Laborgyakorlat
Logikai áramkörök számítógéppel segített tervezése (CAD)

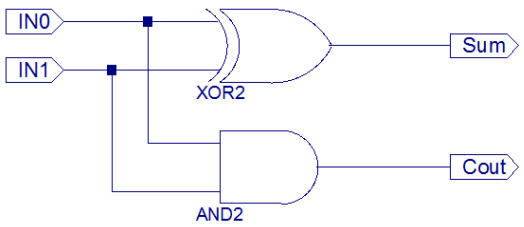
Összeadó áramkör

A legegyszerűbb összeadó két bitet ad össze, és az egy bites eredményt és az átvitelt adja ki a kimenetén, ez a **félösszeadó** (half adder) (1. ábra). A kétbites összeadó igazságtáblázata a következő.

IN0	IN1	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$Sum = IN0 * \overline{IN1} + \overline{IN0} * IN1 = IN0\ xor\ IN1$$

$$Cout = IN0 * IN1$$



1. ábra - Félösszeadó áramkör kapcsolási rajza

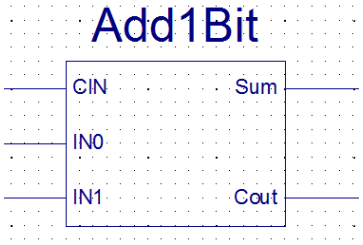
A teljes összeadó (full adder)

A kétbites teljes összeadó (2. ábra, 3. ábra, 4. ábra) igazságtáblázata a következő.

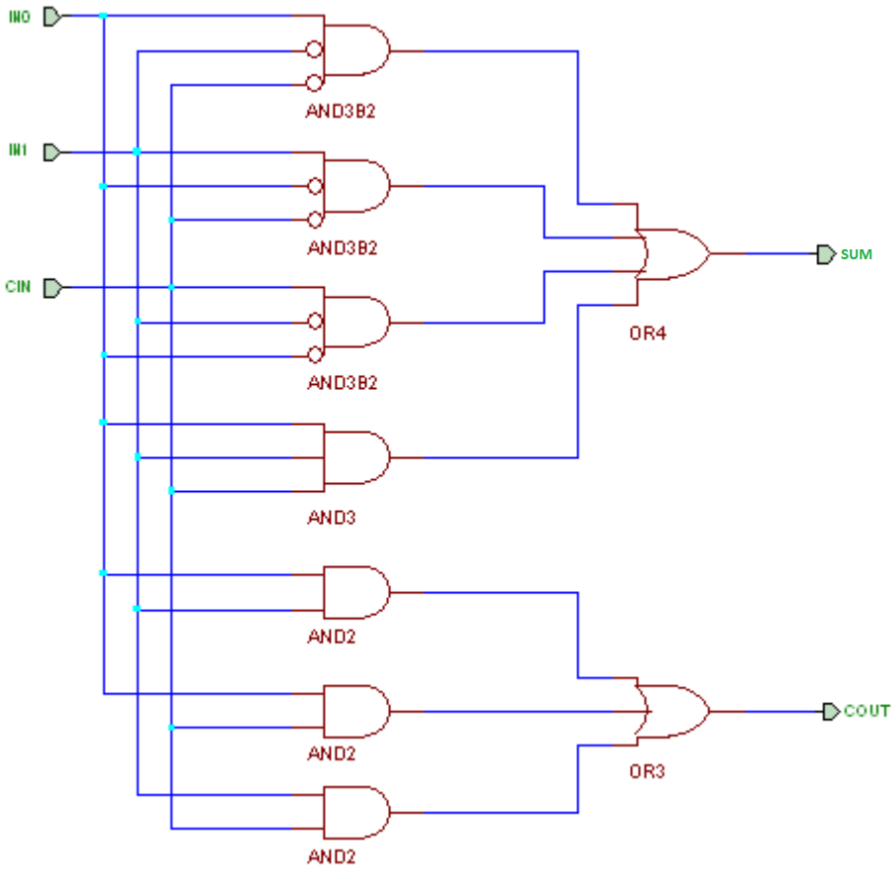
IN0	IN1	CIN	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$Sum = \overline{IN0} * \overline{IN1} * CIN + IN0 * \overline{IN1} * \overline{CIN} + IN0 * IN1 * CIN + \overline{IN0} * IN1 * \overline{CIN}$$
$$= IN0\ xor\ IN1\ xor\ CIN$$

