1. ZH A

1.)FPGA

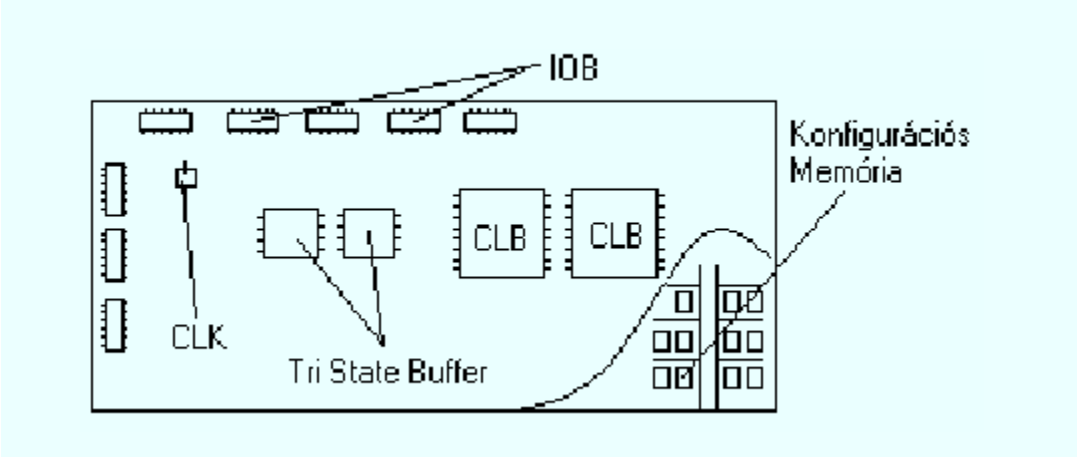
A Xilinx FPGA áramköröknek három alapvető építőeleme van:

CLB: konfigurálható logikai blokk: szükséges logikai kapcsolatok megvalósítása egy logikai tömbben. Tartalmaz 2db. D Flip-Flop-ot és 2db független 4-bemenetű LUT-t tetszőleges (F és G): logikai függvényeket lehet velük megvalósítani.

IOB: programozható be/kimeneti blokk: kapcsolat a belső vonalak és a tokozás lábai (pad) között (lehet bementi /kimeneti / kétirányú (tristate) kapcsolatként definiálni). Szintén D-FF-kel rendelkezik, belső puffere (buffere) van, nem kell az adatokat a külső lábon (pad) tartani. A kimenet lehet negált, vagy ponált.

PI: Programozható összeköttetések: az IOB-k és CLB-k kivezetéseinek (lehetnek TTL v. CMOS) összehuzalozása. Konfigurációs memóriában vannak tárolva az IOB-k, CLB-k és PI-k állapotai. Az összes belső összeköttetés programozható kapcsolókkal valósítható meg. Három típusa: egyszeres / dupla /hosszú összeköttetés.

Ezek mellett a Xilinx tartalmaz tri-state buffereket, CLK órajel-meghajtó áramköröket, és konfigurációs memóriákat is. Minden Xilinx processzorral nagyszámú kapu programozható, 50-150MHz-es tartományban működtethetők, tehát gyorsak. (ezek az adatok a korábbi XC3000, XC4000 családok esetén álltak fenn). Egy Xilinx FPGA általános felépítése a következő:



2.)N=18 bites 2-es komplemens fixpontos rendszer p=3

$$V_{\text{FIXED POINT}} = -b_{N-1} \times 2^{N-p-1} + \sum_{i=0}^{N-2} b_i \times 2^{i-p}$$

Vmin+ =delta r=2^(-p)=2^(-3)=

Vmax= (i=0 ->17)sum 1*2^(i-3)= $\sum_{i=0}^{17} \cdot 2^{i-3}$ 1

Vmin- = -bN-1*2^(N-p-1)= -1*2^(18-3-1)=

3.) szekvenciális hálózatok, hazárd

szekvenciális hálózatok:

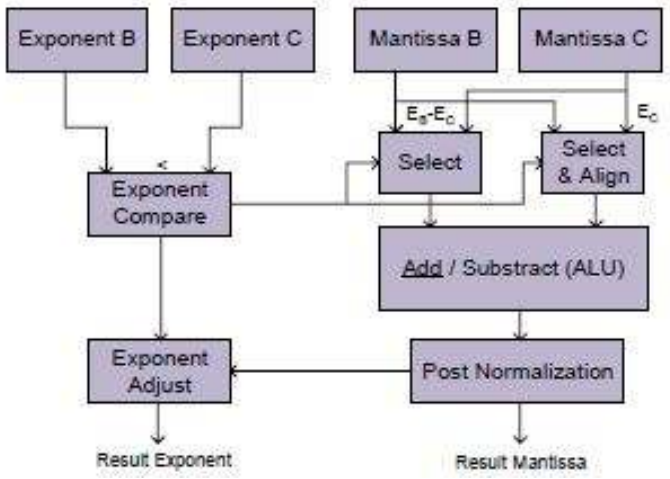
- van visszacsatolás

- korábbi állapot és az aktuális küldő bemenet függvényében határozzuk meg a kimeneteket
- digitális rendszer kontrollálható memóriával
- építőelemei: retesz (latch), flip-flop, regiszter, számláló, memória
- csoportosításuk:
    - 1,órajel nélküli aszinkron hálózatok
    - 2, órajellel vezérelt szinkron hálózatok

<u>hazárd:</u>

- def.: Késleltetés okozta nem-kívánt kimenetek, állapotok.
- Hazárd alakulhat ki, ha egy kapu kimenete a bemenetek változásához képest csak véges időn belül változik (= jelterjedési késleltetés), vagy a logikai kapukat összekötő vezetéken lévő véges jelterjedés miatt (=összeköttetési késleltetés).
- Fajtái:
  - funkcionális/statikus: olyan hazárd jelenségek, amik idővel nem szűnnek meg, ekkor a tervezőnek kell beavatkozni
  - dinamikus: olyan többszintű hálózatokban jöhet létre, ahol a statikus hazárd az alacsonyabb hierarchia szinteken nem lett kiküszöbölve. Megszüntethető szinkronizálással.

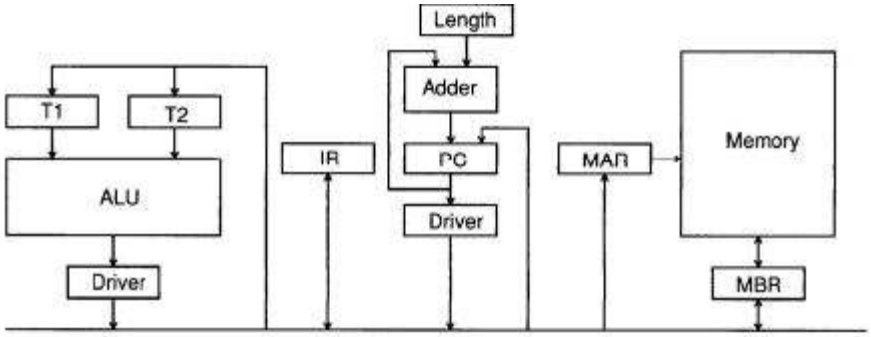
4.)**lebegőpontos összeadó blokkdiagramm**



$A=B+C=M_B*r^{Eb} + M_C*r^{Ec}=(M_B*r^{Eb-Ec}+M_C)*r^{Ec}$

Align: MSB bitek beállítása --> nagyobbik operandus mantisszájához igazítjuk a kisebbét $|E_B-E_C|$-vel jobbra shifteljük

