Beágyazott és Ambiens Rendszerek Laboratórium

BMEVIMIA350

Mérési feladatok az 1., 2. és 3. mérési alkalomhoz

A mérés tárgya: FPGA áramkörök és tervezési rendszereik megismerése

A mérések során egy egyszerű tervezési feladat lépésein keresztül bemutatásra kerül egy korszerű integrált fejlesztőrendszer, amely a programozható logikai eszközök tervezését a specifikációtól a megvalósított terv ellenőrzéséig minden lépésben támogatja.

A tervezés során a **Verilog HDL** hardverleíró nyelven történő tervezési és specifikációs technológiát alkalmazzuk. Az egyes funkciók, műveleti egységek leírásakor törekszünk a **magas szintű**, szintézis orientált leírásra, de amennyiben az áttekinthetőség megkívánja, alkalmazzuk a strukturális leírást is, a szükséges modulok, **hierarchikus egységek** kapcsolatainak specifikálására.

A megvalósítandó feladat: *Számológép tervezése*

A tervezési feladat egy egyszerű számológép megtervezése, amely a laboratóriumban rendelkezésre álló fejlesztőeszközön realizálható. A számológép a 4 alapművelet végrehajtására képes, ezeket a műveleteket 4 bites bemenetei adatokon hajtja végre. Az eredmények megjelenítése a fejlesztő kártya kijelzőin történik.

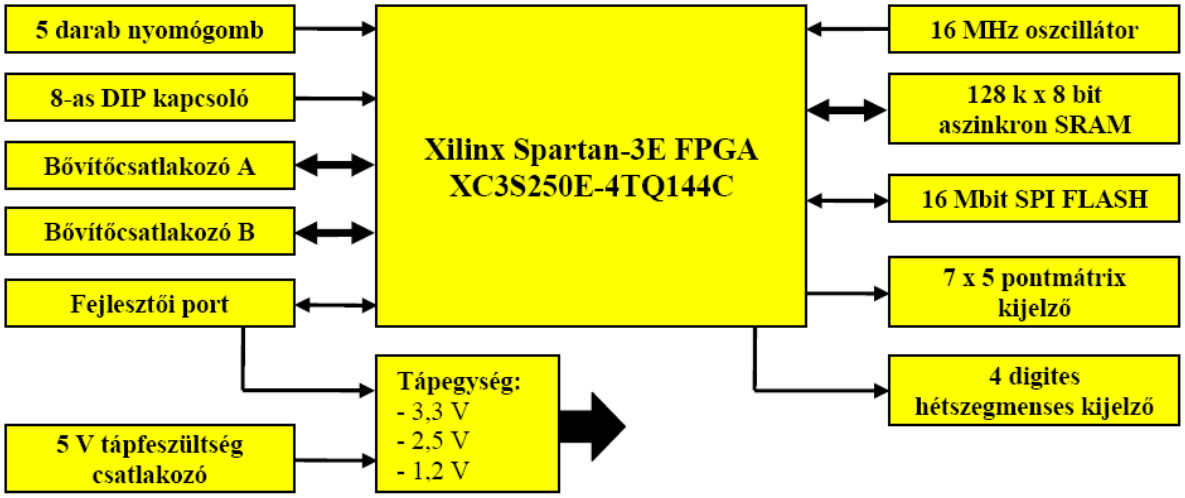
Az egység tervezése során **több egyszerűbb verzió** megvalósítását is elvégezzük, ami segít a teljes technológiai folyamat lépéseinek és a fejlesztőkártya tulajdonságainak készség szintű megismerésében. *Mérési jegyzőkönyvet csak a 3. mérés végén kell leadni, amely az utolsó két tervezési feladatot dokumentálja.*

A számológép általános specifikációja a következő:

- Bemeneti adatok:
  - 4 bites operandusok, beállításuk a fejlesztőkártya kapcsolóval történik
  - A bemeneti adatok pozitív egészként értelmezettek
- Műveletek:
  - Összeadás
  - Kivonás
  - Szorzás
  - Osztás
- Műveletek kijelölése a kártyán található nyomógombokkal történik
- Eredmény megjelenítése
  - LED diódákon, közvetlenül
  - 7 szegmenses kijelzőn

A LOGSYS kártya ismertetése

A LOGSYS Spartan-3E FPGA kártya egy egyszerű felépítésű, elsősorban kezdő felhasználók számára készült FPGA kártya. A kártya blokkvázlata az alábbi ábrán látható.