$$Cout = IN0 * CIN + IN1 * CIN + IN0 * IN1$$

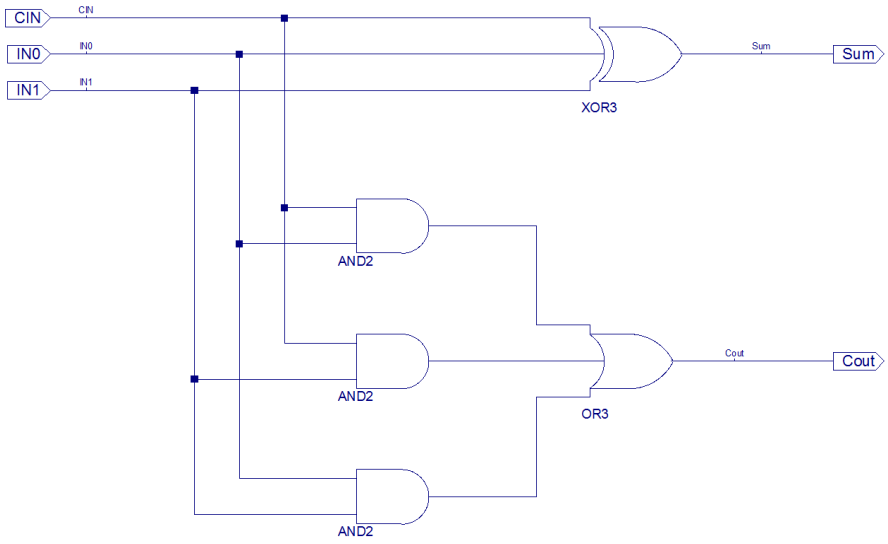


2. ábra - Teljes összeadó áramkör blokkvázlata



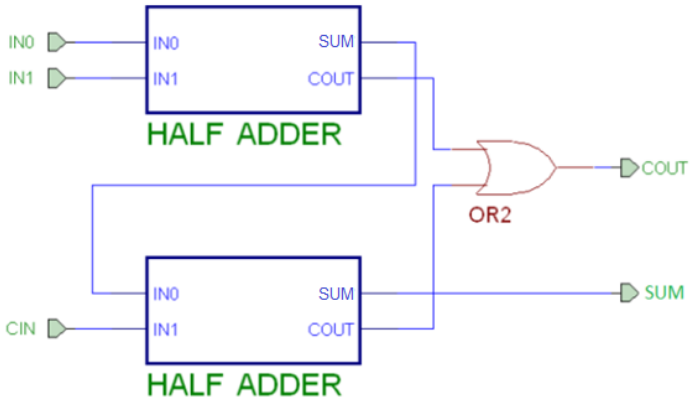
3. ábra - Teljes összeadó áramkör kapcsolási rajza

A teljes összeadó egy másik áramköri megvalósítása.



4. ábra - Teljes összeadó áramkör kapcsolási rajza 2

A teljes összeadó létrehozása két félösszeadó segítségével (5. ábra) (ez utóbbi innen kapta a nevét):



5. ábra - Teljes összeadó áramkör kapcsolási rajza 3

Több bit bites összeadó áramkörben minden helyi értékre kell egy teljes összeadó, az előző helyi érték átvitel kimenete van csatlakoztatva az adott átvitel bemenetére ($COUT_{n-1}$-> CIN_n). Ha kivonást is akarunk végeztetni az összeadónkkal, akkor a második operandus kettes komplemensét (bitenkénti negálás + 1) kell az elsőhöz hozzáadnunk. A negálást exkluzív vagy kapuval, az egyes hozzáadását az első helyi értéken a CIN-re adott egyessel oldhatjuk meg.