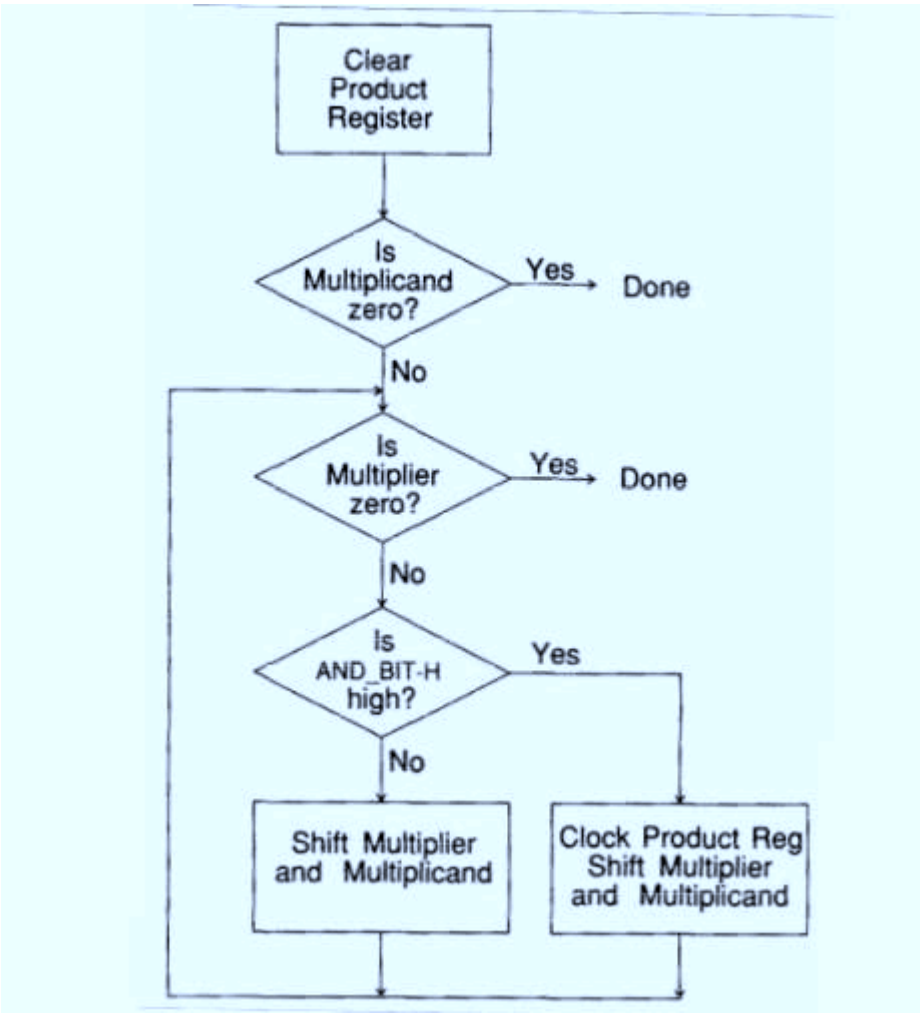
5.) három című számítógép blokkdiagramm

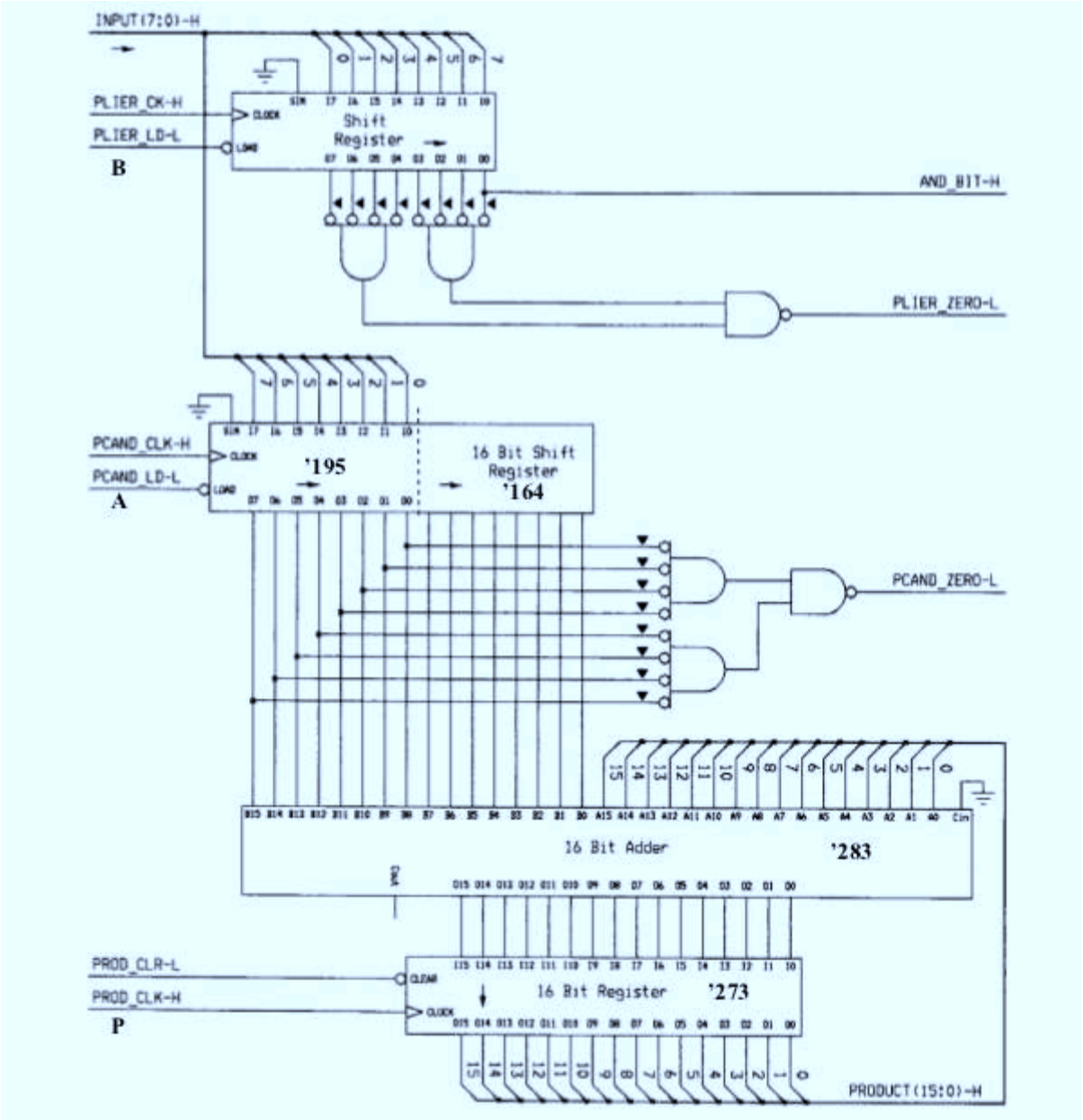


ADD3(operátor) X(operandus1),Y(operandus2),Z(eredmény)

6.) 16*16 bites iteratív szorzóalgoritmus-folymatábra,áramkör,végrehajtási idő

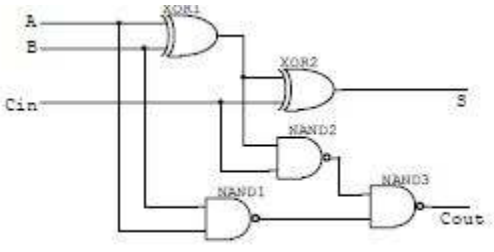
Fordított sorrendű szorzó:



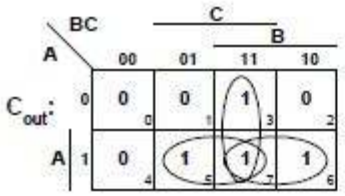


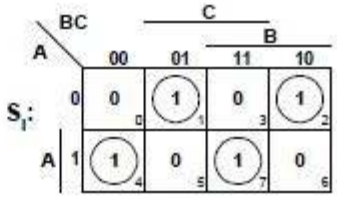
7.) FA igazságtábla, kapcsrajz:

A_i	B_i	Cin	Sum_i	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Karnough táblái:

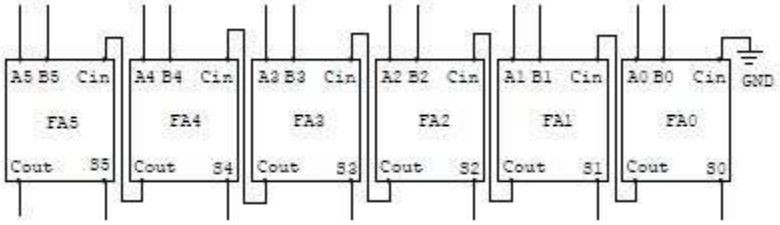




$$C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$$

$$S_i = \overline{A_i} \cdot \overline{B_i} \cdot C_{in} + \overline{A_i} \cdot B_i \cdot \overline{C_{in}} + A_i \cdot \overline{B_i} \cdot \overline{C_{in}} + A_i \cdot B_i \cdot C_{in} = A_i \oplus B_i \oplus C_{in}$$

8 bites RCA kapcsrajz: ez csak egy 6 bites RCA, 8 bites lesz, ha még két FA-t hozzákapcsoltok (FA6 és FA7)



idő ütemezés: T(RCA)=N*t(FA)=N*(2G)=8*2*g=16G (8 bites RCA esetén N=8)

8.) SUB2*X*Y RTL leírása, időszükséglete

Megkockáztatom, hogy SUB=ADD mert a SUB-ról semmi infót nem találok, hogy mit jelent... de mindenesetre ez tuti jó egészen a művelet elvégzéséig

(szerintem u.a. mint az ADD, csak a végén nem összeadod...?)

Példa: Összeadás kétcímű géppel ADD2(*X,*Y)

Időszükségletek feltüntetésével!

