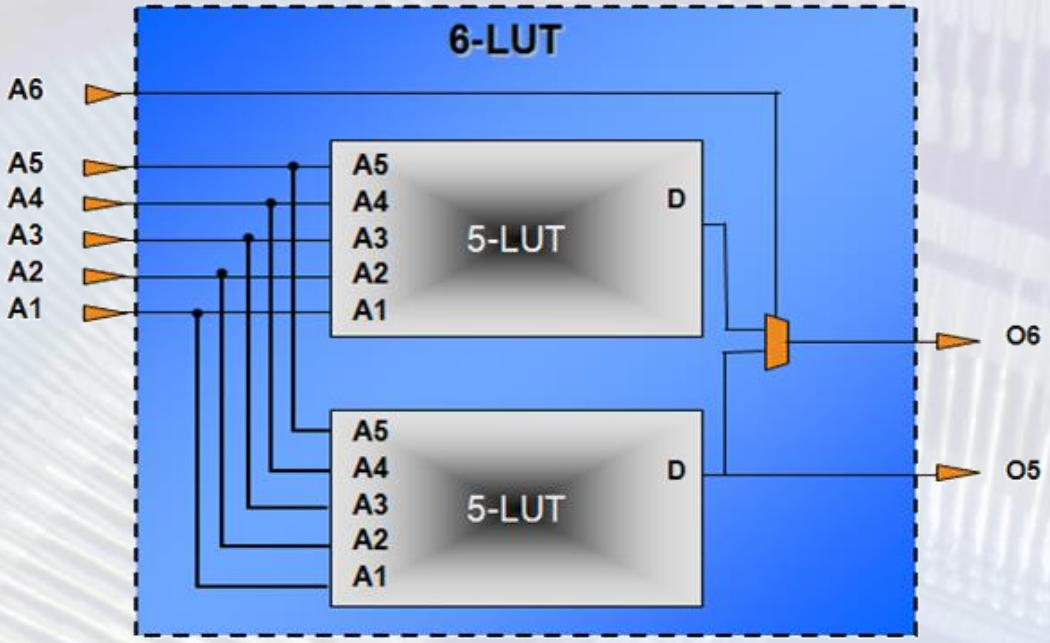
Slice erőforrások

- **4 db 6-bemenetű Look-Up Table (LUT)**
- **Multiplexerek**
- **Carry logika**
- **Shift regiszter (SRL)**
  - (Kaszkád vonalak nincsenek feltüntetve)
- **4 db flip-flop/latch**
- **4 db flip-flop**



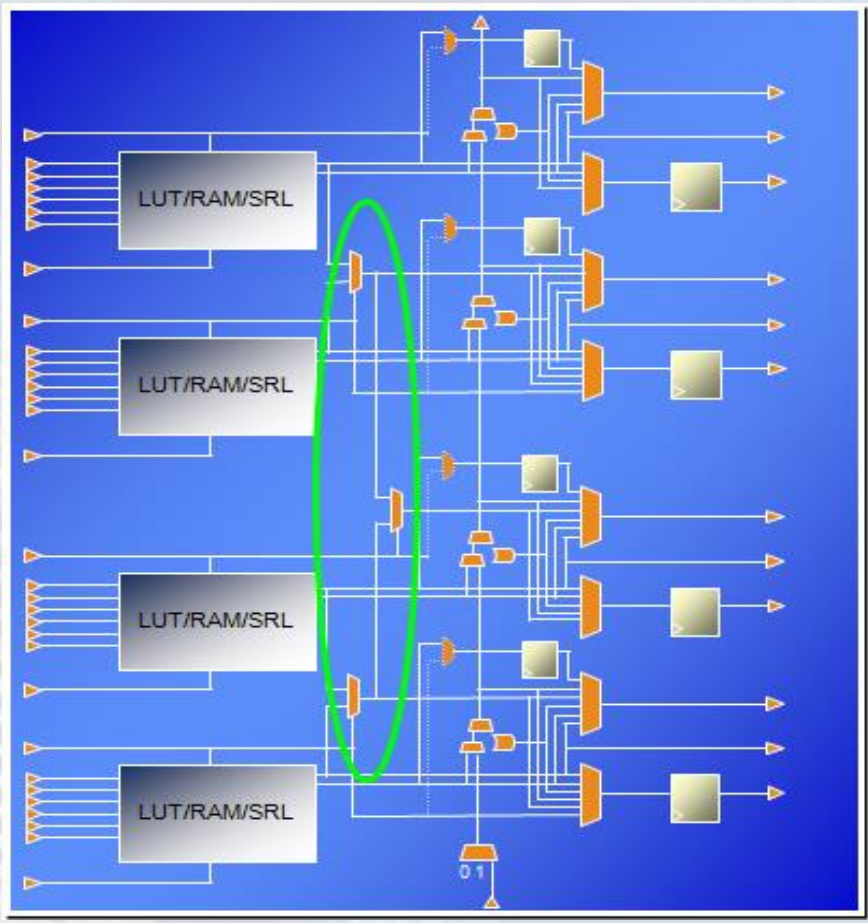
6-bementű, 2-kimentű LUT

- **Bármilyen 6-bemenetű logikai függvény megvalósítható**
- **Használható 2 db 5-bemenetű LUT-ként**
  - A bemenetek közösek
  - A logikai függvények különbözőek



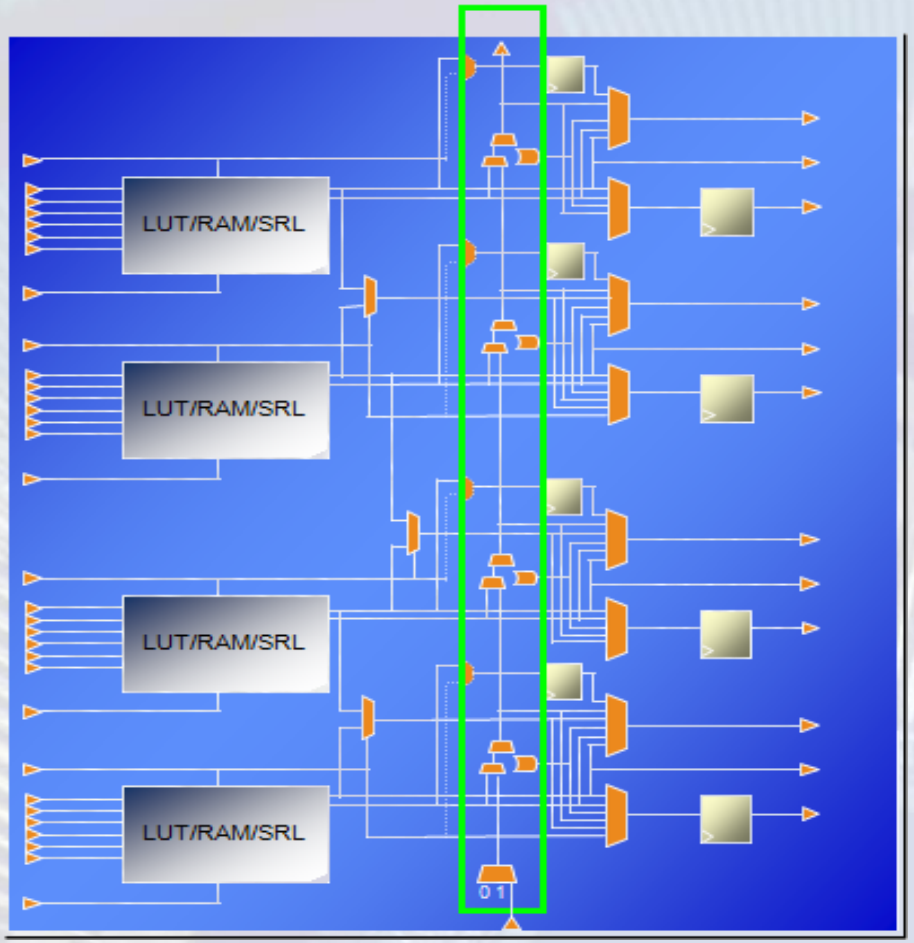
Multiplexerek

- **Az F7MUX két LUT kimenete közül tud választani**
  - Megvalósítható tetszőleges 7-bemenetű logikai függvény
  - Megvalósítható 8:1 Multiplexer
- **Az F8MUX két F7MUX kimenete között tud választani**
  - Megvalósítható tetszőleges 8-bemenetű logikai függvény
  - Megvalósítható 16:1 Multiplexer
- **A MUX-okat a slice BX/CX/DX bemenete vezérli**
- **A MUX-ok kimenete kivezethető kombinációsan, vagy tárolón (flip-flop/latch) keresztül**



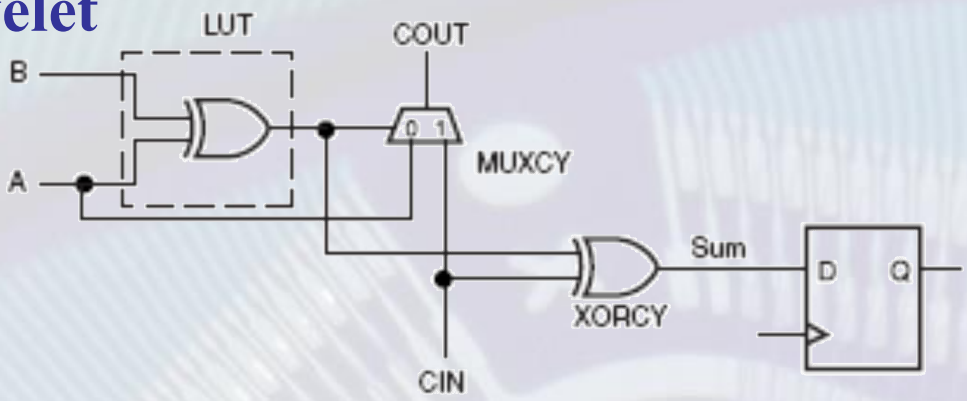
Carry hálózat

- **Dedikált carry logika az összeadás és kivonás gyors elvégzéséhez**
  - A carry függőlegesen terjesztett a slice 4 LUT-ja között
  - Az egy oszlopban levő CLB-k között dedikált huzalozás
- **Carry look-ahead**
  - Minden LUT-hoz carry-lookahead logika
  - Gyors carry terjesztés

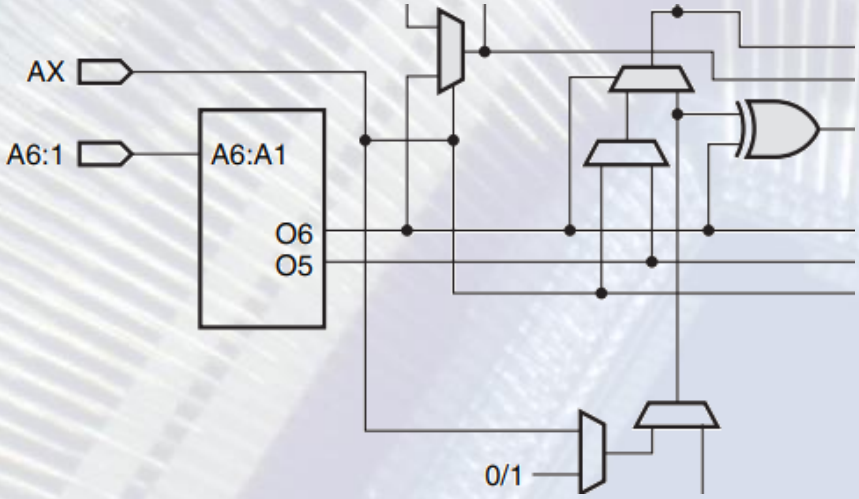


Carry logika

- **ADD/SUB/INC/DEC művelet**

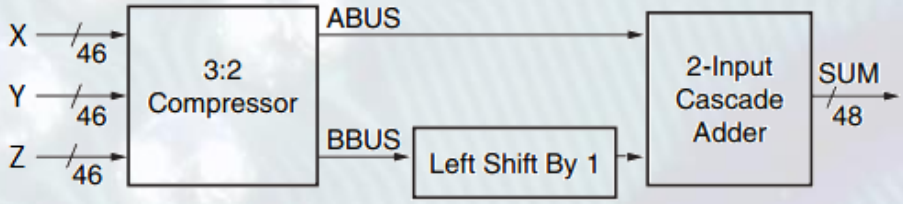


  - SUM=A xor B xor C_i
  - $C_o = AB + AC_i + BC_i$
  - C_o = (A xor B) ? C_i : A;



3-bemenetű összeadó

- **3:2 Compressor (carry-save adder) + összeadó: Add3**



ABUS=X^Y^Z

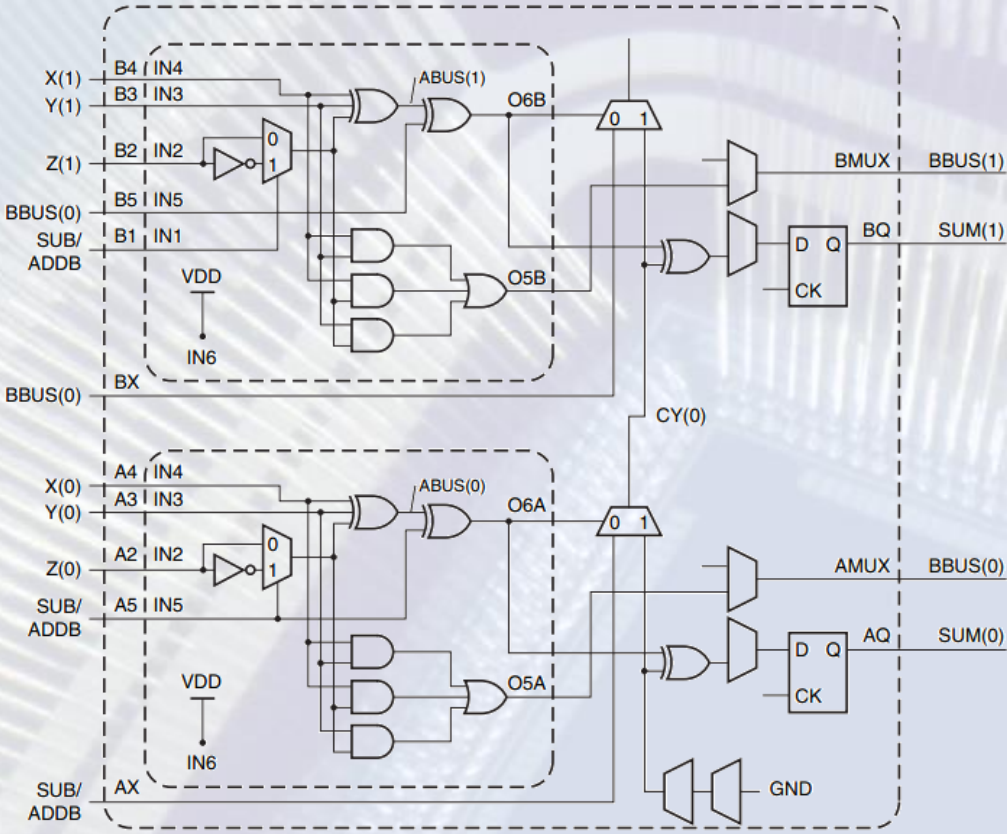
BBUS=(X&Y)|(X&Z)|(Y&Z)

ABUS: adott helyiérték összeg

BBUS: adott helyiérték carry

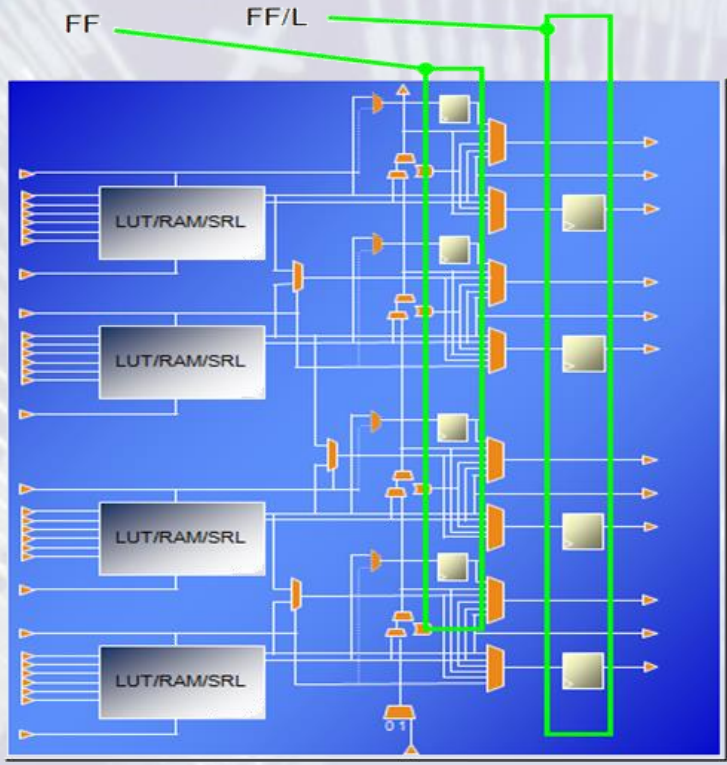
PL:

```
      1011_1001
     +0010_1010
     +0011_1001
-----------------
ABUS: 1010_1010
BBUS: 0011_1001

ABUS:  1010_1010
BBUS:  0011_1001
-----------------
      10001_1100
```



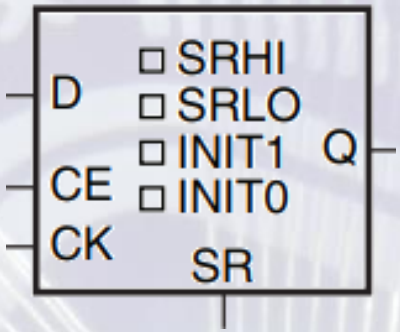
Slice Flip-Flop/Latch

- **Minden slice-ban négy flip-flop/latch (FF/L)**
  - Konfigrálható flip-flop-ként vagy latch-ként (utóbbi ellenjavalt)
  - A D bemenet lehet a LUT O6 kimenete, a carry logika kimenete, a multiplexerek kimenete vagy a slice AX/BX/CX/DX bemenete
- **Ezen kívül minden slice tartalmaz még négy flip-flop-t**
  - A D bemenet lehet a LUT O5 kimenete vagy a slice AX/BX/CX/DX bemenete
  - A carry logika kimenete, illetve a multiplexerek kimenete nem köthető be
- **Ha bármelyik tárolót latch-ként használjuk, a plusz 4 FF nem érhető el.**



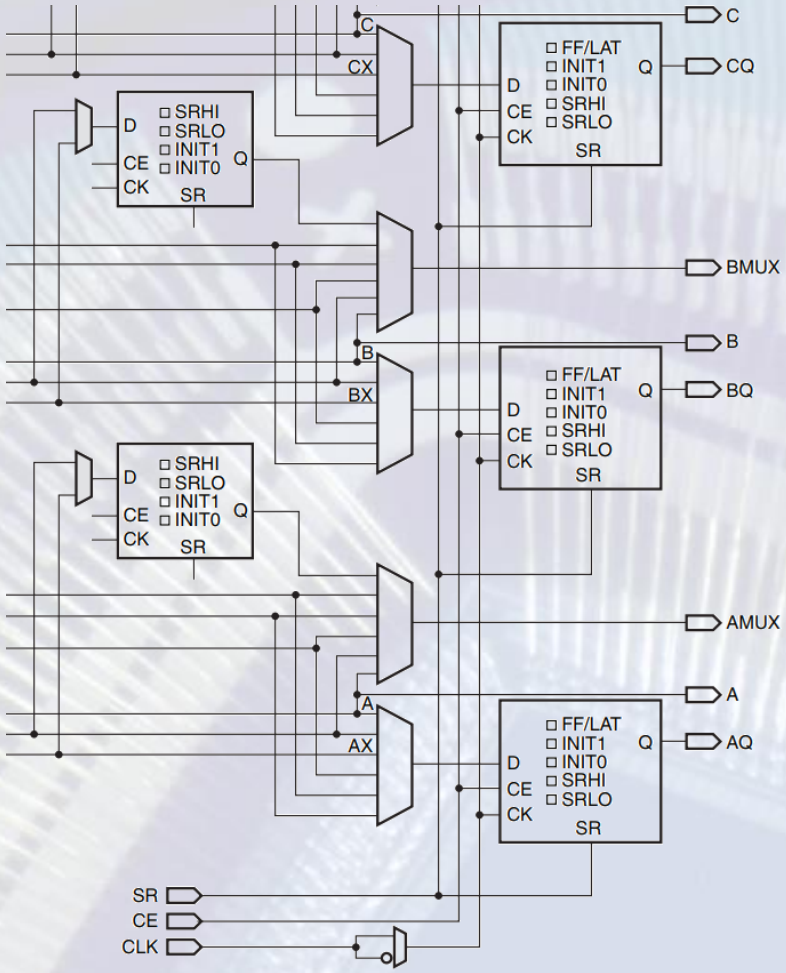
Slice Flip-Flop

- **D flip-flop-ok**
- **Q kimenet**
- **Órajel bemenet**
  - Invertálható a slice bemenetén
- **Aktív magas engedélyező jel (CE)**
- **Aktív magas SR bemenet**
  - Vagy RESET, vagy SET → konfigurációs beállítás
  - Lehet szinkron vagy aszinkron



FF vezérlő jelek

- **Egy slice-ban minden FF-ra közös: CK, SR és CE!**
  - Csak akkor használható ki mindegyik, ha ez a tervre is igaz.
  - CE, SR aktív magas
  - CK: invertálható a slice bemenetén
- **Ha bármelyik FF használ CE-t, akkor mindegyiknek kell**
  - CE az órajelet a slice bemenetén kapuzza
  - Fogyasztást csökkent
- **Ha bármelyik FF használja az SR bemenetet, akkor mindegyiknek ugyanazt kell használnia**
  - Az SR-rel beállított érték konfigurációs paraméter: SRVAL



Distributed SelectRAM memória

- **SLICEM képes memóriát megvalósítani**
- **Szinkron írás, aszinkron olvasás**
  - A slice FF felhasználásával egyszerűen szinkron olvasásra alakítható.
- **Konfigurációk**
  - Single port
    - Egy LUT6 = 64x1 vagy 32x2 RAM
    - Kaszkádosítható 256x1 RAM méretig
  - Dual port (D)
    - 1 olvasás/írás port + 1 csak olvasás port
  - Simple dual port (SDP)
    - 1 csak írás port + 1 csak olvasás port
  - Quad-port (Q)
    - 1 olvasás/írás port + 3 csak olvasás port

Single Port	Dual Port	Simple Dual Port	Quad Port
32x2	32x2**D**	32x6**SDP**	32x2**Q**
32x4	32x4**D**	64x3**SDP**	64x1**Q**
32x6	64x1**D**		
32x8	64x2**D**		
64x1	128x1**D**		
64x2			
64x3			
64x4			
128x1			
128x2			
256x1			

LUT RAM – Verilog

```
module ram16 (
    input clk, we,
    input [3:0] a,
    input [7:0] d,
    output [7:0] o
);

reg [7:0] mem[15:0];
always @ (posedge clk)
if (we)
        mem[a] <= d;

assign o = mem[a];

endmodule
```

LUT RAM – VHDL primitív

- **UNISIM library-ben definiálva**

```
Library UNISIM;
use UNISIM.vcomponents.all;
.....................

RAM32X1D_inst : RAM32X1D
generic map (
   INIT => X"00000000") -- Initial contents of RAM
port map (
   DPO => DPO,      -- Read-only 1-bit data output
   SPO => SPO,      -- R/W 1-bit data output
   A0 => A0,        -- R/W address[0] input bit
   A1 => A1,        -- R/W address[1] input bit
   A2 => A2,        -- R/W address[2] input bit
   A3 => A3,        -- R/W address[3] input bit
   A4 => A4,        -- R/W address[4] input bit
   D => D,          -- Write 1-bit data input
   DPRA0 => DPRA0, -- Read-only address[0] input bit
   DPRA1 => DPRA1, -- Read-only address[1] input bit
   DPRA2 => DPRA2, -- Read-only address[2] input bit
   DPRA3 => DPRA3, -- Read-only address[3] input bit
   DPRA4 => DPRA4, -- Read-only address[4] input bit
   WCLK => WCLK,    -- Write clock input
   WE => WE         -- Write enable input
);
```

LUT RAM – VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

entity ram64 is
port (
   clk  : in  std_logic;
   we   : in  std_logic;
   addr : in  std_logic_vector(5 downto 0);
   din  : in  std_logic_vector(7 downto 0);
   dout : out std_logic_vector(7 downto 0)
);
end;

architecture rtl of ram64 is

type marray  is array (63 downto 0) of
     std_logic_vector(7 downto 0);
signal memory : marray;
……
```

