

FPGA alapú rendszerek fejlesztése

Gyakorlat útmutató

2023.

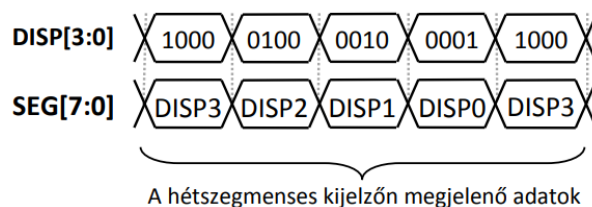
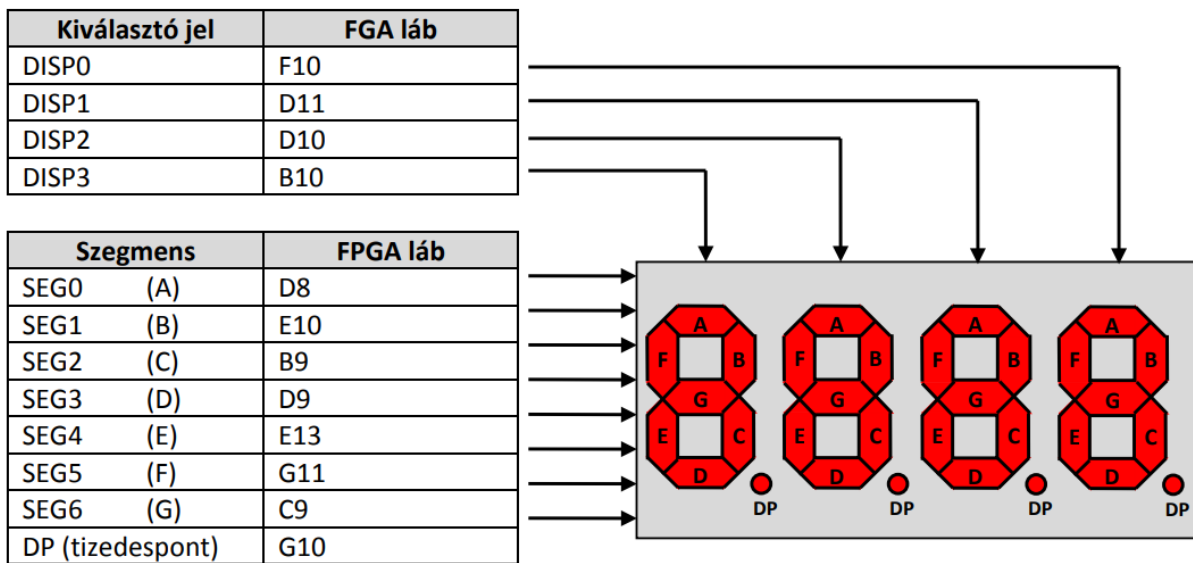
Tartalom

1.	Vivado bevezető – 7-szegmenses vezérlő	2
2.	Audió CODEC illesztése.....	4
3.	Audio FIR szűrő.....	11
4.	RGB → Y átalakítás + logikai analízátor.....	14

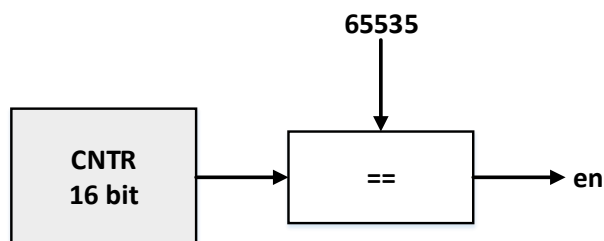
1. Vivado bevezető – 7-segmenses vezérlő

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

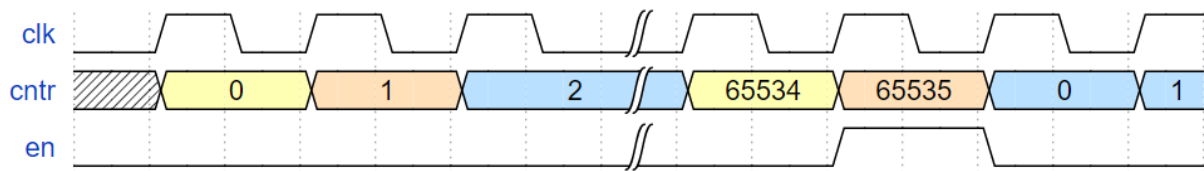
A Logsys Kintex-7 kártyán egy 4-digites, időmultiplexált 7-segmenses kijelzőt találunk. Ennek vezérléséhez az FPGA és a kijelző között 4 digit engedélyező jelet (DSIP0...DSIP3), valamint egy 8 bites szegmens buszt találunk (SEG0...SEG7, DP). Az időmultiplexálás azt jelenti, hogy a szegmens jelek minden 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-segmenses vezérlő LED-jei megfelelően működjenek. Megfelelő választás a kHz nagyságrendű váltás.