T_{MEM}=30ns, T_{ALU}=10ns, T_{REG}=5ns

Fetch: (regiszterek feltöltése, utasításhívások):

PC→MAR	[5ns] PC-ből a következő utasítás címe a MAR-ba töltődik
M[MAR]→MBR	[30ns] Memóriában lévő utasítás beírása az MBR-be
PC+I_len→PC	[5ns] Az utasítás hosszával (I_len) növeli a PC értékét
MBR→IR	[5ns] Majd az MBR-ben lévő adatot az IR-be tesszük

Decode: (a dekódolást általában 0 idejűnek feltételezzük)

Execute: (végrehajtás)

PC→MAR	[5ns] PC-vel a következő (**X**) *címének címére* mutatunk
PC+X_Alen →PC	[5ns] X operandus címének hosszával növeljük a PC-t
M[MAR]→MBR	[30ns] *X* címének címét az MBR-be írjuk
MBR→MAR	[5ns] Ezt a címet a MAR-ba töltjük
M[MAR]→MBR	[30ns] X *címét* megkapjuk az MBR-ben
MBR→MAR	[5ns] X címét a MAR-ba töltjük
M[MAR]→MBR	[30ns] X címén lévő *értékét* megkapjuk MBR-ben
MBR→T1	[5ns] **X értékét** T1-be töltjük
PC→MAR	[5ns] PC-vel a következő (**Y**) *címének címére* mutatunk
PC+Y_Alen →PC	[5ns] Y operandus címének hosszával növeljük a PC-t
M[MAR]→MBR	[30ns] Y címének címét az MBR-be írjuk
MBR→MAR	[5ns] Ezt a címet a MAR-ba töltjük
M[MAR]→MBR	[30ns] Y *címét* megkapjuk az MBR-ben
MBR→MAR	[5ns] Y címét a MAR-ba töltjük
M[MAR]→MBR	[30ns] Y címén lévő *értékét* megkapjuk MBR-ben
MBR→T2	[5ns] **Y értékét** T2-be töltjük
T1 + T2→MBR	[10+5ns] ADD2 művelet elvégzése, MBR-be töltjük
MBR→M[MAR]	[30ns] Eredményt a memóriában tároljuk el (ahol Y volt)

Indirekt címzést használunk itt!

Σ 320ns

9.)CFG (vezérlés folyam) gráf

Ezekből senki sem okos? :\

az egyik diasor végén be van már számozva ez a feladat, de pl mit csinálsz az ELSE-knél (mert csak az else szó is kap egy külön számot, meg utána a köv utasítás is)? leguccsó dia http://www.google.hu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CFkQFjAA&url=http%3A%2F%2Fvirt.uni-pannon.hu%2Findex.php%2Fcomponent%2Fdocman%2Fdoc_download%2F535-vlsi01pldhslprogmod&ei=5cSyT6h9i7z5BtzU0N4I&usg=AFQjCNHVWIGeo_1idt1NI2INSmplKXAtPw&sig2=xb-QqpEwjbca48_t6tv6IQ

jó tudni...

1 if, 2 akkor csinál A, 3 else, 4 különben csinál B, 5 a folytatás

1 -->2 -->5

-->3 -->4

akkor így kell lennie?

hát szerintem nem lesz ilyen hurkos? hárommal előtte van a kidolgozott és ott az ifes az ilyen hurkos... de az else-t akkor hogy rakod bele? ajj mindegy úgyse lehet kettesnél jobb :(

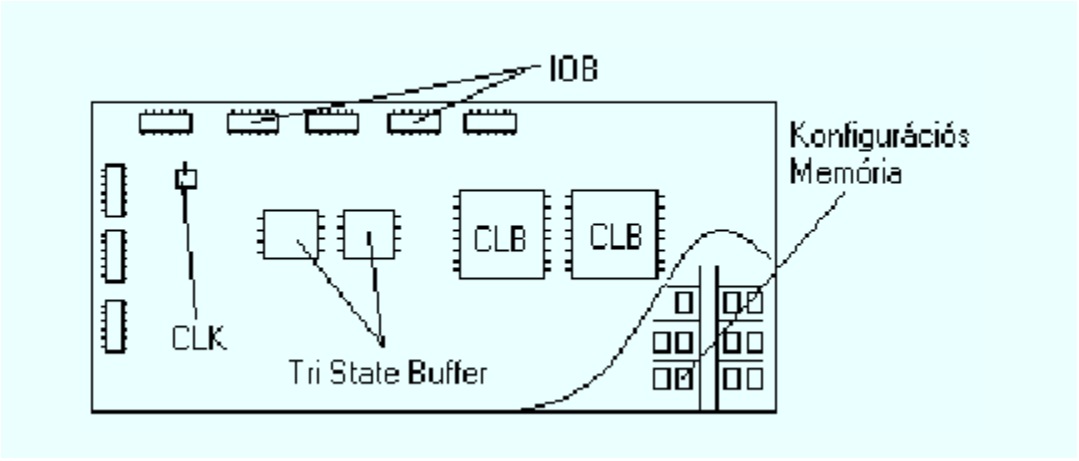
és a dfg-nél mi van a dupla feltételekkel? azt hogy rajzolod le?