```
……

begin

process(clk)
begin
if (clk'event and clk='1') then
  if (we='1') then
    memory(TO_INTEGER(unsigned(addr))) <= din;
  end if;
end if;
end process;

dout <= memory(TO_INTEGER(unsigned(addr)));

end;
```

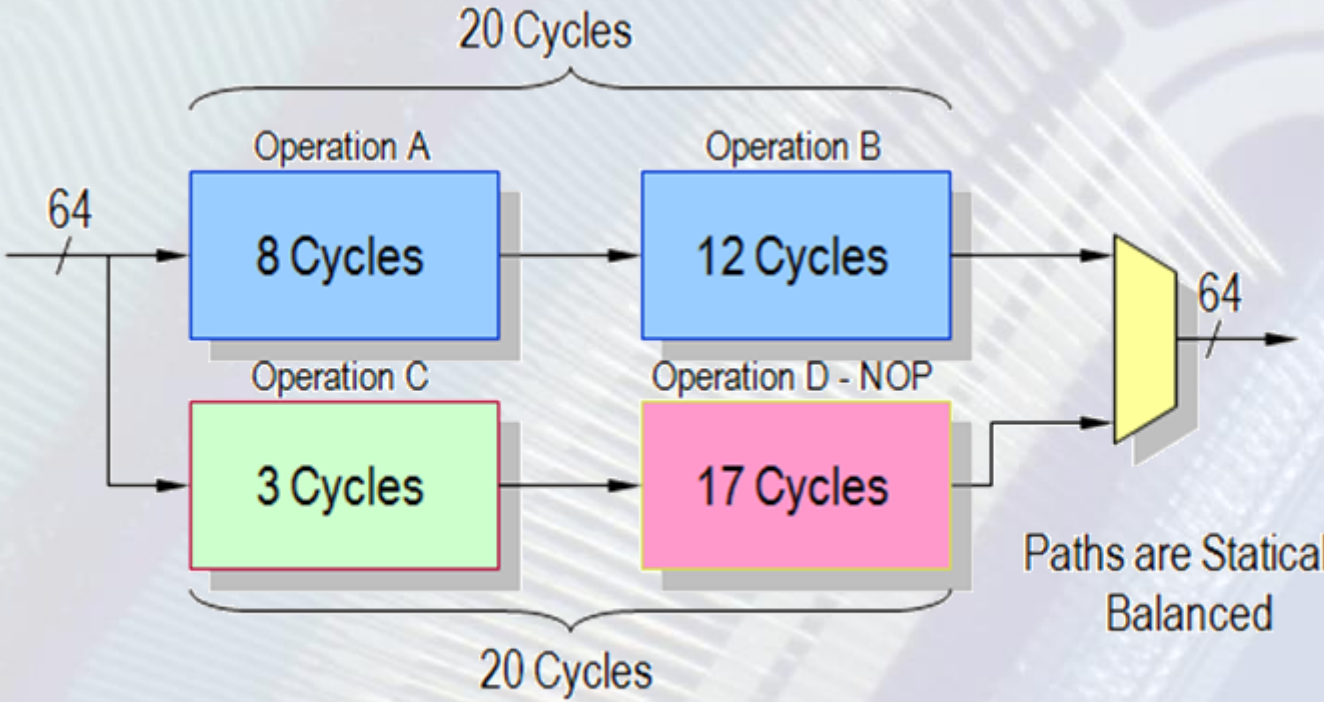
SLICEM shiftregiszter

- **SRL = Shift Register Lut**
- **NINCS RESET bemenete**
- **Sokoldalúan használható**
  - Változtatható hosszúságú shift regiszter
  - Szinkron FIFO
  - Content-Addressable Memory (CAM)
  - Minta generátor
  - Pipeline késleltetés kompenzáció
- **A shift regiszter hosszát a cím bemenet határozza meg**
  - Konstans: fix késleltetés
  - Változó: elasztikus buffer
- **Kaszkádosítható: egy slice-on belül 128x1 shift regiszter**

SRL Configurations in One Slice (4 LUTs)
16x1, 16x2, 16x4, 16x6, 16x8
32x1, 32x2, 32x3, 32x4
64x1, 64x2
96x1
128x1

SRL példa (1)

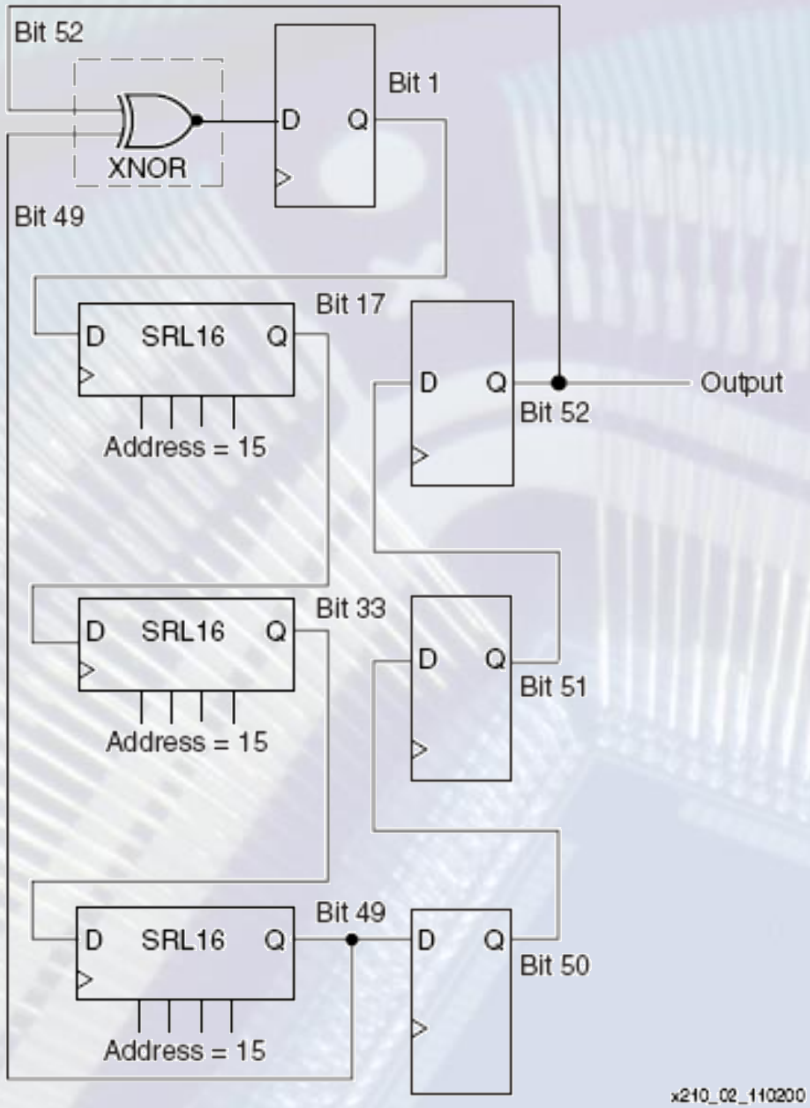
- **Operation D – NOP: az adatút késleltetés kiegyenlítéshez 17 órajel késleltetés**
  - 1,088 flip-flops → 136 slice
  - 64 SRLs → 16 slices



17-stage delay from SRL

SRL példa (2)

- **Pszeudo-random generátor LFSR-rel**
- **XAPP211**
  - polynomials n=3 …. n=168

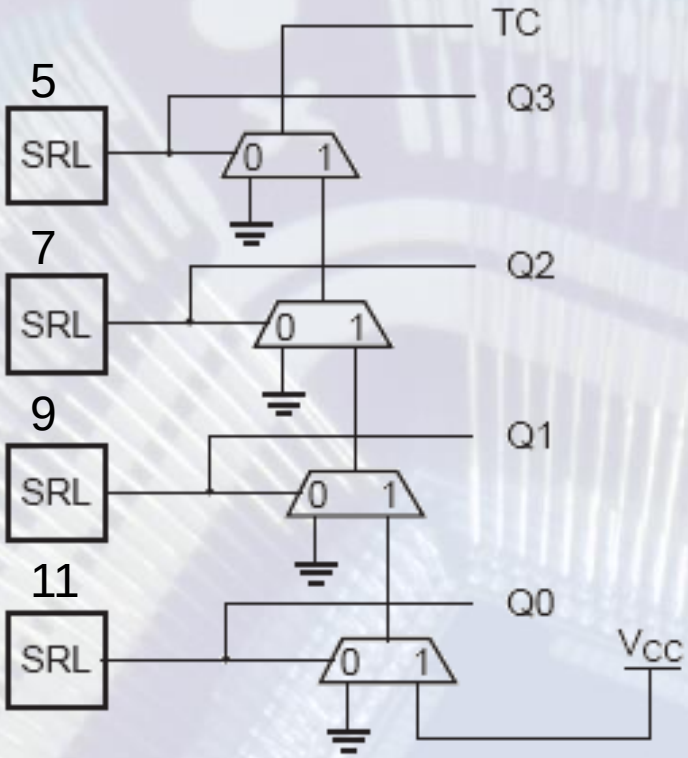


SRL példa (3)

- **Ring & Johnson számlálók**
  - Kezdőérték: bináris 100..000
    - Konfigurációs paraméter, NEM reset
  - Modulus az SRL cím bemenetével választható
  - Nem reset-elhető!
  - Engedélyezhető a CE bemenettel

SRL példa (4)

- **Nagy modulusú számláló**
  - Kaszkádosított SRL
  - SRL-ek hossza: modulus faktorizációja → relatív prímek
- **Pl.: modulus=3465**
  - Faktorizáció: 3465=11*9*7*5
  - Az egyes SRL-ek kimenete minden N-ik órajelben 1
  - Mivel az SRL-ek hosszai relatív prímek, így a TC kimenet csak minden 5*7*9*11= 3465. órajelben lesz 1



LUT SRL – Verilog

```
module SRL(
    input clk, ce, d,
    input [3:0] a,
    output q, q15
);

reg [15:0] shr = 16'h0000;
always @ (posedge clk)
if (ce)
        shr <= {shr[14:0], d};

assign q15 = shr[15];
assign q = shr[a];

endmodule
```

LUT SRL – VHDL

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

entity srl32 is
port (
  clk       : in  std_logic;
  ce        : in  std_logic;
  addr      : in std_logic_vector(4 downto 0);
  d         : in  std_logic;
  q, q31    : out std_logic;);
end;

architecture rtl of srl32 is

signal shr : std_logic_vector(31 downto 0);
……
```

```
begin

process(clk)
begin
if (clk'event and clk='1') then
    if (ce='1') then
        shr <= shr(30 downto 0) & d;
    end if;
end if;
end process;
q   <= shr(TO_INTEGER(unsigned(addr)));
q31 <= shr(31);

end;
```

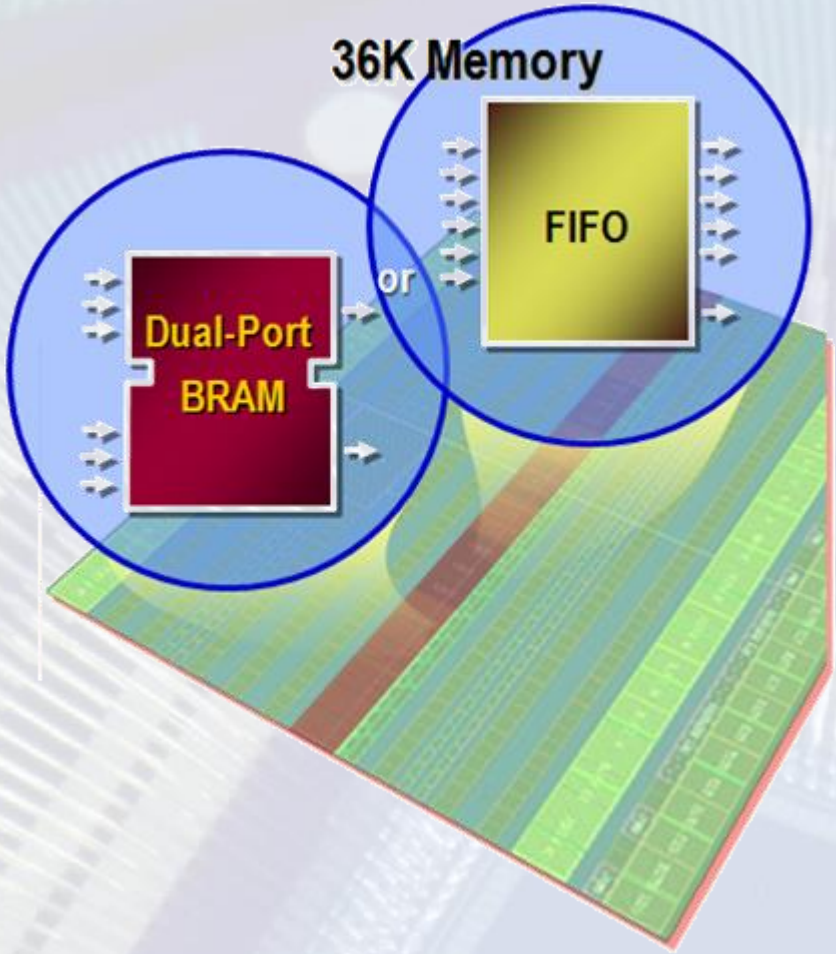
BRAM

Block RAM használat

- **Tetszőleges memória funkciók**
  - Pl. program memória
- **Szó-szélesség konverzió**
  - PAR → SER or SER→PAR,
  - 32 bit → 2 bit, 2 bit → 32 bit
- **Buffer-ek, késleltető vonalak**
  - Pl. videó sor-buffer
- **Általános logika (nagy LUT regiszterezett kimenettel)**
- **Look up table, pl. színusz generáláshoz**
  - Az FPGA konfigurálásakor inicializálható

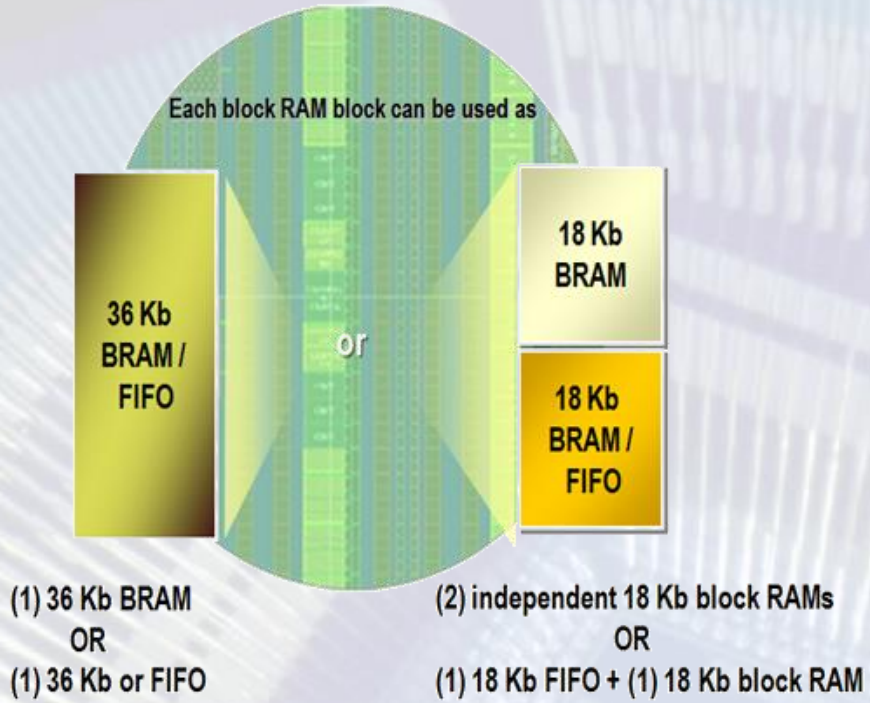
Block RAM és FIFO

- **A 7-es sorozat minden tagjában ugyanolyan**
- **Teljesen szinkron működés**
  - Mind az írás, mind pedig az olvasás
- **Opcionális pipeline regiszter → magasabb elérhető órajel**
- **Két független port**
  - Külön órajel, engedélyezés, írás engedélyezés, cím, adat
  - Portonként állítható szószélesség



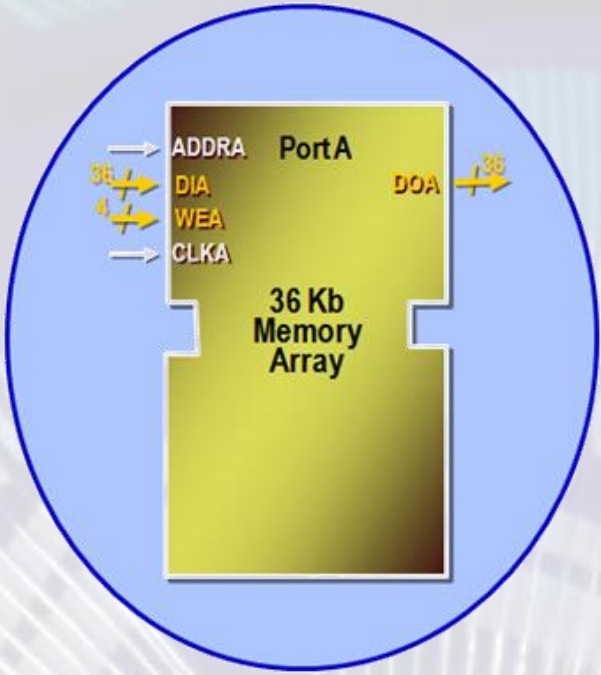
Block RAM és FIFO

- **Több konfigurációs lehetőség**
  - True dual-port, simple dual-port, single-port
- **Integrált kaszkádosítási lehetőség**
- **Byte írás enegedélyezés**
- **Integrált FIFO vezérlő logika**
- **Integrált Hamming hibajavítás**
- **(Külön tápfeszültségről üzemel)**



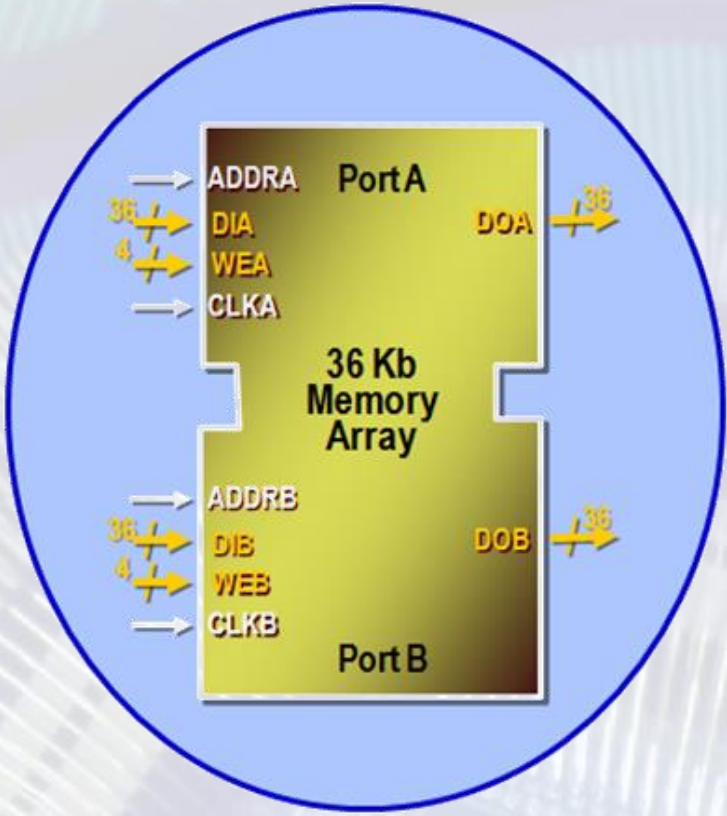
Single-Port Block RAM

- **Egy olvasás/írás port**
  - Órajel: CLKA, Cím: ADDRA,
    Írás engedélyezés: WEA
  - Bemeneti adat: DIA, Kimeneti adat: DOA
- **36-kbit konfigurációk**
  - 32k x 1, 16k x 2, 8k x 4, 4k x 9, 2k x 18, 1k x 36
- **18-kbit konfigurációk**
  - 16k x 1, 8k x 2, 4k x 4, 2k x 9, 1k x 18, 512 x 36
- **Írási mód**
  - WRITE_FIRST: Az írt adat azonnal megjelenik a kimeneten
  - READ_FIRST: A régi érték jelenik meg a kimeneten
  - NO_CHANGE: Nem változik a kimenet
- **Opcionális kimeneti regiszter**



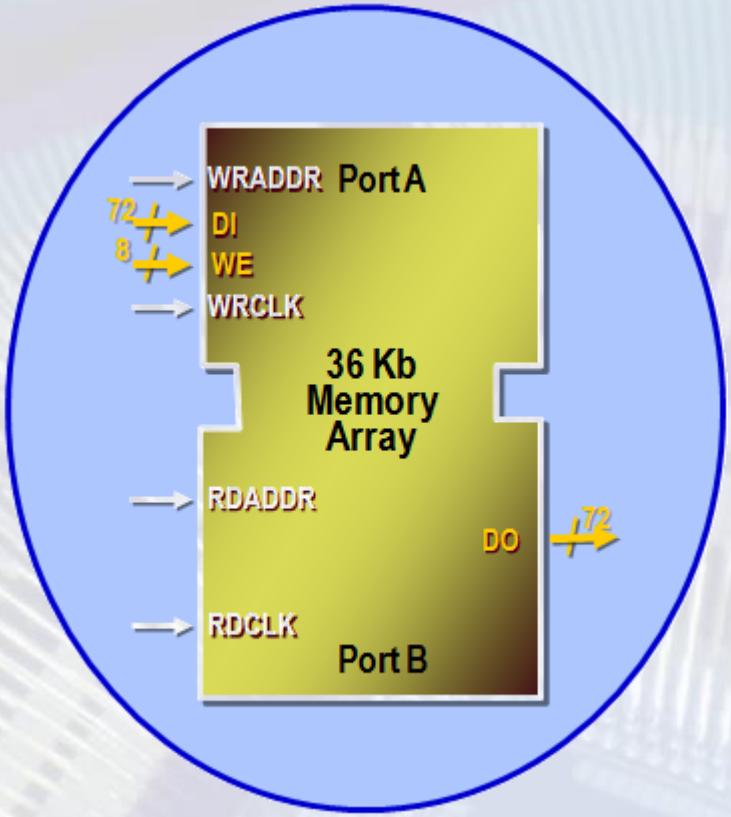
Dual-Port Block RAM

- **Két független írás/olvasás port**
  - Mindkét portnak független vezérlő- és adatvonala van
    - A két port órajele is lehet különböző
  - A két port adatszélessége különbözhet
    - Opciók ugyanazok, mint egy port esetén
  - A két port írási módja lehet különböző
- **NINCS ütközés elkerülés, ha mindkét portról ugyanahhoz az adathoz fordulunk!**
  - Kivéve közös órajel és READ_FIRST üzemmód esetén, ekkor az olvasott adat érvényes
  - Ugyanazt a címet két portról írni tilos.



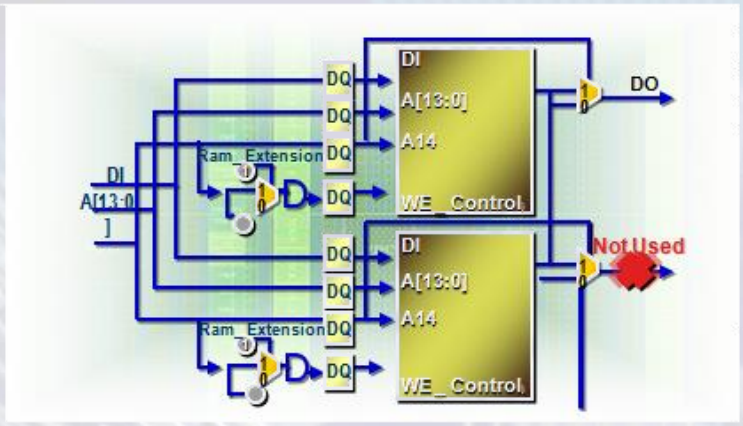
Simple Dual-Port Block RAM

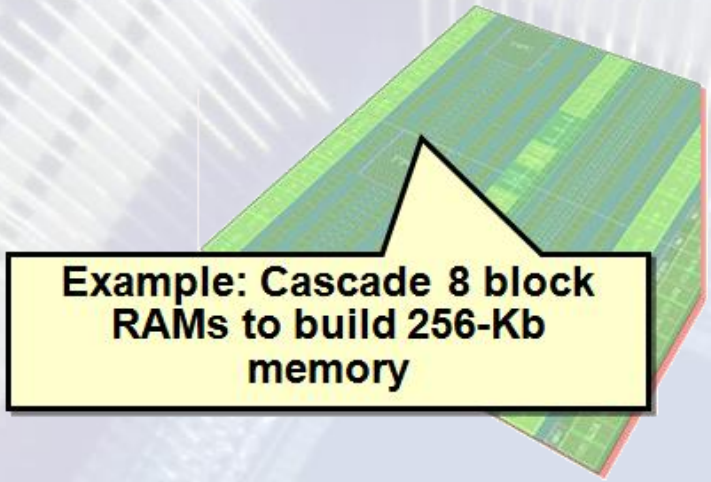
- **Egy olvasási és egy írási port**
  - Független órajel és cím
- **36-kbit-es konfiguráció esetén az egyik portnak 72 bitesnek kell lennie**
  - A másik port lehet: x1, x2, x4, x9, x18, x36, or x72
- **18-kbit-es konfiguráció esetén az egyik portnak 36 bitesnek kell lennie**
  - A másik port lehet: x1, x2, x4, x9, x18, or x36



Block RAM kaszkádosítás

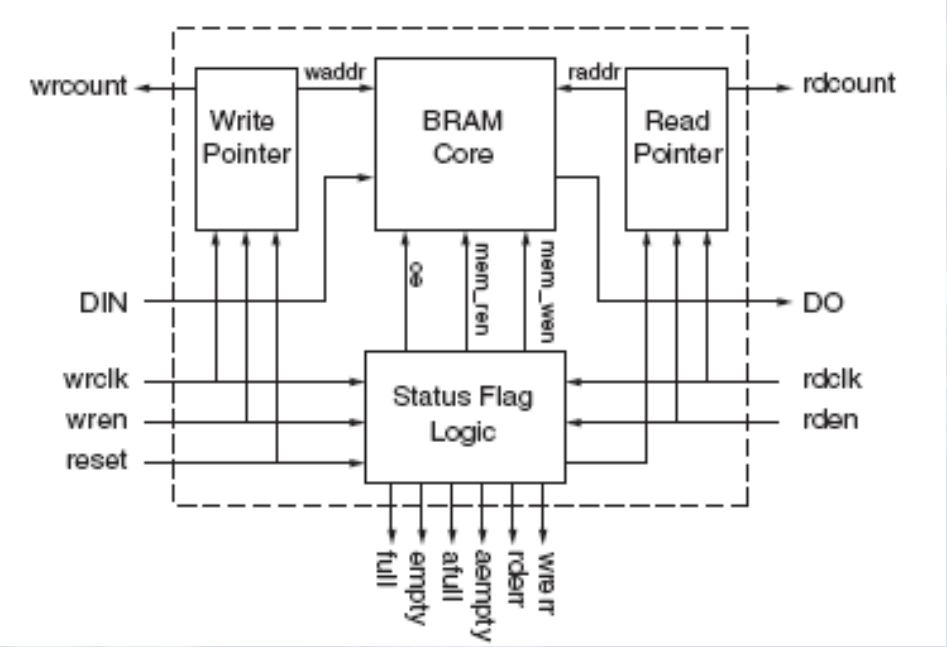
- **Beépített kaszkádosítási lehetőség 64Kx1 konfigurációban**
  - Két vertikálisan szomszédos 32Kx1 BRAM kaszkádosítása CLB logika és teljesítmény veszteség nélkül
- **Nagyobb méretű memóriák**
  - 128Kb, 256Kb, 512Kb, 1 Mb, …
  - CLB logika szükséges a mélység növeléséhez
    - Limitálhatja az elért órajelet
  - Szó-szélesség növelhető BRAM-ok párhuzamosan kapcsolásával





Block RAM FIFO

- **FIFO használat**
  - Nem igényel CLB-t
  - Olyan gyors, mint a BRAM
  - Módok:
    - Normal
    - FWFT: First word fall through
  - Státusz jelek
    - Full, AlmostFull
    - Empty, AlmostEmpty

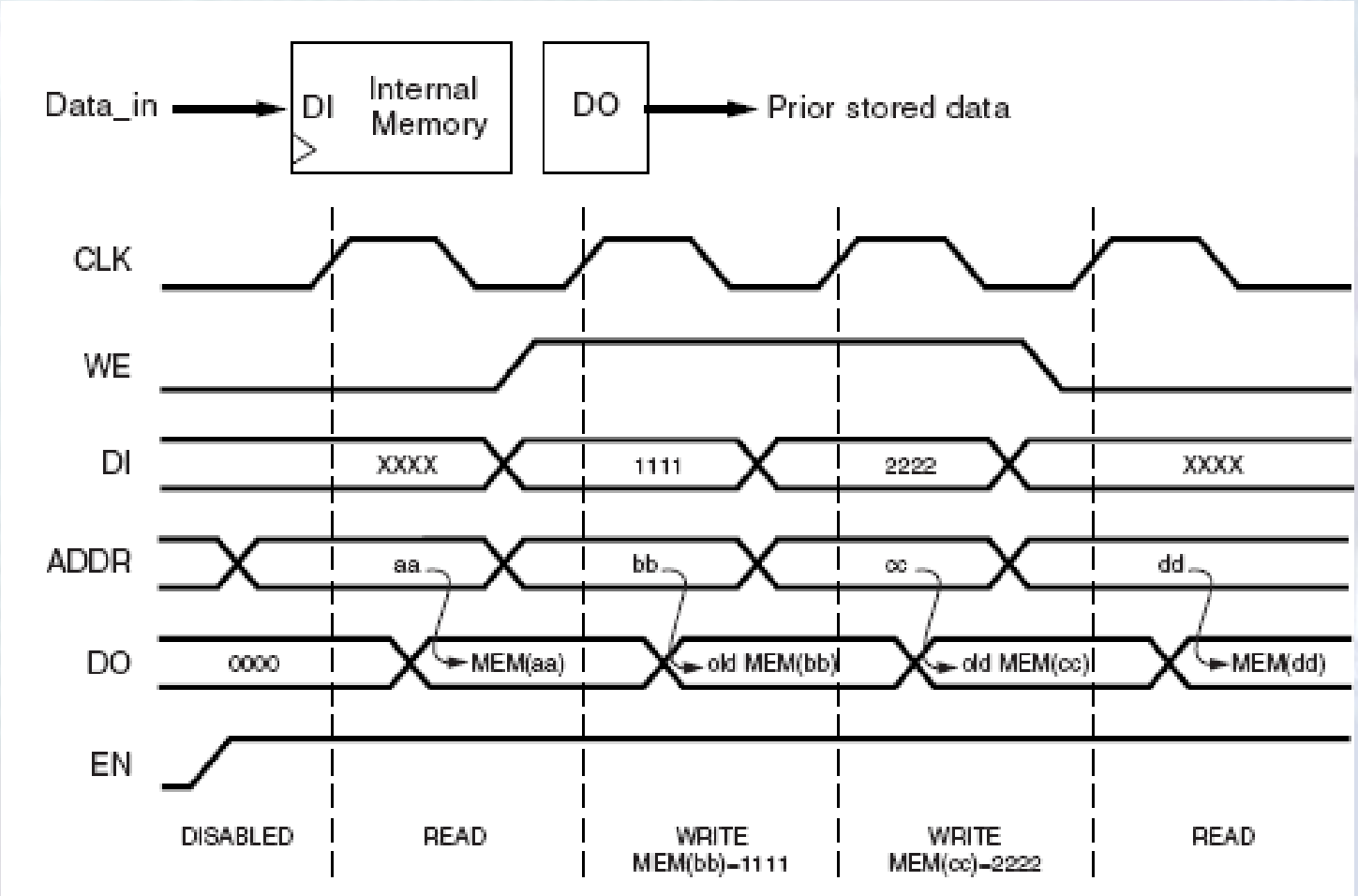


BRAM primitív példányosítás

```
Library UNISIM;
use UNISIM.vcomponents.all;

RAMB36E1_inst : RAMB36E1
generic map (
   DOA_REG => 0,
   DOB_REG => 0,
   INITP_00 => (others=>'0'),
   INIT_00 => (others=>'0'),
   INIT_A => X"000000000",
   INIT_FILE => "NONE",
   RAM_MODE => "TDP",
   READ_WIDTH_A => 0,    -- 0-72
   READ_WIDTH_B => 0,    -- 0-36
   WRITE_WIDTH_A => 0,   -- 0-36
   WRITE_WIDTH_B => 0,   -- 0-72
   WRITE_MODE_A => "WRITE_FIRST",
   WRITE_MODE_B => "WRITE_FIRST"
)
```

```
port map (
   CASCADEOUTA => CASCADEOUTA,     -- 1-bit output: A port cascade
   CASCADEOUTB => CASCADEOUTB,     -- 1-bit output: B port cascade
   DBITERR => DBITERR,             -- 1-bit output: Double bit error status
   ECCPARITY => ECCPARITY,         -- 8-bit output: Generated error correction parity
   RDADDRECC => RDADDRECC,         -- 9-bit output: ECC read address
   SBITERR => SBITERR,             -- 1-bit output: Single bit error status
   DOADO => DOADO,                 -- 32-bit output: A port data/LSB data
   DOPADOP => DOPADOP,             -- 4-bit output: A port parity/LSB parity
   DOBDO => DOBDO,                 -- 32-bit output: B port data/MSB data
   DOPBDOP => DOPBDOP,             -- 4-bit output: B port parity/MSB parity
   CASCADEINA => CASCADEINA,       -- 1-bit input: A port cascade
   CASCADEINB => CASCADEINB,       -- 1-bit input: B port cascade
   INJECTDBITERR => INJECTDBITERR, -- 1-bit input: Inject a double bit error
   INJECTSBITERR => INJECTSBITERR, -- 1-bit input: Inject a single bit error
   ADDRARDADDR => ADDRARDADDR,     -- 16-bit input: A port address/Read address
   CLKARDCLK => CLKARDCLK,         -- 1-bit input: A port clock/Read clock
   ENARDEN => ENARDEN,             -- 1-bit input: A port enable/Read enable
   REGCEAREGCE => REGCEAREGCE,     -- 1-bit input: A port register enable/Register enable
   RSTRAMARSTRAM => RSTRAMARSTRAM, -- 1-bit input: A port set/reset
   RSTREGARSTREG => RSTREGARSTREG, -- 1-bit input: A port register set/reset
   WEA => WEA,                     -- 4-bit input: A port write enable
   DIADI => DIADI,                 -- 32-bit input: A port data/LSB data
   DIPADIP => DIPADIP,             -- 4-bit input: A port parity/LSB parity
   ADDRBWRADDR => ADDRBWRADDR,     -- 16-bit input: B port address/Write address
   CLKBWRCLK => CLKBWRCLK,         -- 1-bit input: B port clock/Write clock
   ENBWREN => ENBWREN,             -- 1-bit input: B port enable/Write enable
   REGCEB => REGCEB,               -- 1-bit input: B port register enable
   RSTRAMB => RSTRAMB,             -- 1-bit input: B port set/reset
   RSTREGB => RSTREGB,             -- 1-bit input: B port register set/reset
   WEBWE => WEBWE,                 -- 8-bit input: B port write enable/Write enable
   DIBDI => DIBDI,                 -- 32-bit input: B port data/MSB data
   DIPBDIP => DIPBDIP              -- 4-bit input: B port parity/MSB parity
);
```

