Logikai tervezés gyakorlatok (2021.)

Tartalom

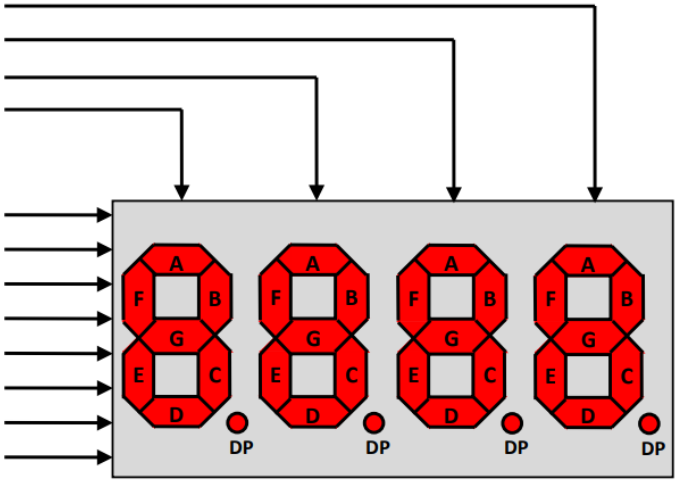
1. 7-szegmenses vezérlő

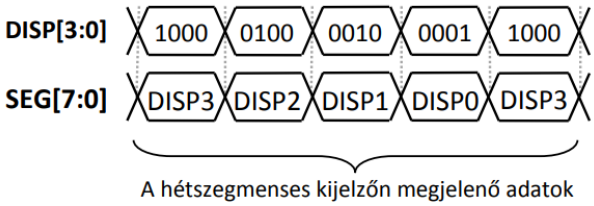
Vezette mérés, amelyen egy 7-szegmenses kijelző vezérlőjének implementációján keresztül megismerkedünk a Xilinx Vivado fejlesztői környezettel.

A Logsys Kintex-7 kártyán egy 4-digites, időmultiplexált 7-szegmenses kijelzőt találunk. Ennek vezérléséhez az FPGA és a kijelző között 4 digit engedélyező jelet (DSIP0…DSIP3), valamint egy 8 bites szegmens buszt találunk (SEG0…SEG7, DP). Az időmultiplexálás azt jelenti, hogy a szegmens jelek minden digitre közösek, így adott időpillanatban mindig csak egyetlen digit aktív. A szegmens vonalakra mindig az aktív digiten megjeleníteni kívánt érték szegmens kódjait adjuk. A digitek közötti váltásnak elég gyorsnak kell lennie ahhoz, hogy a szemünk ne érzékelje ezt (>>60 Hz), de elég lassúnak ahhoz, hogy a 7-szegmenses vezérlő LED-jei megfelelően működjenek. Megfelelő választás a kHz nagyságrendű váltás.

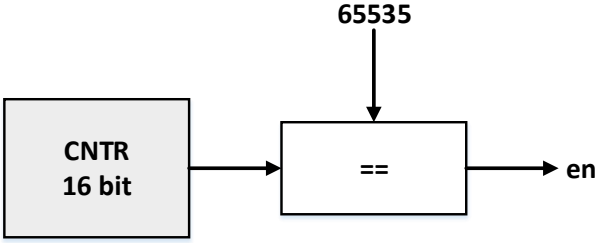
Kiválasztó jel	FGA láb
DISP0	F10
DISP1	D11
DISP2	D10
DISP3	B10

Szegmens	FPGA láb
SEG0 (A)	D8
SEG1 (B)	E10
SEG2 (C)	B9
SEG3 (D)	D9
SEG4 (E)	E13
SEG5 (F)	G11
SEG6 (G)	C9
DP (tizedespont)	G10

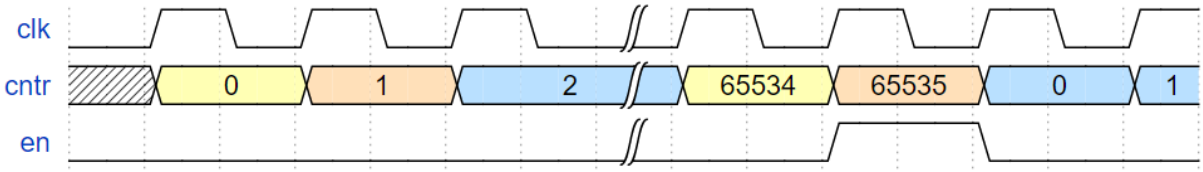




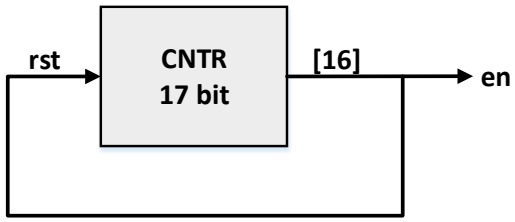
Az FPGA órajel forrása 100 MHz frekvenciájú, ahhoz hogy a megfelelő frekvenciájú jelváltásokat generálni tudjuk, egy ~kHz frekvenciájú engedélyező jelre van szükség, ami az órajelnek 100.000-ed része. Mivel nem kritérium pontosan 1 kHz-es frekvencia előállítása, így az egyszerűség kedvéért a legközelebbi kettő hatvánnyal, 65.536-tal osztunk. Ezt legegyszerűbben egy 16 bites számlálóval tehetjük meg:



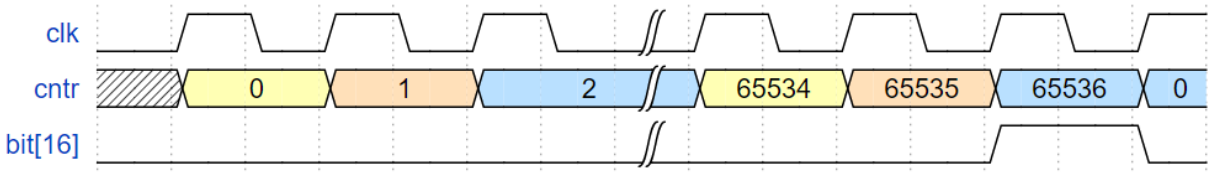
A generált hullámforma:



A fenti blokkvázlatban szerepel egy 16 bites komparátor, ami elhagyható amennyiben a számlálót 17 bitesre egészítjük ki, és a számlálót reset-eljük amikor a legfölső bitje 1.



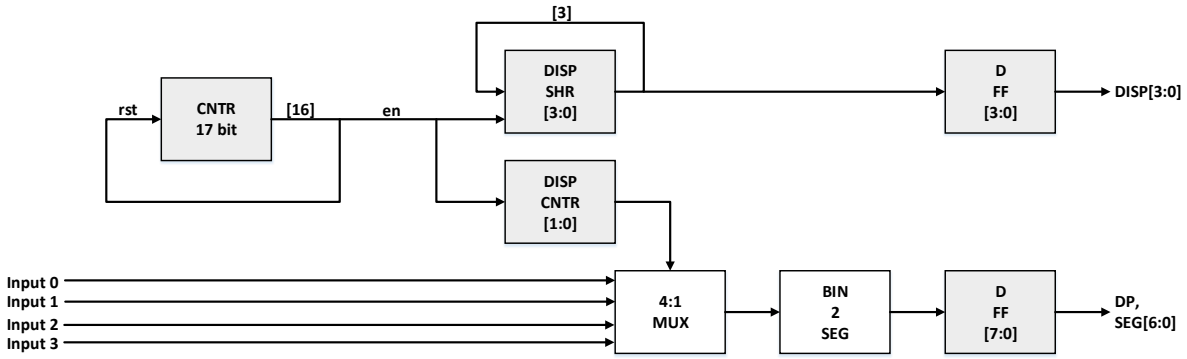
Az így létrejövő hullámforma:



A megfelelő frekvenciájú engedélyező jel generálása mellett az alábbi komponensekre van szükség:

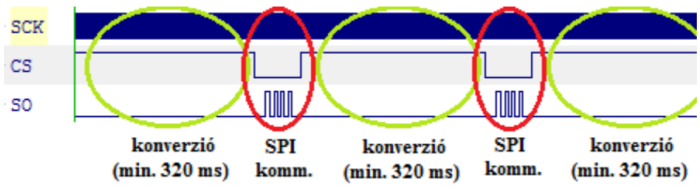
- 4 bites visszacsatolt shift regiszter a DISP[3:0] jel generálásához. Periodikusan a 0001→0010→0100→1000→0001… értékeket veszi fel.
- 2 bites számláló, amely a DISP shift regiszterrel szinkronban jár, mindig azt mutatja, hogy hányadik digit van kiválasztva.
- 4 bites 4:1 multiplexer, amely a 4 bemeneti adatból kiválasztja az aktív digitnek megfelelőt.
- Bináris → szegmens enkóder, amely a 4 bites bináris értékből előállítja a szegmens kódot, azaz megadja, hogy melyik szegmenseknek kell világítania.

A teljes blokkvázlat tehát:

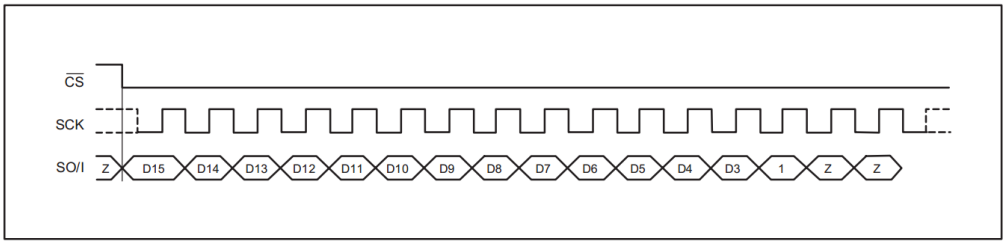


2. TMP121 SPI interfész

A hőmérő adatlapja alapján a kommunikációs ciklus az alábbi:



Az új hőmérséklet érték előállítása 320 ms ideig tart, ez alatt a CS jelnek 1 értékűnek kell lennie. Ezt követően kiolvasható a hőmérséklet az alábbi időzítési diagramnak megfelelően:



Tehát minden egyes SPI átvitel során 16 bitet olvasunk, amiből az első 13 bit az érvényes adat.
Időzítési kritériumok:

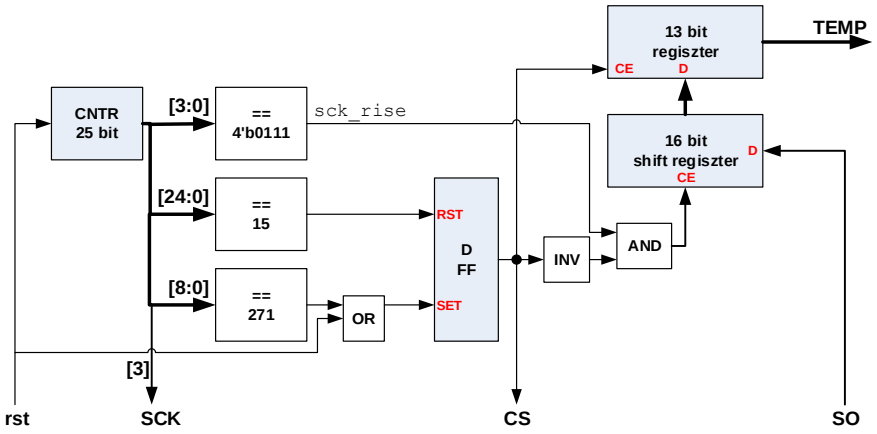
PARAMETER		MIN	MAX	UNITS
SCK Period	t_1	100		ns
SCK Falling Edge to Output Data Delay	t_2		30	ns
$\overline{CS}$ to Rising Edge SCK Set-Up Time	t_3	40		ns
$\overline{CS}$ to Output Data Delay	t_4		30	ns
$\overline{CS}$ Rising Edge to Output High Impedance	t_5		30	ns

A fentiek alapján megfontolásaink:

- A teljes ciklus ~320 ms, azaz 32.000.000 100 MHz-es órajel. 25 bites szabadon futó számlálót (cntr) használva az időzítésre az SPI kommunikáció + várakozás 33.554.432rendszerórajel lesz.
- A hőmérő órajelének minimális periódusideje 100 ns, azaz maximális frekvenciája 10 MHz. A 100 MHz-es rendszerórajelet 16-tal leosztva 6,25 MHz-es órajel adódik, azaz az SCK periódusideje 160 ns lesz.
- 16-tal történő órajel osztásra a szabadon futó 25 bites számláló cntr[3] indexű bitje használható.
- A CS jel váltásának legalább 40 ns-mal meg kell előznie az SCK felfutó élét (setup time) → ha az SCK lefutó élénél változtatjuk, akkor ez bőven teljesül.
- A hőmérő az SCK lefutó élére adja ki az adatot, kimeneti jelterjedési ideje 30 ns → SCK felfutó élénél stabil az adat, itt mintavételezhető.
- Mivel az SCK megegyezik cntr[3]-mal, a felfutó és lefutó éleket jelző engedélyező jelek:
  - felfutó él: cntr[3:0]==3'b0111
  - lefutó él: cntr[3:0]==3'b1111
- Az SPI kommunikációt a teljes ciklus elején végezzük. A CS-t SCK legelső lefutó élekor váltjuk 0-ba, ez cntr==15. A komparáláskor a teljes 25 bites számlálót használni kell, mert a CS 0-ba állítása csak egyszer tehető meg a teljes periódus alatt.
- Az SPI kommunikáció 16 SCK, ami 16*16=256 rendszerórajel. Ilyen hosszú CS pulzust kell generálni, azaz ha cntr==15 esetén állítottuk 0-ba, akkor cntr==15+256=271 esetén kell 1-be állítani. Ehhez elegendő a számláló alsó 9 bitje, hiszen az nem okoz problémát ha a számláló nagyobb értékeinél többször 1-t írunk a CS-t megvalósító FF-ba, hiszen az már úgyis 1 lesz.

- A hőmérő az MSB bitet küldi először, így a soros→párhuzamos átalakításra egy balra shiftelő shiftregiszterrel oldható meg.
- Annak érdekében, hogy a kimeneten mindig érvényes adat legyen, a shiftregiszter tartalmát át kell írni egy kimeneti regiszterbe. Erre minden olyan időpont megfelelő, amikor a shiftregiszterben érvényes adat van. Ez lehet pl. a CS felfutó éle (ekkor ciklusonként egyszer írjuk a kimeneti regisztert), de az is megfelelő ha CS==1 esetén mindig frissítjük a kimeneti regisztert, hiszen ekkor a shiftregiszter nem változik.

A megvalósítandó blokkvázlat tehát:



A hőmérő által küldött adat kettes komplemens, 4 bit törtrésszel:

TEMPERATURE (°C)	DIGITAL OUTPUT(1) (BINARY)	HEX
150	0100 1011 0000 0000	4B00
125	0011 1110 1000 0000	3E80
25	0000 1100 1000 0000	0C80
0.0625	0000 0000 0000 1000	0008
0	0000 0000 0000 0000	0000
−0.0625	1111 1111 1111 1000	FFF8
−25	1111 0011 1000 0000	F380
−55	1110 0100 1000 0000	E480
(1) The last two bits are high impedance and are shown as 00 in the table.		