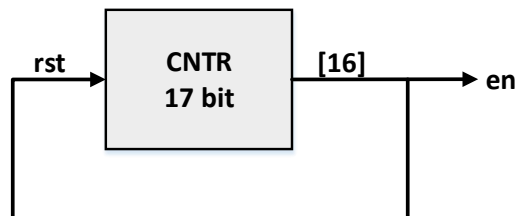
Az FPGA órajel forrása 100 MHz frekvenciájú, ahhoz hogy a megfelelő frekvenciájú jelváltásokat generálni tudjuk, egy ~kHz frekvenciájú engedélyező jelre van szükség, ami az órajelnek 100.000-ed része. Mivel nem kritérium pontosan 1 kHz-es frekvencia előállítása, így az egyszerűség kedvéért a legközelebbi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



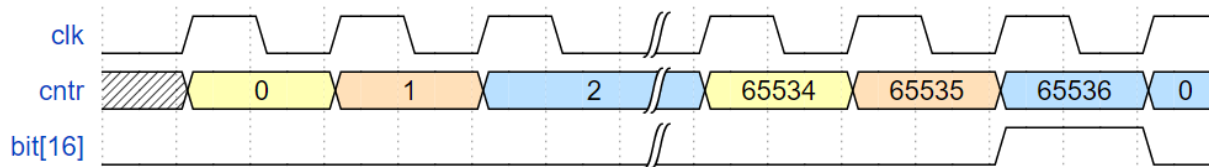
A generált hullámforma:



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



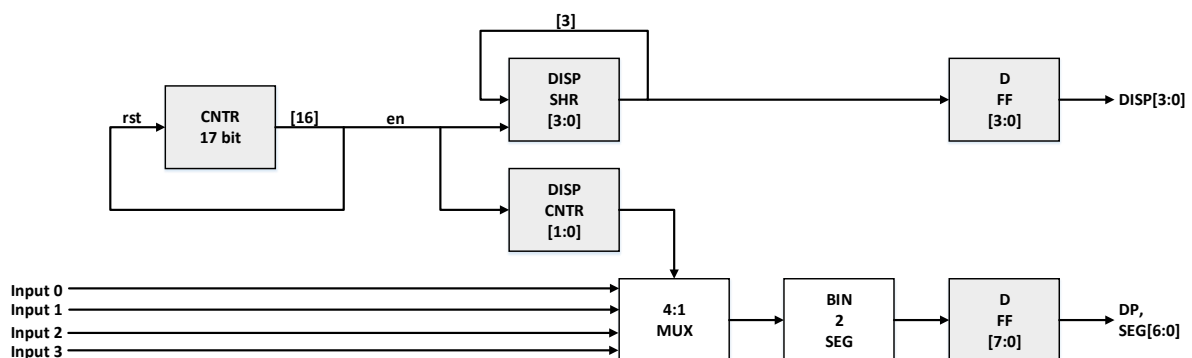
Az így létrejövő hullámforma:



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. Audió CODEC illesztése

A gyakorlat során egy sztereó audió CODEC (coder-decoder) illesztünk az FPGA-hoz. A megtervezett modul célja, hogy a CODEC-kel történő „alacsony szintű” kommunikációt elrejtse az azt használó tervező mérnök elől, számára egy egyszerű, könnyen használható interfészt biztosítson, amin keresztül a CODEC-ből érkező, illetve a CODEC felé továbbított audió adatok egyszerűen kezelhetők.

Jelen gyakorlat során a CODEC-ből az ADC-vel mintavételezett és kvantált adatokat az FPGA-ban egy regiszteren keresztül visszacsatoljuk és a CODEC DAC-jának bemenetére továbbítjuk (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 konfiguráció során a következő paramétereket kell beállítanunk.

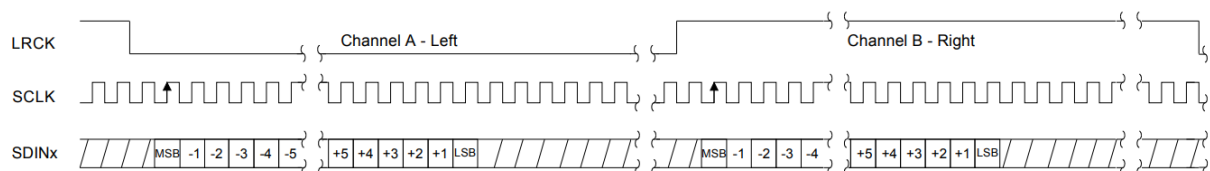
- Slave vagy master mód.
 - Slave módban minden órajel (MCLK, SCLK, LRCK) bemenet a CODEC számára.
 - Master módban az MCLK bemenet, a többi órajelet a CODEC szolgáltatja.
 - Az órajelek elvárt frekvencia-viszonyait az alábbi táblázat mutatja.