Block RAM – Read first



BlockRAM – Read First

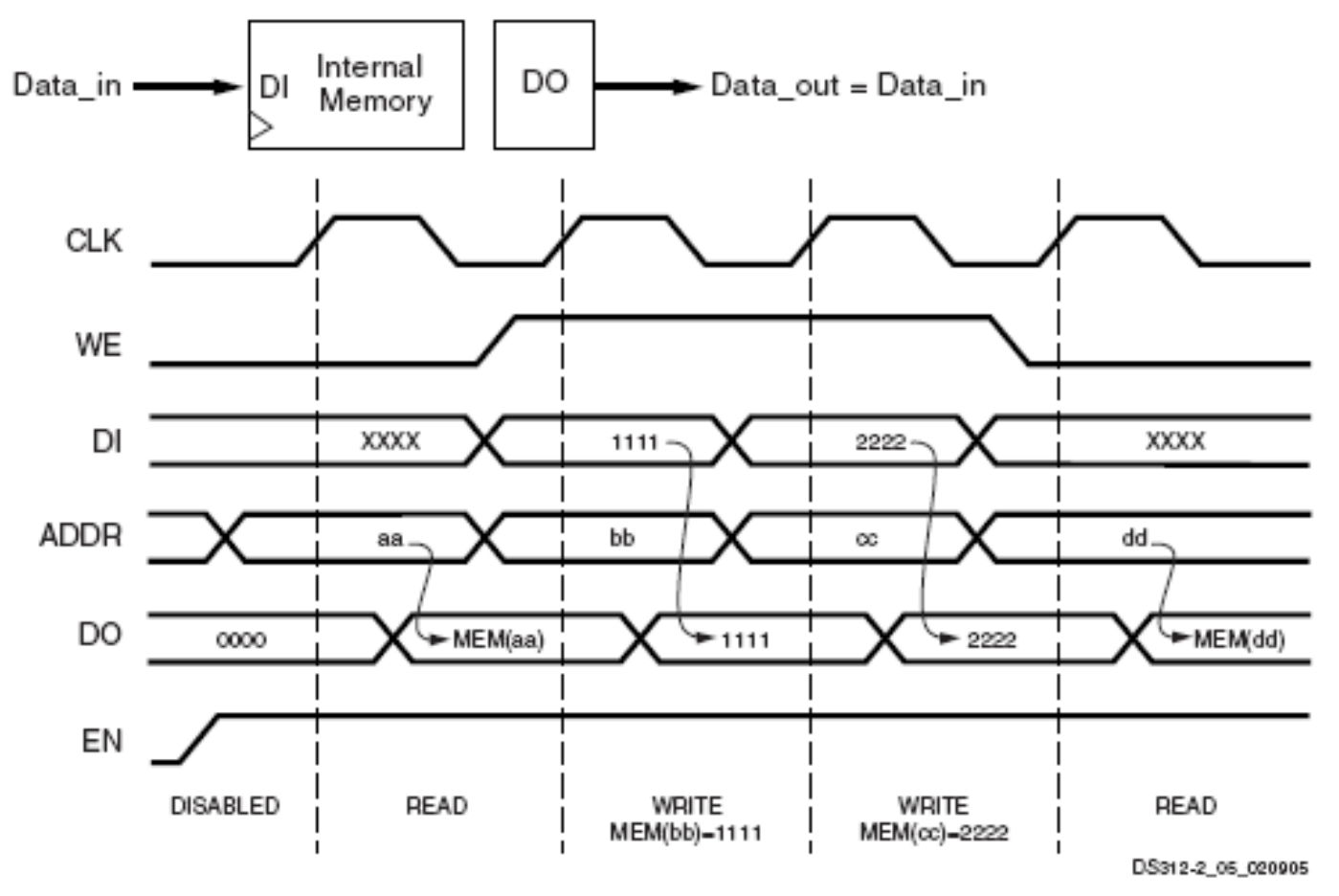
```
module sp_ram(input clk, input we, input en,
              input [10:0] addr, input [ 7:0] din,
              output [7:0] dout);

reg [7:0] memory[2047:0];
reg [7:0] dout_reg;

always @ (posedge clk)
if (en)
begin
   if (we)
      memory[addr] <= din;
   dout_reg <= memory[addr];
end

assign dout = dout_reg;

endmodule
```

BlockRAM – Read First

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


entity ram is
port (
  clk, ce, we : in  std_logic;
  addr : in  std_logic_vector(8 downto 0);
  din  : in  std_logic_vector(35 downto 0);
  dout : out std_logic_vector(35 downto 0)
);
end;

architecture rtl of ram is
type marray  is array (511 downto 0) of
             std_logic_vector(35 downto 0);
signal memory : marray;
```

```
begin

process(clk)
begin
if (clk'event and clk='1') then
  if (ce='1') then
    if (we='1') then
      memory(TO_INTEGER(unsigned(addr))) <= din;
    end if;
    dout <= memory(TO_INTEGER(unsigned(addr)));
  end if;
end if;
end process;

end;
```

Block RAM – Write first



BlockRAM – Write First

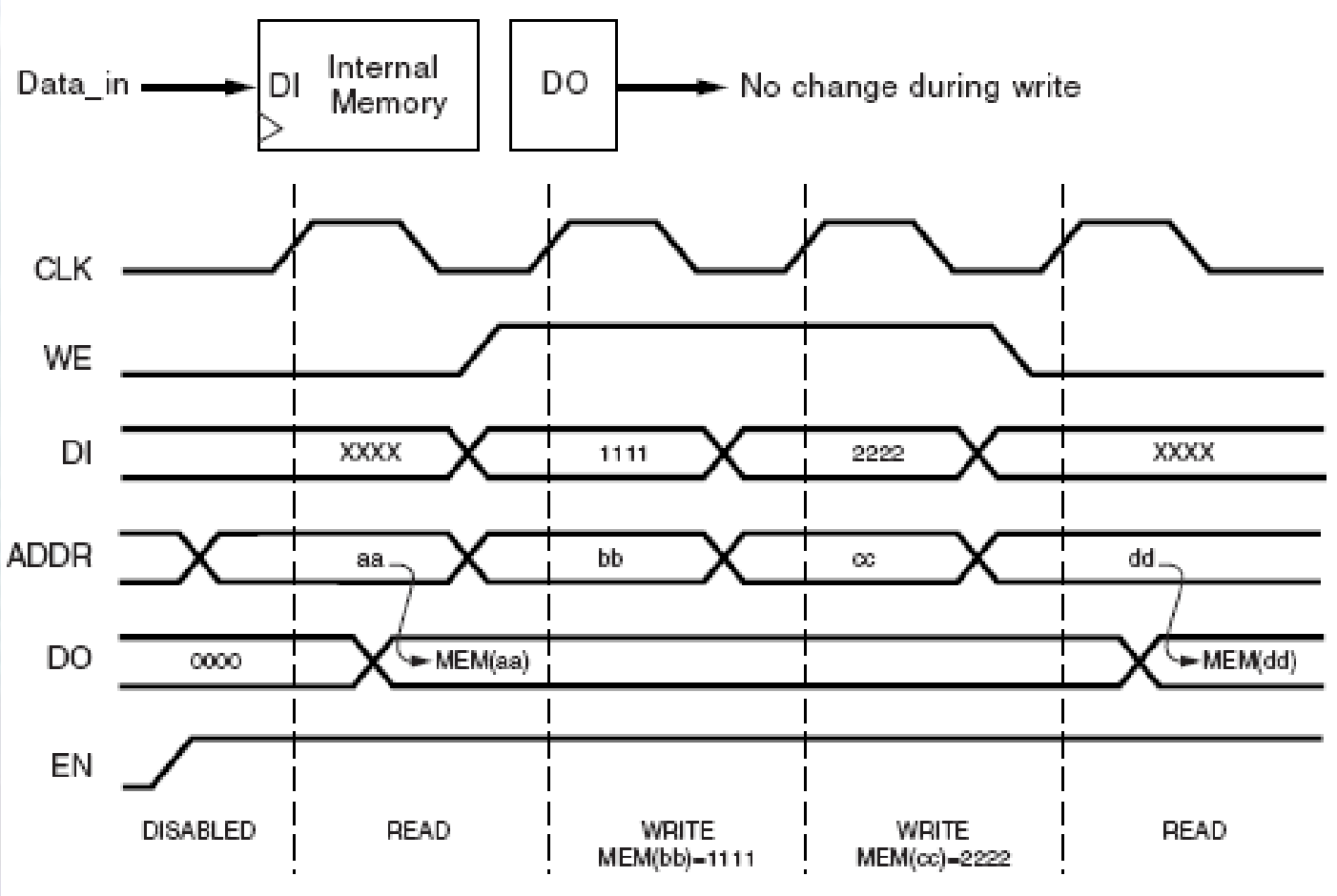
```
module sp_ram(input clk, input we, input en,
              input [10:0] addr, input [ 7:0] din,
              output [7:0] dout);

reg [7:0] memory[2047:0];
reg [7:0] dout_reg;

always @ (posedge clk)
if (en)
begin
   if (we)
      memory[addr] = din;
   dout_reg = memory[addr];
end

assign dout = dout_reg;

endmodule
```

BlockRAM – Write First

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


entity ram is
port (
  clk, ce, we : in  std_logic;
  addr : in  std_logic_vector(8 downto 0);
  din  : in  std_logic_vector(35 downto 0);
  dout : out std_logic_vector(35 downto 0)
);
end;

architecture rtl of ram is
type marray  is array (511 downto 0) of
             std_logic_vector(35 downto 0);
signal memory : marray;
```

```
begin

process(clk)
begin
if (clk'event and clk='1') then
  if (ce='1') then
    if (we='1') then
      memory(TO_INTEGER(unsigned(addr))) <= din;
      dout <= din;
    else
      dout <= memory(TO_INTEGER(unsigned(addr)));
    end if;
  end if;
end if;
end process;

end;
```

Block RAM – No change



BlockRAM – No Change

```
module sp_ram(input clk, input we, input en,
              input [10:0] addr, input [ 7:0] din,
              output [7:0] dout);

reg [7:0] memory[2047:0];
reg [7:0] dout_reg;

always @ (posedge clk)
if (en)
begin
   if (we)
      memory[addr] <= din;
   else
      dout_reg <= memory[addr];
end

assign dout = dout_reg;

endmodule
```

BlockRAM – No Change

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


entity ram is
port (
  clk, ce, we : in  std_logic;
  addr : in  std_logic_vector(8 downto 0);
  din  : in  std_logic_vector(35 downto 0);
  dout : out std_logic_vector(35 downto 0)
);
end;

architecture rtl of ram is
type marray  is array (511 downto 0) of
             std_logic_vector(35 downto 0);
signal memory : marray;
```

```
begin

process(clk)
begin
if (clk'event and clk='1') then
  if (ce='1') then
    if (we='1') then
      memory(TO_INTEGER(unsigned(addr))) <= din;
    else
      dout <= memory(TO_INTEGER(unsigned(addr)));
    end if;
  end if;
end if;
end process;

end;
```

DP BlockRAM

```
module dp_ram(input clk_a, we_a, en_a, clk_b, we_b, en_b,
              input [10:0] addr_a, addr_b,
              input [ 7:0] din_a, din_b, output [7:0] dout_a, dout_b);

reg [7:0] memory[2047:0];
reg [7:0] dout_reg_a, dout_reg_b;

always @ (posedge clk_a)
if (en_a)
begin
   if (we_a)
      memory[addr_a] <= din_a;
   dout_reg_a <= memory[addr_a];
end
assign dout_a = dout_reg_a;

always @ (posedge clk_b)
if (en_b)
begin
   if (we_b)
      memory[addr_b] <= din_b;
   dout_reg_b <= memory[addr_b];
end
assign dout_b = dout_reg_b;
endmodule
```

DP BlockRAM – VHDL (1)

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

entity bram is
generic(
   ADDR_WIDTH : integer := 12;
   DATA_WIDTH : integer := 18);
port(
   clk_a      : in  std_logic;
   rst_a      : in  std_logic;
   ce_a       : in  std_logic;
   we_a       : in  std_logic;
   addr_a     : in  std_logic_vector(ADDR_WIDTH-1 downto 0);
   din_a      : in  std_logic_vector(DATA_WIDTH-1 downto 0);
   dout_a     : out std_logic_vector(DATA_WIDTH-1 downto 0);
   clk_b      : in  std_logic;
   rst_b      : in  std_logic;
   ce_b       : in  std_logic;
   we_b       : in  std_logic;
   addr_b     : in  std_logic_vector(ADDR_WIDTH-1 downto 0);
   din_b      : in  std_logic_vector(DATA_WIDTH-1 downto 0);
   dout_b     : out std_logic_vector(DATA_WIDTH-1 downto 0));
end bram;
```

DP BlockRAM – VHDL (2)

```
architecture rtl of bram is

type array_WbyD is array ((2**ADDR_WIDTH)-1 downto 0) of
std_logic_vector(DATA_WIDTH-1 downto 0);
shared variable memory : array_WbyD := (others=>(others=>'0'));

signal dout_reg_a0 : std_logic_vector(DATA_WIDTH-1 downto 0);
signal dout_reg_a1 : std_logic_vector(DATA_WIDTH-1 downto 0);

signal dout_reg_b0 : std_logic_vector(DATA_WIDTH-1 downto 0);
signal dout_reg_b1 : std_logic_vector(DATA_WIDTH-1 downto 0);

begin
```

DP BlockRAM – VHDL (3)

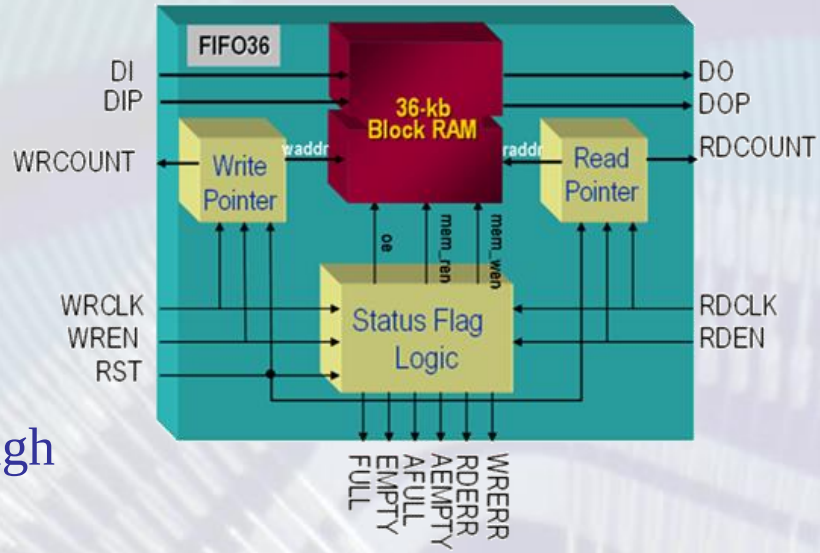
```
-- PORT A
process(clk_a)
begin
if (clk_a'event and clk_a='1') then
  if (ce_a='1') then
    if (rst_a='1') then
      dout_reg_a0 <= (others=>'0');
      dout_reg_a1 <= (others=>'0');
    else
      dout_reg_a0 <= memory(TO_INTEGER(unsigned(addr_a)));
      dout_reg_a1 <= dout_reg_a0;
    end if;
    if (we_a='1') then
      memory(TO_INTEGER(unsigned(addr_a))) := din_a;
    end if;
  end if;
end if;
end process;
dout_a <= dout_reg_a1;
```

DP BlockRAM – VHDL (4)

```
-- PORT B
process(clk_b)
begin
if (clk_b'event and clk_b='1') then
  if (ce_b='1') then
    if (rst_b='1') then
      dout_reg_b0 <= (others=>'0');
      dout_reg_b1 <= (others=>'0');
    else
      dout_reg_b0 <= memory(TO_INTEGER(unsigned(addr_b)));
      dout_reg_b1 <= dout_reg_b0;
    end if;
    if (we_b='1') then
      memory(TO_INTEGER(unsigned(addr_b))) := din_b;
    end if;
  end if;
end if;
end process;
dout_b <= dout_reg_b1;

end rtl;
```

FIFO

- **Tulajdonságok**
  - Közös vagy különböző (aszinkron) írás/olvasás órajelek
  - Státusz kimenetek
    - Full, empty, programozható almost-full/empty
  - Standard vagy First Word Fall Through (FWFT)
- **Méretek**
  - 36-Kb BRAM: 8Kx4, 4Kx9, 2Kx18, 1Kx36, 512x72
  - 18-Kb BRAM: 4Kx4, 2Kx9, 1Kx18, 512x36
  - Az írási és olvasási adatszélesség megegyezik!
- **X72 adatméret: beépített hibajavítás**

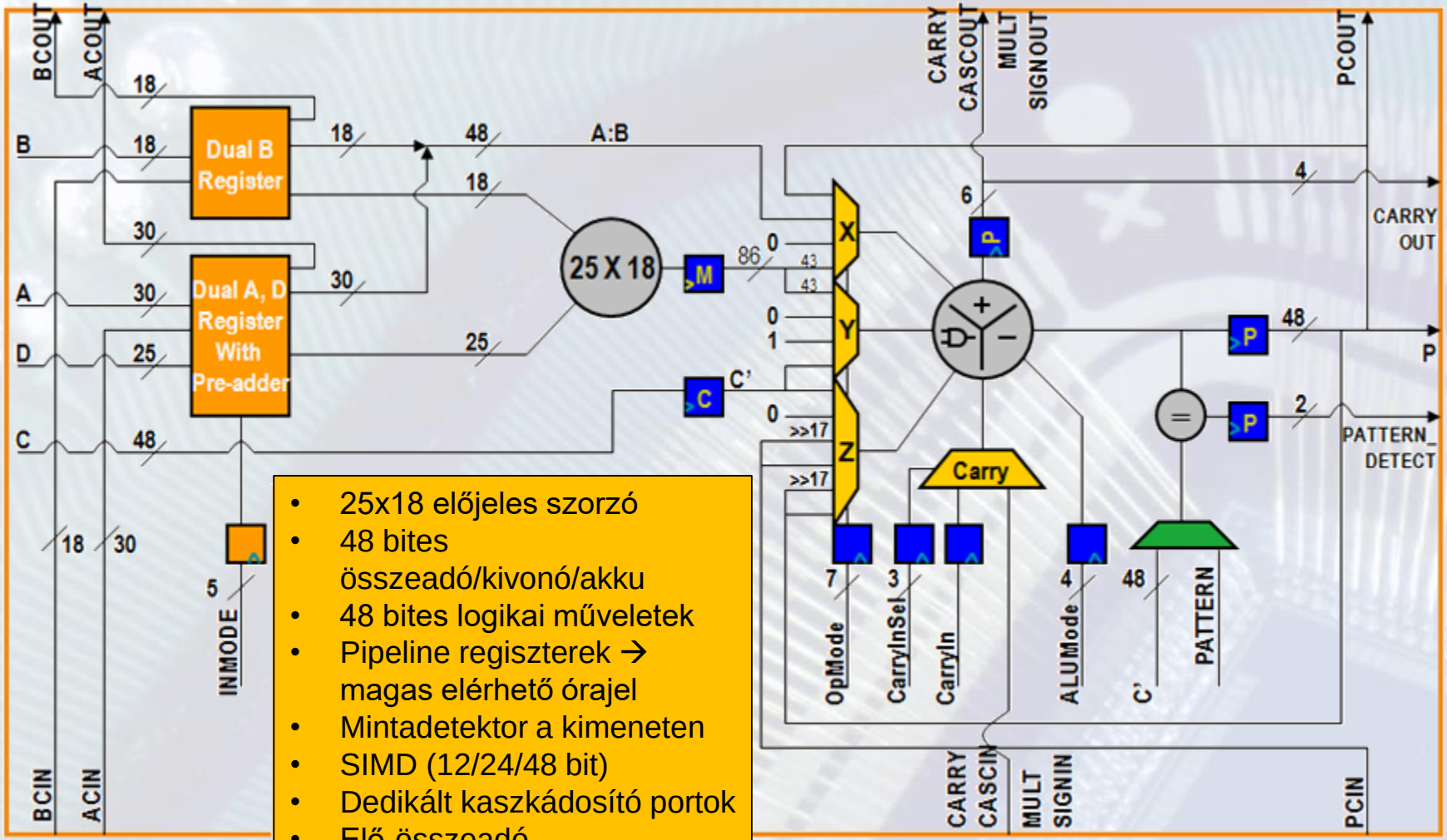


18k FIFO primitív

```
Library UNISIM;use UNISIM.vcomponents.all;

FIFO18E1_inst : FIFO18E1
generic map (
   ALMOST_EMPTY_OFFSET => X"0080",    -- Sets the almost empty threshold
   ALMOST_FULL_OFFSET => X"0080",     -- Sets almost full threshold
   DATA_WIDTH => 4,                   -- Sets data width to 4-36
   DO_REG => 1,                       -- Enable output register (1-0) Must be 1 if EN_SYN = FALSE
   EN_SYN => FALSE,                   -- Specifies FIFO as dual-clock (FALSE) or Synchronous (TRUE)
   FIFO_MODE => "FIFO18",             -- Sets mode to FIFO18 or FIFO18_36
   FIRST_WORD_FALL_THROUGH => FALSE, -- Sets the FIFO FWFT to FALSE, TRUE
   INIT => X"000000000",              -- Initial values on output port
   SIM_DEVICE => "7SERIES",           -- Must be set to "7SERIES" for simulation behavior
   SRVAL => X"000000000"              -- Set/Reset value for output port
)
port map (
   DO => DO,                     -- 32-bit output: Data output
   DOP => DOP,                   -- 4-bit output: Parity data output
   ALMOSTEMPTY => ALMOSTEMPTY, -- 1-bit output: Almost empty flag
   ALMOSTFULL => ALMOSTFULL,    -- 1-bit output: Almost full flag
   EMPTY => EMPTY,               -- 1-bit output: Empty flag
   FULL => FULL,                 -- 1-bit output: Full flag
   RDCOUNT => RDCOUNT,           -- 12-bit output: Read count
   RDERR => RDERR,               -- 1-bit output: Read error
   WRCOUNT => WRCOUNT,           -- 12-bit output: Write count
   WRERR => WRERR,               -- 1-bit output: Write error
   RDCLK => RDCLK,               -- 1-bit input: Read clock
   RDEN => RDEN,                 -- 1-bit input: Read enable
   REGCE => REGCE,               -- 1-bit input: Clock enable
   RST => RST,                   -- 1-bit input: Asynchronous Reset
   RSTREG => RSTREG,             -- 1-bit input: Output register set/reset
   WRCLK => WRCLK,               -- 1-bit input: Write clock
   WREN => WREN,                 -- 1-bit input: Write enable
   DI => DI,                     -- 32-bit input: Data input
   DIP => DIP                    -- 4-bit input: Parity input
);
```

DSP blokk (DSP48E1)

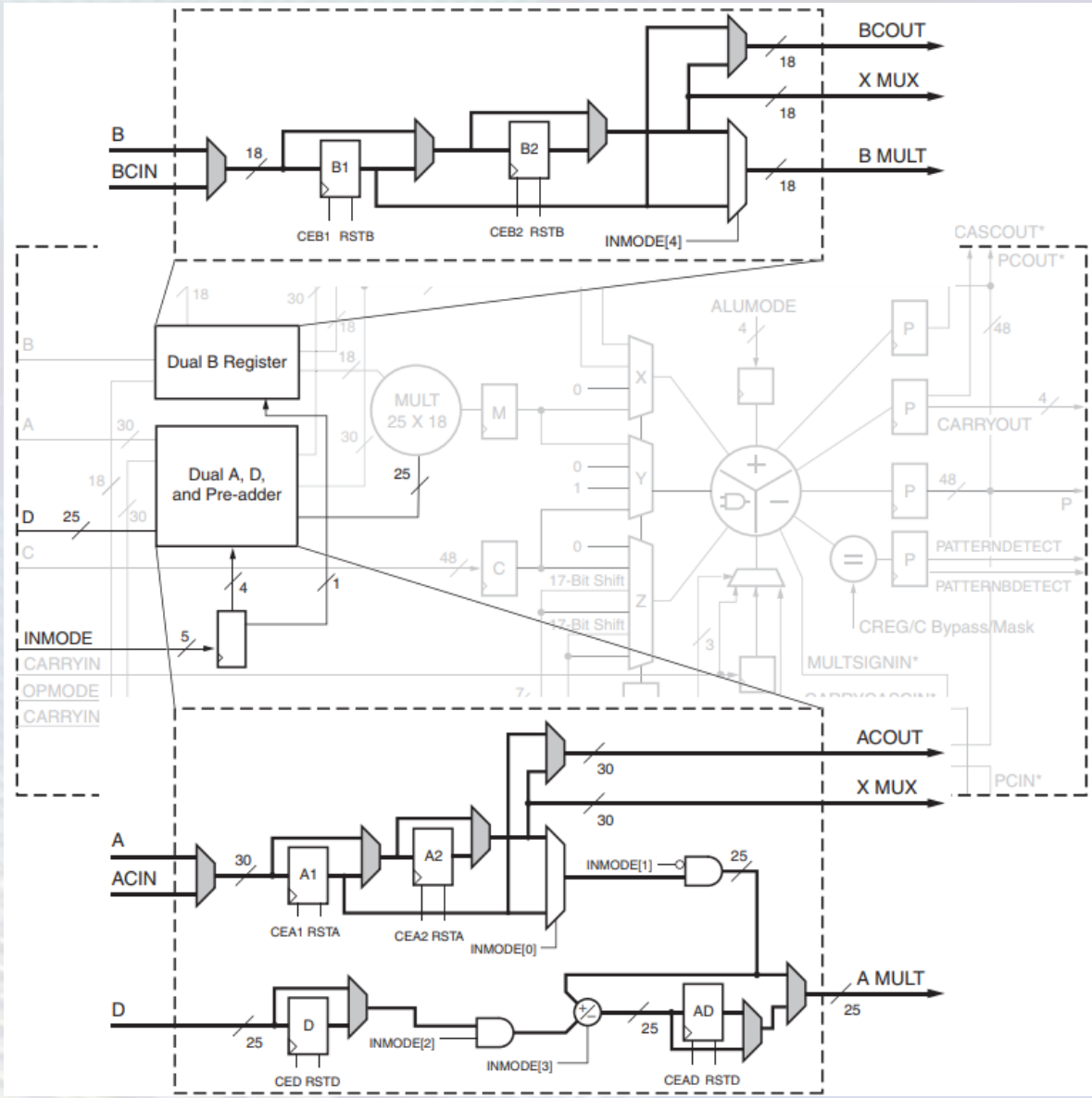
DSP48E1



- 25x18 előjeles szorzó
- 48 bites összeadó/kivonó/akku
- 48 bites logikai műveletek
- Pipeline regiszterek → magas elérhető órajel
- Mintadetektor a kimeneten
- SIMD (12/24/48 bit)
- Dedikált kaszkádosító portok
- Elő-összeadó