A Logsys kártyán 4 digites hétszegmenses kijelző van, ezen az előjelet, az egészrész két bitjét és a törtrészt jelenítjük meg, először hexadecimálisan, majd decimálisan (BCD). Utóbbihoz szükség van egy bináris→BCD átalakításra. Megvalósítási lehetőségek:

- ROM-ban tároljuk a BCD értékeket, a ROM-t pedig a bináris hőmérséklet értékkel címezzük.
- SHIFT-ADD3 (lásd PDF a tárgy honlapon) kombinációs logikaként. Mivel az átalakításra sok órajel áll rendelkezésre, így feleslegesen nagy az erőforrásigénye.
- SHIFT-ADD3 szekvenciális megvalósítása. Minden órajelben 1 bitet számolunk ki, így egyetlen komparátor-összeadó (+plusz vezérlő logika) szükséges. PREFERÁLT MEGOLDÁS.

3. Audió CODEC illesztése

A gyakorlat során egy sztereó audió CODEC (coder-decoder) illesztünk az FPGA-hoz. A CODEC-ben található ADC kimenetét az FPGA-ban egy regiszteren keresztül visszacsatoljuk a CODEC DAC bemenetére (későbbi gyakorlaton a direkt visszacsatolás helyett FIR szűrőt valósítunk meg az FPGA-ban).

A Kintex-7 kártyán található CODEC típusa Cirrus Logic CS4270.
(Adatlap: https://d3uzseaevmutz1.cloudfront.net/pubs/proDatasheet/CS4270_F1.pdf). Beállítástól függetlenül igaz, hogy a CODEC audió interfésze az alábbi jeleket tartalmazza:

- MCLK: CODEC master clock.
- SCLK: bit clock. A soros adatinterfész időzítő jele.
- LRCK: left-right clock. Jobb/bal csatorna kiválasztó jele. Frekvenciája megegyezik a mintavételi frekvenciával.
- SDOUT: Az ADC soros adatkimenete.
- SDIN: A DAC soros adatbemenete.

A CODEC konfigurációjára az alábbi lehetőségeink vannak:

- Stand-alone mód: a konfigurációs lábak megfelelő logikai értékre történő állítása.
- Szoftver mód: SPI vagy I2C interfészen keresztül. A CODEC akkor kerül szoftver módba, ha a nRST bemenet 1-be állítását követően 1.045 mintavételi időn belül érvényes SPI vagy I2C tranzakcióval a „power down” regiszter bitet beállítjuk.

Az egyszerűség kedvéért mi most az első megoldással élünk. A konfigurácó során a következő paramétereket kell beállítanunk.

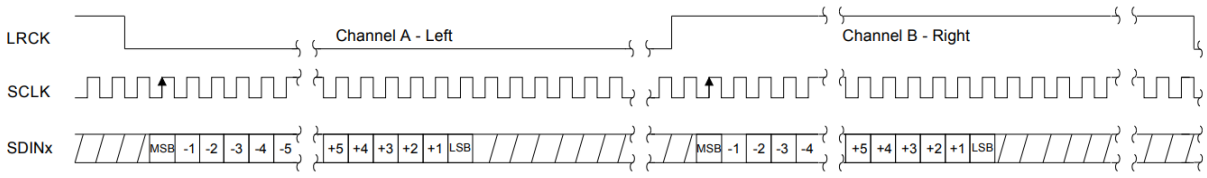
- Slave vagy master mód.
    - Slave módban minden órajel (MCLK, SCLK, LRCK) bemenet a CODEC számára.
    - Master módban az MCLK bemenet, a többi órajelet a CODEC szolgáltatja.
    - Az órajelek elvárt frekvencia-viszonyait az alábbi táblázat muatatja.

Master Mode					
	MCLK/LRCK	SCLK/LRCK	LRCK	MDIV2	MDIV1
Single-Speed	256	64	Fs	0	0
	384 (Note 22)	64	Fs	0	1
	512	64	Fs	1	0
	1,024	64	Fs	1	1
Double-Speed	128	64	Fs	0	0
	192 (Note 22)	64	Fs	0	1
	256	64	Fs	1	0
	512	64	Fs	1	1
Quad-Speed	64	64	Fs	0	0
	96 (Note 22)	64	Fs	0	1
	128	64	Fs	1	0
	256	64	Fs	1	1
Slave Mode					
	MCLK/LRCK	SCLK/LRCK	LRCK	MDIV2	MDIV1
Single-Speed	256	32, 48, 64, 128	Fs	0	0
	384 (Note 22)	32, 48, 64, 96	Fs	0	1
	512	32, 48, 64, 128	Fs	1	0
	1,024	32, 48, 64, 96	Fs	1	1
Double-Speed	128	32, 48, 64	Fs	0	0
	192 (Note 22)	32, 48, 64	Fs	0	1
	256	32, 48, 64	Fs	1	0
	512	32, 48, 64	Fs	1	1
Quad-Speed	64	32, 48, 64	Fs	0	0
	96 (Note 22)	32, 48, 64	Fs	0	1
	128	32, 48, 64	Fs	1	0
	256	32, 48, 64	Fs	1	1

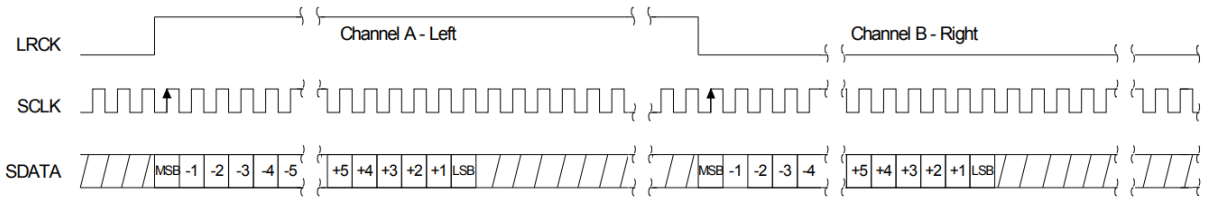
- Single-speed, double-speed vagy quad-speed üzemmód. A beállítandó üzemmódot egyértelműen meghatározza a mintavételi frekvencia.

Mode	Sampling Frequency
Single-Speed	4-54 kHz
Double-Speed	50-108 kHz
Quad-Speed	100-216 kHz

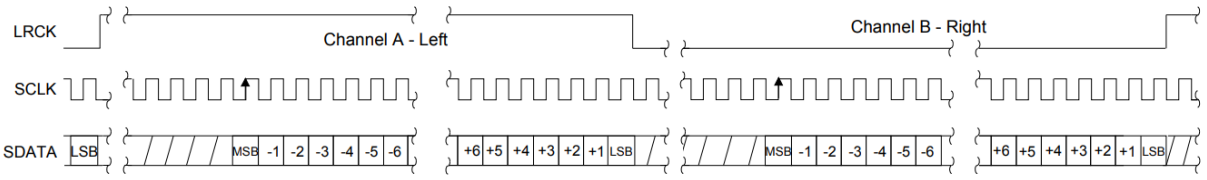
- Audió interfész üzemmódja: I2S, Right-Justified vagy Left-Justified.
    - I2S üzemmód



    - Left-Justified üzemmód



    - Right-Justified üzemmód (Stand-alone módban nem érhető el!!)