A sorba kapcsolt összeadókkal történő megoldást ripple carry módnak hívjuk. Ennek a megoldásnak az a hátránya, hogy meg kell várnunk, amíg az egyes átvitelek végighaladnak az összes összeadón. *Ezért alkalmaznak gyorsítókat (look ahead carry).*

Egy 4 bites összeadó-kivonó kapcsolási rajzát a 6. ábra szemlélteti (adder-subtractor with ripple carry out).

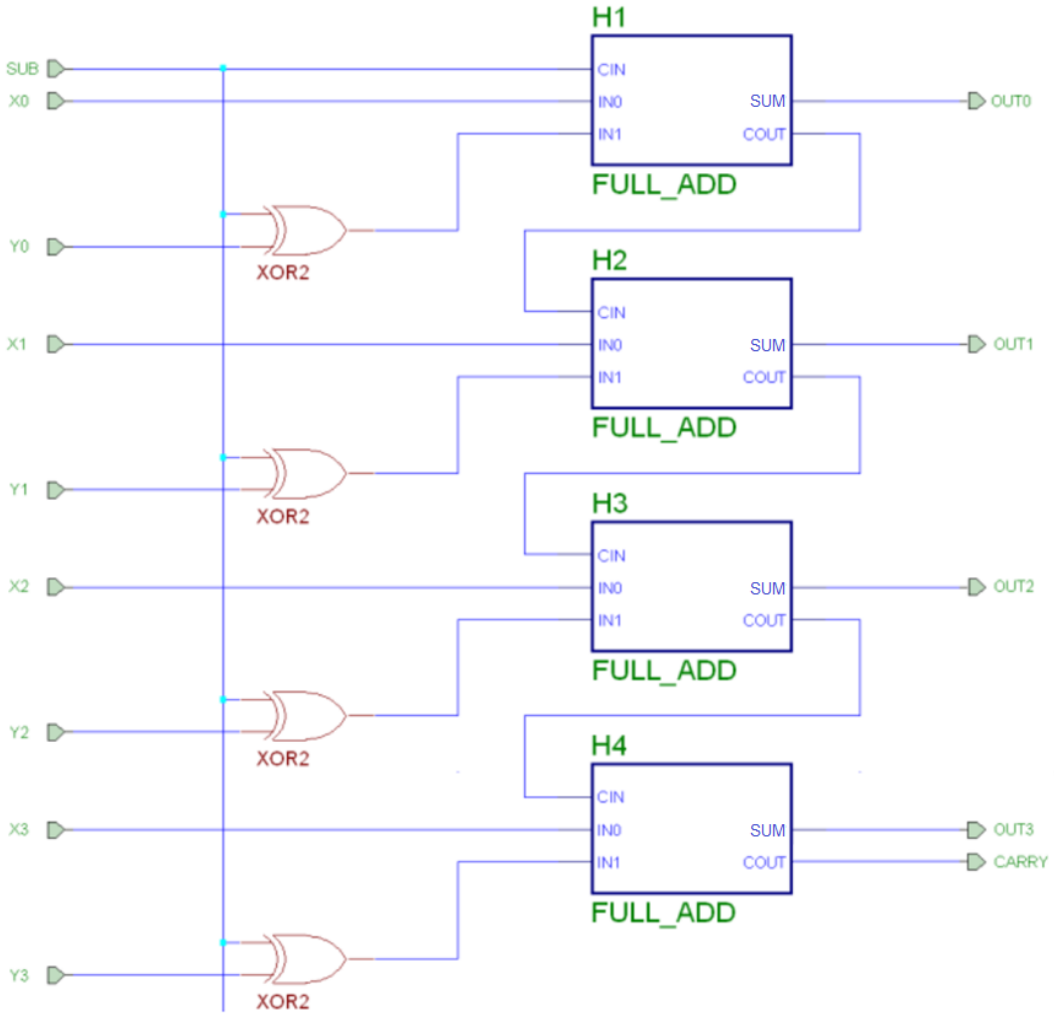
Összeadáskor 4-bites pozitív számokat tudunk összeadni a 4-bites összeadóval és a kimeneti átvitel bit is az eredmény része (ez lesz a legmagasabb helyi értékű eredmény bit).