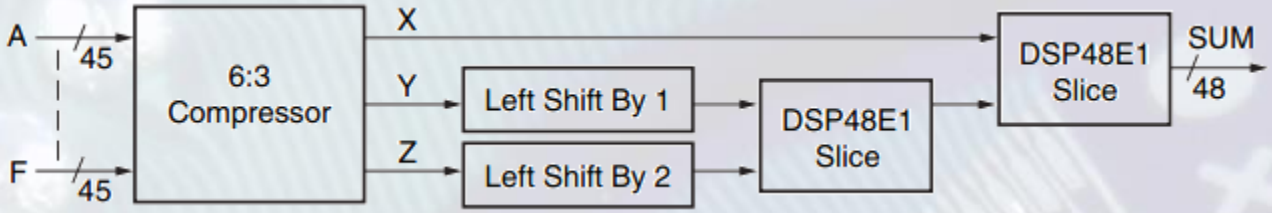
DSP48E1 elő-összeadó

- **B bemenet: 2 FF szint**
- **A bemenet: 2 FF szint**
- **D bemenet: egy FF szint az elő-összeadó előtt**
- **AD FF: elő-összeadó kimeneti FF**



6-bemenetű összeadó DSP48-cal

- **6:3 Compressor + 2 DSP48s**

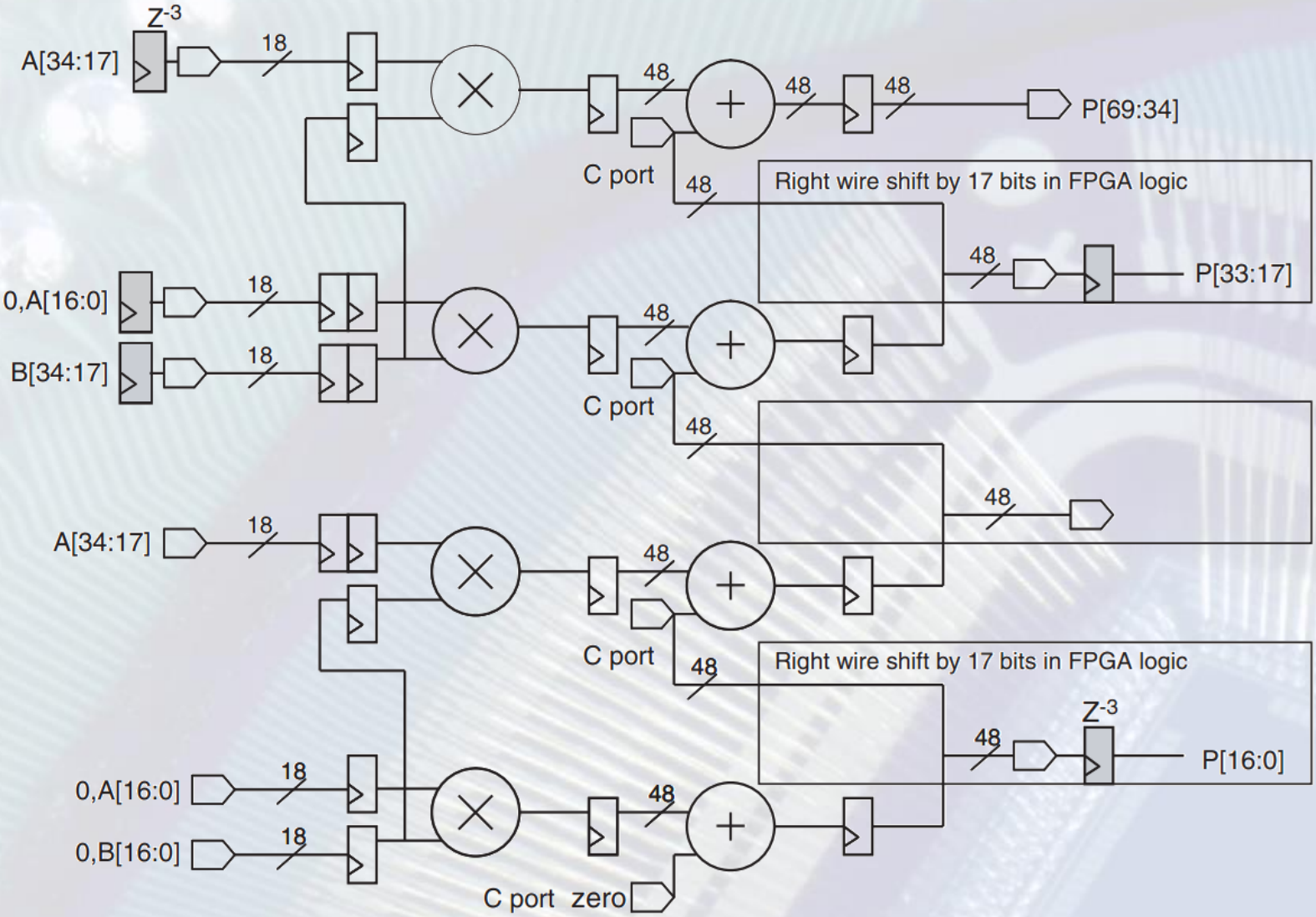


X = A^B^C^C^D^E^F

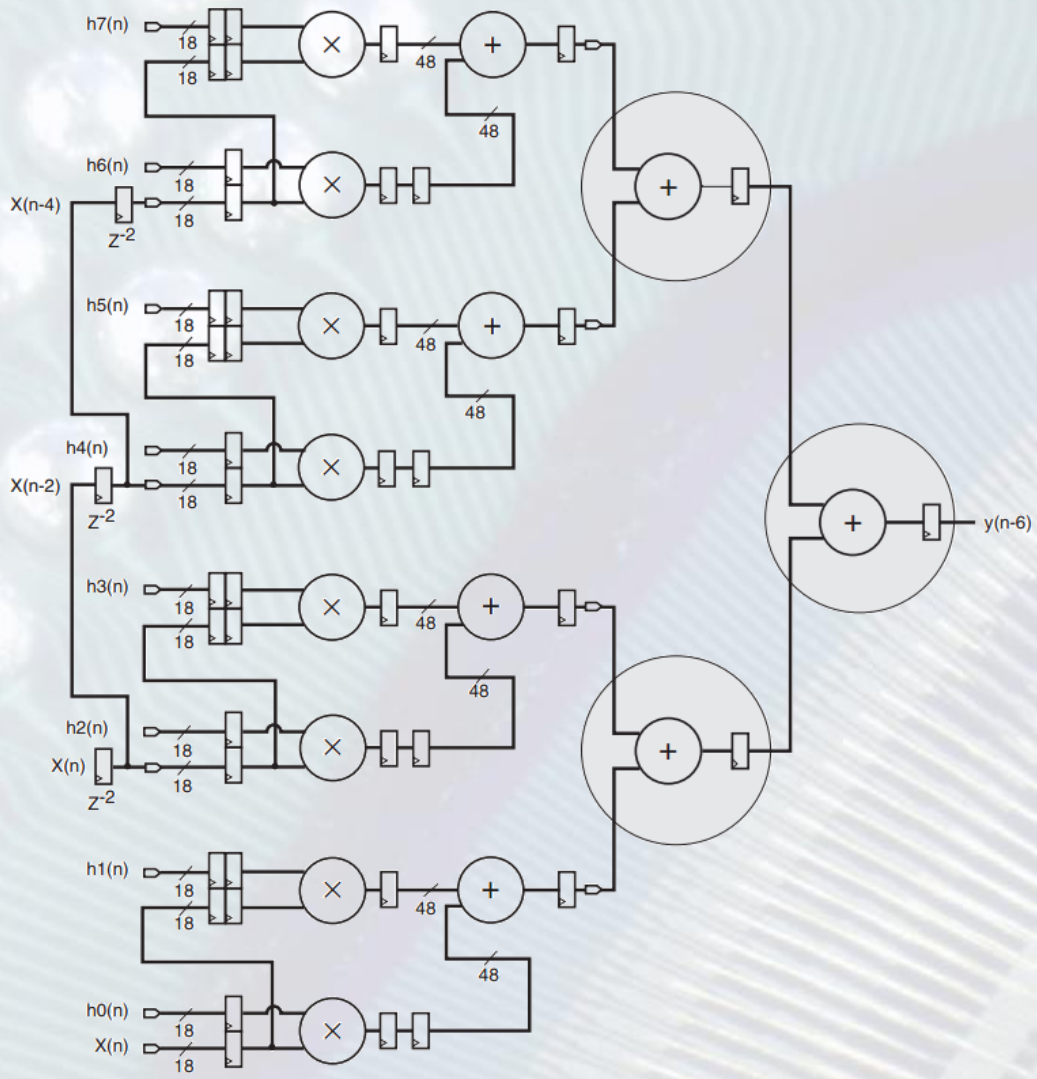
Y = AB^AC^AD^AE^AF^
BC^BD^BE^BF^
CD^CE^CF^DE^DF^EF

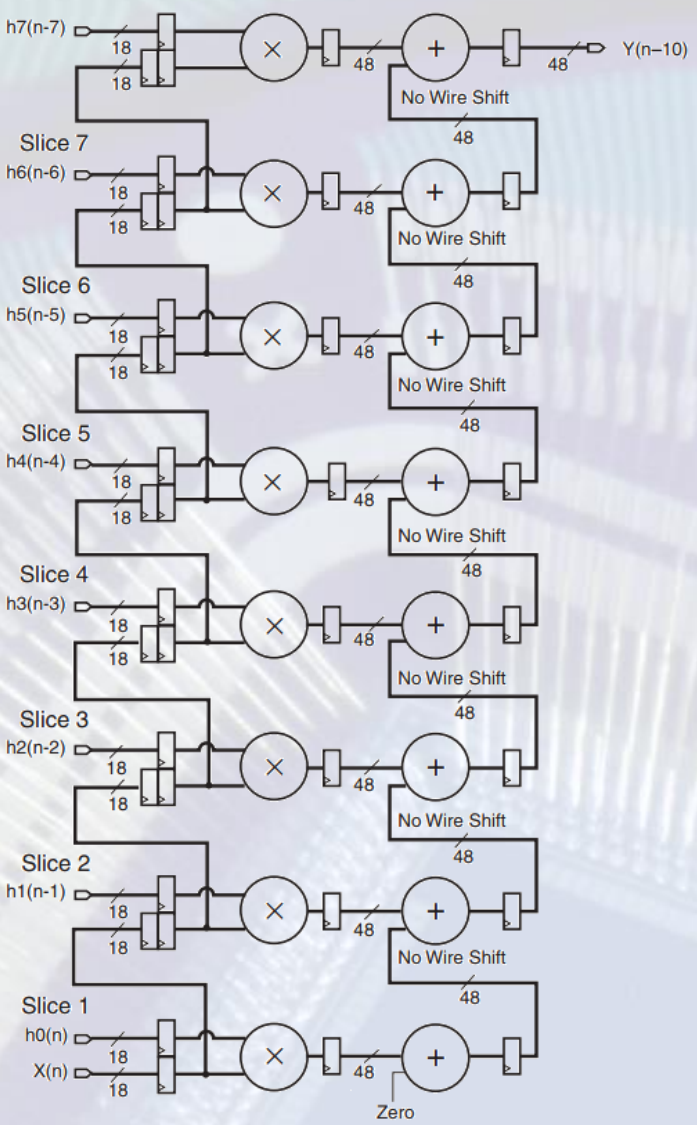
Z = ABCD | ABCE | ABCF | ABDE | ABDF | ABEF |
ACDE | ACDF | ACEF | ADEF | BCDE | BCDF |
BCEF | BDEF | CDEF

35x35 bit szorzó



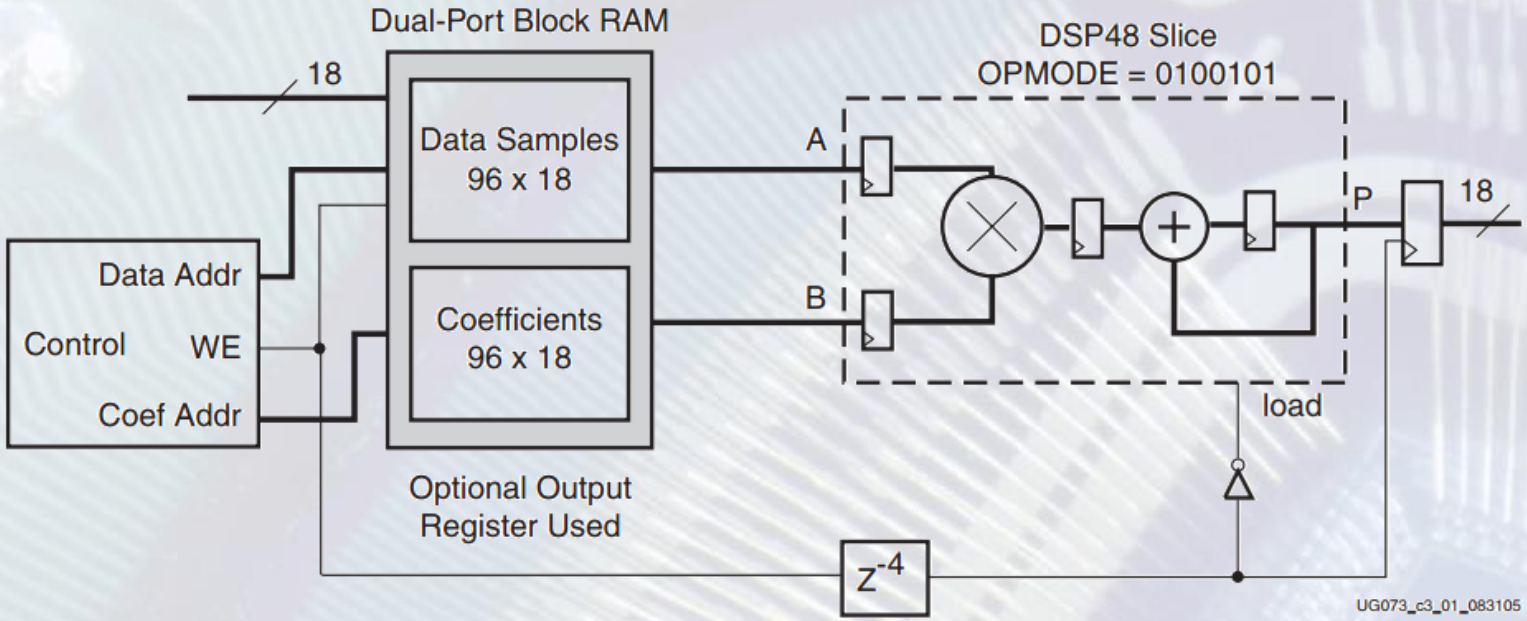
DSP – Adder Tree vs. cascade





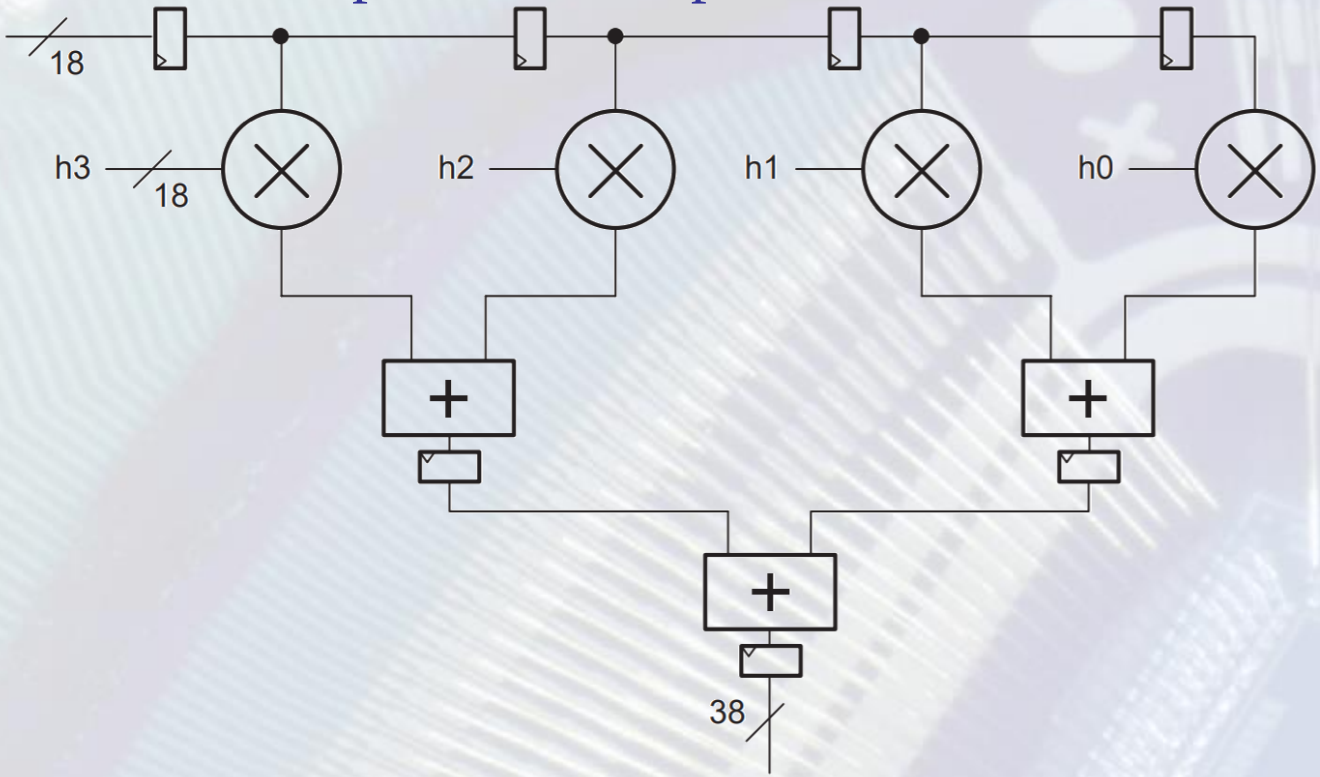
1-MAC FIR

- **Mintavételi frekvencia << órajel (fs*N<fclk)**



Párhuzamos FIR (1)

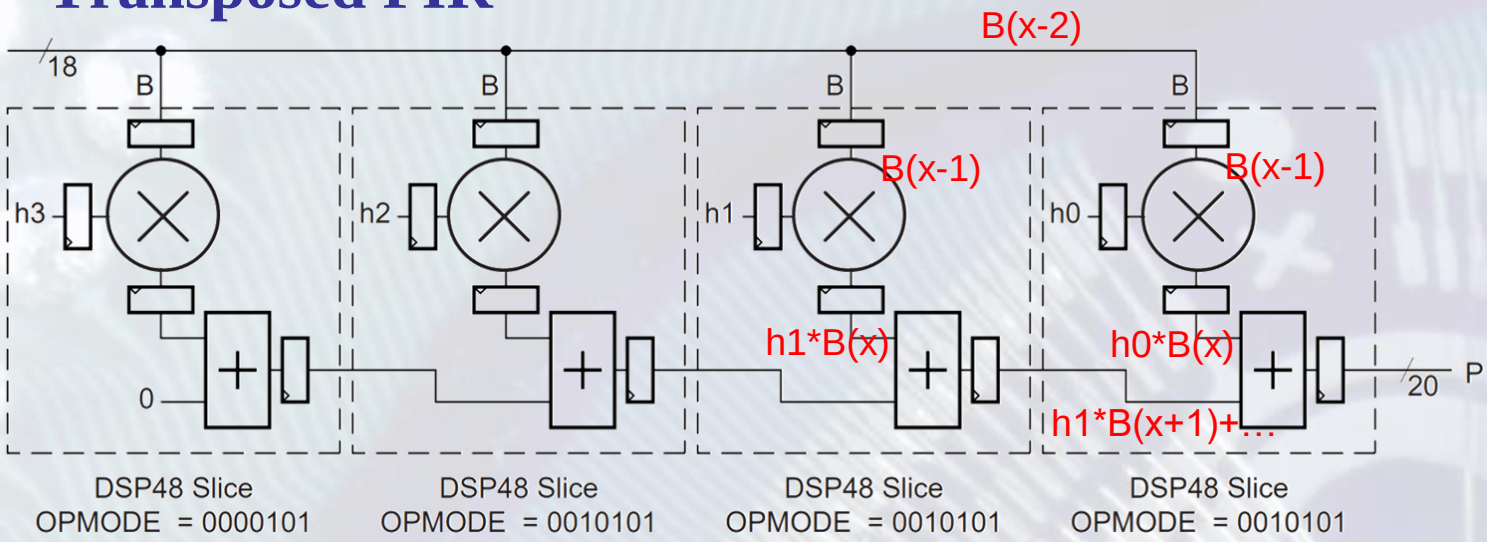
- **Mintavételi frekvencia ~= órajel**
  - Konvolúció képletének leképzése:



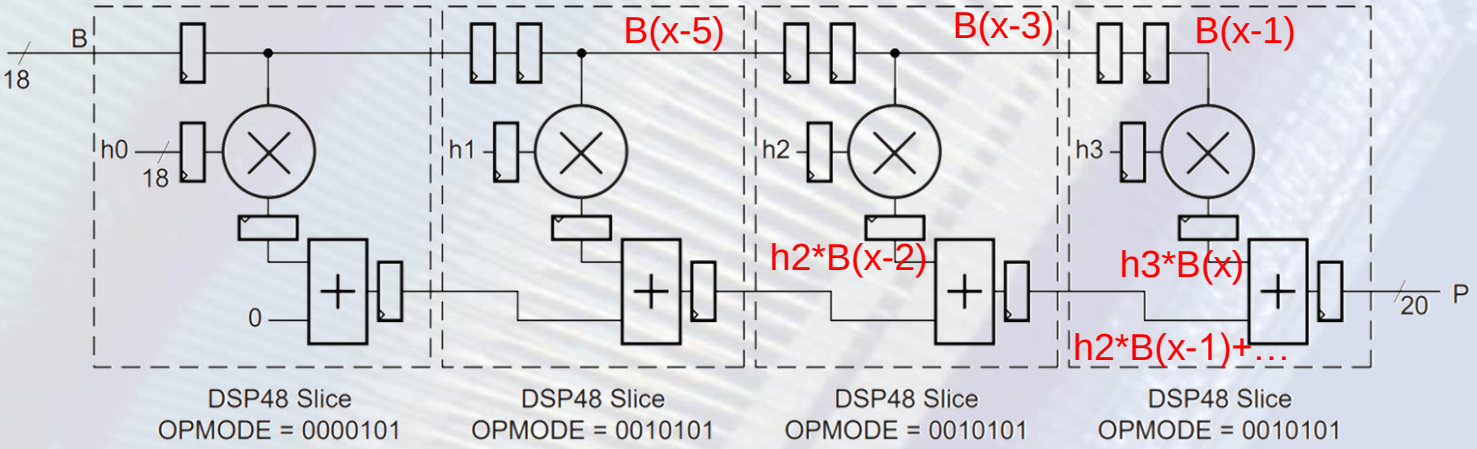
  - Nem illeszkedik jól a DSP blokk felépítéséhez

Párhuzamos FIR (2)

