Laborgyakorlat
Logikai áramkörök számítógéppel segített tervezése (CAD)

Bevezetés

A laborgyakorlatok alapvető célja a tárgy későbbi laborgyakorlataihoz szükséges ismeretek átadása, az azokban szereplő korszerű tervezési és vizsgálati eszközök, módszerek első bemutatása.

Napjainkban egyre szélesebb körben alkalmazzák a digitális elektronikus eszközöket. Az eszközök bonyolultsága is egyre nő, nagyon gyakori az olyan alkalmazás, amelynél egyetlen IC tokba több százezer vagy akár több millió kaput integrálnak. Ezek az áramkörök hagyományos papír-ceruza módszerekkel már nem tervezhetők, számítógéppel segített tervezésre (CAD) van szükség. A laborgyakorlatok során a **Xilinx ISE WebPack** fejlesztő rendszert és annak a kapcsolási rajz alapú logikai áramkörtervezését használjuk.

A laboratóriumi gyakorlatok célja, hogy a hallgatók

- ismerjék meg a laborgyakorlatok során használt hardver és szoftver eszközöket;
- ismerjék meg a Xilinx ISE Webpack FPGA fejlesztő környezetet és használatát;
- sajátítsák el a kapcsolási rajz alapú logikai áramkörtervezés alapjait;
- önállóan készítsenek el egy kapcsolási rajz alapú példaalkalmazásokat, majd szimulációval és valós hardveren ellenőrizzék azok működését.

Aritmetikai és Logikai Egység (ALU)

A számítástechnikában az aritmetikai-logikai egység (rövidítése ALU az angol Arithmetic Logic Unit alapján) aritmetikai és logikai műveleteket elvégző digitális áramkör. Alapvető eleme a számítógép központi vezérlőegységének. Neumann János alkotta meg az ALU fogalmát 1945-ben, amikor cikkében beszámolt új számítógépéről, az EDVAC-ról. Egy tipikus Neumann-féle CPU belső szerkezetének részében az ALU saját maga végzi az összeadást, a kivonást és más egyszerű műveleteket az inputjain, így adva át az eredményt az output regiszternek.

Az első integrált aritmetikai logikai egység a 74-es TTL sorozat 74181[1] típusú áramköre volt, ami megalapozta a mai modern CPU-k kialakulását.

[1] http://en.wikipedia.org/wiki/74181

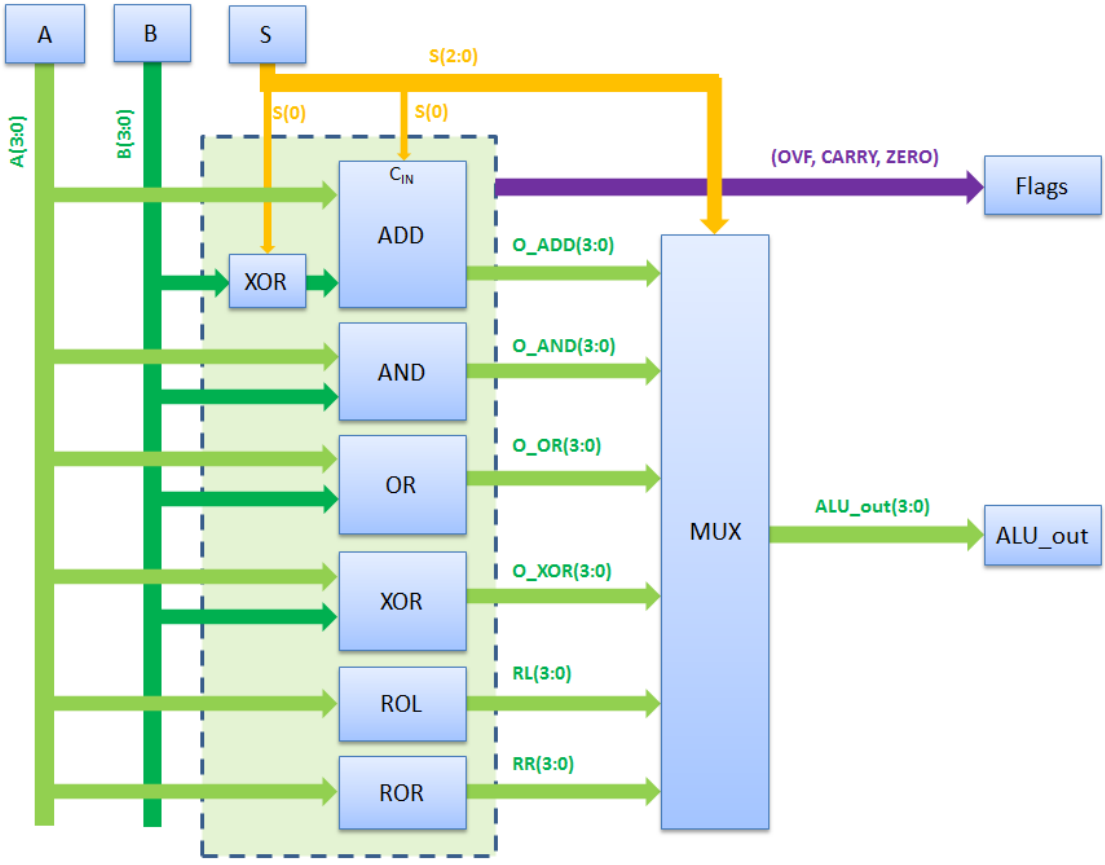
Egy ALU általában a következő alapműveletek végrehajtására képes:

- összeadás és kivonás (fixpontos, lebegőpontos számokkal)
- szorzás, osztás (fixpontos, lebegőpontos számokkal)
- léptetések
- bitszintű logikai műveletek

A műveletek eredményével kapcsolatos egyéb információkat (pl.: túlcsordulás, az eredmény nulla, carry átvitel) is szolgáltat.

A laborgyakorlat során megvalósított ALU

Az egyszerűség kedvéért a laborgyakorlaton csak a legalapvetőbb műveleteket valósítjuk meg az ALU-ban. A laborgyakorlat során megépítésre kerülő ALU blokkvázlata az alábbi ábrán látható.



A modul bemenete két 4 bites szám (**A, B**). Az **S** bemenet az ALU vezérlését végzi. Ezzel lehet kiválasztani, hogy éppen milyen műveletet szeretnénk végezni a két bemeneti értéken. Az **ALU_out** kimeneten az eredmény jelenik meg. A **Flags** kimeneten a műveletvégzés eredményét jelző (Carry, Overflow, Zero) információk jelennek meg.

Az ALU részei

- **ADD:** Két négybites szám összeadására képes teljes összeadó. Ha kivonást is akarunk végeztetni az összeadóval, akkor a második operandus (**B**) kettes komplemenést (bitenként negált +1) kell az elsőhöz (**A**) hozzáadnunk. A negálást exkluzív vagy kapuval, az egyes hozzáadását az első helyi értéken a C_{IN}-re adott egyessel oldhatjuk meg.
- **AND, OR, XOR:** A két négybites bemenet bitenkénti és, vagy, kizáró-vagy kapcsolatát adja eredményül.
- **ROL, ROR:** Az **A** bemenet egy bittel történő balra, illetve jobbra forgatása.
- **MUX:** A multiplexer a vezérlő jeleknek megfelelő műveletvégző egység kimeneti jelét választja ki. Az ALU-ban minden műveletvégző egység kimenetén megjelenik az adott műveleti eredmény, ezért szükséges a multiplexer, ami csak a megfelelő műveleti eredményt adja a kimenetre. Ez lesz az ALU kimeneti értéke.

A blokksémán látható, hogy a multiplexer több négybites adatvonal kiválasztását végzi, majd az eredmény egy szintén négybites adatvonalra kerül. Megvalósítása:

- Négy darab multiplexer, a négybites busz helyi értékeinek kiválasztására.
- Másik lehetőség az **ALU_out_mux**[2] makró használata. Ez egy 8-ról 1-re multiplexer, 4 bites adatcsatornákkal.

A következő táblázat az ALU kimeneti jelét mutatja a vezérlőjeleinek függvényében.