1. ZH B

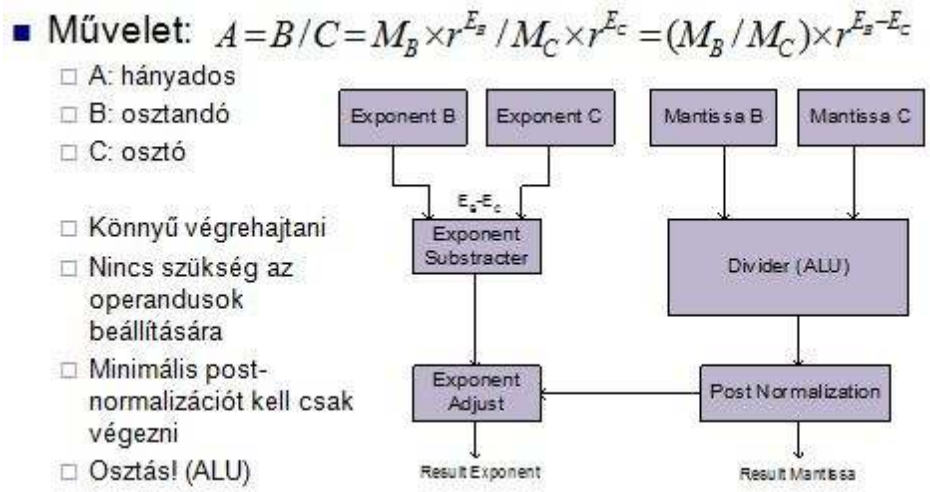
1.) Kombinációs hálózatok megadási módjai:
igazságtábla, algebrai kifejezés, elvi logikai vázlat
Minimális alak megkeresési módjai:
Algebrai módszer (Boole), kifejtési módszer, grafikus módszer (Karnough-, igazság-tábla), normálformák (DNF, KNF), számjegyes minimalizálás (would be expected, it is necessary to include it in the minimized boolean equation. In some cases, the essential prime implicants do not cover all minterms, in which case additional procedures for chart reduction can be employed. The simplest "additional procedure" is trial and error, but a more systematic way is Petrick's Method. In the current examp módszer)

2.) 2-es komplemens fixpontos rendszer: N=16 bit, p=5
V= -bN-1*2^(N-p-1)+ (i=0 ->N-2)sum bi*2^(i-p)
Vmin+ =delta r=2^(-p)=1/32
Vmax= (i=0 ->14)sum 1*2^(i-5)=2^9+…2^(-5)= (2^10 -1) + ((2^5)-1)/(2^5)=1023,96875
Vmin- = -bN-1*2^(N-p-1)= -1*2^(16-5-1)=-1024
3.) FPGA slice felépítése: szerintem ez nem az fpga, csak egy CLB blokk (lsd. A/1 kidolgozása)
A Xilinx FPGA áramköröknek három alapvető építőeleme van:
CLB: konfigurálható logikai blokk: szükséges logikai kapcsolatok megvalósítása egy logikai tömbben. Tartalmaz 2db. D Flip-Flop-ot és 2db független 4-bemenetű LUT-t tetszőleges (F és G): logikai függvényeket lehet velük megvalósítani.
IOB: programozható be/kimeneti blokk: kapcsolat a belső vonalak és a tokozás lábai (pad) között (lehet bementi /kimeneti / kétirányú (tristate) kapcsolatként definiálni). Szintén D-FF-kel rendelkezik, belső puffere (buffere) van, nem kell az adatokat a külső lábon (pad) tartani. A kimenet lehet negált, vagy ponált.
PI: Programozható összeköttetések: az IOB-k és CLB-k kivezetéseinek (lehetnek TTL v. CMOS)