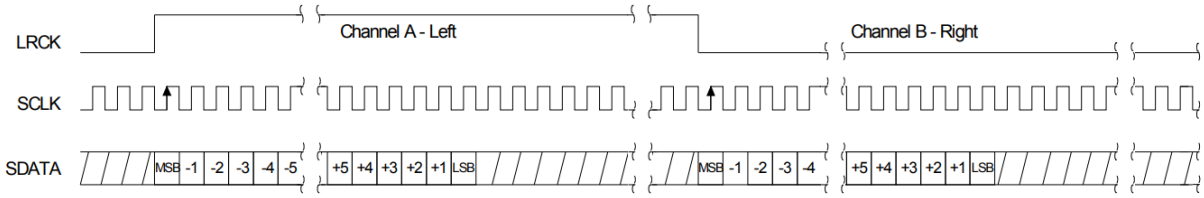
A gyakorlat során az alábbi beállításokkal fogjuk használni a CODEC-t:

- 192 kHz mintavételi frekvencia → Quad mode.
- MCLK/LRCK=256.
- SCLK = 64*FS.
- Stand-alone mód: konfiguráció lábak segítségével.
- Slave mód, azaz minden órajelet az FPGA szolgáltat.
- Left-Justified audió interfész.

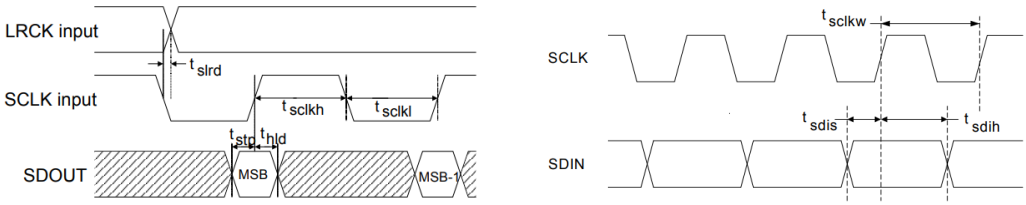
Ehhez az alábi konfigurációs láb beállítások szükségesek:

- Quad mode: {M1, M0} = 2’b11.
- MCLK/LRCK: {MDIV2, MDIV1} = 2’b11.
- Slave_mód: Az SDOUT lábat le kell húzni (a lehúzó ellenállás megtalálható a NYÁK-on).
- Left-Jsutified: I2S/LJ lábat 0-ba kell húzni.

A Left-Justified üzemmód időzítési diagramja:



A soros interfész időzítési adatai:



Parameter		Symbol	Min	Typ	Max	Unit
Sample Rate	Single-Speed Mode	Fs	4	-	54	kHz
	Double-Speed Mode	Fs	50	-	108	kHz
	Quad-Speed Mode	Fs	100	-	216	kHz
MCLK Specifications						
MCLK Frequency (Note 15)	Stand-Alone Mode	fmclk	1.024	-	55.296	MHz
	Serial Control Port Mode	fmclk	1.024	-	55.296	MHz
MCLK Duty Cycle			40	50	60	ns
Slave Mode						
LRCK Duty Cycle			40	50	60	%
SCLK Period (Note 15)	Single-Speed Mode	t_{sclkw}	$\frac{1}{(128)Fs}$	-	-	s
	Double-Speed Mode	t_{sclkw}	$\frac{1}{(64)Fs}$	-	-	s
	Quad-Speed Mode	t_{sclkw}	$\frac{1}{(64)Fs}$	-	-	s
SCLK Duty Cycle			45	50	55	ns
SCLK falling to LRCK edge		t_{slrd}	-20	-	20	ns
SDOUT valid before SCLK rising		t_{stp}	10	-	-	ns
SDOUT valid after SCLK rising		t_{hld}	5	-	-	ns
SDIN valid to SCLK rising setup time		t_{sdis}	16	-	-	ns
SCLK rising to SDIN hold time		t_{sdih}	20	-	-	ns

Megfontolásaink:

- A gyakorlaton kb. 192 kHz-es mintavételi frekvencia elérése a cél, az MCLK = 256 * fs beállítást fogjuk használni, azaz MCLK = 256 * 192 kHz = 49,152 MHz.
- A kiszámolt MCLK kétszerese 98,304 MHz, ami igen közel esik a Kintex-7 kártya oszcillátornak 100 MHz-es frekvenciájához. Amennyiben ezt használjuk rendszerórajelként, úgy 195,3125 kHz-es mintavételi frekvencia adódik. Ez ugyan nem szabványos audió frekvencia, de a CODEC szempontjából még megfelelő, így ezt a megoldást fogjuk használni. Ebben az esetben az MCLK a rendszerórajel fele.
- Egy csatorna érvényes adata 24 bit, egy LRCK fél-periódus alatt 32 bit kerül átvitelre. Left-Justified módban az LRCK élt követő első 24 bit az érvényes adat. A teljes LRCK periódus alatt 64 SCLK van, azaz SCLK = 64 * LRCK.
- Összegezve:

    - MCLK = CLK / 2.
    - LRCK = MCLK / 256 = CLK / (256*2)
    - SCLK = LRCK * 64 = CLK / 8
- Az LRCK jel váltása az SCLK váltásával együtt történhet (-20…20 ns tűréssel egybe esnek).
- A DAC az SCLK felfutó élére mintavételezi a bemeneti soros adatot, ezt legalább ezen él előtt 16 ns-mal ki kell adni (setup time), illetve 20 ns-ig még ott kell tartani (hold time). Az SCLK peridódusidejének fele ennél jóval nagyobb, így SCLK lefutó éle megfelelő időpont az adatkiadásra.
- A CODEC adatkimenete az SCLK előtt legalább 10 ns-mal már érvényes, utána pedig 5 ns-ig még biztosan érvényes marad, így az SCLK felfutó élére mintavételezhető.
- Észrevehető, hogy minden, a CODEC számára előállított órajel (ezek az FPGA szempontjából NEM órajelek, hanem egyszerű kimeneti jelek) a rendszerórajel (CLK) 2 hatványad része, így ezek egyetlen számláló megfelelő bitjeinek kivezetésével generálhatók. Konkrétan:
    - LRCK = CLK / 512 → bit[8]
    - SCLK = CLK / 8 → bit[2]
    - MCLK = CLK / 2 → bit[0]
- A bemeneti soros → párhuzamos, illetve a kimeneti párhuzamos → soros átalakítás megoldható 1-1 shift regiszterrel.
- Szükséges még a két csatornára 1-1 „shift regiszter érvényes” jel (egy rendszer órajel hosszúságú pulzus).
    - Ezek generálhatók az ütemező számláló azon részéből, ami bit számlálóként értelmezhető (0…31 között számol), tehát olyan, mintha a generált SCLK-ra számolna → bit[7:3].
    - Az így generált jel 1 SCLK hosszúságú, ahhoz hogy ez egyetlen CLK idejű legyen, szükséges feltétel még a SCLK felfutó élét jelző impulzus.
    - Azt, hogy a bemeneti shiftregiszter melyik csatorna adatát tartalmazza, a generált LRCK jelből lehet eldönteni.
- A kimeneti shiftregiszter töltését engedélyező jelet hasonló megfontolások alapján lehet generálni.
- Az FPGA konfigurációja, illetve a globális reset után reset jelet generálunk a CODEC-nek, majd várunk legalább 1.045 mintavételi időt, hogy a CODEC biztosan stand-alone módban legyen és érvényes kimenetet generáljon.