Amikor kivonást valósítunk meg, akkor – mivel áttérünk a 2-es komplemens kódú számábrázolásra – az eredmény kettes komplemens kódú lesz.

4-bites, 2-es komplemens kódú számábrázoláskor 4-biten **-8..+7** számtartományt tudunk előjel helyesen ábrázolni. A legmagasabb helyi értékű bit (2^3 bit) az előjel bit, ami pozitív szám esetén 0, negatív szám esetén 1 értékű. Negatív eredmény esetén a 2^3 bit 1 értékű lesz, az eredményt a következőképpen lehet kiszámítani (10-es számrendszerbe átszámítva):

$$\textbf{pld. } (0100)_2 - (0110)_2 = (1110)_2 = -1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = -2_{10}$$

Az esetleg keletkező (CARRY) átvitelt pedig kivonáskor figyelmen kívül kell hagynunk. [6]



6. ábra - 4 bites összeadó-kivonó áramkör

Look ahead carry out

Fentebb láttuk, hogy **Cout**=In0*Cin+ In1*Cin+In0*In1. Legyen az i-edik helyi értéken In0=x_i, In1=y_i.

$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$

$c_{i+1} = x_i y_i + c_i (x_i + y_i)$

$g_i = x_i y_i$

$p_i = x_i + y_i$