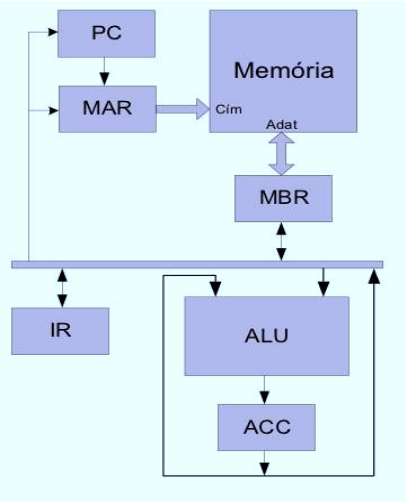
összehuzalozása. Konfigurációs memóriában vannak tárolva az IOB-k, CLB-k és PI-k állapotai. Az összes belső összeköttetés programozható kapcsolókkal valósítható meg. Három típusa: egyszeres / dupla /hosszú összeköttetés.
Ezek mellett a Xilinx tartalmaz tri-state buffereket, CLK órajel-meghajtó áramköröket, és konfigurációs memóriákat is. Minden Xilinx processzorral nagyszámú kapu programozható, 50-150MHz-es tartományban működtethetők, tehát gyorsak. (ezek az adatok a korábbi XC3000, XC4000 családok esetén álltak fenn). Egy Xilinx FPGA általános felépítése a következő:



4.) Lebegőpontos osztó blokk diagramja:

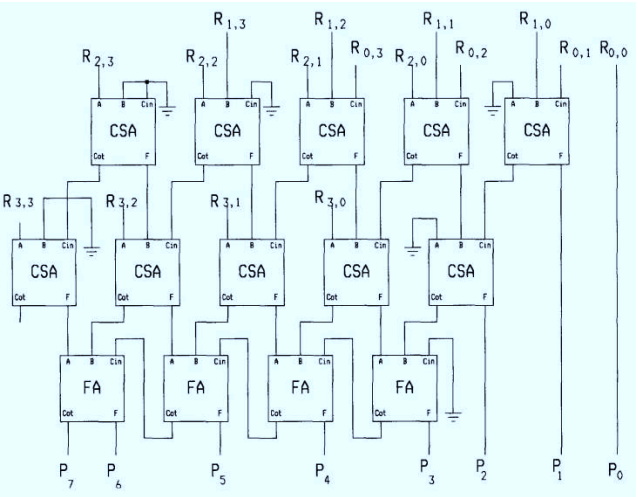


5.) Egy című számítógép blokk diagramja:



- ***MAR: Memory Address Register*** *(Memória-cím Regiszter)*: információ helyét azonosítja adott memóriacím alapján.
- ***MBR: Memory Buffer Register*** *(Memória Puffer Regiszter)*: tárolja a memóriába bevitt, ill. érkező információt. Egy adott memóriacímen lévő adat kiolvasásakor az ott lévő bejegyzés törlődhet (destruktív memória)
- ***PC: Program Counter*** *(Programszámláló)*: a soron következő (végrehajtandó) utasítás helyét azonosítja. Azon gépeknél, amelyek egy utasítást tárolnak memóriaterületenként, az utasítás végrehajtása után a PC értékét 1-el kell növelni (increment), mint egy számlálót.
- ***IR: Instruction Register*** *(Utasítás Regiszter)*: tárolja az éppen végrehajtás alatt álló utasítást. Engedélyezi a gép vezérlő részeinek, hogy a regiszterek, memóriák, aritmetikai egységek vezérlő vonalait a végrehajtáshoz szükséges működési módba állítsák. Az IR olyan széles, hogy az utasítás műveleti kódja ill. a hozzá tartozó egyéb utasítások ideiglenes másolatai eltárolhatók legyenek.
- ***ACC: Accumulator regiszter*** *(tároló regiszter):* eredmény ideiglenes tárolására használjuk (összes adatkezeléshez tartozó utasítás tárolása).