1. ábra: A Logsys kártya felépítése

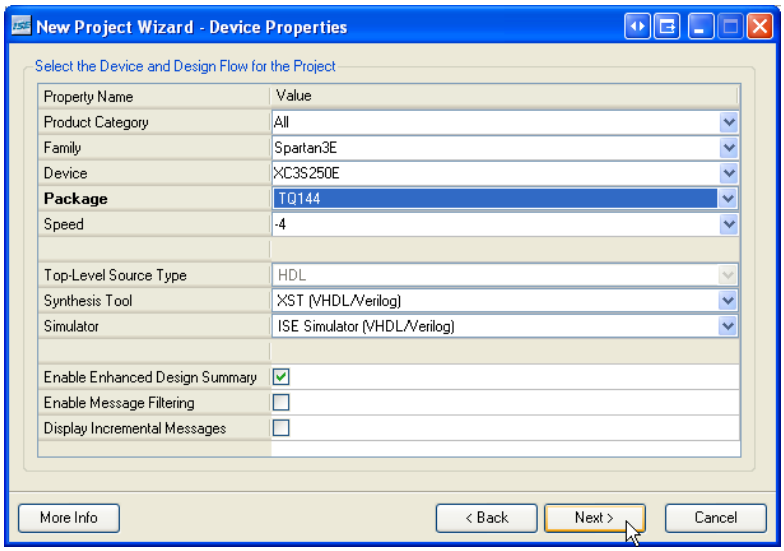
A kártya részletes leírása az alábbi helyen érhető el:

http://logsys.mit.bme.hu/download/LOGSYS_SP3E_FPGA_Board.pdf

A LOGSYS honlapja:

http://logsys.mit.bme.hu/

Fontos! Az projekteket mindig az alábbi beállításokkal fordítsuk:



A felhasznált perifériák

LEDek:

A LOGSYS Spartan-3E FPGA kártyán található 8 darab LED bekötését az alábbi táblázat mutatja. A LED-ek LD0-tól LD7-ig vannak számozva, a bal szélső LED az LD7, a jobb szélső LED az LD0. A LED-ek vezérlő jelei aktív magas szintűek.

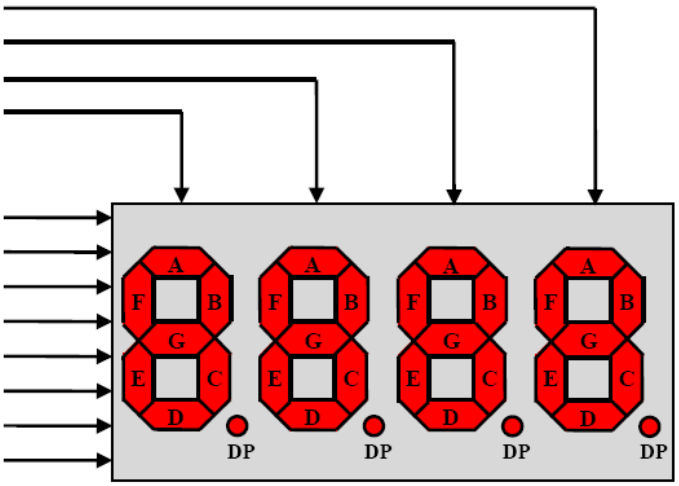
LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA láb	P43	P50	P51	P52	P53	P54	P58	P59

Hétszegmenses kijelző:

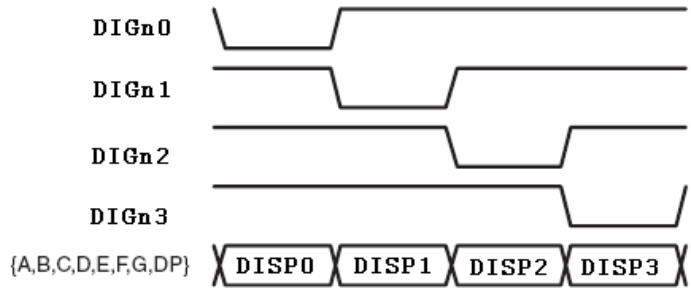
A LOGSYS Spartan-3E FPGA kártyán található 4 digites hétszegmenses kijelző bekötését a az alábbi ábra mutatja. A karakterek DIG0-tól DIG3-ig vannak számozva, a bal szélső karakter a DIG3, a jobb szélső karakter a DIG0. A hétszegmenses kijelző **minden vezérlő jele aktív alacsony** szintű. **A szegmens jelek és a pontmátrix kijelző sor jelei közösek.**