$c_{i+1} = g_i + p_i c_i$

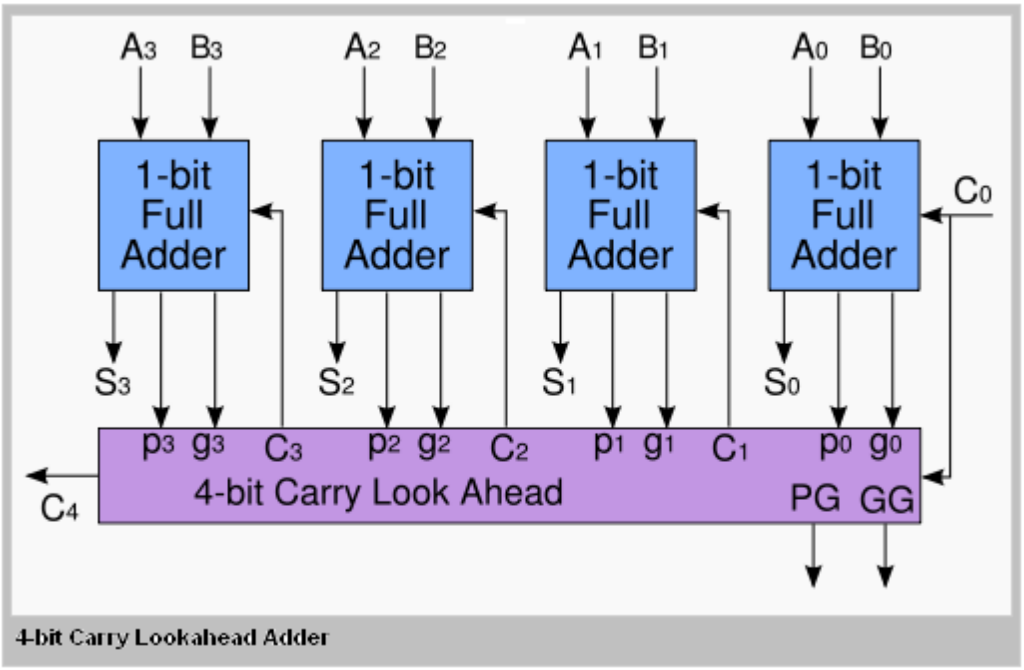
$c_1 = g_0 + p_0 c_0$

$c_2 = g_1 + p_1 c_1$

$c_2 = g_1 + p_1 (g_0 + p_0 c_0) = g_1 + p_1 g_0 + p_1 p_0 c_0$

$c_3 = g_2 + p_2 c_2 = g_2 + p_2(g_1 + p_1 g_0 + p_1 p_0 c_0) = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$

$c_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_0$



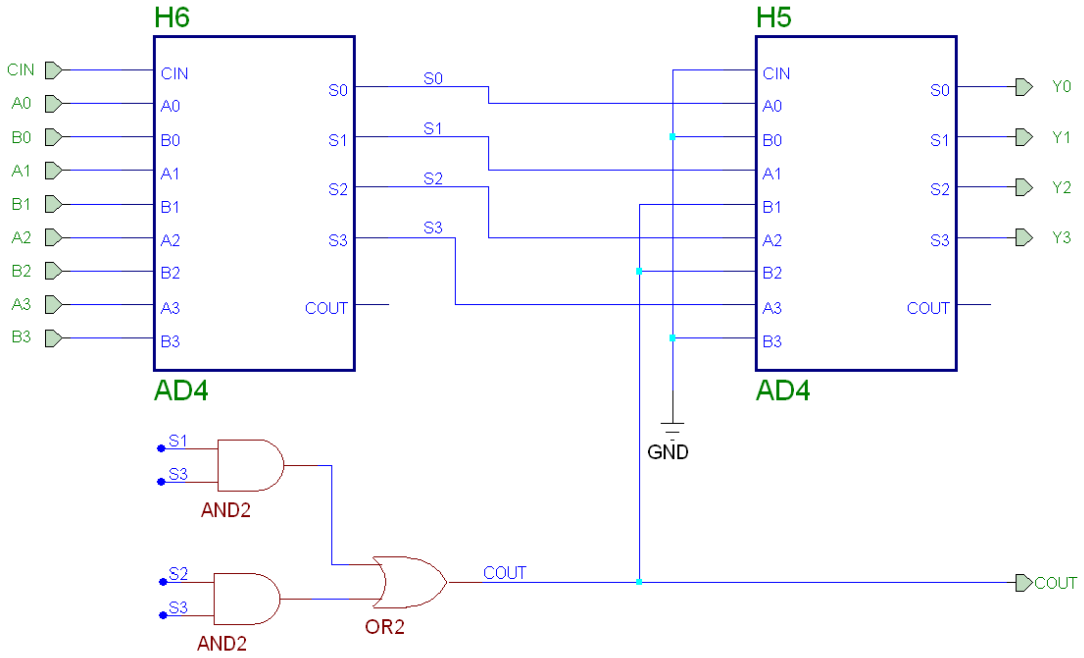
7. ábra - 4 bites összeadó, Look ahead carry out

Látható, hogy az átvitelek képzésében az előző átvitelek nem vesznek részt, ezeket az összeadandók bemeneti bitjeiből számolja ki a gyorsító. Természetesen szükségünk van a megfelelő bemenetszámú kapuáramkörökre.

A BCD összeadó

A decimális összeadónál is használhatunk hexa összeadót (8. ábra). A bemenet itt is 2db 4 bites operandusból és a bemenő átvitelből áll, de az operandusok maximális értéke 9 lehet, és ez a tervező felelőssége. Átvitel akkor keletkezik, amikor az összeg 10 vagy annál nagyobb, maximálisan 19 lehet. Ha az összeg eléri vagy meghaladja a tízes értéket, ki kell