- **Transposed FIR**

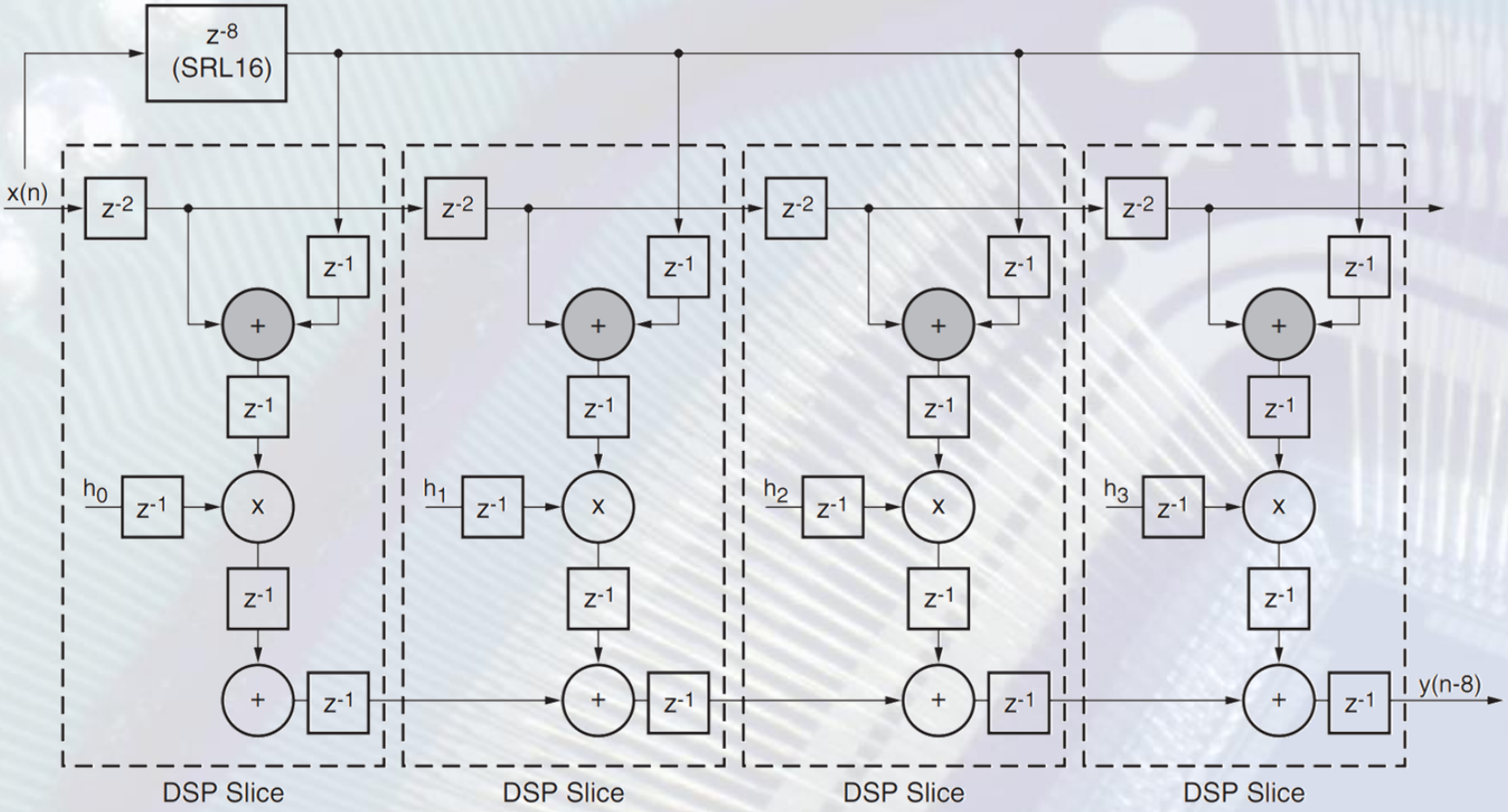


- **Systolic FIR**



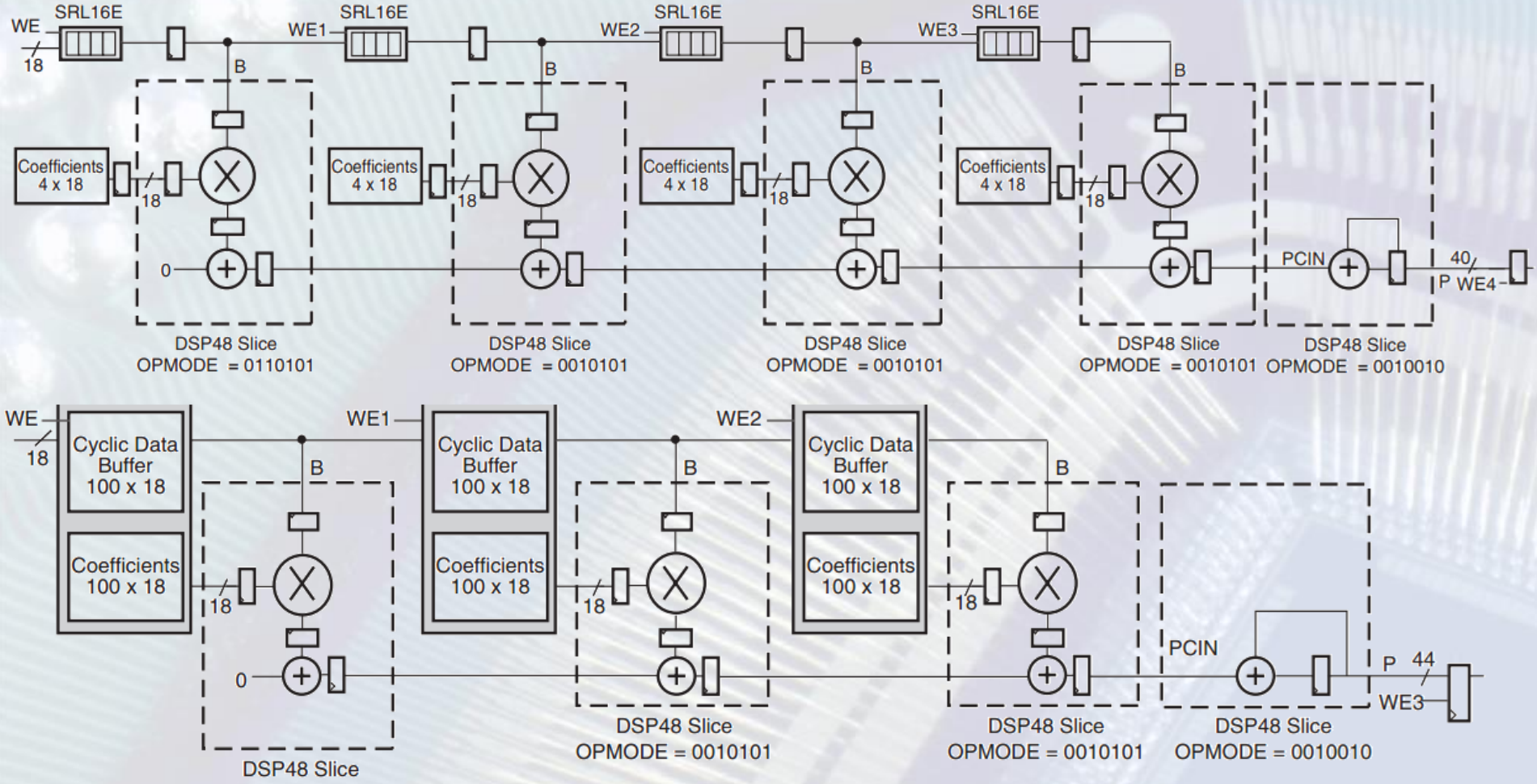
Szimmetrikus szűrő

- **Systolic FIR + pre-adder**



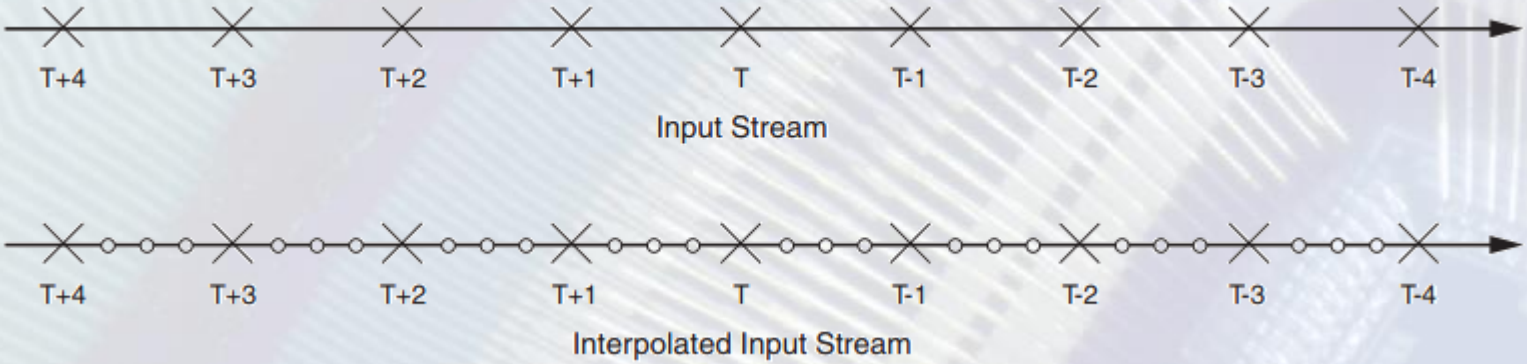
Részben párhuzamos FIR

- **Egyéb esetekben**



Interpoláció (1)

- **Bemeneti mintavételi frekvencia egész számú többszörözése (N-szerezése)**
  - Új minták beillesztése (N-1 db 0)
  - Aluláteresztő szűrő Fout/2/N vágási frekvenciával
    - A 0 értékű mintákkal NEM szorzunk!



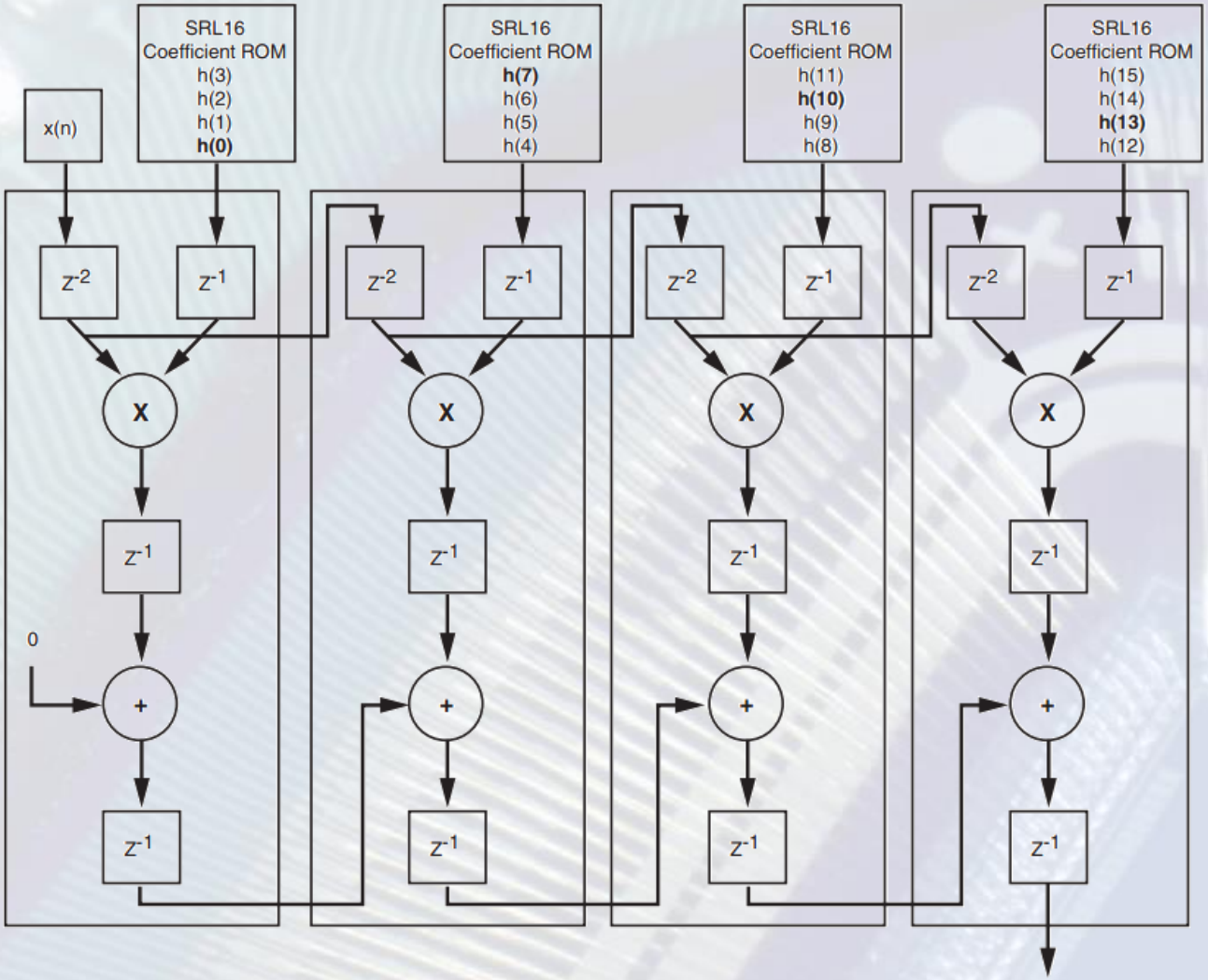
Interpoláció (2)

- **Kimeneti minták generálása:**
  - 16 együttható, 4x interpoláció → 16/4=4 MAC

	h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15
clk1	x1	0	0	0	x2	0	0	0	x3	0	0	0	x4	0	0	0
2		x1	0	0	0	x2	0	0	0	x3	0	0	0	x4	0	0
3			x1	0	0	0	x2	0	0	0	x3	0	0	0	x4	0
4				x1	0	0	0	x2	0	0	0	x3	0	0	0	x4
5					x1	0	0	0	x2	0	0	0	x3	0	0	0
6						x1	0	0	0	x2	0	0	0	x3	0	0
7							x1	0	0	0	x2	0	0	0	x3	0
8								x1	0	0	0	x2	0	0	0	x3
9									x1	0	0	0	x2	0	0	0

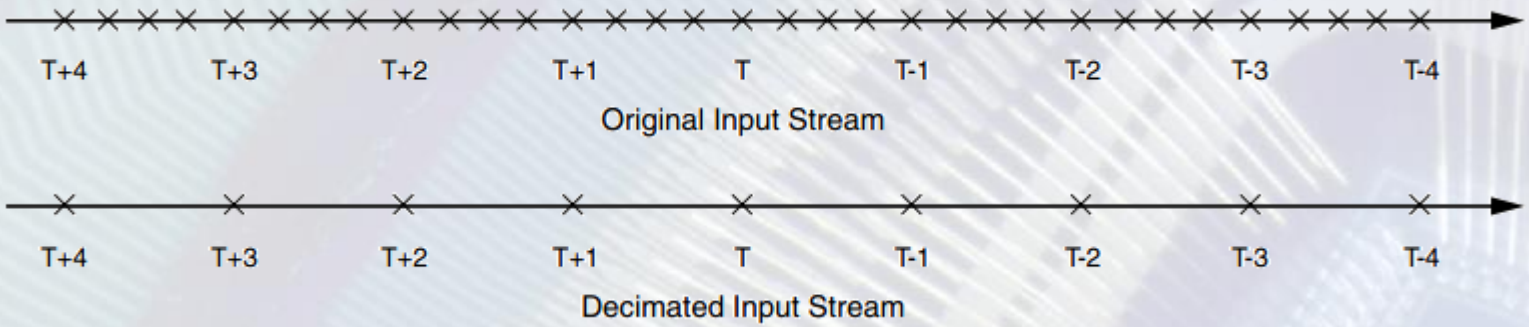
- **Ez a Polyphase filter**

Párhuzamos interpoláló FIR



Decimálás (1)

- **Mintavételi frekvencia N-ed részére csökkentése**
  - Aluláteresztő szűrés Fin/2/N-re
  - Minden N mintából N-1 eldobása
- **Azt eldobott mintákat felesleges kiszámolni!**



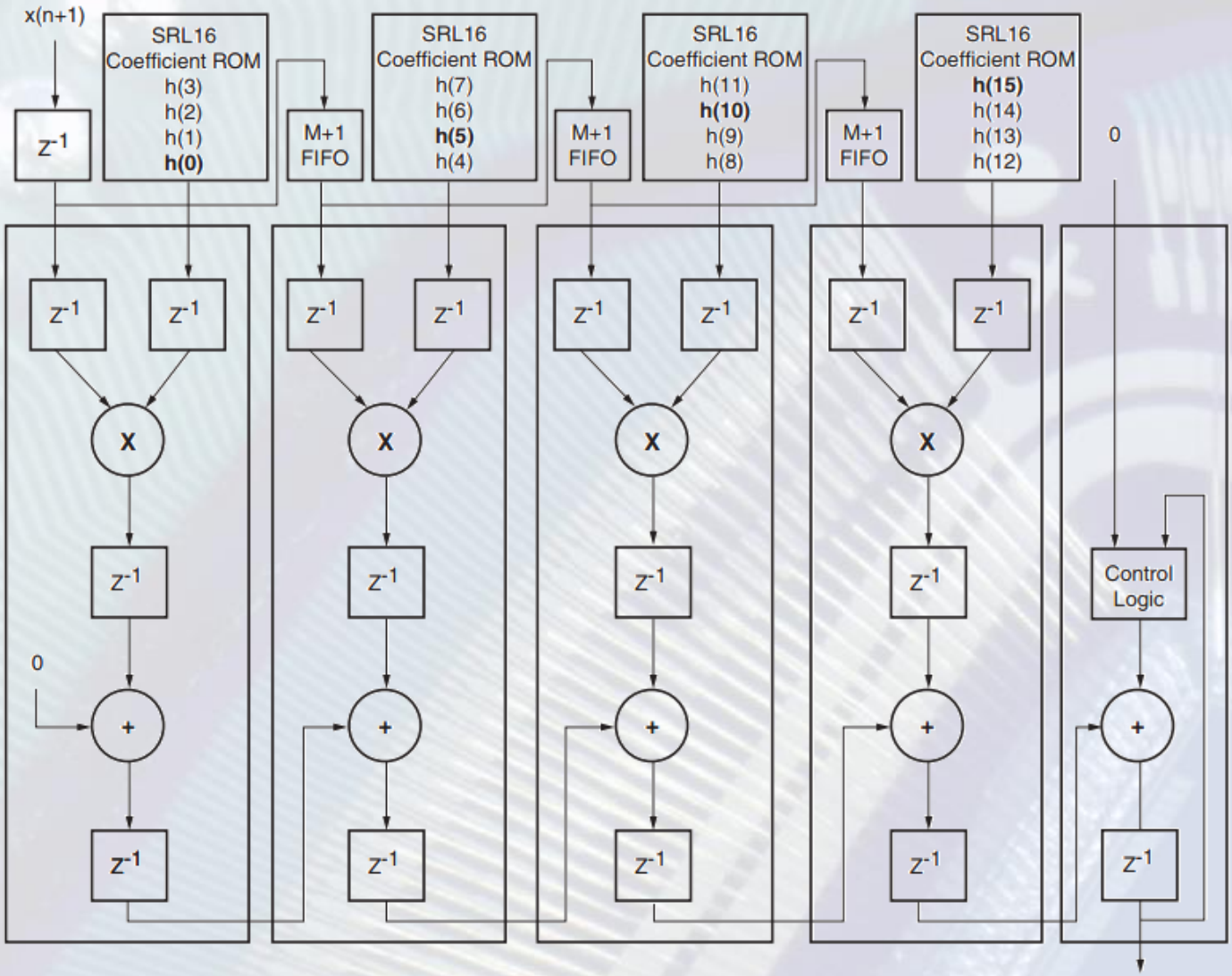
Decimálás (2)

- **Kimeneti minták generálása:**
  - 16 együttható, 4x decimálás → 16/4=4 MAC

	out1 ↑				out2 ↑				out3 ↑				out4 ↑			
coefficient	h0	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13	h14	h15
clock1	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
clock2		x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
clock3			x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14
clock4				x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13
clock5					x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12
clock6						x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
clock7							x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
clock8								x1	x2	x3	x4	x5	x6	x7	x8	x9
clock9									x1	x2	x3	x4	x5	x6	x7	x8

inputs

Párhuzamos decimáló FIR

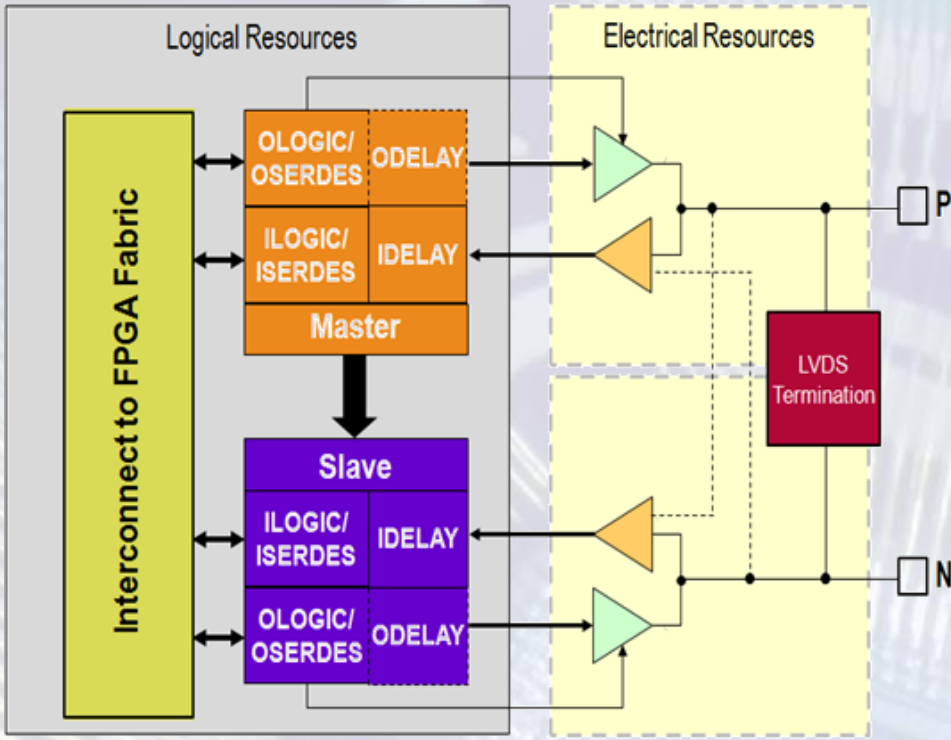


I/O interfész kihívások

- **Nagy sebességű működés, jelintegritás**
  - Forrás-szinkron interfészek (source synchronous)
  - Rendszer-szinkron interfészek (közös rendszer órajel)
  - Vonalak megfelelő terminációja (visszaverődések elkerülése)
- **Párhuzamos buszok meghajtása és mintavételezése**
  - Busz skew és órajel skew kompenzáció
  - Soros→párhuzamos és párhuzamos→soros konverzió
  - Nagy adatsebesség (> 1 Gbps) elérése
- **Single Data Rate (SDR) és Double Data Rate (DDR) interfészek**
- **Sok különböző I/O szabvány**
  - Feszültség, meghajtás erősség, protokoll

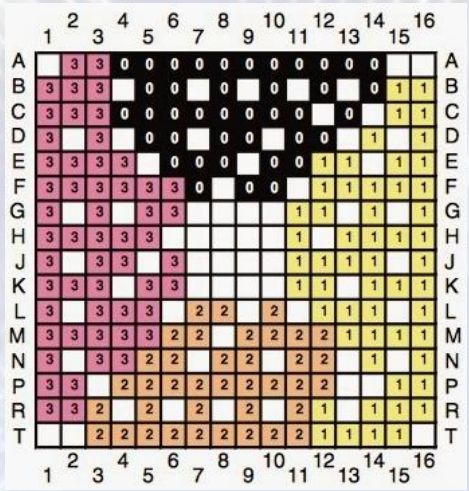
7-es sorozatú FPGA I/O

- **I/O feszültségek széles skálája**
  - 1,2V… 3,3V működés
- **Különböző I/O szabványok támogatása**
  - Unipoláris és differenciális
  - Referencia feszültség
  - 3-állapotú vonalak
- **Nagy frekvenciás interfészek**
  - Max. 1600 Mbps LVDS
  - Max. 1866 Mbps unipoláris DDR3 memóriához
- **Memória interfész**
  - Hardver támogatás DDR3 és QDRII+ memóriákhoz
- **Digitálisan vezérelt impedancia**
- **Fogyasztás csökkentés**



I/O erőforrások

- **Several I/O standards, e.g.**
  - Unipoláris (Single ended)
    - LVCMOS, LVTTL, HSTL, SSTL, PCI
  - Differenciális (Differential)
    - LVDS, RSDS, TMDS, Differential HSTL & SSTL
  - Vccaux (per bank)
    - IOB meghajtók, erősítők, komparátorok, LVDS bias generátor
  - Vcco (per bank)
    - I/O feszültség
  - Kimeneti meghajtás erősség & slew rate
  - PULLUP, PULLDOWN, KEEPER



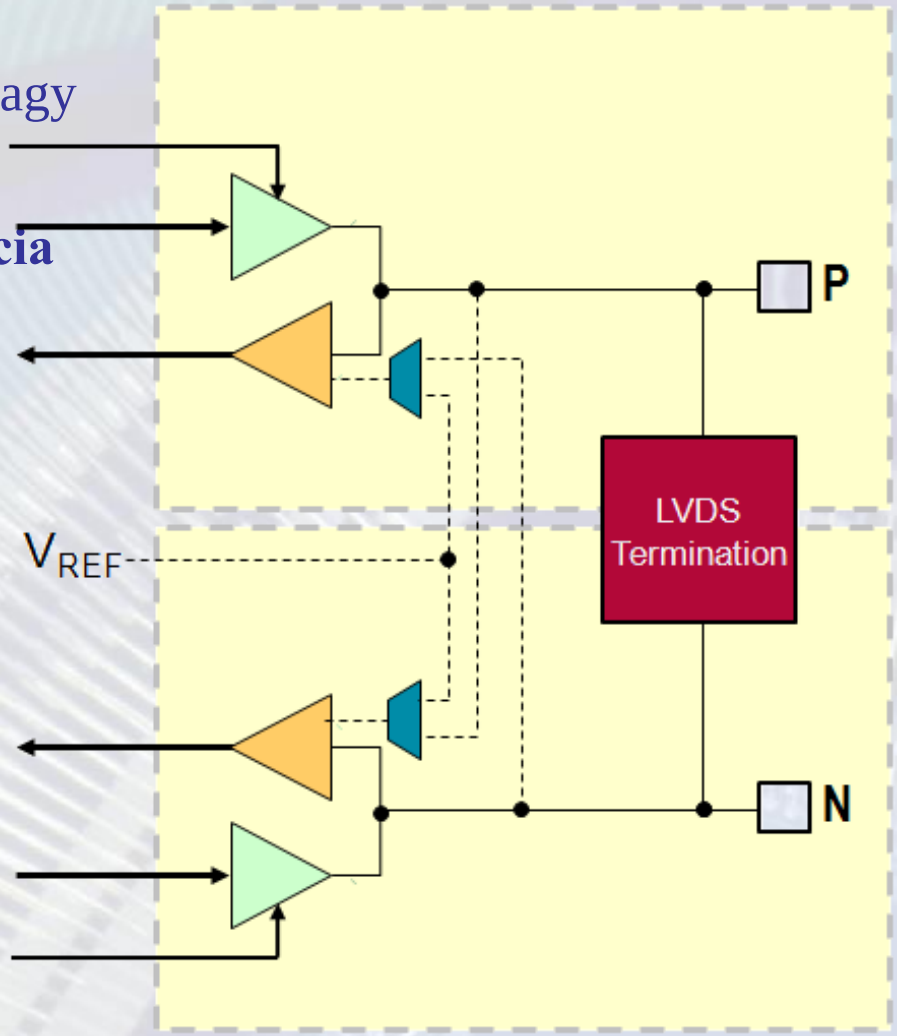
I/O típusok

- **Két féle I/ blokk**
  - High Range (HR)
    - Max. 3,3V Vcco I/O sztenderd
  - High Performance (HP)
    - Max. 1,8V Vcco
    - Nagy átviteli sebesség
    - ODELAY és DCI támogatás

I/O	Artix-7	Kintex-7	Virtex-7	Virtex-7 XT/HT
High Range	Minden	Sok	Néhány	
High Performance		Néhány	Sok	Minden

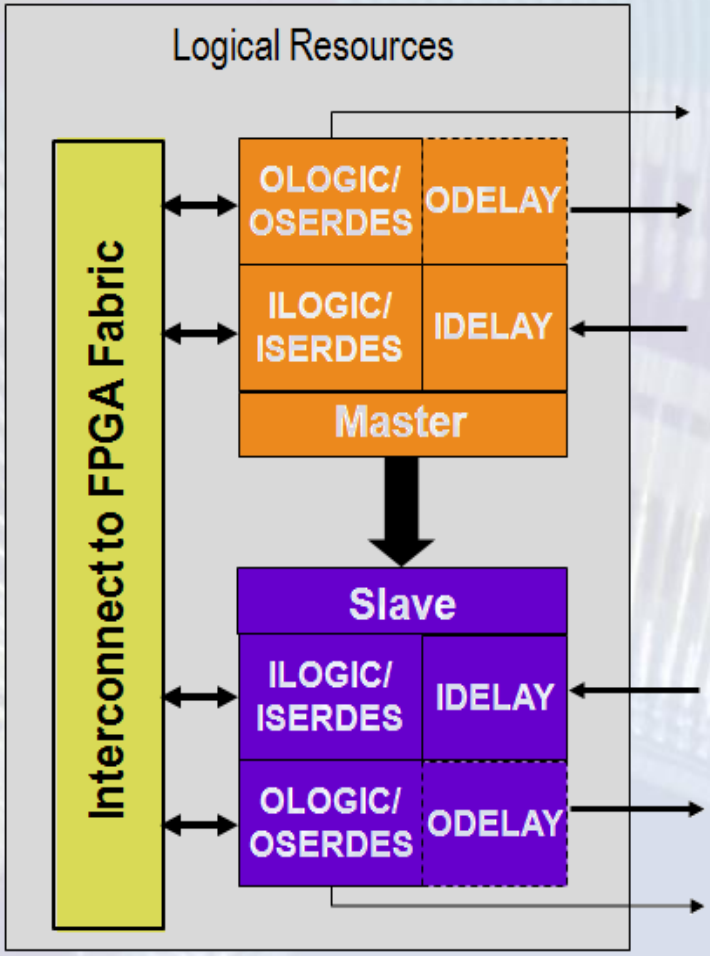
I/O erőforrások (elektromos)

- **P és N lábak konfigurációja**
  - Egyedi (különálló) unipoláris I/O vagy
  - Differenciális pár
- **Vevő sztenderd CMOS vagy referencia feszültség alapú**
  - Sztenderd CMOS CMOS
    - Logikai 0 föld „közelében”
    - Logikai 1 VCCO „közelében”
- **Referencia feszültség (VREF) alapú**
  - Logikai 0 VREF alatt
  - Logikai 1 VREF felett
- **Differenciális**
  - Logikai 0: VP < VN
  - Logikai 1: VP > VN



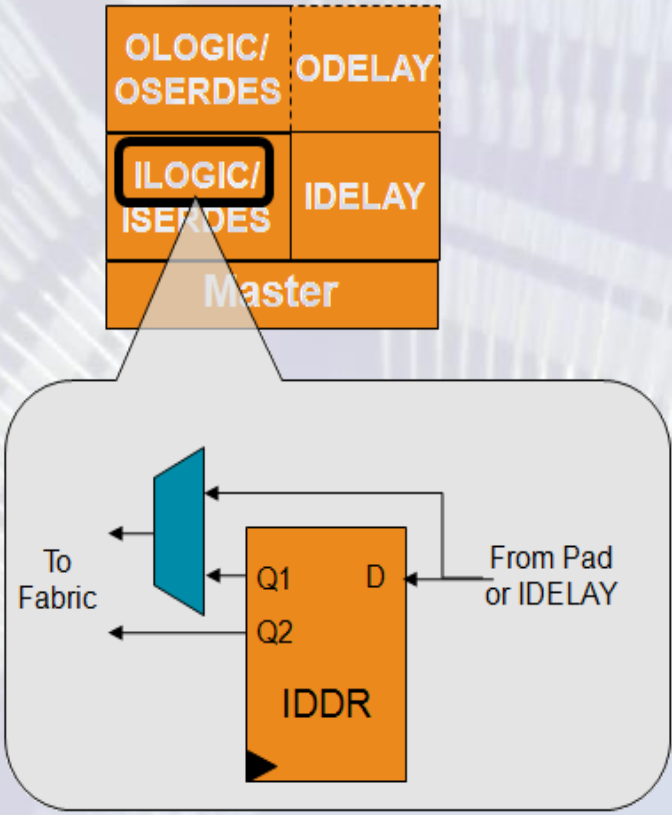
I/O erőforrások (logika)