Kiválasztó			Kimenet	Leírás
S2	S1	S0		
0	0	0	Y = A + B	Az 'A' és 'B' bemenet összege.
0	0	1	Y = A - B	Az 'A' és 'B' bemenet különbsége
0	1	0	Y(i) = A(i) & B(i)	Az 'A' és 'B' bemenet bitenkénti ÉS kapcsolata
0	1	1	Y(2:0) = A(3:1), Y(3) = A(0)	Az 'A' bemenet rotálása jobbra.
1	0	0	Y(i) = A(i) | B(i)	Az 'A' és 'B' bemenet bitenkénti VAGY kapcsolata
1	0	1	Y(3:1) = A(2:0), Y(0) = A(3)	Az 'A' bemenet rotálása balra.
1	1	0	Y(i) = A(i) ^ B(i)	Az 'A' és 'B' bemenet bitenkénti XOR kapcsolata
1	1	1	-	-

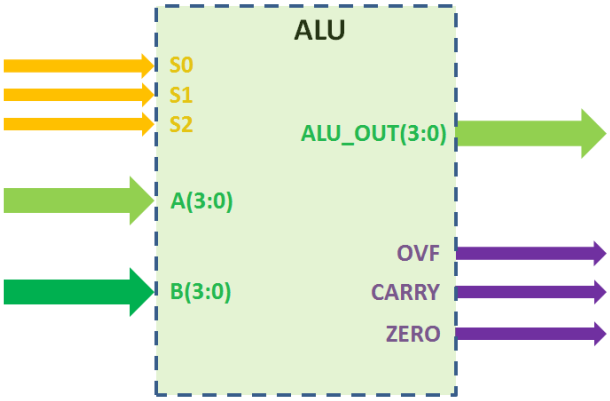
A laborgyakorlaton megépített ALU három **Flag** bitet tartalmaz.

- **Overflow:** Előjeles túlcsordulás. Előjeles számok esetében az összeadás, vagy a kivonás eredménye nem ábrázolható az adott számtartományon.
- **Carry**: Átvitel előjel nélküli műveleteknél. Az összeadás eredménye nem ábrázolható az adott számtartományban.
- **Zero:** Az ALU művelet eredmény nulla lett.

[2] A makró fájlok a laborgyakorlatok alatt megtalálhatóak az FTP* szerveren. A működése és használata a makro_kieg.pdf -ben található.

Laborfeladat:

Készítsen el egy négybites aritmetikai és logikai egységet, a következő műveletek végrehajtására: összeadás, kivonás, ÉS, VAGY, kizáró-VAGY, forgatás jobbra, forgatás balra. Az ALU építésénél felhasználható, a szimbólumok között megtalálható **ADD4** összeadó áramkör, és az FTP-ről letölthető **ALU_out_mux** multiplexer. A kész ALU áramkörből készítsen makrót az alábbi ábrának megfelelően.