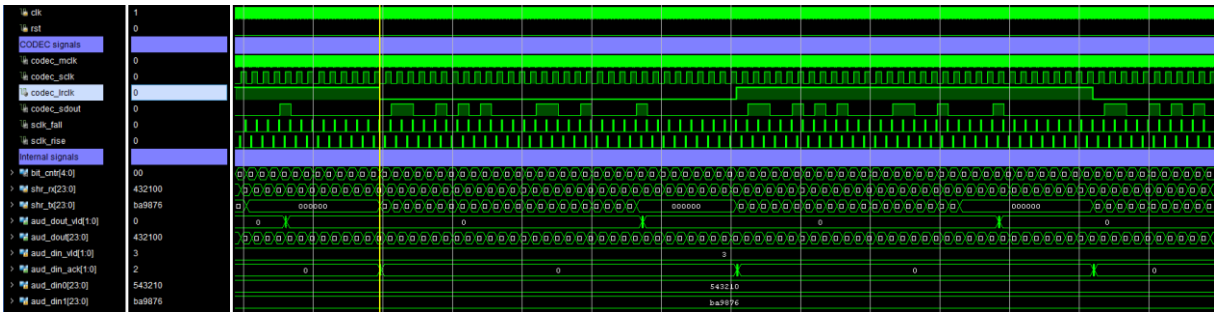
A modul portjai:

- clk: Bemenet; rendszerórajel.
- rst: Bemenet; globális reset, aktív magas.
- codec_m0: Kimenet; CODEC konfigurációs láb.
- codec_m1: Kimenet; CODEC konfigurációs láb.
- codec_i2s: Kimenet; CODEC konfigurációs láb.
- codec_mdiv1: Kimenet; CODEC konfigurációs láb.
- codec_mdiv2: Kimenet; CODEC konfigurációs láb.
- codec_rstn: Kimenet; a CODEC aktív alacsony reset jele.
- codec_mclk: Kimenet; a CODEC MCLK órajele.
- codec_lrclk: Kimenet; a CODEC LRCK órajele.
- codec_sclk: Kimenet; a CODEC SCLK órajele.
- codec_sdin: Kimenet; a CODEC soros adatbemenete.
- codec_sdout: Bemenet; a CODEC soros adatkimenete.

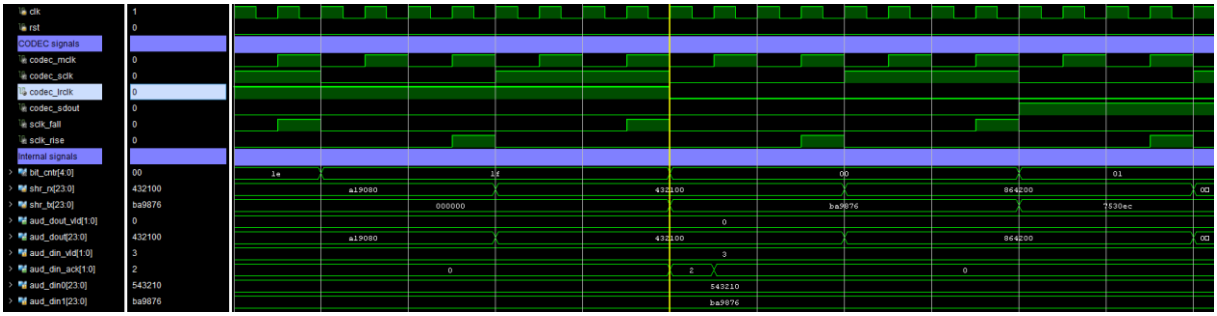
- aud_dout_vld: 2 bites kimenet; a CODEC-től fogadott párhuzamos adat érvényes (mindkét csatornára 1-1 bit), 1 rendszerórajel hosszúságú impulzus
- aud_dout: 24 bites kimenet; a CODEC-től fogadott párhuzamos adat. Értéke akkor érvényes, ha valamelyik aud_dout_vld jel 1 értékű.
- aud_din_vld: 2 bites bemenet; DAC bemeneti adat (aud_din0, illetve aud_din1) érvényes.
- aud_din_ack: 2 bites kimenet; azt jelzi, hogy az I2S interfész a megfelelő (0. vagy 1. csatorna) adatot beolvasta.
- aud_din0: 24 bites bemenet; a DAC 0. csatorna párhuzamos adata.
- aud_din1: 24 bites bemenet; a DAC 1. csatorna párhuzamos adata.

Hullámformák:

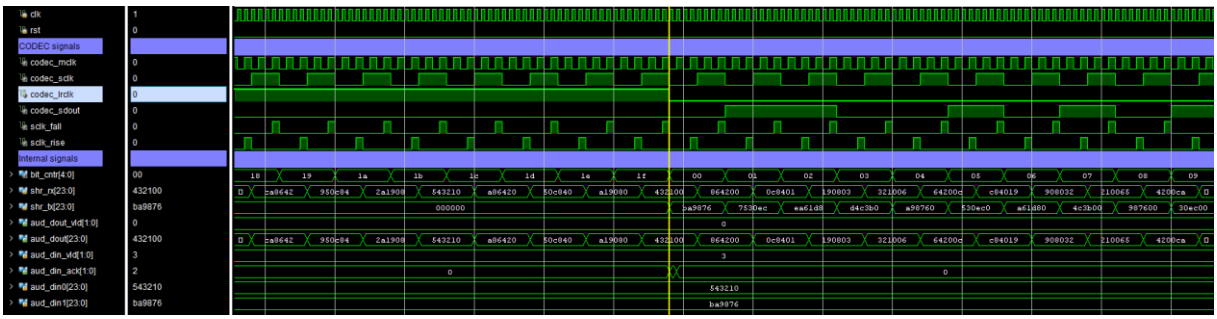
Egy teljes LRC periódus:



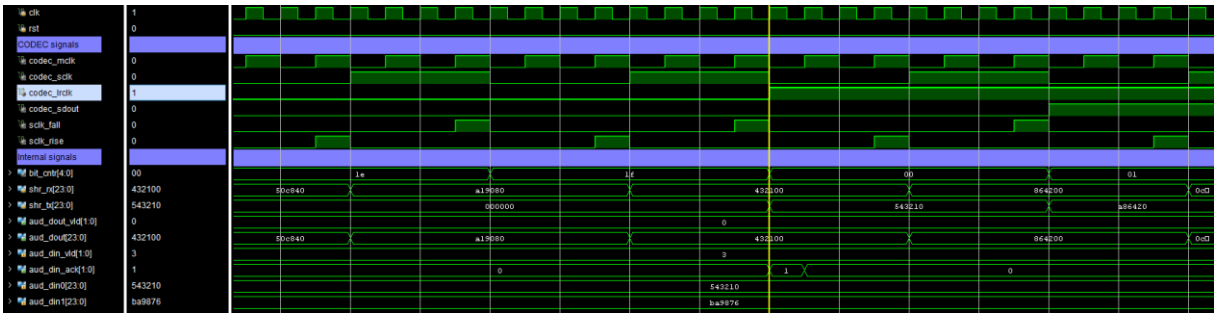
LRC lefutó éle:



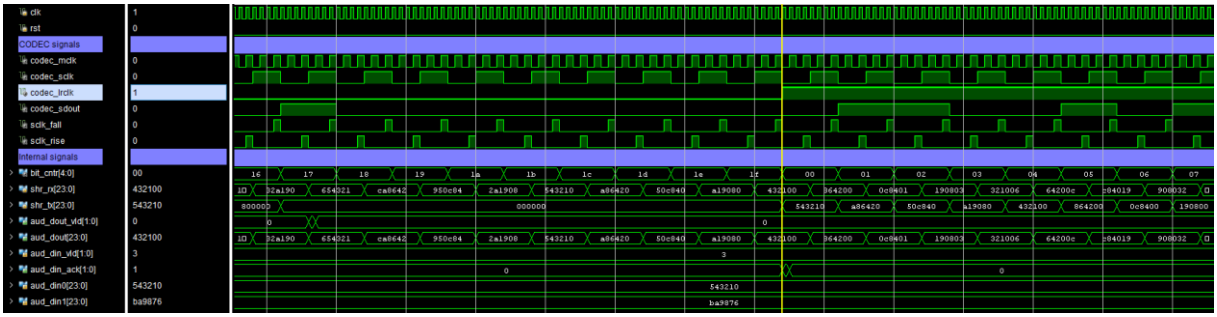
Ugyanez kissé messzebbről nézve:



LRC felfutó éle:



És messzebbről:

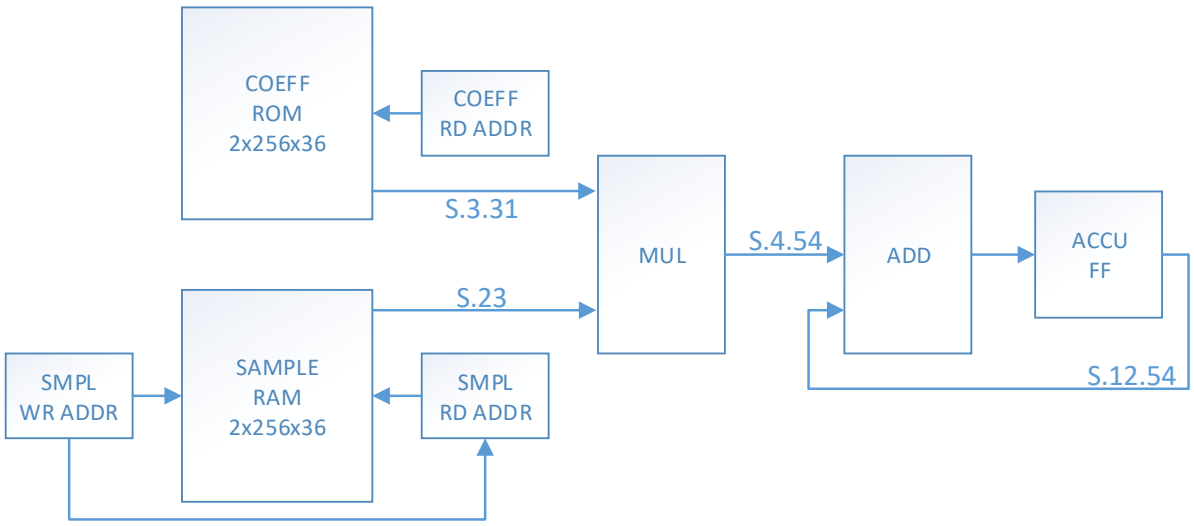


4. FIR szűrő

A 4. gyakorlaton a 3. gyakorlaton megvalósított, loopback módban módban működtetett CODEC interfészt egészítjük ki egy FIR szűrővel, azaz az ADC által digitalizált adatokat szűrjük, majd a DAC felé továbbítjuk.

A FIR szűrés egy N pontos konvolúció: $y_k = \sum_{i=0}^{N-1} x_{k-i} * c_{N-i-1}$, ahol y a kimeneti minta, x a bemeneti minták sorozata, c pedig az együtthatókat tartalmazó tömb. Azaz szemléletesen: az utolsó N darab mintát páronként szorozzuk egy N elemű együttható tömb elemeivel, majd a részszorzatokat összegezzük. A k-adik kimeneti minta előállításához a [k-N+1) …. k] indexű mintákat használjuk, míg a (k+1)-ik kimenethez a [k-N+2 …. k+1] indexűeket, azaz a legrégebbi mintát eldobjuk, az új mintát pedig behelyezzük a mintákat tároló tömbbe. Ez láthatóan egy N elemű shift regiszter tömb, aminek minden eleme 1-1 minta. Erőforrás takarékosság szempontjából sok esetben hatékonyabb a mintatárat memóriában megvalósítani – ennek optimális megoldása az N elemű cirkuláris buffer, amelyet folyamatosan (inkrementálisan) címezve írunk. Amennyiben a cím eléri az (N-1)-t, következő értéke 0 lesz. Ha N kettő hatvány, akkor ez FPGA realizációnál automatikusan megoldódik megfelelő szélességű címszámlálót használva. Adott időpillanatban, amikor az írási cím A, akkor ezen a címen a legújabb adat van, az A-1 címen az egyel régebbi, és így tovább; az A+1 címen a legrégebbi adat található. Ha a legújabb mintától kezdve a legrégebbig haladva szeretnénk összeszorozni a minta-együttható párokat, akkor az együttható tár címzése minden kimeneti minta előállításánál N-1 → 0 értékeket jár be, míg a mintatár címzését az aktuális minta címétől kell kezdeni és dekrementálni. Tehát [A, A-1, …. 0, N-1 … A+1] a címzés.

A megvalósítandó szűrő párhuzamossági fokát a jel mintavételi frekvenciája (f_s) és a működési frekvencia (f_{clk}) határozza meg. Egy csatorna feldolgozásakor két bemeneti minta között $\frac{f_{clk}}{f_s}$ órajel telik el, tehát órajelben számolva ennyi idő van a feladat elvégzésére. Az előző gyakorlathoz képest az FPGA működési frekvenciáját megnöveljük 200 MHz-re (f_s marad ~195 kHz), így a jelenlegi rendszerben: $\frac{f_{clk}}{f_s} = 1024$. Mivel két csatornát kell feldolgozni, így egy csatornára 512 órajel jut. A szűrőnk fokszáma 256, így ahhoz, hogy 512 órajel alatt kiszámítsunk 256 részszorzatot egyetlen szorzó hardver is bőven elegendő, azaz a feldolgozás szekvenciális. (Amennyiben pl. a mintavételi frekvencia megegyezne a működési frekvenciával, teljesen párhuzamos rendszerre lenne szükség, azaz csatornánként 256 szorzót használnánk). Egyszerűsített blokkvázlat a fentiek alapján:



Adatformátumok:

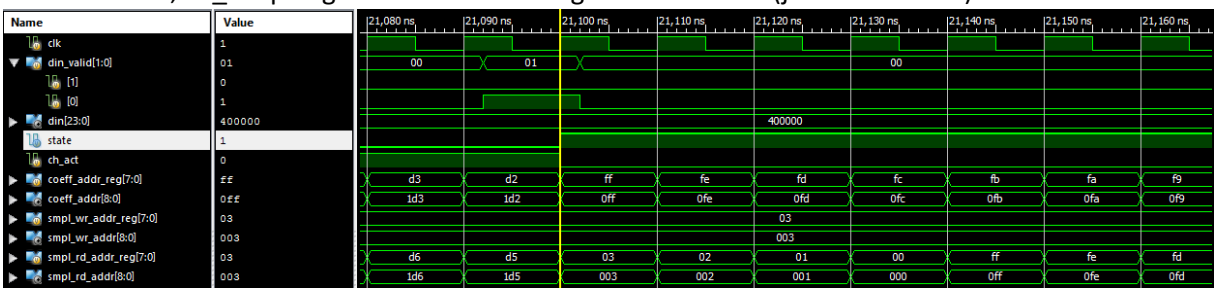
- A bemeneti minták 24 bites előjeles adatok, ezeket előjeles, csak törtrészt tartalmazó fixpontos számokként értelmezzük: azaz 23 bitnyi törtrész van, a formátum tehát s.23
- Az együtthatók (1-es DC erősítést feltételezve) jóval kisebbek, mint 1, így alapvetően ezeket is fixpontosként ábrázoljuk. Részben önkényesen, részben az FPGA tulajdonságait figyelembe véve 35 bites, s.3.31 formátumú értékeket használunk.
- A minta és az együttható szorzata: s.23*s.3.31→s.4.54
- Annak érdekében, hogy a 256 szorzat akkumulálásánál ne léphessen fel túlcsordulás az összeadónak $\log_2 256=8$ bittel szélesebbnek kell lennie, így formátuma s.12.54.
- A kimeneti minták a bemenethez hasonlóan s.23 formátumúak, ezt az akku formátumából a törtrészek tekintetében csonkolással, az egész rész tekintetében szaturációval állítjuk elő.