Kiválasztó jel	FPGA láb
DIGn0	P40
DIGn1	P35
DIGn2	P26
DIGn3	P22

Szegmens	FPGA láb
SEGn0 (A)	P16
SEGn1 (B)	P17
SEGn2 (C)	P20
SEGn3 (D)	P21
SEGn4 (E)	P31
SEGn5 (F)	P32
SEGn6 (G)	P33
DP (tizedespont)	P34



Mivel az egyes digitek szegmensvezérlő jelei közösítve vannak, így több digites kijelzés csak idő-multiplexálás segítségével valósítható meg.



Először, a jobb oldali digitre írunk ki. A DIGn0 jel **aktív alacsony** kiválasztásával (DIGn0 = 0, DIGn1=DIGn2=DIGn3= 1), a szegmensvezérlő vonalakra (A-G) az első digiten megjelenítendő számnak megfelelő vezérlést adjuk. (Például „0”-ás számjegy kirajzolásához, a szegmensvezérlő jelek az alábbiak: A=B=C=D=E=F =0, G=1).

Ezután a második digitet választjuk ki (DIGn0 = 1, DIGn1=0, DIGn2= DIGn3= 1), miközben az ennek megfelelő szegmensvezérlést biztosítjuk.

A négy digitet a fenti módon egymás után ciklikusan választjuk ki. Amennyiben a kiválasztás frekvenciája nagyobb mint a szem tehetetlensége, úgy a ciklusokat nem érzékeljük, a kijelzés folyamatosnak tűnik. Ehhez a vezérlésnek ~500 Hz-nél nagyobb frekvenciával kell járnia. (A multiplexálás frekvenciájának az áramkör szab határt, ami ~10kHz.)

DIP kapcsoló:

A LOGSYS Spartan-3E FPGA kártyán található 8-as DIP kapcsoló bekötését az alábbi táblázat mutatja. A kapcsolók 0-tól 7-ig vannak számozva, a bal szélső kapcsoló sorszáma a 7, a jobb szélső kapcsoló sorszáma a 0. Az adott FPGA láb a kapcsoló alsó állásában logikai alacsony szintű, a kapcsoló felső állásában pedig logikai magas szintű lesz.

Kapcsoló	7	6	5	4	3	2	1	0
FPGA láb	P47	P48	P69	P78	P84	P89	P95	P101

Nyomógombok:

A LOGSYS Spartan-3E FPGA kártyán található 5 darab nyomógomb bekötését az alábbi táblázat mutatja. A nyomógombok jelölése balról jobbra rendre BTN3-BTN0 és RST. Az adott FPGA lábra logikai magas szint kerül a nyomógomb megnyomása esetén. Az RST gomb elsősorban az alaphelyzetbe állításra szolgál, de tetszőlegesen is felhasználható.

Nyomógomb	BTN3	BTN2	BTN1	BTN0	RST
FPGA láb	P12	P24	P36	P38	P41

Órajel források:

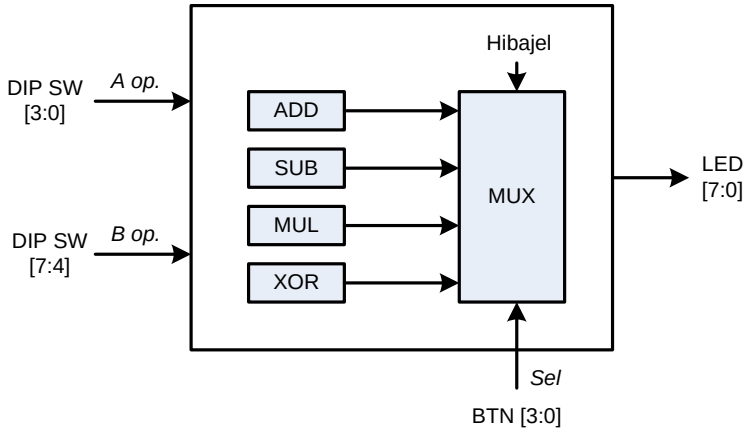
Az FPGA a kártyán lévő 16 MHz-es oszcillátortól és a **fejlesztői port CLK** (P129) vonaláról kaphat órajelet. A fejlesztés során célszerű a fejlesztői port órajelét használni, amely frekvenciája változtatható Az órajel források bekötését az alábbi táblázat mutatja.