adnunk az átvitel jelet a következő helyi érték felé (3 kapuval megoldva), és a bináris értékből le kell vonnunk tízet. Miután modulo 16-ban dolgozunk, ez ekvivalens a 6 hozzáadásával, és ezt egyszerűbb megvalósítani. Ha az összeadás értéke 9 vagy kisebb, a második összeadóval még hozzáadunk 0-t (látszólag feleslegesen). A COUT=0, amely a H5 második operandusának B1 és B2 bemenetére kerül, a B0, B3 és a Cin fixen nullára van kötve. Ha azonban a H6-on keletkezett összeg 10 vagy annál nagyobb, a COUT=1, ez kerül a H6 B1 és B2 bemenetére, így H6 második operandusa 6 lesz.



8. ábra - BCD összeadó

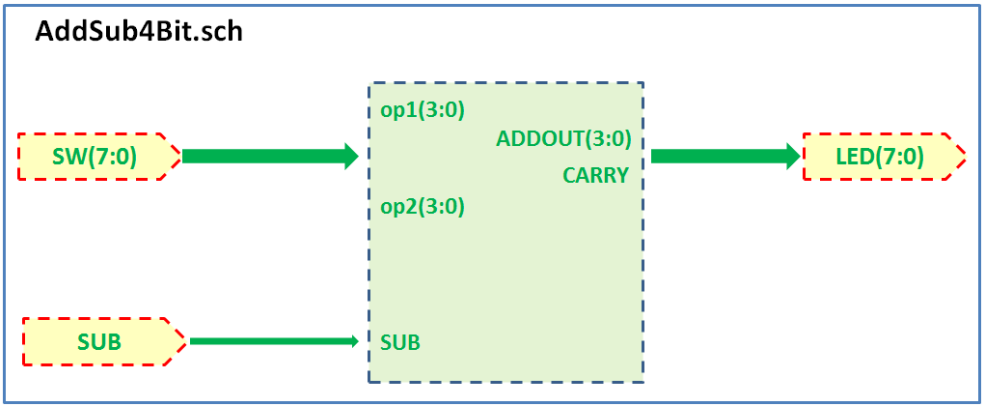
Laborfeladat:

Tervezzen egy 4 bites összeadót-kivonót (9. ábra)! A projekt neve legyen **AddSub4Bit**!

A projekt létrehozása után először készítse el az **Add1bit** nevű modult (makrót), majd ezután az **AddSub4Bit** főmodult (ugyanaz legyen a neve, mint a projektnek). Rajzolt elvi kapcsolási rajzokat kell készíteni az eddig tanultak alapján! [4.ábra] [10.ábra].

Az első operandus **(op1)** a 4 bal oldali SW kapcsoló, a második **(op2)** a 4 jobb oldali legyen. A helyi értékek az operandusokon belül jobbról balra nőnek. A kivonás a **SUB** jel logikai egyes értékére, a jobb szélső nyomógomb (BTN0) lenyomásakor történjen. Az összeadás vagy kivonás eredményét a négy jobbszélső LED-en jelenítse meg, míg a legnagyobb helyi értékű LED-en az átvitel bit értéke látszódjon.

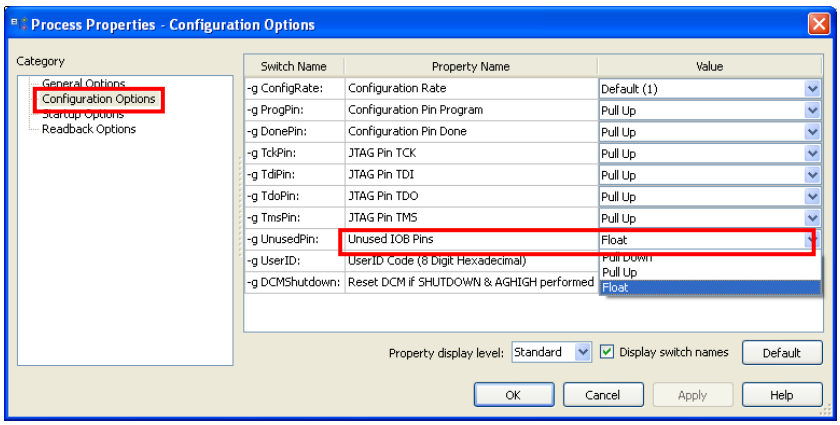
Végezze el az áramkör szimulációját a kapott szimulációs fájl segítségével. Amennyiben az áramkör működése megfelelő, implementálja azt, a kapott bit kiterjesztésű fájlt töltse le a Basys2 kártyára, és a kapcsolók segítségével próbálja ki az áramkör működését.