- **I/O páronként két blokk**
  - Master és slave
  - Különálló vagy összekapcsolt működés
- **Minden blokk tartalmaz**
  - ILOGIC/ISERDES
    - SDR, DDR, nagy sebességű soros→párhuzamos átalakító
  - OLOGIC/OSERDES
    - SDR, DDR, nagy sebességű párhuzamos→soros átalakító
  - IDELAY
    - Finom hangolású bemeneti késleltetés
  - ODELAY
    - Finom hangolású kimeneti késleltetés
    - CSAK HP I/O blokkban



ILOGIC: bemeneti SDR és DDR logika

- **Két típusú ILOGIC blokk**
  - ILOGICE2 a High Performance I/O bankban
  - ILOGICE3 a High Range I/O bankban
- **ILOGIC bemenete a bemeneti buffer kimenete**
  - Vagy közvetlenül, vagy IDELAY-en keresztül
- **ILOGIC kimenete az FPGA CLB-khez kapcsolódik**
  - Közvetlenül (nincs mintavételezés az I/O blokkban)
  - IDDR flip-flop-on keresztül
  - SDR módban fel- vagy lefutó élre mintavételezve
  - DDR módban mindkét élre mintavételez

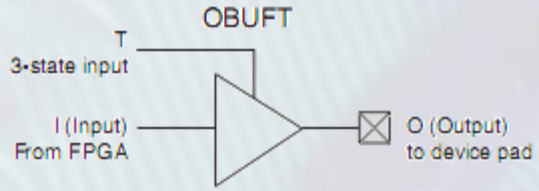


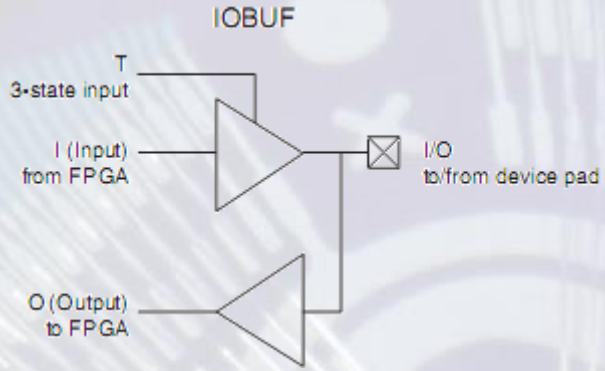
DCI

- **Digitally Controlled Impedance**
  - Chip-en belüli impedancia illesztett lezárás
  - Egy külső referencia ellenállás értékét „másolja”
    - Ellenállás: VRN/VRP lábak között
  - Folyamatos kompenzáció
    - Hőmérséklet
    - Feszültség
  - Soros és/vagy párhuzamos lezáró ellenállás

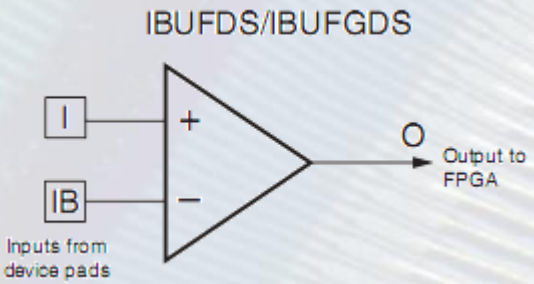
I/O primitívek

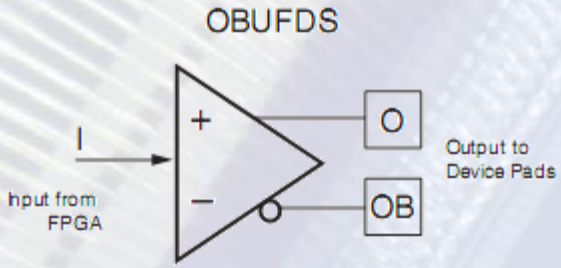
- **Single ended**
  - IBUF, IBUFG, OBUF, OBUFT, IOBUF





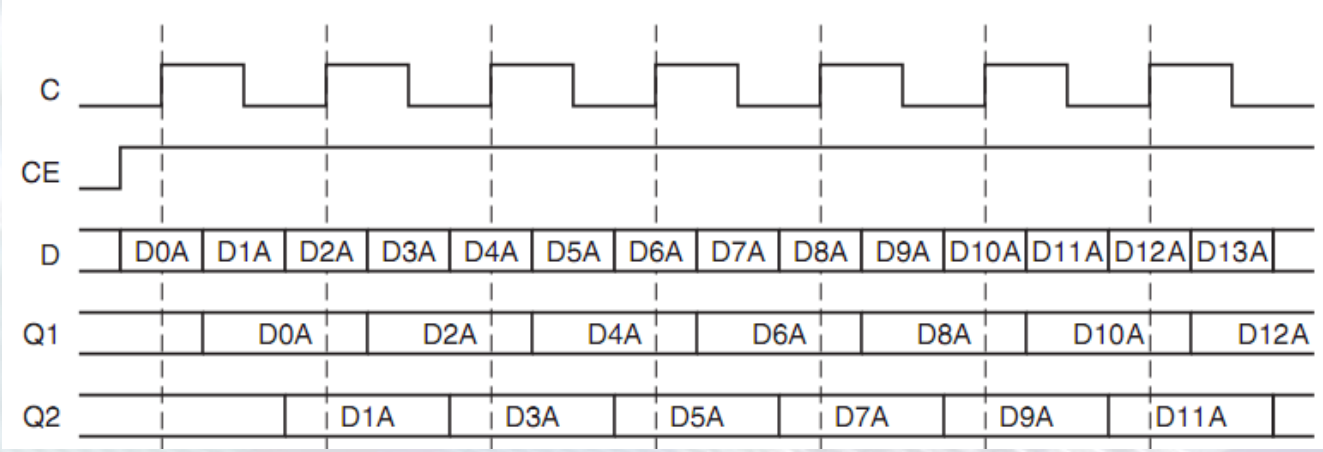
- **Differenciális**
  - IBUFDS, IBUFGDS, OBUFDS, OBUFTDS, …





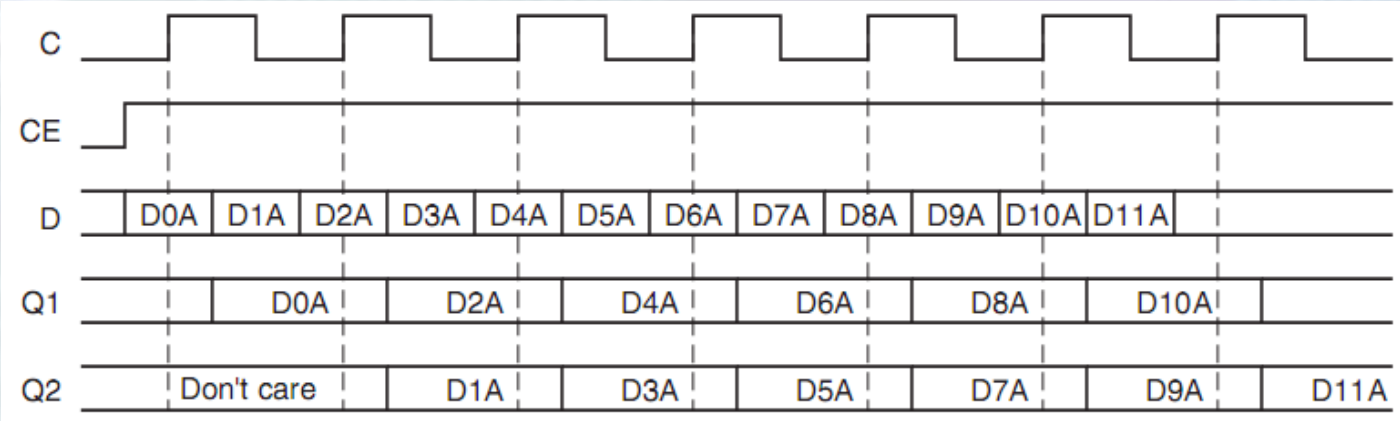
IDDR FF

- **DDR OPPOSITE_EDGE**

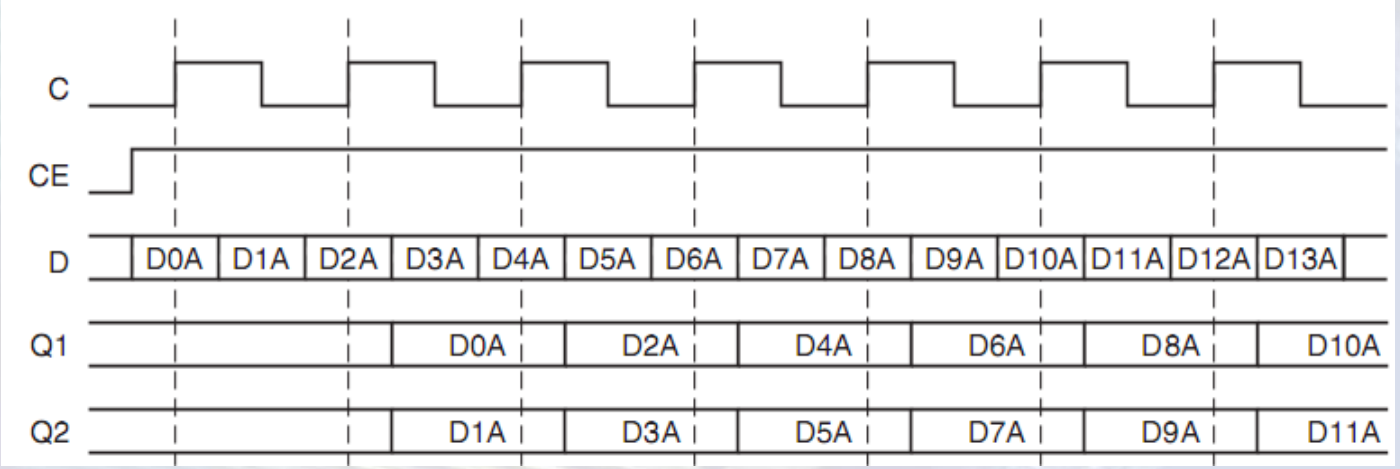


IDDR FF

- **DDR SAME_EDGE**

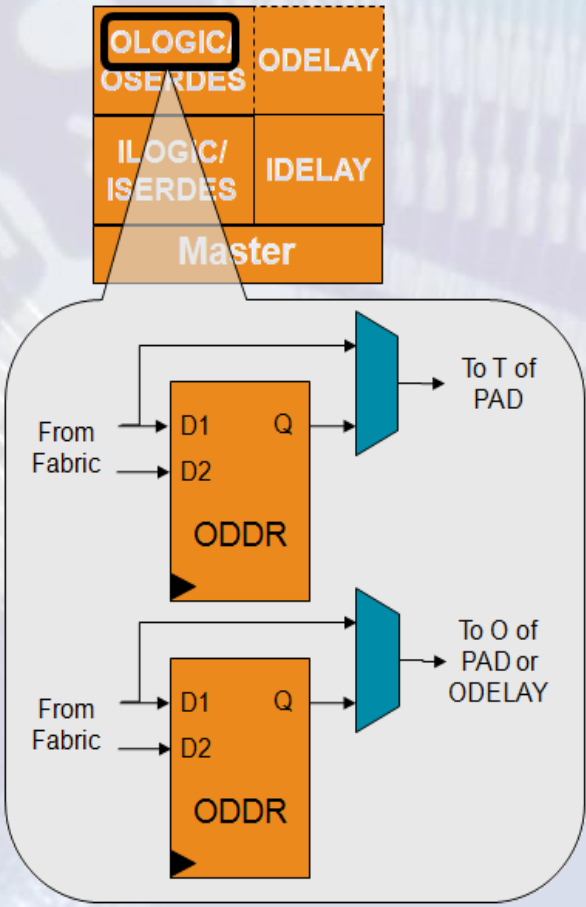


- **DDR SAME_EDGE_PIPELINED**



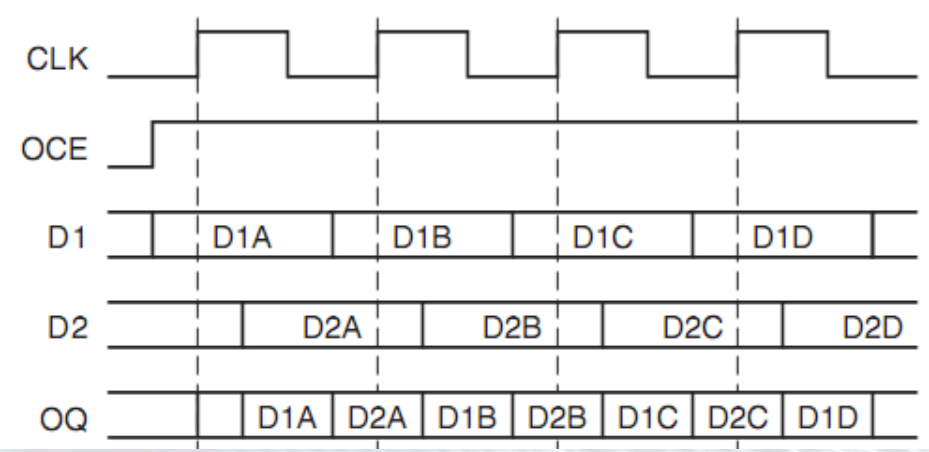
OLOGIC: kimeneti SDR és DDR logika

- **OLOGICE2 a HP bankokban, OLOGICE3 a HR bankokban**
- **Az OLOGIC kimenete a kimeneti bufferre csatlakozik**
  - Közvetlenül
  - ODELAY-en keresztül (csak HP bankokban)
- **Output is driven directly from the fabric**
  - Közvetlenül (nincs FF), SDR FF-on kersztül vagy DDR FF-on keresztül
- **Minden OLOGIC blokkban két ODDR FF**
  - Adatút
  - 3-állapotú meghajtás
  - Közös órajel és reset
- **Csak SAME_EDGE vagy OPPOSITE_EDGE mód**

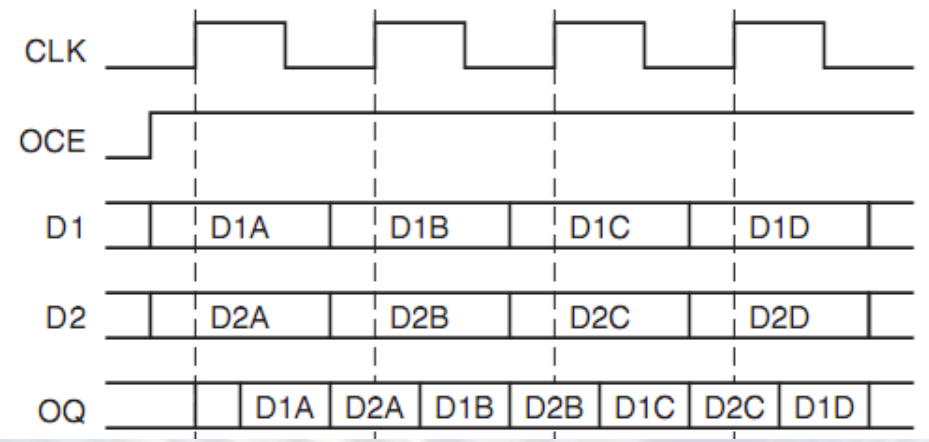


ODDR FF

- **OPPOSITE_EDGE**

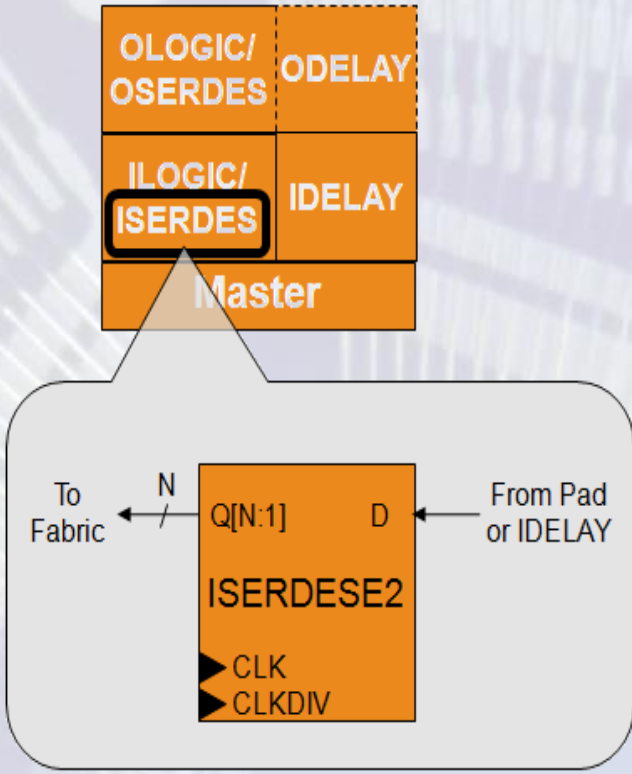


- **SAME_EDGE**



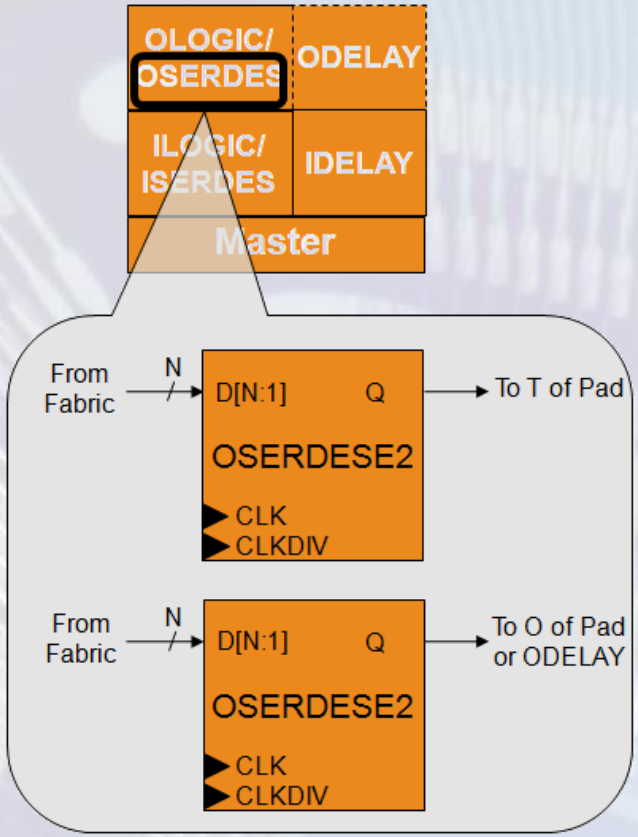
ISERDES: bemeneti soros→párhuzamos konverzió

- **Bemenet közvetlenül a bemeneti bufferből vagy IDELAY-en keresztül**
  - D-t a nagy sebességű órajelre mintavételezi (CLK)
  - SDR és DDR mód
- **A párhuzamos adatkimenet a CLB-khez csatlakozik**
  - Q kimenet a leosztott órajellel (CLKDIV) szinkron
- **CLK és CLKDIV órajelek fázisa azonos!**
  - Párhuzamosítás
    - Single data rate: 2, 3, 4, 5, 6, 7, 8
    - Double data rate: 4, 6, 8
- **Két ISERDES kaszkádosítható**
  - Double data rate: 10, 14
  - Két I/O blokkot igényel!
- **BITSLIP: szó határ állítás**



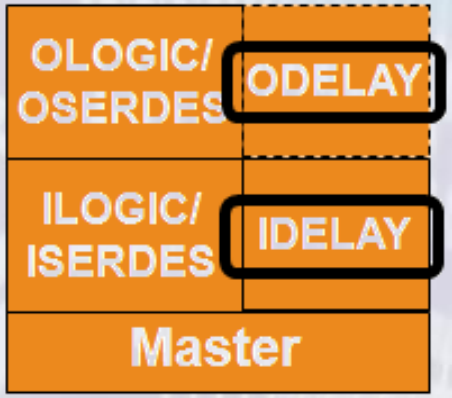
OSERDES: kimeneti párhuzamos→soros átlakítás

- **Kimenete a kimeneti buffer-t vagy ODELAY-t hajtja**
  - Q kimenet a nagy sebességű soros (CLK) órajellel szinkron
  - SDR és DDR üzemmód
- **Párhuzamos adat a CLB-ktől érkezik**
  - D bemenet a leosztott órajellel (CLKDIV) szinkron
- **CLK és CLKDIV órajelek fázisa azonos!**
- **Párhuzamos→soros átalakítás**
  - Single data rate: 2, 3, 4, 5, 6, 7, 8
  - Double data rate: 4, 6, 8
- **Kaszkádosítható (két I/O blokk)**
  - Double data rate: 10, 14
- **Külön 3-állapotú sorosító**
  - Sorosítási faktor: 4
  - Az órajelek közösek



IDELAY és ODELAY

- **Külön késleltetővonal a bemeneten és kimeneten**
  - IDELAY: HR és HP bankokban
  - ODELAY: csak HP bankokban!
- **Kalibráció az IDELAYCTRL primitívvel**
  - Hőmérséklettől és feszültségtől függetlenül állandó
- **IDELAY és ODELAY: azonos tulajdonságok**
  - IDELAY elérhető a CLB-kből
- **Késleltető TAP számlálója elérhető a CLB-kből**
  - Monitorozás; inkrementálás és dekrementálás; adott értékre állítás (0…31)
- **Referencia frekvencia: 200 MHz (vagy 3-as speed grade-ben lehet 300 MHz is)**
  - Késleltetés 78 ps (vagy 52 ps) TAP-enként
  - Maximum ~2,5 ns



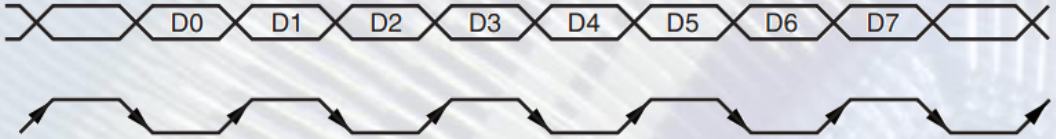
ISERDES órajel kezelés

- **A nagy sebességű órajelnek (CLK) és a leosztott órajelnek (CLKDIV) azonos fázisúnak KELL lennie**
- **Kötött órajel meghajtás**
  - PLL/MMCM nélkül:
    - CLK BUFIO-n keresztül
    - CLK_DIV BUFR-en keresztül (tud órajelet osztani)
  - Mindkét órajel ugyanabból az MMCM-ből
    - Egy BUFIO és egy BUFR
    - Két BUFG

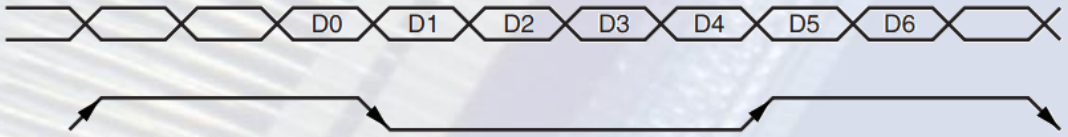
FPGA	Speed Grade	Global Clock Network Maximum (BUFG)	I/O Clock Network Maximum (BUFIO)	Regional Clock Network Maximum (BUFR)	Horizontal Clock Network Maximum (BUFH)
Artix®-7	-1	464 MHz	600 MHz	315 MHz	464 MHz
Artix-7	-2	550 MHz	680 MHz	375 MHz	550 MHz
Kintex®-7	-1	625 MHz	710 MHz	450 MHz	625 MHz
Kintex-7	-2	710 MHz	800 MHz	540 MHz	710 MHz
Virtex®-7	-1	625 MHz	710 MHz	450 MHz	625 MHz
Virtex-7	-2	710 MHz	800 MHz	540 MHz	710 MHz

Forrás szinkron interfész

- **Forrás szinkron: az adatforrás az adat mellett órajelet is továbbít**
  - „Edge aligned”: az órajel éle egybe esik az adatváltással
  - „Center aligned”: az órajel éle az adat bitidő közepére illesztett
- **Továbbított órajel frekvenciája**
  - Megegyezik (vagy DDR esetben fele) az adatsebességgel – pl. RAM, MIPI



  - A továbbított órajel le van osztva: pl. HDMI

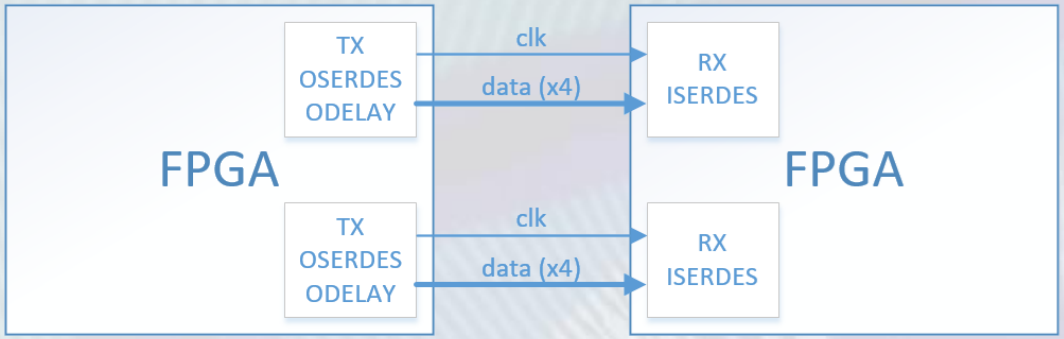


Forrás szinkron center aligned DDR interfész (1)

- **Kimenet**
  - Az adat kiadása a párhuzmos és a bit órajel 0° fázisával történik
  - Az órajel kimenetet a bit órajel 90°-os fázisával generáljuk, fix “01010101” adatmintával
- **Bemenet**
  - Elvileg egyszerű, a bejövő órajelre az adat mintavételezhető → ritkán igaz → ilyenkor is célszerű beépíteni vagy kimeneti vagy bemeneti késleltetést, és hangolni az interfészt

Forrás szinkron center aligned DDR interfész (2)

- **Rendszer:**



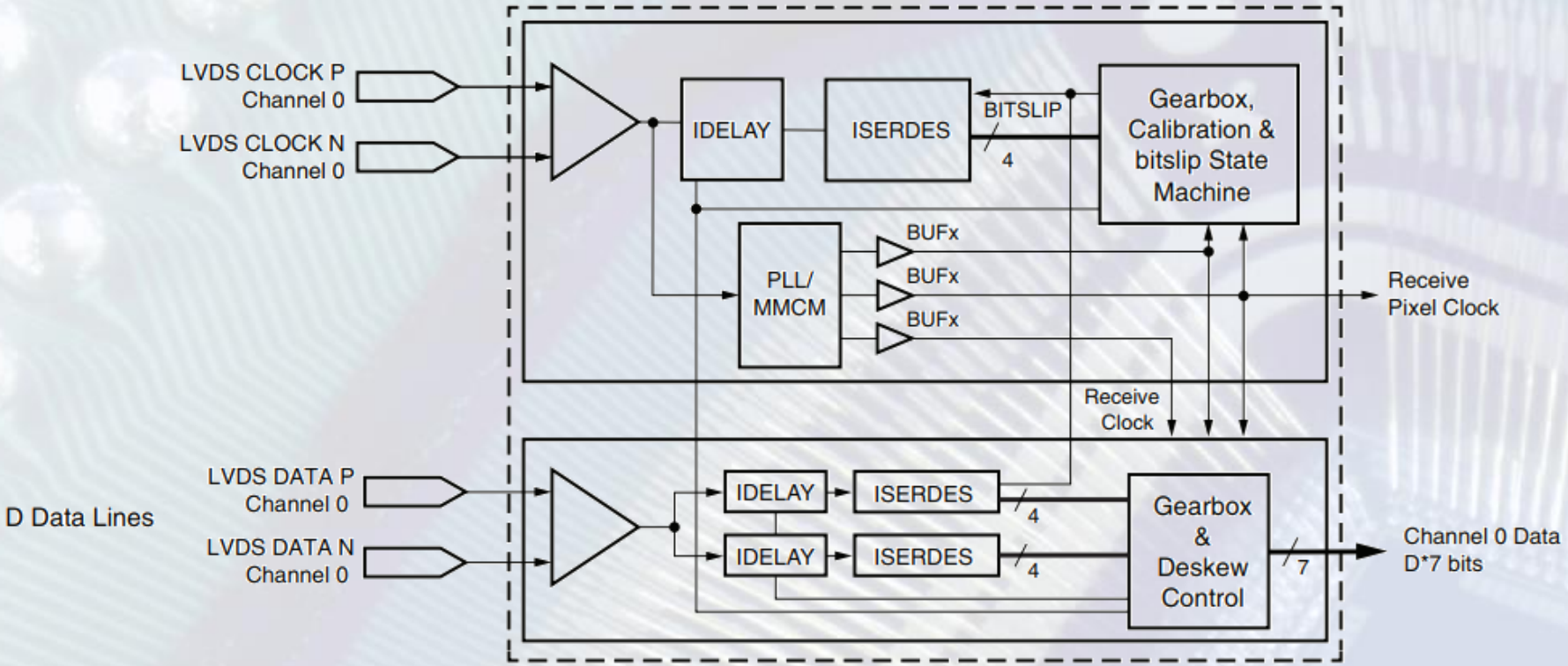
- **Küldő FPGA:**
  - Adat: 0x55
  - Bit sebesség: 800 Mbit/s → 1,25 ns bitidő
    - Generált órajel késleltetés: 15 TAP → 1,17 ns
    - Adat ODELAY késleltetés: 0…31 tartomány → 0…2,418 ns
- **Fogadó FPGA által vett adat:**