Órajel forrás	**FPGA láb**
16MHz-es oszcillátor	P56
Fejlesztői port CLK vonala	P129

Tervezési feladatok

1. Tervezési feladat: CALC_1 nevű egység:

Miután a bemenetei operandusokat a kapcsolókon beállítottuk, a kiválasztott művelet típusának megfelelő nyomógomb megnyomása után az eredményt *közvetlen bináris formában a 8 db LED* diódán jelenítjük meg. Az egyes műveletek: Összeadás, kivonás, szorzás, ***kizáró vagy*** (XOR). Az esetleges *hibás eredményt* az összes LED bekapcsolásával jelezzük. Hibának számít a kivonás negatív eredménye.



Megjegyzés: Mind a négy alapművelet a többféleképen megvalósítható a digitális technikában tanultak szerint:

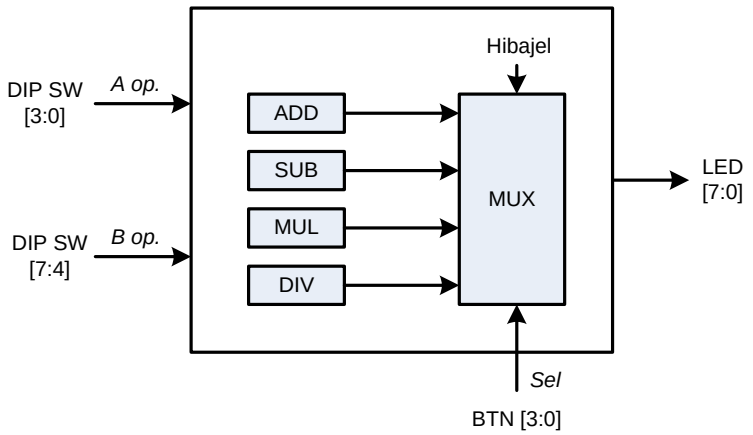
1. Egy triviális megoldás lenne (amit most nem használunk), hogy 4 db **műveleti táblázatot** definiálunk, amiket memóriák valósítanak meg. Bár a 2 db 4 bites operandus miatt a táblázat csak 256 adatot tartalmazna, ezt a megoldást elvetjük.
2. Az egyes műveleteket elemei, **bitszintű algoritmus**ai alapján is felépíthetők az egységek, azonban bizonyos esetekben a fejlesztőrendszer a kapuszintű leírásból már nem képes felismerni a valódi műveletet, ezért nem képes az eszközben esetleg erre rendelkezésre álló speciális, beépített erőforrásokat felhasználni.
3. Ezt elkerülendő, törekedjünk a **magas szintű műveleti operátorok** használatára, ahol pedig a logikai hálózatot generáló szintézer támogatása nem áll rendelkezésre, (például a későbbiekben megvalósításra kerülő osztás művelete) ott a probléma végiggondolásával, egy jó minőségű megoldás megtervezésével oldjuk meg a feladatot.

Feladat: Az elkészített CALC_1 számológép forráskódját *szimulációval* ellenőrizzük. Amennyiben az eredmények helyesek, fordítsuk le, generáljuk a konfigurációs adatokat és *ellenőrizzük a működést* a mérőpanelen.

Fontos! Az összes tervezett rendszer minden modulja **globálisan szinkron** legyen. (Minden always block ***ugyanannak*** az órajelnek a felfutó élét használja.) A szekvenciális modulok tartalmazzanak egy **közös reset** jelet is.

2**. Tervezési feladat:** CALC_2 nevű egység:

Módosítsuk a CALC_1 nevű egységet úgy, hogy az alábbi műveleteket támogassa: Összeadás, kivonás, szorzás, ***osztás*****.** Bővítsük a hibakezelő egységet, hogy figyelje a nullával történő osztást is.