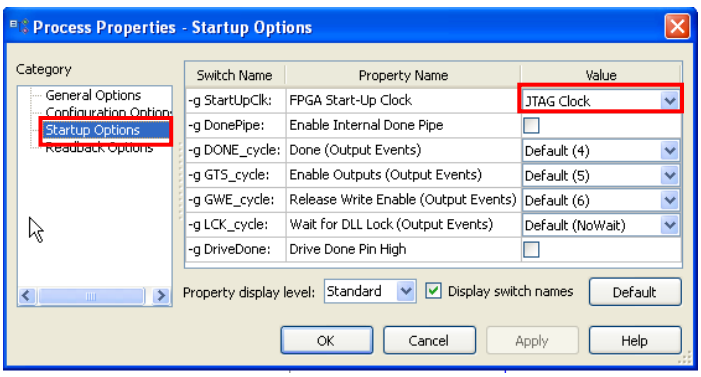
9. ábra - Összeadó-kivonó áramkör kapcsolás blokkvázlata

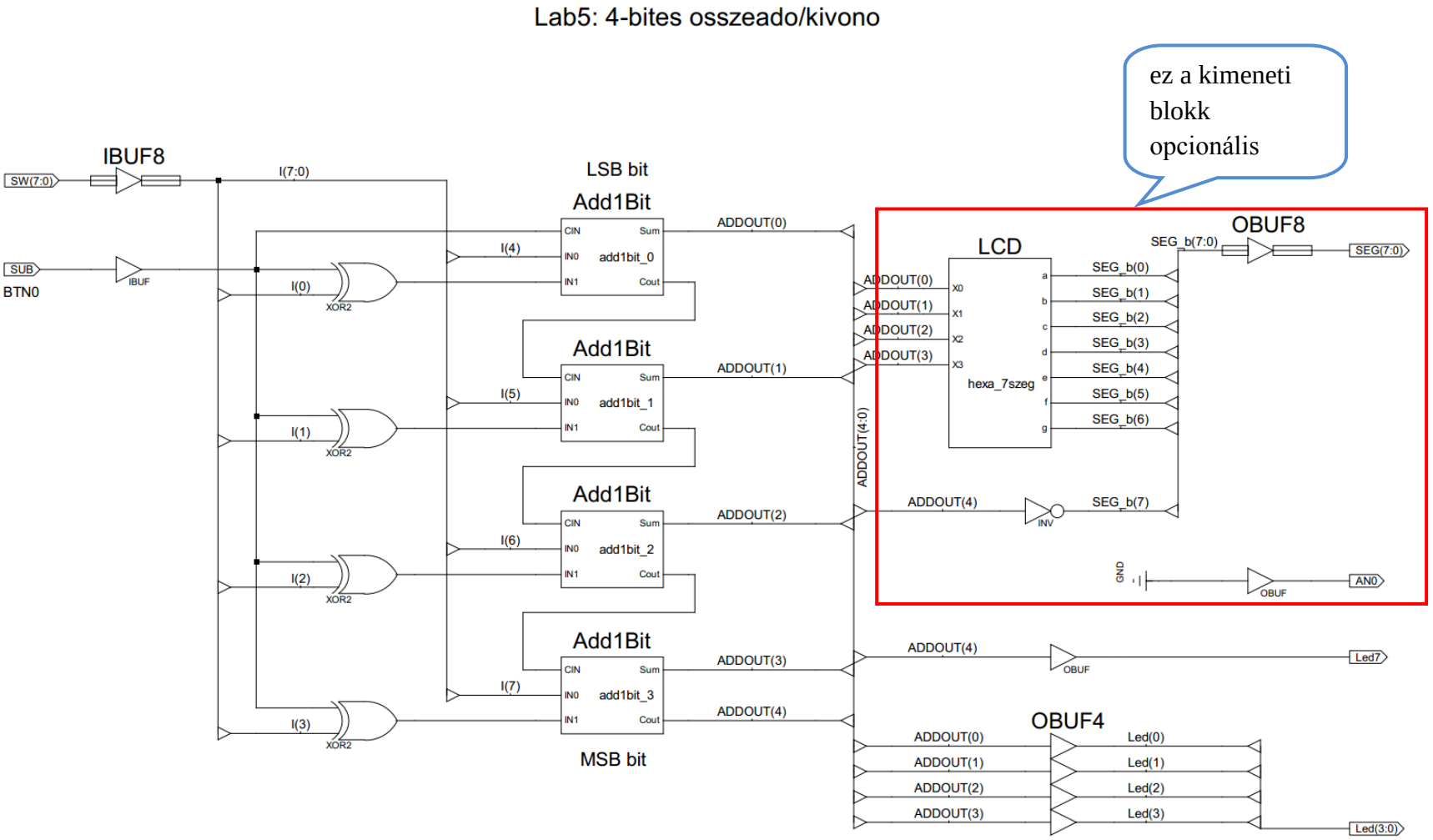
A feladathoz szükséges port nevek (ucf file)			
Port név	**Busz**	**CP132 tokozás**	**Leírás**
SUB	-	NET "sub" LOC = ""G12";	A BTN0 nyomógomb lenyomásakor a bemeneti operandusok kivonása történik. ADDOUT = op1 – op2
SW(7:0)	8 bit	NET "SW<7>" LOC = "N3; NET "SW<6>" LOC = "E2"; NET "SW<5>" LOC = "F3; NET "SW<4>" LOC = "G3"; NET "SW<3>" LOC = "B4; NET "SW<2>" LOC = "K3; NET "SW<1>" LOC = "L3"; NET "SW<0>" LOC = "P11";	Összeadó bemenetei SW(3:0) – 2. operandus, op2 SW(7:4) – 1. operandus, op1
LED(7:0)	8 bit	NET "Led<3>" LOC = "P6"; NET "Led<2>" LOC = "P7"; NET "Led<1>" LOC = "M11"; NET "Led<0>" LOC = "M5"; NET "Led<7>" LOC = "G1";	LED(3:0) – Összeadó-kivonó áramkör kimenete, ADDOUT LED(7) – CARRY flag értéke

A nem használt LED-eket és hétszegmenses kijelzőket le kell tiltani az ISE „Generate Programming File” parancs jobbgombos „Process Properties” menüjében (Float beállítás):



Állítsa be a JTAG clock opciót is!





10. ábra Addsub4bit elvi kapcsolási rajza (ez a főmodul)

Alkalmazandó műszerek és eszközök

- PC számítógép
- Digilent Basys2 Spartan-3E FPGA mérőpanel
- Digilent Adept konfiguráló szoftver

Hivatkozások, felkészüléshez ajánlott irodalom

[1] FPGA fejlesztés a Xilinx ISE Webpack-ben, Elektronikus formában a tantárgy honlapján
[2] Digilent Basys2 Board Reference Manual, Elektronikus formában a tantárgy honlapján
[3] Spartan-3E Libraries Guide for Schematic Designs, Elektronikus formában a tantárgy honlapján
[4] Kóré László: Digitális elektronika I. BMF 1121
[5] Arató Péter: Logikai rendszerek tervezése, Tankönyvkiadó
[6] Zsom Gyula: Digitális technika I, Műszaki Könyvkiadó