Egyéb megfontolások:

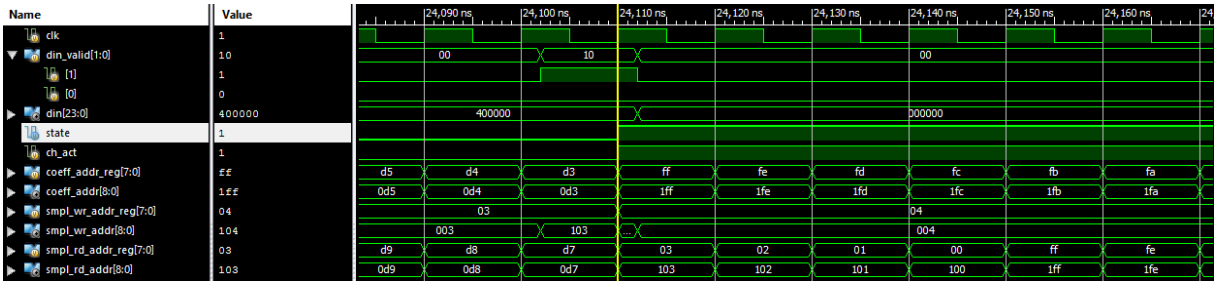
- Mind az együttható, mind pedig a mintatár két csatorna adatát tárolja. Az első 256 (0….255) cím tartozik a 0. csatornához, a második 256 (256….511) pedig az 1. csatornához.
- A minták írását az ADC interfésztől kapott adc_valid jel bitjeinek vagy kapcsolata engedélyezi.
- Az írási címszámláló növelését mintavételi periódusonként egyszer kell elvégezni (a két csatorna adott bemeneti mintáját a saját memória területen belül ugyanarra a címre kell írni), azaz ezt adc_valid[1] engedélyezi. A csatornához tartozó 256 elem címzéséhez 8 bites címszámlálóra van szükség, a teljes 512 elemű memória címzéséhez szükséges plusz egy MSB bitet adc_valid[1], szolgáltatja (azaz a 0. csatorna „alulra”, az 1. csatorna „felülre” íródik).
- Az új minta beírásakor az aktuális írási cím átmásolódik az olvasási címszámálóba, majd ezután 256 ütemeig ez dekrementálódik. Ugyanekkor az együttható olvasási címszámlálója 255-re inicializálódik, majd lefele számol.
- A memóriák olvasási címe a minta beírást követő 256 órajelben érvényes, így egy „cím érvényes” jel generálható úgy, hogy a mintatár írásakor 1-be állítunk egy FF-t, majd ha az együttható címszámláló elérte a 0-t, akkor 0-ba állítjuk.
- A minta írás megkezdésekor el kell tárolni, hogy melyik csatorna adatát dolgozzuk fel, ez a bit lesz az olvasási címek MSB bitje.
- A memóriaolvasásnak 1 órajel késleltetése van, valamint az alkalmazott 35x35 bites szorzó is rendelkezik viszonylag nagy késleltetéssel (adott bemenethez tartozó kimenet ennyi órajel múlva jelenik meg), ez utóbbi a HDL kód alapján meghatározható.
- Az akkumulátort akkor kell engedélyezni, amikor a szorzó kimenete érvényes – ehhez a „cím érvényes” jel megfelelő órajellel késleltetett verziója megfelelő (→shift regiszter).
- Az akkumulátort minden egyes konvolúció megkezdése előtt reset-elni kell. Erre minden olyan időpont megfelelő, ami megelőzi az első érvényes részszorzat megjelenését, de később van, mint az előző konvolúció befejezése. Ilyen pl. a bemeneti memória írásának engedélyezése. Még jobb – nem jár órajel veszteséggel – az a megoldás, hogy az első érvényes akkumulátor bemenet órajelében „resetel-jük” az akkumulátort; de nem 0-ba állítjuk, hanem akkumulálás nélkül beleírjuk a bemeneti értéket.
- Az akkumulátor az engedélyező jelének 0-ba váltásakor érvényes adatot tartalmaz, így ezen jel lefutó élének detektálásával generálható a kimeneti valid jel (ez is csatornánként 1 bit). Amennyiben a szaturáció plusz egy pipeline szintet jelent, úgy ezt a jelet is késleltetni kell még egy órajellel.

Hullámformák

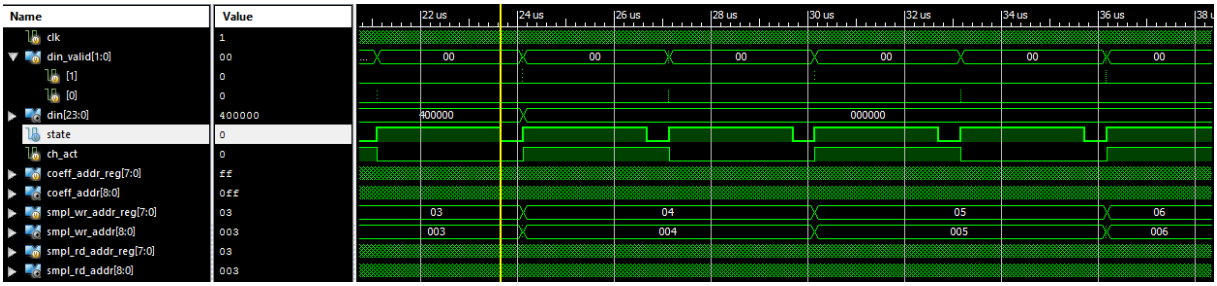
1. 0. csatorna feldolgozásának megkezdése → írási címszámláló (smpl_rd_addr_reg) nem nő; együttható címszámláló (coeff_addr_reg) 255-ről indul; minta olvasási címszámláló (smpl_rd_addr_reg) az írási címről – 0x3 – indul. state=1 jelenti, hogy érvényesek az olvasási címek, ch_act pedig az aktuálisan feldolgozott csatornát (jelen esetben 0).



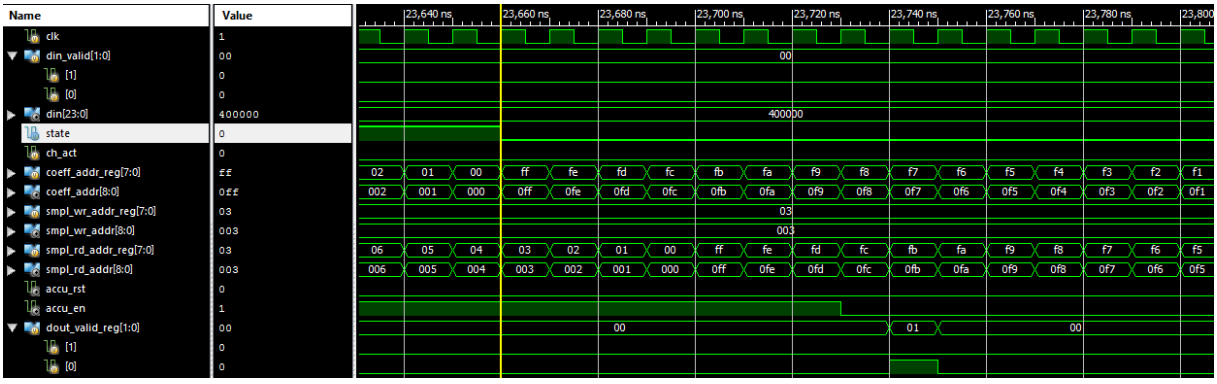
2. 1. csatorna feldolgozásának megkezdése → nő az írási címszámláló