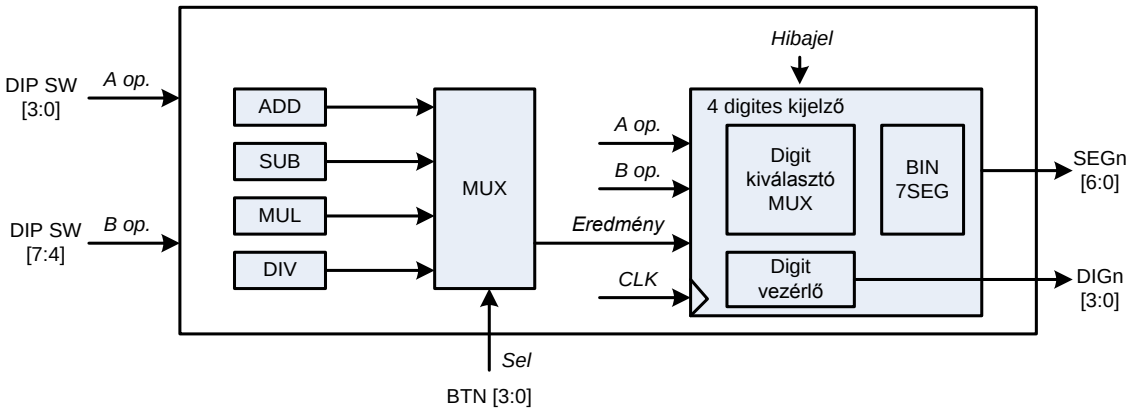


A bináris osztás több módon is megvalósítható:

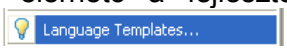
1. Felépíthetünk egy valódi **kombinációs** osztó hálózatot, „papír ceruza módszerrel”.
2. Mivel csak 4 bites számokról van szó, létrehozhatunk egy nagy „case” szerkezetet amely tartalmazza az eredményt.
3. Készíthetünk egy **szekvenciális** megvalósítást is. (Osztandóból ciklikusan kivonjuk az osztót, amíg az osztandó nagyobb mint az osztó; miközben számoljuk a ciklusokat. A hányados egyenlő a ciklus számmal.)

3**. Tervezési feladat:** CALC_3 nevű egység:

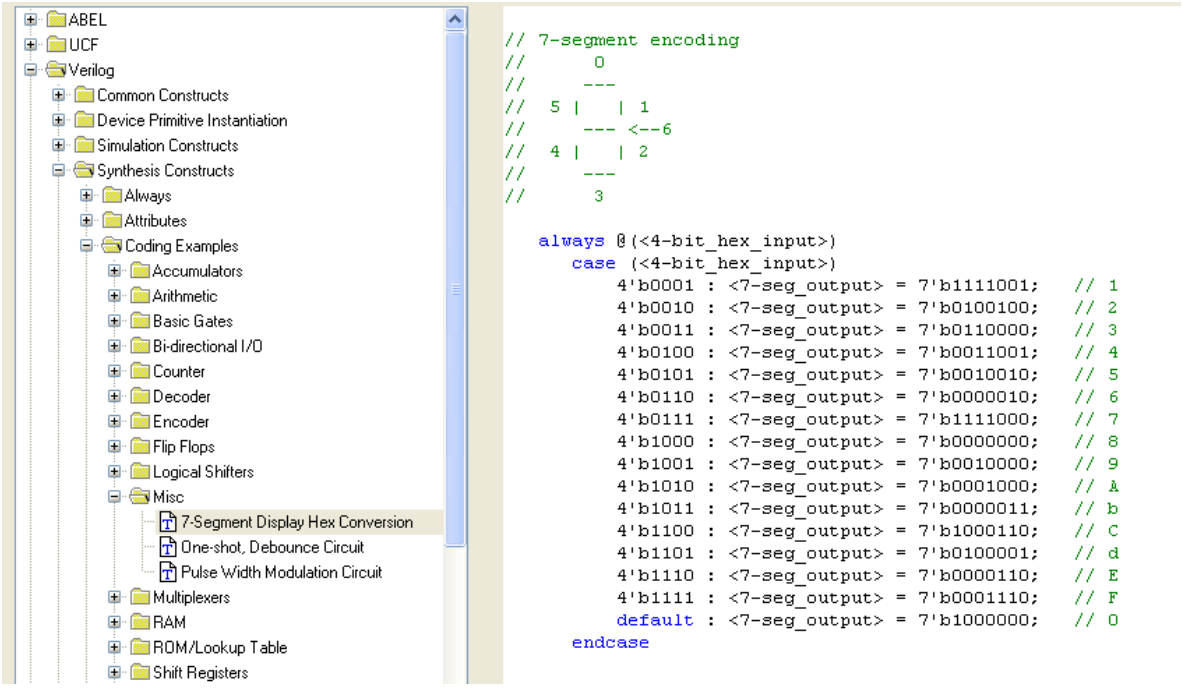
Az elkészített CALC_2 a technikai specifikációt teljesíti, de használata, az eredmények leolvasása kényelmetlen. Egészítsük ki a számológépet *numerikus kijelzéssel*.