	Delay TAP																														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CH0	0xaa	0xaa	0xaa	0xaa	0xaa	Err	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	Err	0xaa	0xaa	0xaa	0xaa	0xaa	0xaa	0xaa	0xaa	0xaa	0xaa
CH1	0xaa	0xaa	0xaa	0xaa	0xaa	0xaa	Err	Err	Err	Err	Err	Err	Err	Err	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	0x55	Err	Err	Err	Err	Err	Err	Err	Err	0xaa

Forrás szinkron center aligned DDR interfész (3)

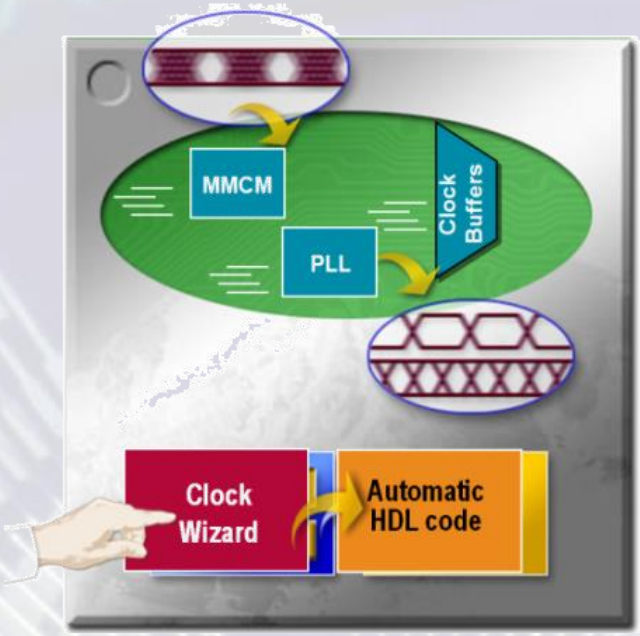
- **Mintavételezés kalibrációja adatvonalanként külön**
  - Ismert adat küldése és fogadása
    - Tipikusan 0x55 vagy 0xAA, mert ezek nem szó-határ függőek
    - Memóriák esetében sokszor van teszt adat generálási mód
  - Jó mintavételezési tartomány meghatározása
    - Küldő vagy fogadó fél késleltetésének hangolása
  - Késleltetés beállítása a megfelelő tartomány közepére
- **Szó-határ meghatározása**
  - Alkalmas adat küldése (pl. 0x01)
    - Ekkor a bitek mintavételezése már jó!
  - Szó határ illesztés: amíg a vevő 0x01 adatot nem vesz minden adatvonalon
    - Pl. Xilinx: ISERDES BITSLIP
    - Megvalósítható CLB-ben is

DDR forrás szinkron bemenet



7-es sorozat órajel kezlés

- **Global clock buffers**
  - Sok szinkron elem meghajtására optimalizált (high fanout) órajel terjesztő hálózat
- **Regionális órajel terjesztő hálózat**
  - Alacsonyabb skew
- **Órajel régiók**
  - Minden órajel régió 50 CLB magas, horizontálisan az eszköz felét fedi le
- **Clock management tile (CMT)**
  - Egy Mixed-Mode Clock Manager (MMCM) és egy Phase Locked Loop (PLL)
  - Frekvencia szintézis, de-skew, jutter szűrés
  - Nagy átfogott frekvencia tartomány (de kötött!)
- **Primitívként vagy Clocking Wizard-on keresztül**



Clock-Capable bemenetek

- **Minden szinkron tervhez kell legalább egy külső órajel forrás**
  - Ezt be kell vezetni az FPGA-ba
- **A 7-es családban minden I/O bank tartalmaz clock-capable bemeneti lábakat**
  - Ezek normál I/O lábak, de dedikált huzalozással a belső órajel erőforrásokhoz
  - Minden I/O bankban 4 clock capable láb
    - 2x Multi-Region Clock Capable (MRCC)
    - 2x Single Region Clock Capable (SRCC)

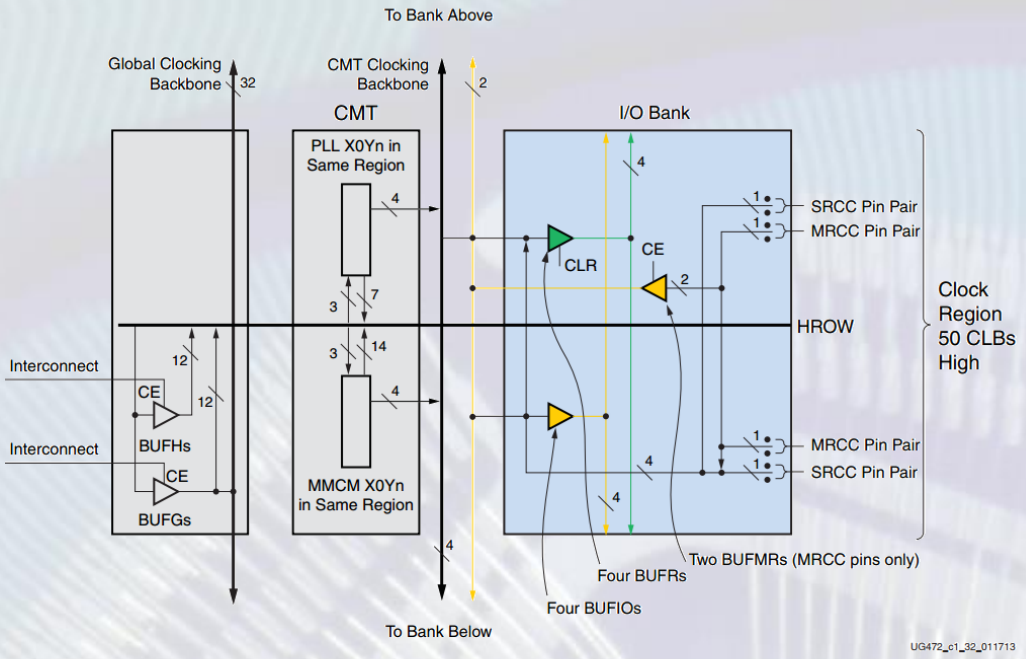
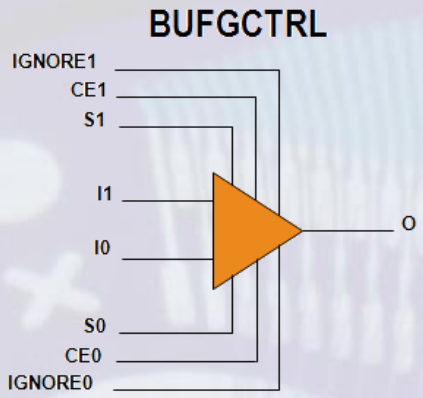


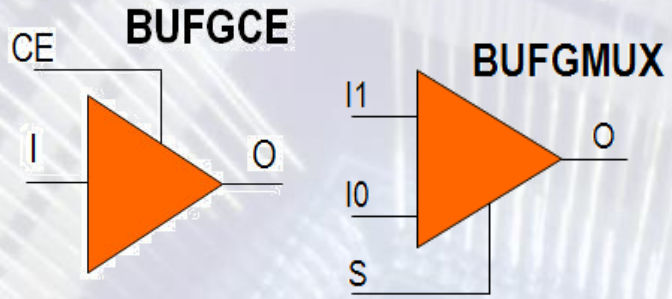
Figure 1-3: **Single Clock Region (Right Side of the Device)**

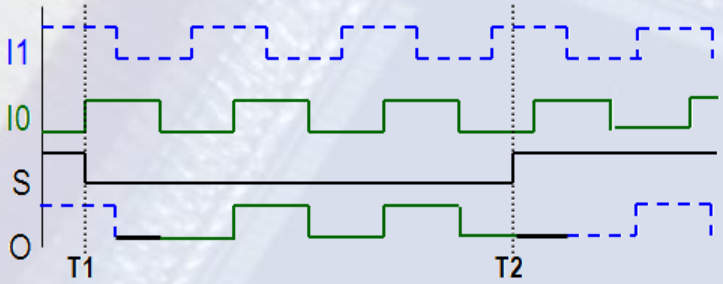
- Minden CC bemenet lehet különálló single-ended órajel bemenet vagy párosával egy differenciális órajel bemenet

Global Clock Buffer (BUFGCTRL)

- **A BUFGCTRL (BUFG) az eszköz közepén helyezkedik el**
- **Forrása lehet**
  - Clock-capable I/O (CCIO) ugyanabból az eszköz-félből
  - CMT kimenet ugyanabból az eszköz-félből
  - Gigabit transceiver órajel ugyanabból az eszköz-félből
  - Másik BUFG, BUFR, huzalozás
- **A BUFGCTRL a vertikális órajel terjesztő hálózatot hajtja meg**
- **BUFGCTRL verziók**
  - Egyszerű órajel buffer (BUFG)
  - Órajel multiplexer (BUFGMUX or BUFGMUX_CTRL)
  - Engedélyezhető órajel buffer (BUFGCE)

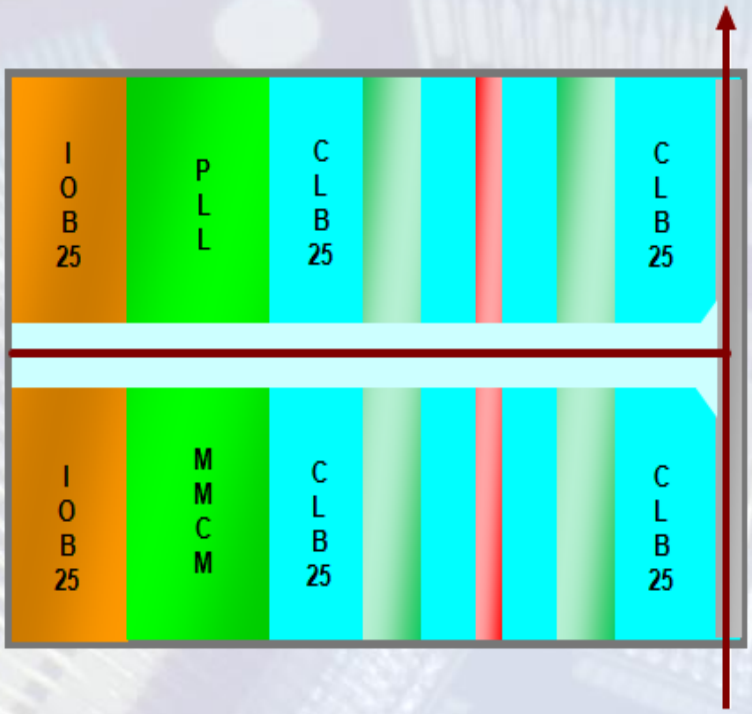






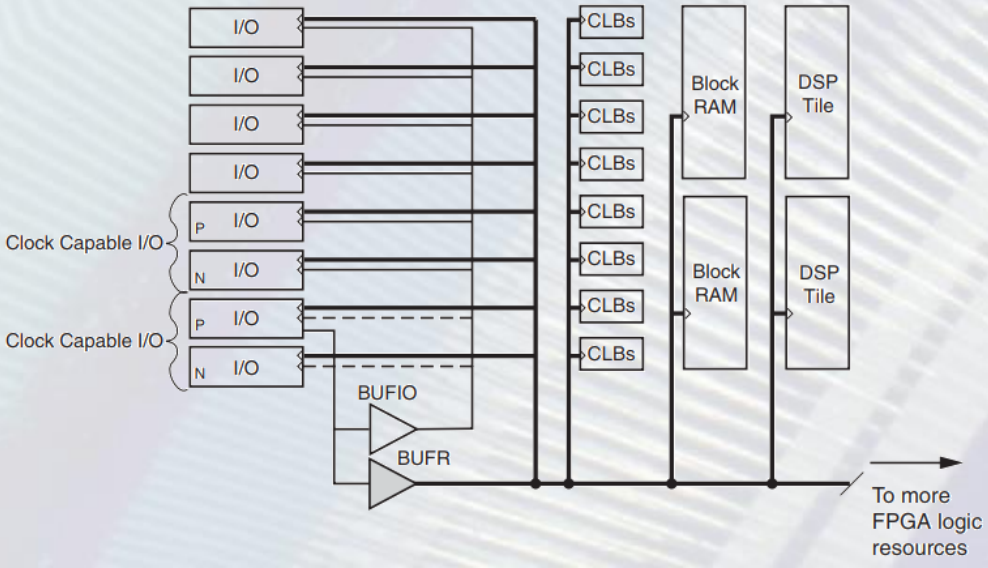
7-es sorozat órajel régiók

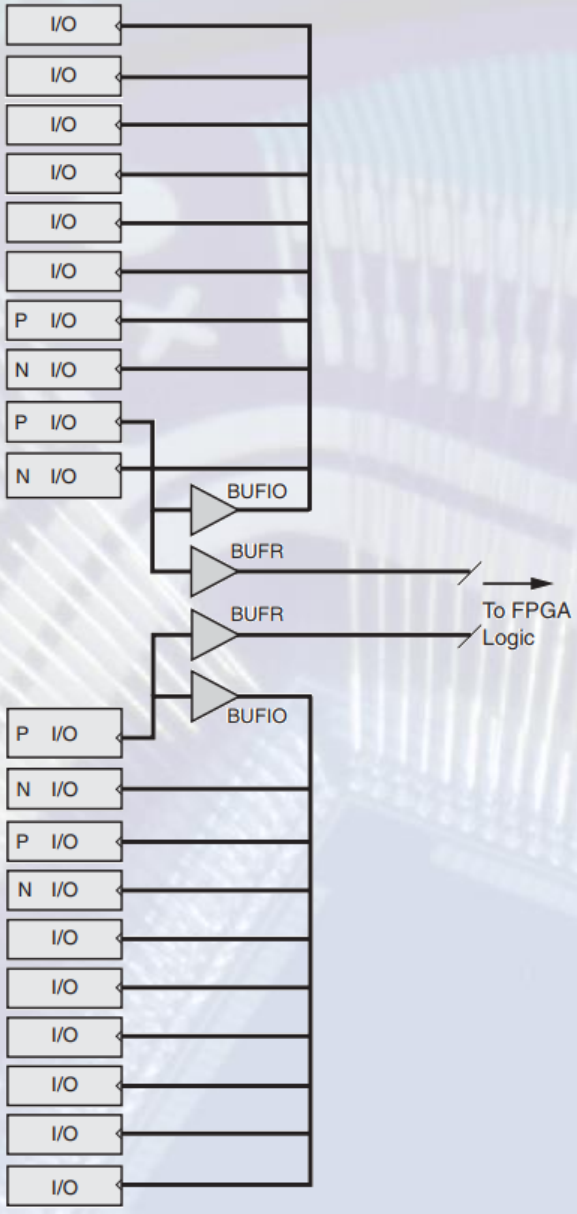
- **Az előző családoknál nagyobb régiók**
  - 50 CLB és 50 I/O magas
  - Mérete megegyezik a I/O bank méretével
  - Eszköz szélességének fele
  - Eszköztől függően 2–24 régió
- **Erőforrások egy régióban**
  - 12 órajel terjesztő hálózat
    - Meghajtás: BUFH (horizontális)
  - 4 regionális órajel terjesztő hálózat
    - Meghajtás: BUFR
      - I/O, CLB, RAM, DSP
  - 4 I/O órajel terjesztő hálózat
    - Meghajtás: BUFIO
      - Csak I/O, forrása CC láb vagy MMCM



BUFIO & BUFR

- **BUFIO**
  - Forrása CC láb vagy MMCM
  - A szomszédos I/O lábaknál használható
  - NEM használható az FPGA egyéb erőforrásaiban
- **BUFR**
  - Ugyanaz lehet a forrása, mint a BUFR-nek
  - Használható az FPGA egyéb erőforrásaihoz



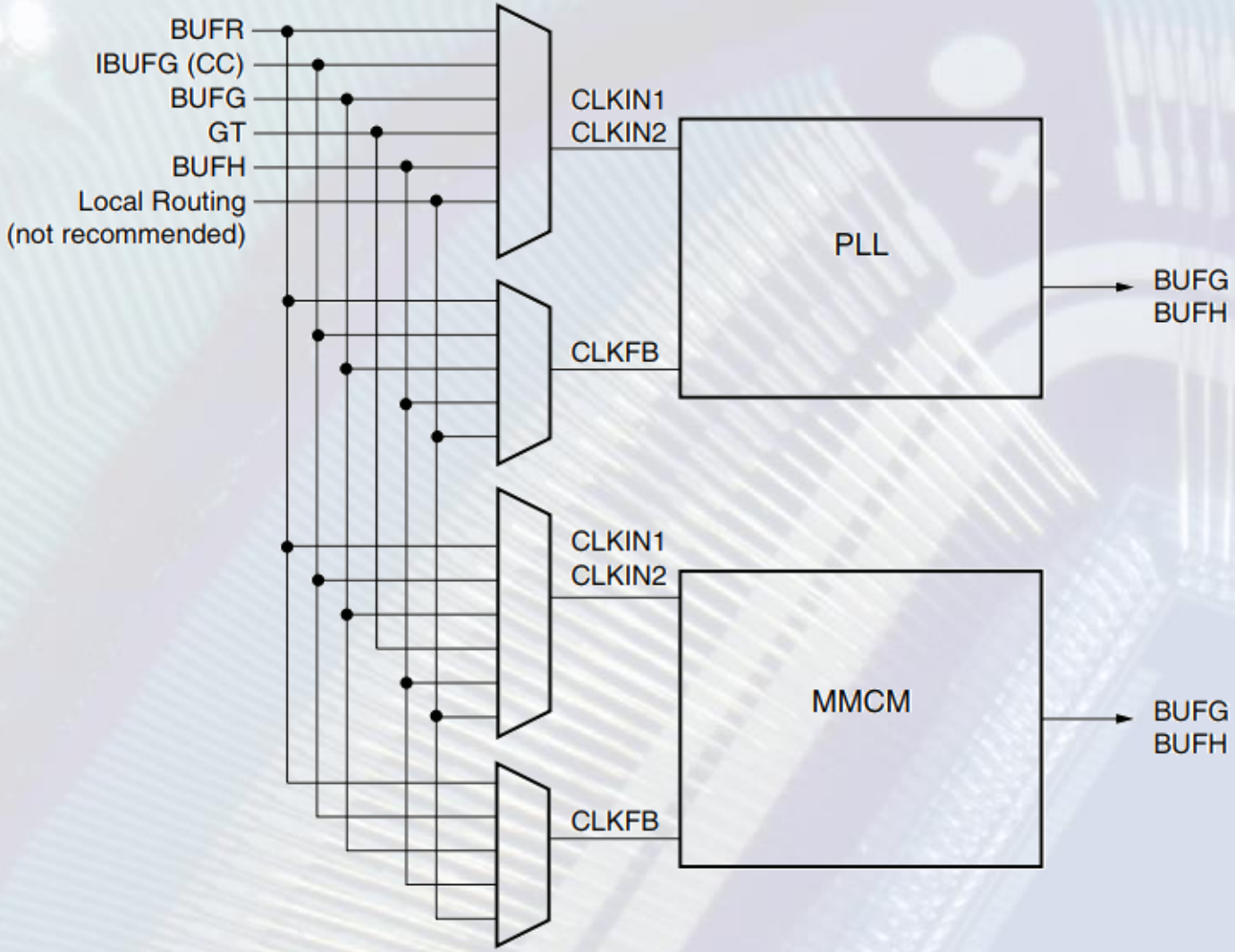


Órajel bufferek – összefoglaló

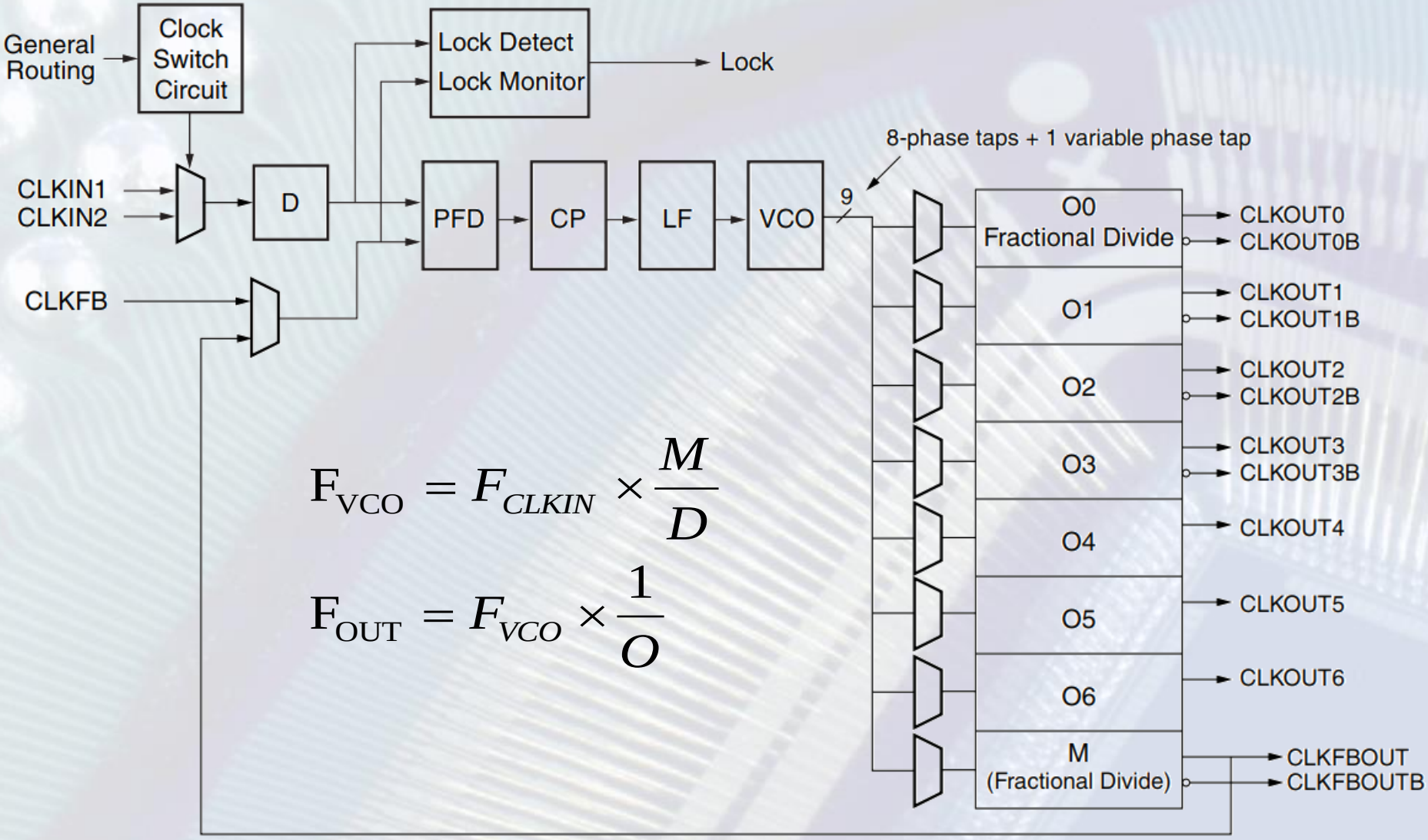
- **BUFG**
  - Kimenete elérhető az eszköz teljes felületén
  - Meghajthatja: CC I/O, MMCM, PLL, BUFG, BUFR, transceiver, huzalozás
- **BUFR**
  - Kimenete egy órajel régióban érhető el
  - Meghajthatja: CC I/O, MMCM 0…3 kimenetek, huzalozás
  - Tud órajelet osztani
- **BUFH**
  - Kimenete egy órajel régióban érhető el
  - Meghajthatja: CC I/O, MMCM, PLL, BUFG, transceiver, huzalozás
- **BUFIO**
  - Kimenete a szomszédos I/O blokkokban érhető el
  - Forrása CC I/O vagy MMCM 0…3 kimenetek

Clock Management Tile

- **Egy MMCM-t és egy PLL-t tartalmaz**



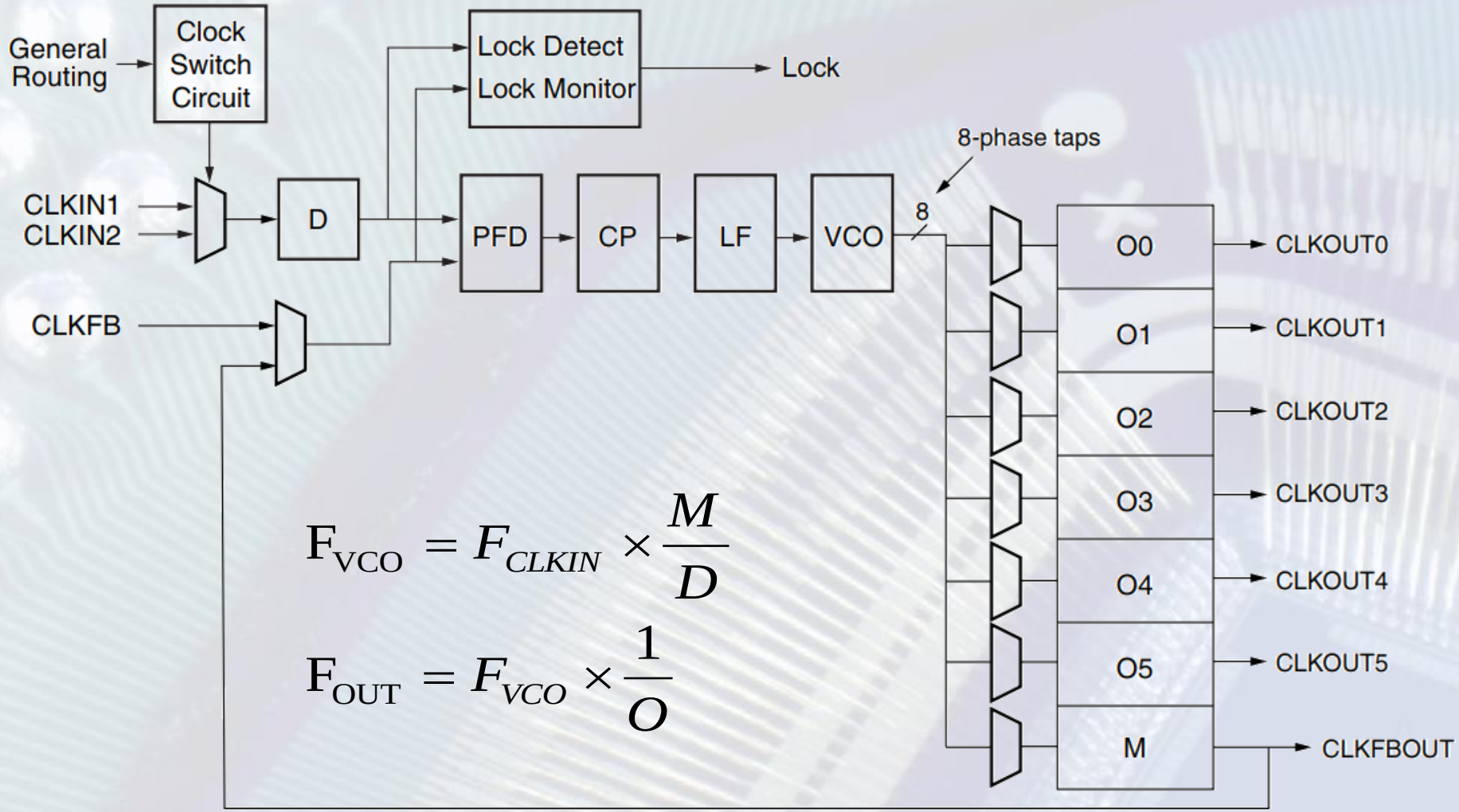
MMCM



$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

$$F_{OUT} = F_{VCO} \times \frac{1}{O}$$

PLL



$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

$$F_{OUT} = F_{VCO} \times \frac{1}{O}$$

CMT időzítési követelmények

- **MMCM vs. PLL**
  - Tört osztás az MMCM-ben
  - Nagyobb osztók az MMCM-ben
  - Időzítési követelmények különbözőek!
- **Kintex-7, -2**
  - BUFG: 710 MHz, BUFIO: 800 MHz, BUFR: 540 MHz