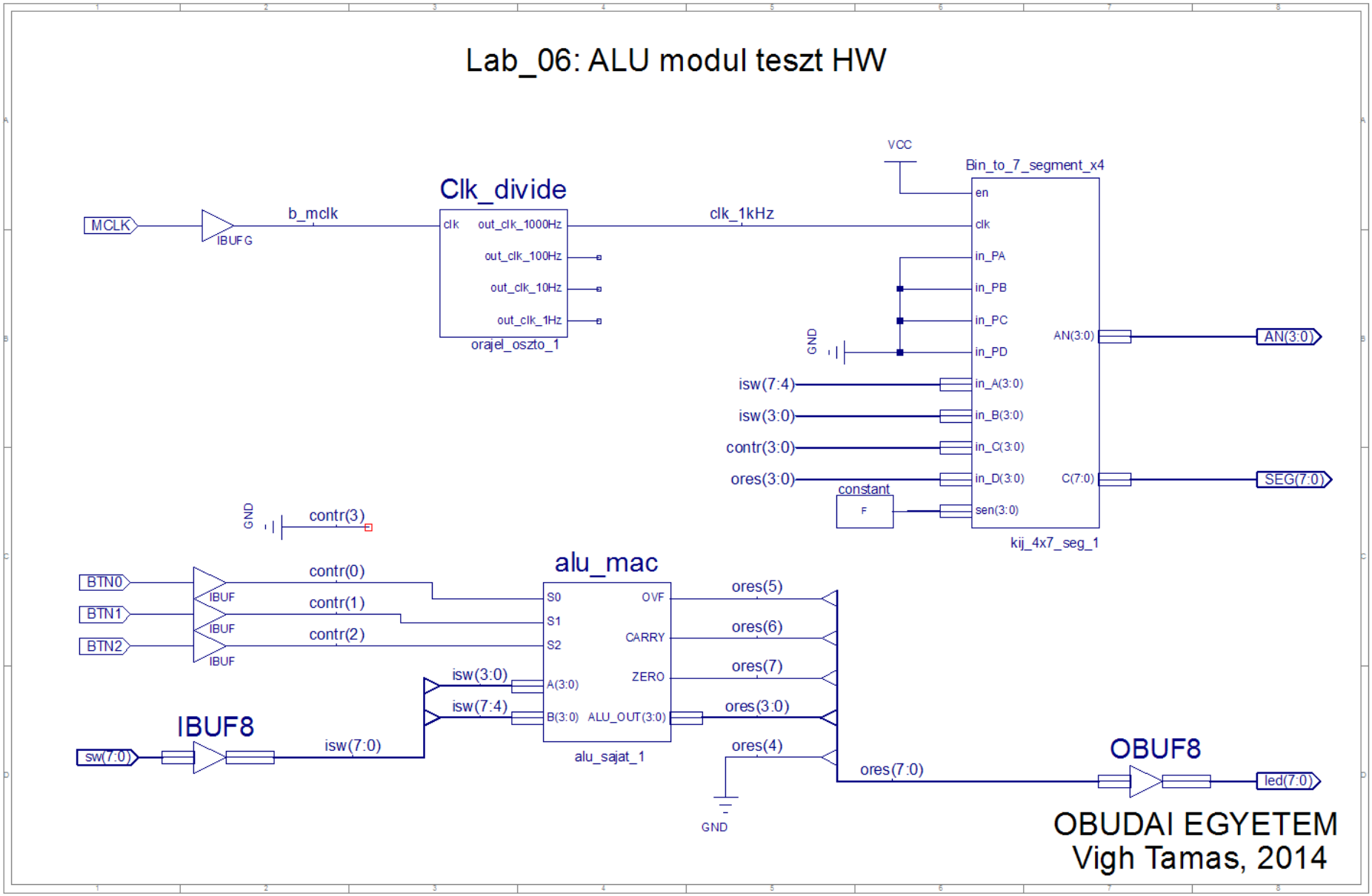
A makró ki-bemeneteit a következőképpen társítsa portokhoz.

Macro	Port (ucf file)
S0, S1, S2	BTN0, BTN1, BTN2
A(3:0)	SW(3:0)
B(3:0)	SW(7:4)
ALU_OUT(3:0)	LED(3:0)
OVF	LED(5)
CARRY	LED(6)
ZERO	LED(7)

A szimulációhoz használja a kiadott tesztfájlt.

Kiegészítő feladat: A bemeneti A, és B számot, a vezérlőbiteket és az eredményt jelenítse meg hétszegmenses kijelzőkön. A feladathoz szükséges makrók: **Clk_divide**, **Bin_to_7_segment_x4**[3].

[3] A makró fájlok a laborgyakorlatok alatt megtalálhatóak az FTP* szerveren. A működése és használata a makro_kieg.pdf -ben található.


Lab_06: ALU modul teszt HW
MCLK
IBUFG
b_mclk
Clk_divide
clk
out_clk_1000Hz
out_clk_100Hz
out_clk_10Hz
out_clk_1Hz
orajel_oszto_1
clk_1kHz
VCC
Bin_to_7_segment_x4
en
clk
in_PA
in_PB
in_PC
in_PD
GND
AN(3:0)
AN(3:0)
isw(7:4)
in_A(3:0)
isw(3:0)
in_B(3:0)
contr(3:0)
in_C(3:0)
ores(3:0)
in_D(3:0)
C(7:0)
SEG(7:0)
constant
F
sen(3:0)
kij_4x7_seg_1
GND
contr(3)
alu_mac
BTN0
BTN1
BTN2
IBUF
IBUF
IBUF
contr(0)
contr(1)
contr(2)
S0
S1
S2
OVF
CARRY
ZERO
A(3:0)
B(3:0)
ALU_OUT(3:0)
isw(3:0)
isw(7:4)
ores(5)
ores(6)
ores(7)
ores(3:0)
ores(4)
IBUF8
sw(7:0)
isw(7:0)
alu_sajat_1
GND
ores(7:0)
OBUF8
led(7:0)
OBUDAI EGYETEM
Vigh Tamas, 2014

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