Készítsük el azt a kijelző egységet, amely a 4 digiten balról jobbra haladva egy-egy digiten kijelzi a két bemeneti operandust és az utolsó két digiten pedig az eredmény értékét. A kijelzés úgy működjön, hogy amíg valamelyik műveleti gombot nem nyomtuk meg, az eredmény digitek *maradjanak sötétek*. A 7 szegmenses kijelző a bináris értékeket **hexadecimális** formában jelenítse meg. A hibás vagy nem értelmezhető eredmény esetén az utolsó két digit az *„EE”* hibajelzést tartalmazza.

A Bináris – 7 szegmenses kijelző átkódoló modul (*BIN7SEG*) elérhető a fejlesztői környezetben is. Elérése: **Edit Menu > Language Templates…** 

. Verilog > Synthesis Constructs> Coding Examples > Misc > 7-Segment Display Hex.

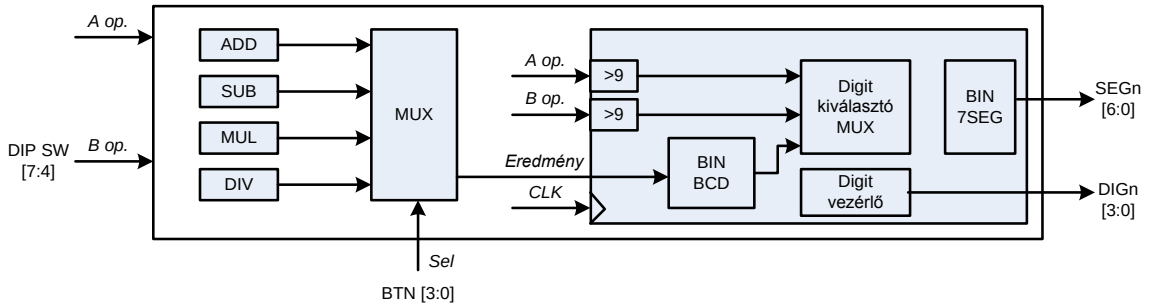


4. Tervezési feladat: CALC_4 nevű egység:

A CALC_3 egység szolgáltatása mérnöki szemmel már egészen kielégítő, azonban átlagos felhasználó számára esetleg zavaró a hexadecimális eredmény. Alakítsuk át a rendszert, úgy, hogy a bemeneti adatok is és az eredmények is **tízes számrendszerben** jelenjenek meg.

Ennek érdekében a CALC_4 számológép a bemenetein beállított kapcsolóállásokból csak a 0 … 9 tartományba eső értékeket fogadja el és jeleníti meg, az ennél nagyobb (de beállítható) értékekre a bemeneti adatok kijelzője egy „E” hibajelzést ad. Az így korlátozott bemeneti adattartomány esetén, az eredmény minden művelet esetén kisebb, mint decimális 100, tehát 2 decimális digiten ábrázolható.

A CALC_4 számológép tartalmazzon egy bináris-BCD kódkonvertert, ami lehetővé teszi, hogy a 2 utolsó eredmény digit értelmezése a szokásos decimális számrendszernek feleljen meg.



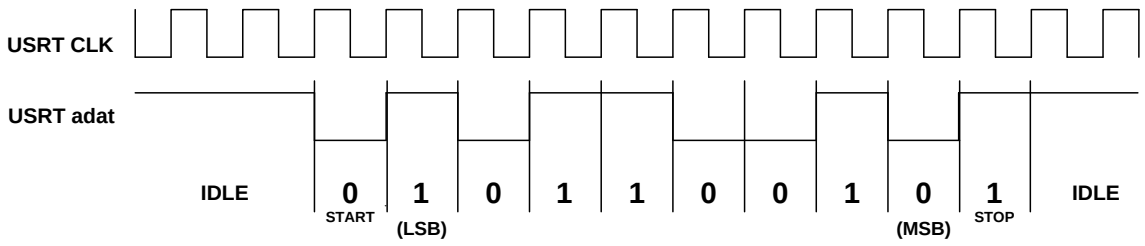
Megjegyzés: A bináris-BCD és a BCD-bináris kódkonverziók nem egyszerű műveletek. Az osztáshoz hasonlóan a szintézer a feladatra nem ad automatikus megoldást.