	MMCM	PLL
Input	10…933 MHz	19…933 MHz
PFD	10…500 MHz	19…500 MHz
VCO	600…1440 MHz	800…1866 MHz
Output	4.69…933 MHz	6.25…933 MHz

Órajel kimenet

- **Leosztott rendszer órajel**

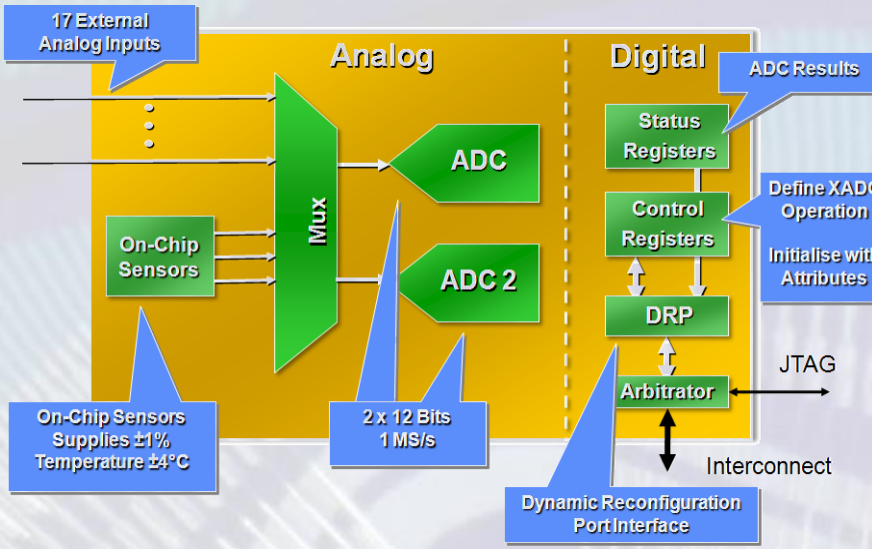
```
process(clk)
begin
if (clk'event and clk='1')
then
   cntr <= cntr + 1;
end if;
end process;
clk_d4 <= cntr(1);
```

- **Rendszer órajellel megegyező frekvenciájú kimenet**

```
ODDR_clk: ODDR2
port map(
      Q => clk_pin
      C0 => clk,
      C1 => not clk,
      CE => '1',
      D0 => '1',
      D1 => '0',
      R => '0',
      S => '0'
);
```

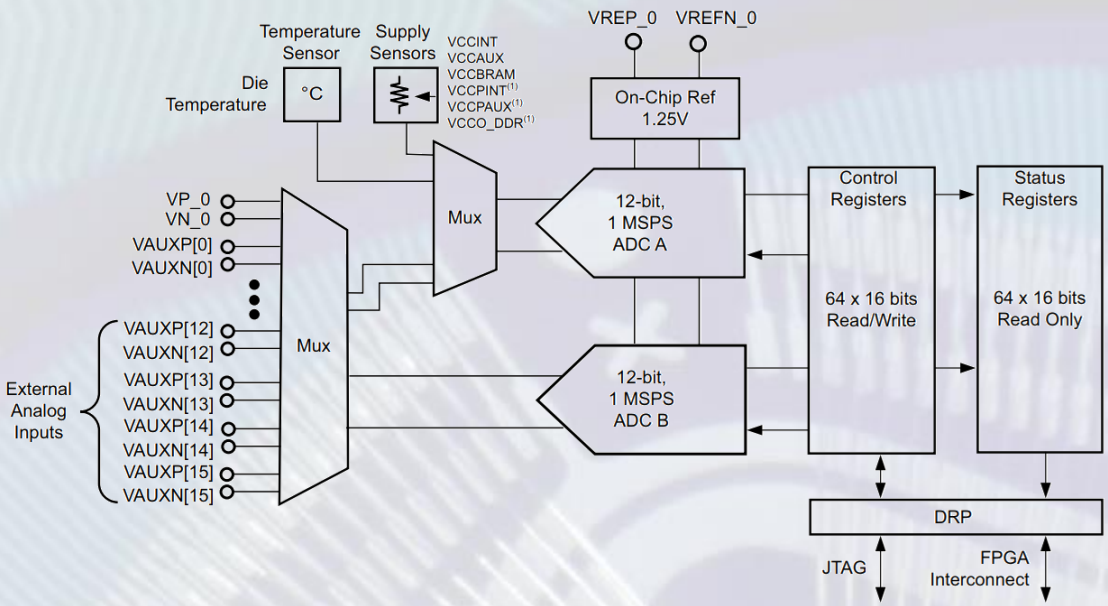
XADC

- **XADC: újdonság a 7-es szériában**
  - Két 12-bit 1Msps ADC, szenzorok a chip-en, 17 analóg bemenet, programozható jelkondícionálás
  - 1V bemeneti tartomány
  - 16-bit eredmény
  - Beépített erősítés és offszet kalibráció
  - Egyedi erősítő az ADC-khez
- **Felxibilis bemenetek**
  - Differenciális analóg bemenet
  - Unipoláris, bipoláris és differenciális bemeneti módok



XADC Block Diagram

- **Chip-en belüli szenzorok**
  - Chip hőmérséklet
  - Feszültségek
  - JTAG-en keresztül akkor is elérhető, ha a tervben nincs példányosítva
- **I/O lábak**
  - Dedikált GND, táp- és referencia feszültségek
  - Analóg bemenetek normál I/O lábként is használhatók
- **Dinamikus rekonfigurációs port**
  - Vezérlés és státusz az FPGA tervből



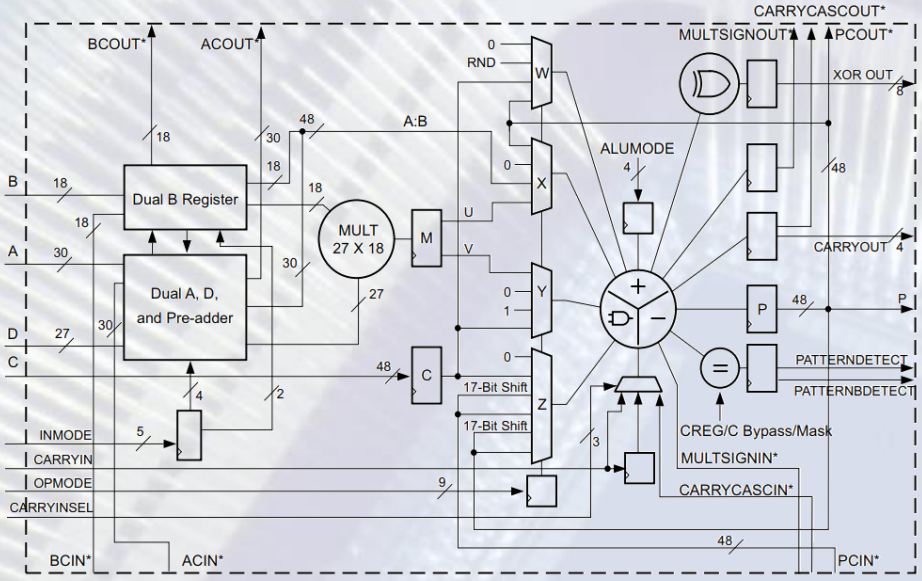
Xilinx UltraScale & UltraScale+ áttekintés

Xilinx UltraScale & UltraScale+

	Kintex UltraScale	Kintex UltraScale+	Virtex Ultarscale	Virtex Ultrascale+	Zynq UltraScale+ MPSoc	Zynq UltraScale+ RFSoC
CPU	N/A	N/A	N/A	N/A	CG: 2x Cortex A53 EG: 4x Cortex A53 EV: 4x Cortex A53	4x Cortex A53
RT CPU	N/A	N/A	N/A	N/A	2x Cortex R5	2x Cortex R5
GPU	N/A	N/A	N/A	N/A	Mali-400 MP2	N/A
Video ENC/DEC	N/A	N/A	N/A	N/A	4k@60 H.264, H.265	N/A
RF ADC, DAC SD-FEC	N/A	N/A	N/A	N/A	N/A	Igen
UltraRAM	N/A	0-36	N/A	90-360	0-36	13-22
DSP	768-5520	1368-3528	600-2880	2280-12288	240-3528	3145-4272
Transceiver	12-64 16,3 Gb/s	16-76 32,75 Gb/s	36-120 30,5 Gb/s	32-128 58 Gb/s	0-72 32,75 Gb/s	8-16 32,75 Gb/s
DDR4 (Mbit/s)	2400	2666	2400	2666	2666	2666
HBM RAM	N/A	N/A	N/A	0-8 GB	N/A	N/A

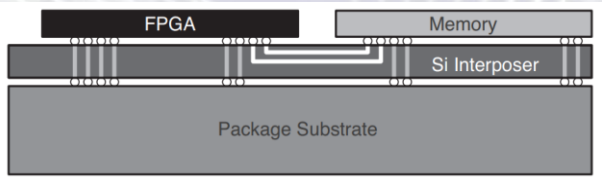
UltraScale+ erőforrások

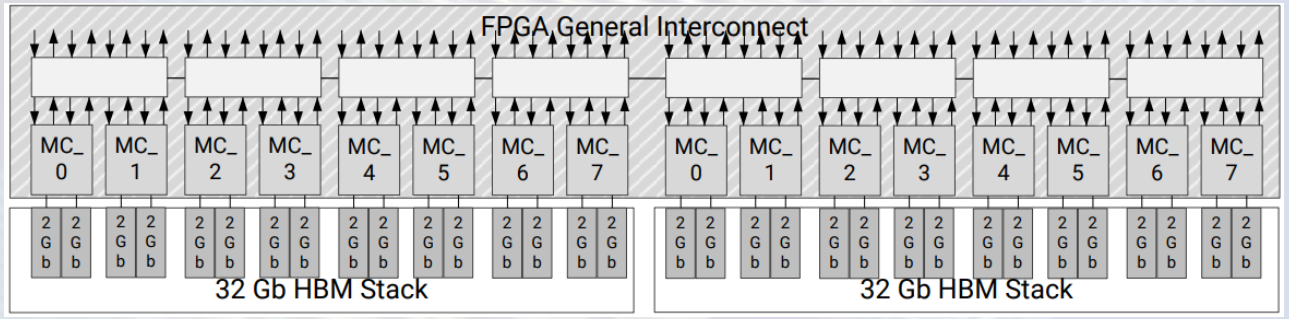
- **CLB: nagyon hasonló a 7-es sorozathoz, de**
  - 1 slice/CLB, 8 LUT6, 16 FF/Latch
  - 4 CE/slice
  - Külön órajel és WE a LUT RAM-hoz
  - Több kimenet: LUT, MUX, 2x FF
- **DSP: DSP48E2**
  - 27 x 18 bites szorzó
  - 27 bites elő-összeadó
  - ALU 4 bemenetű



Memóriák

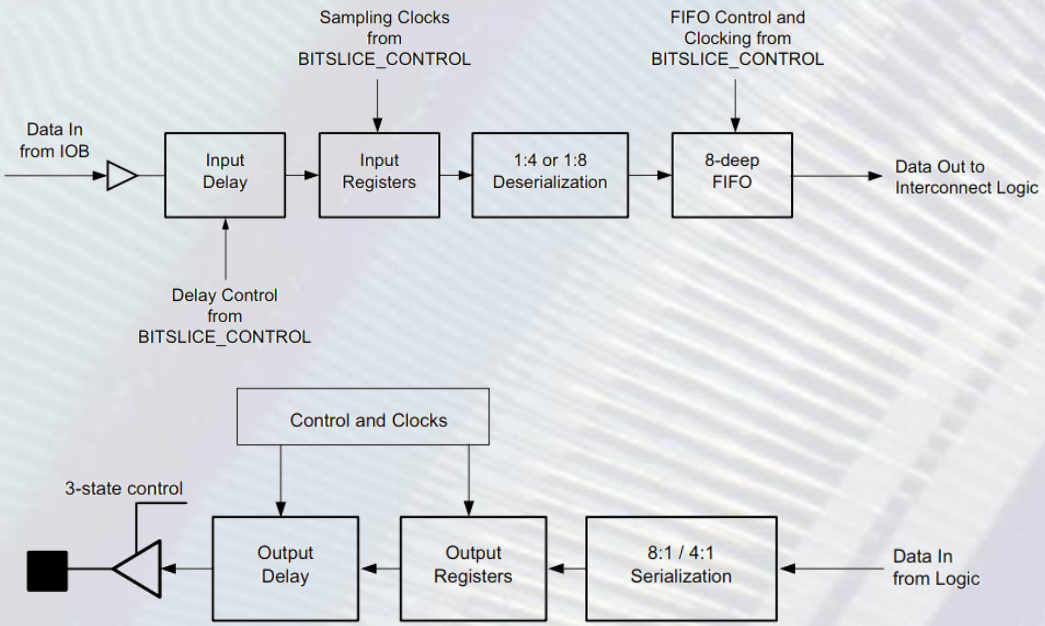
- **BRAM/FIFO: apróbb változások**
- **UltraRAM**
  - 288 kbit, true dual-port
    - 4k x 72 bit, ECC
    - Írás: mint a BRAM, vagy külön adat és paritás
  - Kaszkádosítható (max ~100 Mbit)
- **HBM: a tokozáson intergált külső memória**
  - 0-8 GB méret, 460 GB/s sávszélesség
  - Dedikált vezérlő
    - 16 db 256 bites AXI3 interfész, 16x16 crossbar

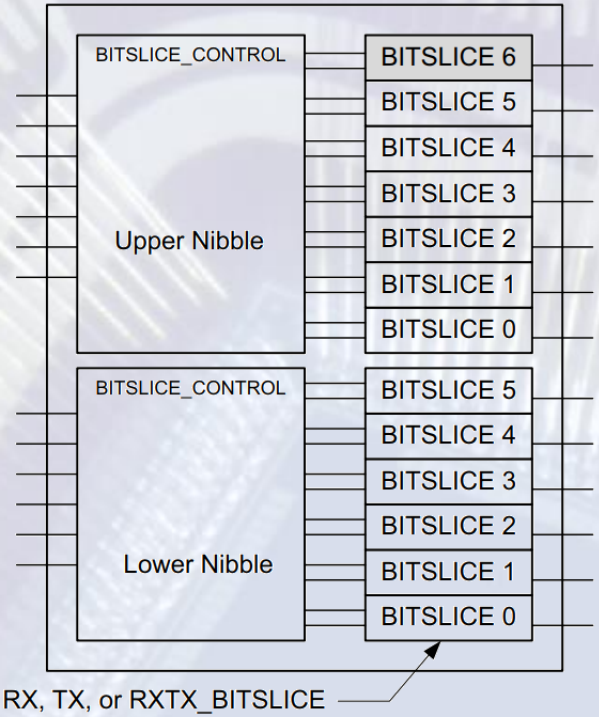




Általános I/O

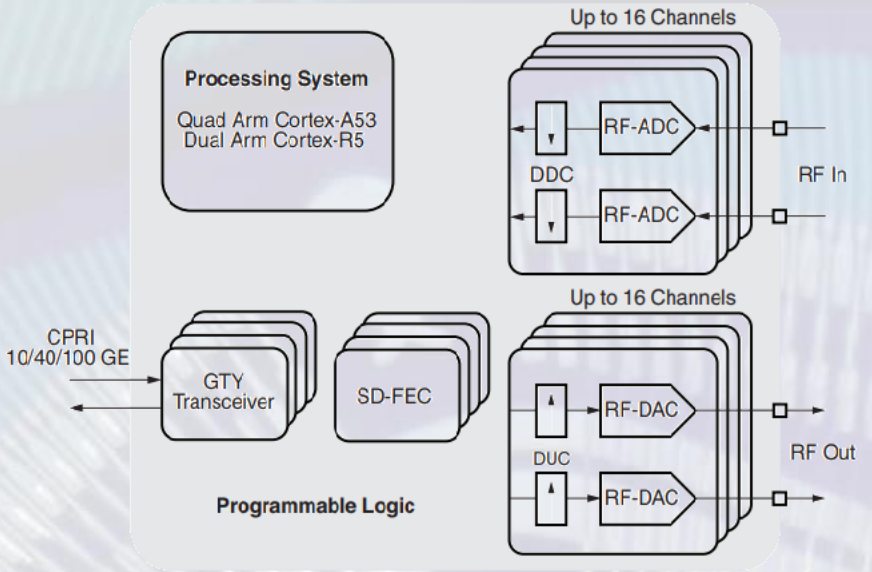
- **MIPI DPHY támogatás a HP I/O bankokban**
- **Rendelkezésre áll a 7-es sorozatból ismert primitív készlet**
  - IDELAY, ISERDES, ODELAY, OSERDES
  - “Component mode”: max 1250 Mb/s
- **Új primitív: BITSLICE (“Native mode”)**
  - Max. 1600 Mbit/s





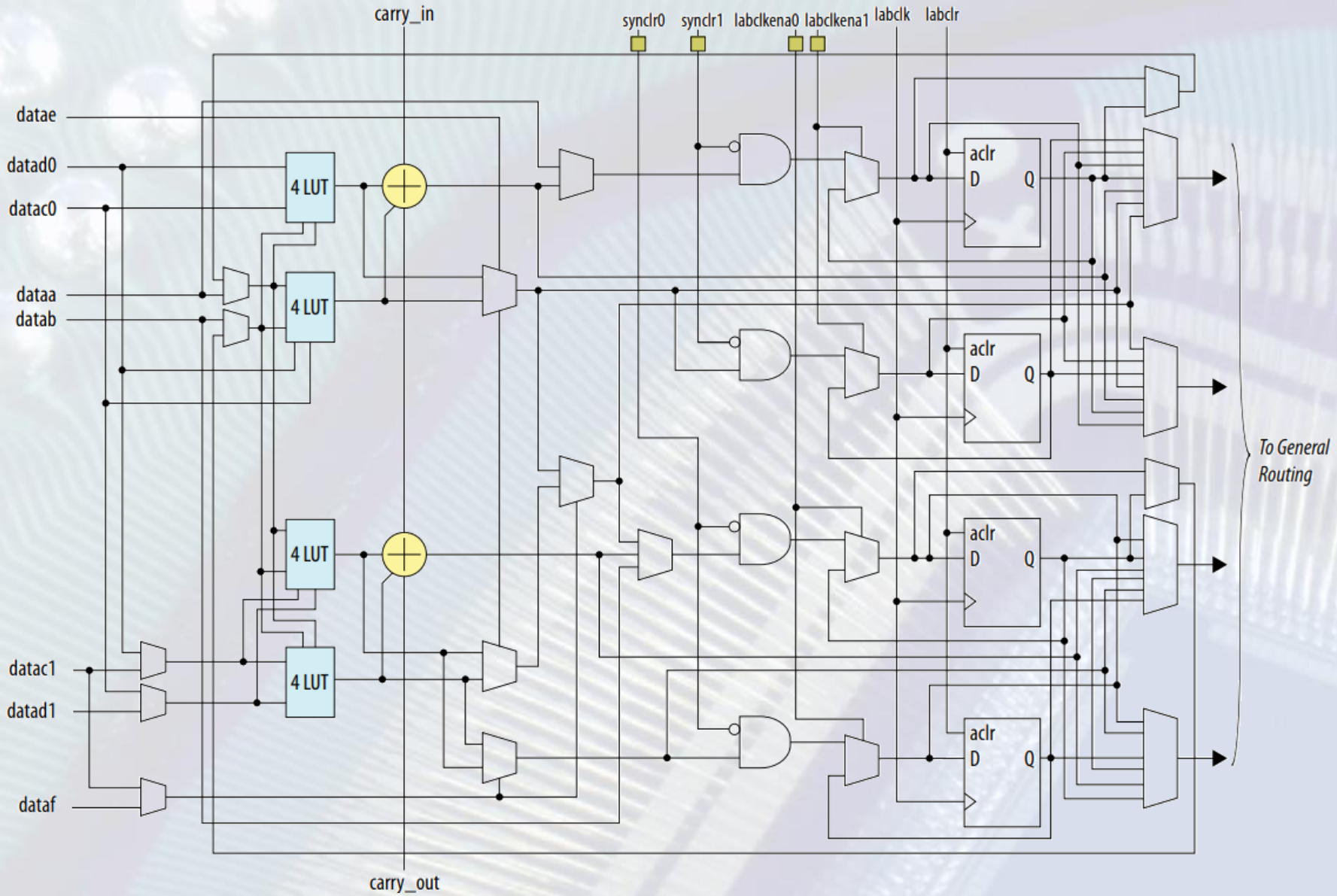
Zynq UltraScale+ RFSOC

- **RF ADC**
  - Valós vagy I/Q
  - 12 bit @ 2,275 vagy 4 GS/s
  - 14 bit @ 2,5 vagy 5 GS/s
- **RF DAC**
  - 14 bit @ 6,554 vagy 10 GS/s
- **48 bit NCO (numerically controlled oscillator)**
- **Complex mixer**
- **Forward error correction (FEC) decoder**
- **Low-density parity-check (LDPC) encoder**

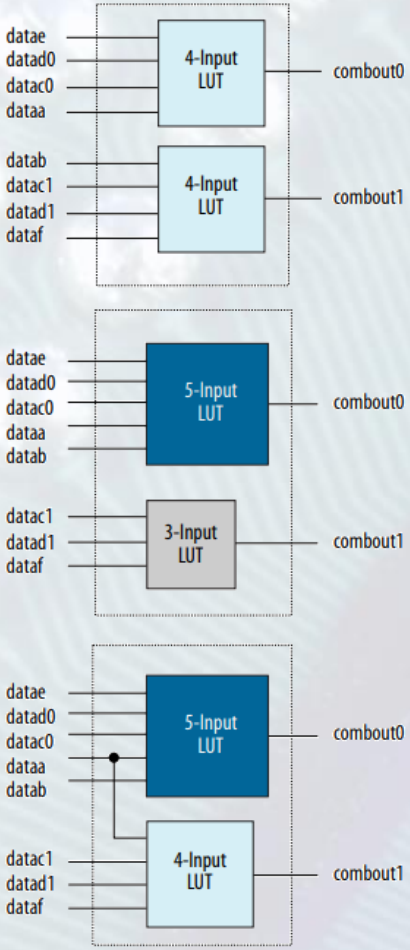


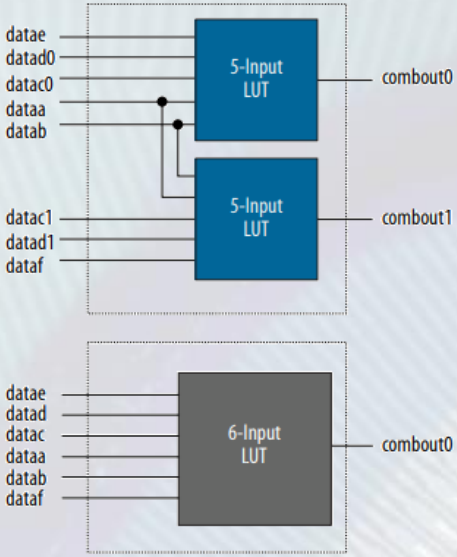
Intel Stratix 10 áttekintés

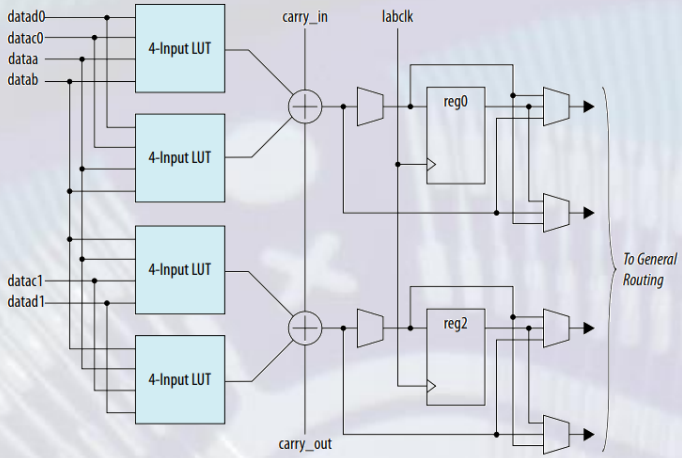
Stratix 10 ALM

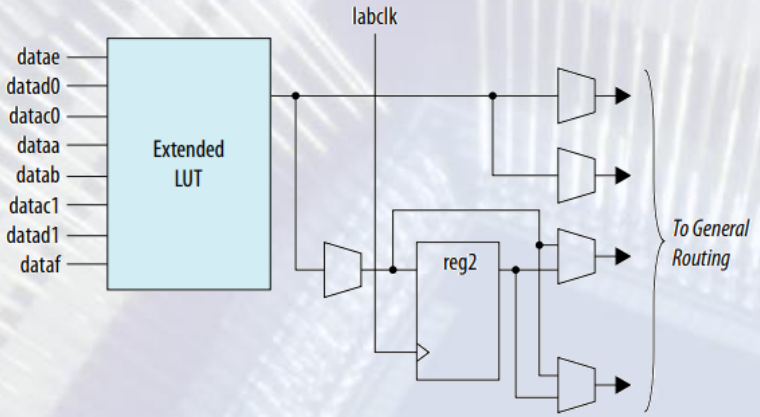


Stratix 10 ALM módok







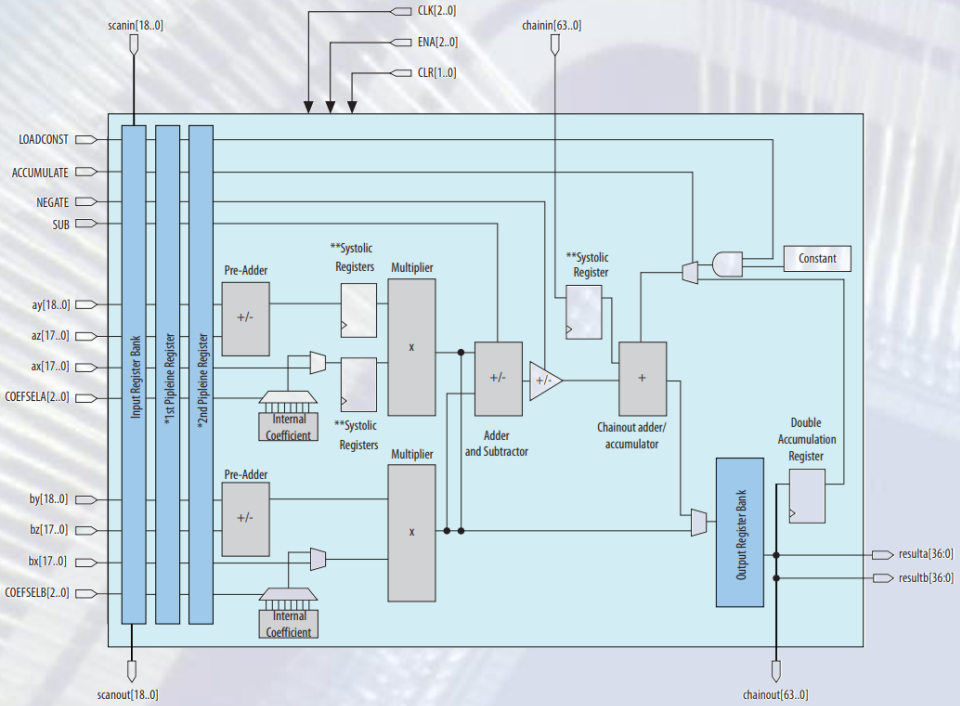


Stratix 10 memóriák

- **Chip-en belüli memóriák**
  - MLAB: 640 bit (ALM alapú)
    - x16, x18, x20, 32 mély
    - ~LUT RAM, cask kevésbé szabadon konfigurálható
  - M20K: 20 kbit, simple/true dual port, simple quad port
    - Hasonló funkcionalítás, mint a 18 kbit BRAM
  - eSRAM: 47,25 Mbit simple dual-port
    - 1 channel
      - 42 db 2k x 72 bit SRAM (144 kb vagy 128 kb ECC-vel)
      - Simple dual port
    - eSRAM: 8 független channel
- **Külső memóriák**
  - Dedikált memória vezérlő
  - Stratix 10 MX: HBM

Stratix 10 DSP

- **Fix pontos**
  - 2 db 18x19 bites szorzó (vagy 1 27x27)
  - Pre-adder
  - 64-bit akkumulátor
  - 8 mély együttható tár mindkét szorzóhoz
- **Lebegőpontos (!)**
  - 1 db MAC



Stratix 10 I/O

- **Normál i/O lábak bankokban**
  - Kb. ugyanazok a lehetőségek mint Xilinx esetén
  - Unipoláris vagy differenciális
  - Több I/O szabvány
  - I/O órajel generálás, SERDES, fázis illesztés (DPA)
  - Max 1250 – 1600 Mb/s