6.) 8*8 bites szorzó blokk diagramja CSA-ekkel:
-asszem ez csak 4*4 bites, de ha jól sejtem, akkor annyi változik, h minden sorba 9 CSA kell

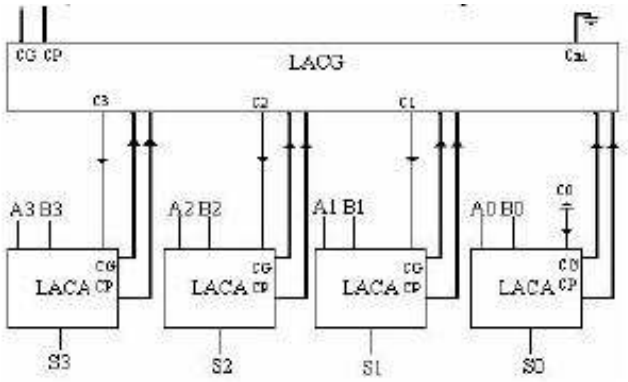


Szorzás időszükséglete, ha kapu késleltetése G:
időszükséglete mindig 2G

7.) Allokáció definíciója magas szintű szintézis esetén:

Allokáció: az erőforrások feladatokhoz rendelése
erőforrás típusok meghatározása,
erőforrások számának meghatározása
erőforrás: műveletvégző
adattároló
adatút (MUX, BUSZ)

8.) 16 bites LACA összeadó blokk diagramja



na ebből 4
összeadás időszükséglete 4 bites LACG-vel:
$TLACA=2+4*((\log_b N)-1)$
b=4, N=16
TLACA=2+4=6

9.) mem hozzáférés ideje: 20ns, regből regbe másolás: 4ns, kivonás 6ns
ADD3(X,Y,Z) RTL leírása:
Fetch: (regiszterek feltöltése, utasításhívások):

PC®MAR	[4ns] PC-ből a következő utasítás címe a MAR-ba töltődik
M[MAR]®MBR	[20ns] Memóriában lévő utasítás beírása az MBR-be
PC+I_len®PC	[4ns] Az utasítás hosszával (I_len) növeli a PC értékét
MBR®IR	[4ns] Majd az MBR-ben lévő adatot az IR-be tesszük

Decode: (a dekódolást általában 0 idejűnek feltételezzük)
Execute: (végrehajtás)

PC®MAR	[4ns] PC-vel a következő (**X**) címre mutatunk
PC+X_Alen ®PC	[4ns] X operandus címének hosszával növeljük a PC-t
M[MAR]®MBR	[20ns] Ezt az *X* címet az MBR-be írjuk
MBR®MAR	[4ns] Ez a cím lesz az X operandus címe
M[MAR]®MBR	[20ns] X címen lévő értéket az MBR-be töltjük
MBR®T1	[4ns] **X értékét** T1-be töltjük
PC®MAR	[4ns] PC-vel a következő (**Y**) címre mutatunk
PC+Y_Alen ®PC	[4ns] Y operandus címének hosszával növeljük a PC-t
M[MAR]®MBR	[20ns] Ezt a Y címet az MBR-be írjuk
MBR®MAR	[4ns] Ez a cím lesz az Y operandus címe
M[MAR]®MBR	[20ns] Y Címen lévő értéket az MBR-be töltjük
MBR®T2	[4ns] **Y értékét** T2-be töltjük
PC®MAR	[4ns] PC-vel a következő (**Z**) címre mutatunk
PC+Z_Alen ®PC	[4ns] Z operandus címének hosszával növeljük a PC-t
M[MAR]®MBR	[20ns] a Z eredmény címét az MBR-be írjuk
MBR®MAR	[4ns] majd a MAR-ba töltjük
T1 + T2®MBR	[6+4ns] ADD2 művelet elvégzése, MBR-be töltjük
MBR®M[MAR]	[20ns] Eredményt a memóriában tároljuk el (ahol Z volt)

időszükséglete: 206s (ha jól számoltam)

10.) VHDL szubrutin DFG (adatfolyam) gráfja:

```
LED_PROC : process (Bus2IP_Clk) is
begin
--1             if Bus2IP_Clk' event and Bus2IP_Clk='1' then      MIT KELL CSINÁLNI, HA AND VAN?!
--2                     if Bus2IP_Reset='1' then
--3                             LED_i<="0000";
--4                     else
--5                             if Bus2IP_WrCE(0)='1' then
--6                                     LED_i<= Bus2IP_Data(0 to 3);
--7                             else
--8                                     LED_i<=LED_i;
--9                             end if;
--10                    end if;
--11            end if;
end process LED_PROC;
```

Ha jól emlékszem,akkor az 1-esbe 3 nyíl megy: "Bus2IP_Clk"," '1' " és "event" és egyesből ki egy nyíl,legalábbis az egyik gyakvezsrác ezt mondta anno