3. Működési szekvencia „távolról” nézve



4. Konvolúció vége: kimenet érvényes (dout_valid) generálása.



5. ChipScope – FIR szűrő

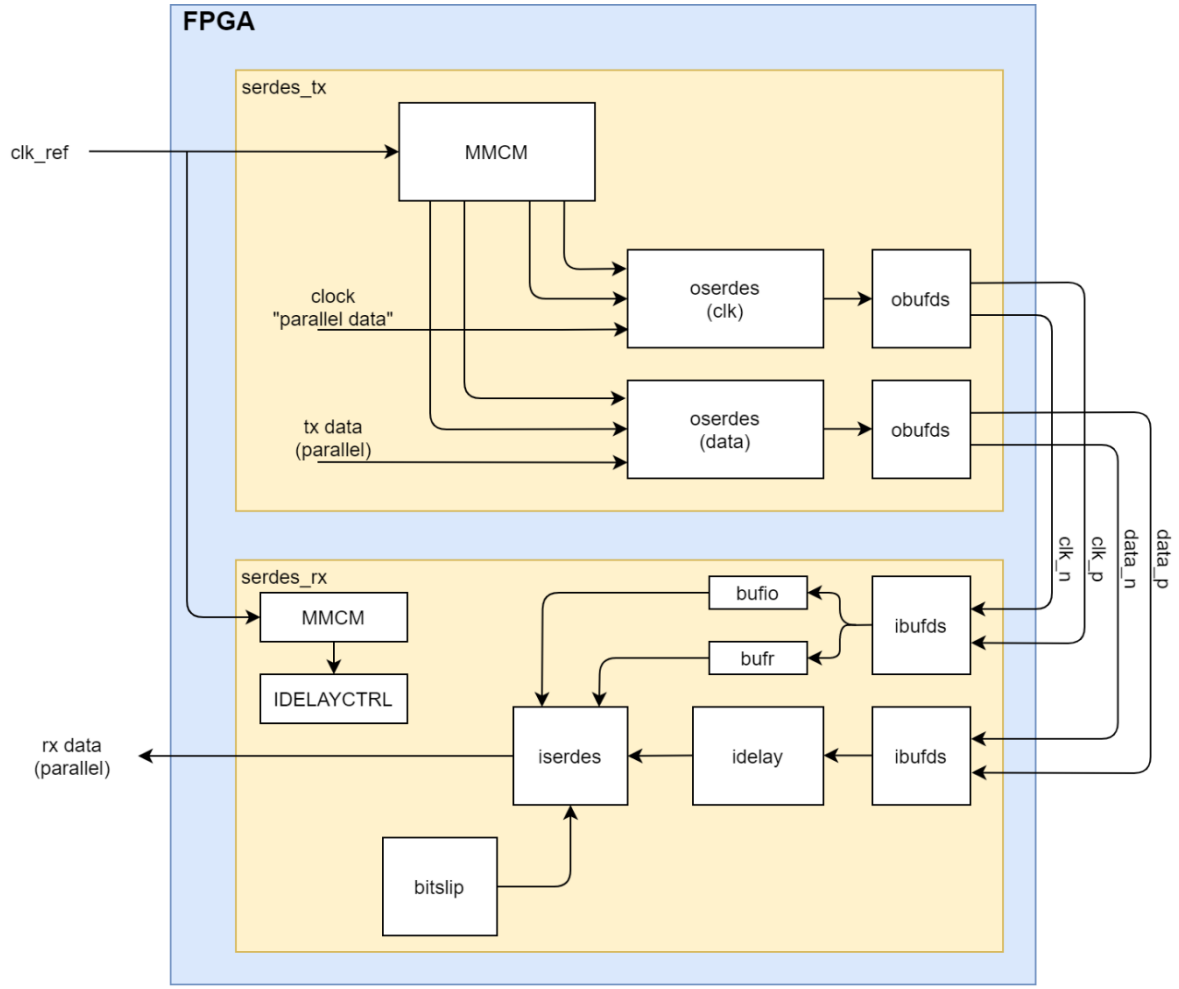
Lépések:

- Ha szimulációban megfelelően működik a FIR szűrő, akkor az analizátorban vizsgálni kívánt jelekhez adja hozzá a mark_debug szintézis attributumot. Ezen jelek:
    - accu
    - ch_act
    - coeff_addr_reg
    - coeff_rom_dout
    - din
    - din_valid
    - dout_reg
    - dout_valid_reg
    - smpl_ram_dout
    - smpl_addr_reg
    - state
    - Pl. (* mark_debug = "true" *) reg [1:0] ch_act;
- Szintetizálja a tervet.
- Nyissa meg a szintetizált tervet.
    - Klikkeljen a Setup Debug opcióra.
    - Ellenőrizze, hogy a „Nets to Debug” ablakban megjelenik az összes jel, amit mark_debug-gal megjelölt.
    - Sample Data Depth legyen 1024.
    - Engedélyezze a Capture control opciót.
- Kösse össze a CODEC kártya audió bementét a PC line out kimenetével, generáljon Audacity-ben egy szinusz jelet, és játssza végtelenítve.
- Vizsgálja meg:
    - Egy szűrési ciklus végrehajtását.
    - A FIR szűrő által kapott adatbemenet analóg hullámformáját.
    - A FIR szűrő kimenetének analóg hullámformáját.

6. SERDES

A gyakorlat során egy nagy sebességű forrás szinkron adó-, illetve vevő kialakításával ismerkedünk meg, amelyet post-implementation szimulációval vizsgálunk.

A vizsgált rendszer blokkvázlata az alábbi:



Felkészülés:

- Nézze át a gyakorlaton használt projekt kódját, vesse össze a fenti blokkvázlattal és az előadáson elhangzottakkal!

A gyakorlat nagyrészt vezetett, az alábbi kérdésekkel:

- A constraint fájl tartalma alapján határozza meg, hogy mennyi az FPGA bementi órajelének frekvenciája.
- A transmitter „source synchrounous center aligned” módot valósít meg.
  - Az oserdes-ek DDR üzemmódban működnek, 8:1 sorosítási faktorral.
  - A kimenő órajel éle az adat bitidő közepére kell essen, azaz az órajel fázisa az adathoz képest 90°-kal el van tolva.
- Határozza meg, hogy 800 Mbit/s kimeneti adatsebesség eléréséhez az MMCM modulnak milyen frekvenciájú és fázishelyzetű órajeleket kell generálnia, és ennek megfelelően hogy kell beállítani az MMCM modul paramétereit.
- Gondolja át, hogy mit kell az OSERDES párhuzamos adatbemenetére kötni a megfelelő órajel hullámforma előállításához.

- Végezzen viselkedési szimulációt és ellenőrizze, hogy a kimeneti hullámformák (clk_p, clk_n, data_p, data_n) megfelelnek-e az elvártnak.
- Implementálja a rendszert.
- Sikeres implementáció után „post-implementation timing simulation” használatával ellenőrizze, hogy a működés megfelel-e az elvártaknak:
    - Vizsgálja meg a transmitter által generált jelek hullámformáit.
    - Adja hozzá a szimuláció hullámforma ablakához a receiver IBUFDS blokkjainak kimeneteit, a BUFIO és BUFR kimeneteit, valamint az adat IDELAY kimentét. Ezután vizsgálja meg a jelek időzítési viszonyait.
- Ellenőrizze, hogy az időzítés analízis által mutatott késleltetések megfelelnek-e a szimulációban látottaknak.
- Mekkora az ISERDES órajel- és az adat bemenete közötti késleltetés különbség? Hogyan kompenzálható ez?