5. Tervezési feladat: CALC_5 nevű egység:

A CALC_5 egység az eddigiekben megtervezett egység jelentősebb módosítása az alábbiak szerint:

- A bemeneti operandusokat és a végrehajtandó műveletet a nyomógombok és kapcsolók helyett egy USRT interfészen keresztül kapja az egység. Mindkét operandus legfeljebb 2 decimális számjegyet tartalmazhat. A terminal programban megadott értékeket egy ENTER validálja. Azaz egy bemeneti szekvencia: 1-2 digites decimális szám; ENTER; műveleti kód; ENTER; 1-2 digites decimális szám. ESC megnyomására az USRT vevő alaphelyzetbe áll, és újra az első operandust várja.
- Az eredmény megjelenítése a 4 darab 7-szegmenses kijelzőn történik, decimális formátumban. Az első érvényes bemeneti kombináció érkezéséig csak a 8. szegmens (pont) világítson. Normál működés során érvényes végeredmény esetén a 4 digit az eredményt mutassa, újabb operandus-műveleti kód fogadása alatt viszont újfent csak a 4 pont legyen bekapcsolva.

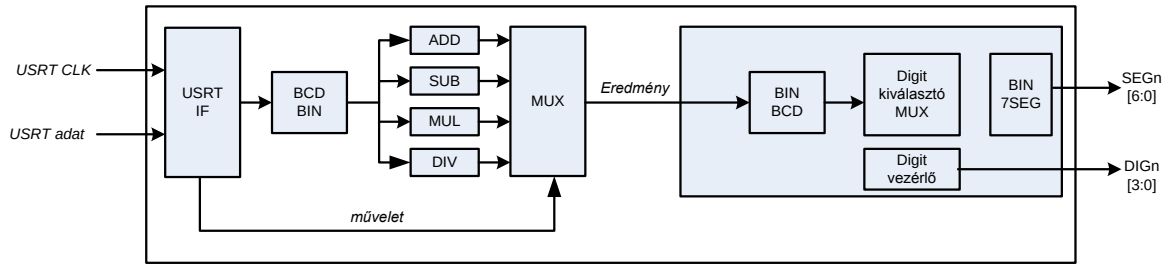
Az USRT vevő egység szabványos keretformátum vételére legyen képes az alábbi beállításokkal: 8 adatbit, nincs paritás, 1 stop bit. Az adó (LOGSYS kábel) az adatokat az USRT órajel felfutó élére küldi, így vevő oldalon a lefutó él környékén történő mintavételezés megfelelő. A rendszer többi moduljához hasonlóan az USRT vevő esetében is elvárás a teljesen szinkron rendszer tervezése, azaz az USRT órajel lefutó élét a rendszer órajel segítségével kell detektálni, az közvetlenül nem köthető FF-ok órajel bemenetére! Egy keret átvitelét az alábbi időzítési diagram szemlélteti (egy keretben egy ASCII karakter átvitele történik):



A szükséges ASCII karakterek kódjai:

billentyű	**ASCII kód**
0 – 9	0x30 – 0x39
+	0x2B
-	0x2D
*	0x2A
/	0x2F
ENTER	LOGSYS GUI beállítás függő: \r: 0x0D (kocsi vissza) \n: 0x0A (új sor) \r \n: 0x0D, majd 0x0A
ESC	0x1B

A megvalósítandó rendszer blokkvázlata tehát:



USRT lábkiosztás:

- USRT CLK: LOGSYS kábel CLK (P129)
- USRT RX (bemeneti adat): LOGSYS kábel MOSI (P120)
- USRT TX (kimeneti adat, fix logikai 1): LOGSYS kábel MISO (P143)

Ellenőrző kérdések:

1. Milyen előnyei vannak a hardver leíró nyelv használatának?
2. Mit jelent a hierarchikus tervezés, mire ügyeljünk a kapcsolatok specifikálásakor?
3. Mi a különbség a strukturális és a viselkedési leírás között?
4. Mit jelent a nem blokkoló értékadás?
5. Specifikáljon egy 4 bites pozitív élvezérelt regisztert aszinkron aktív magas RST-el!
6. Mi az időmultiplex vezérlés lényege?
7. Milyen erőforrások találhatók az FPGA-kban?
8. Mit jellemzi az SRAM technológiájú FPGA-kat?