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

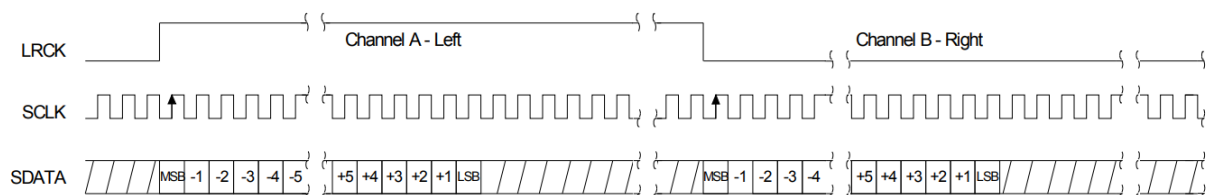
- Single-speed, double-speed vagy quad-speed üzemmód. A beállítandó üzemmódot egyértelműen meghatározza a mintavételi frekvencia.

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

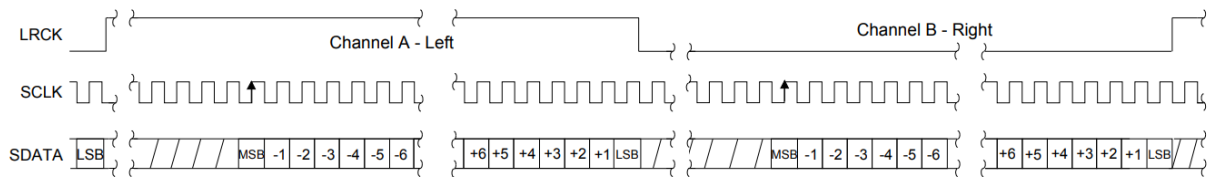
- Audió interfész üzemmódja: I2S, Right-Justified vagy Left-Justified.
 - I2S üzemmód



- Left-Justified üzemmód



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



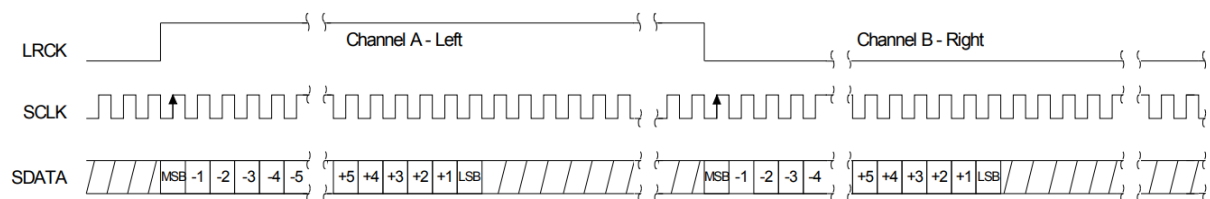
A gyakorlat során az alábbi beállításokkal fogjuk használni a CODEC-t:

- 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.

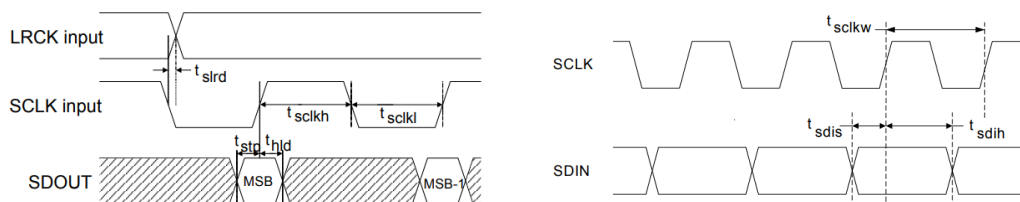
Ehhez az alábbi konfigurációs láb beállítások szükségesek:

- 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-Justified: 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 Double-Speed Mode Quad-Speed Mode	Fs Fs Fs	4 50 100	- - -	54 108 216 kHz kHz kHz
MCLK Specifications					
MCLK Frequency (Note 15)	Stand-Alone Mode Serial Control Port Mode	fmclk fmclk	1.024 1.024	- -	55.296 55.296 MHz MHz
MCLK Duty Cycle			40	50	60 ns
Slave Mode					
LRCK Duty Cycle			40	50	60 %
SCLK Period (Note 15)	Single-Speed Mode Double-Speed Mode Quad-Speed Mode	t _{sclkw} t _{sclkw} t _{sclkw}	$\frac{1}{(128)Fs}$ $\frac{1}{(64)Fs}$ $\frac{1}{(64)Fs}$	- - -	- - - s s 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 _{sdlh}	20	-	- ns

Megfontolásaink:

- A modul használó tervező mérnök felé egyszerű interfész:
 - CODEC ADC → FPGA irány:
 - A CODEC által küldött párhuzamos adat
 - Mindkét csatornára 1-1 adat érvényes jel, amely 1 rendszer órajel hosszúságú impulzussal jelzi az adat érvényességét.
 - FPGA → CODEC DAC
 - A CODEC felé küldendő párhuzamos adat, illetve az ennek érvényességét jelző bit
 - „Acknowledge” jel, amely jelzi, hogy a modul a felhasználó által szolgáltatott adatot felhasználta.
- 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).

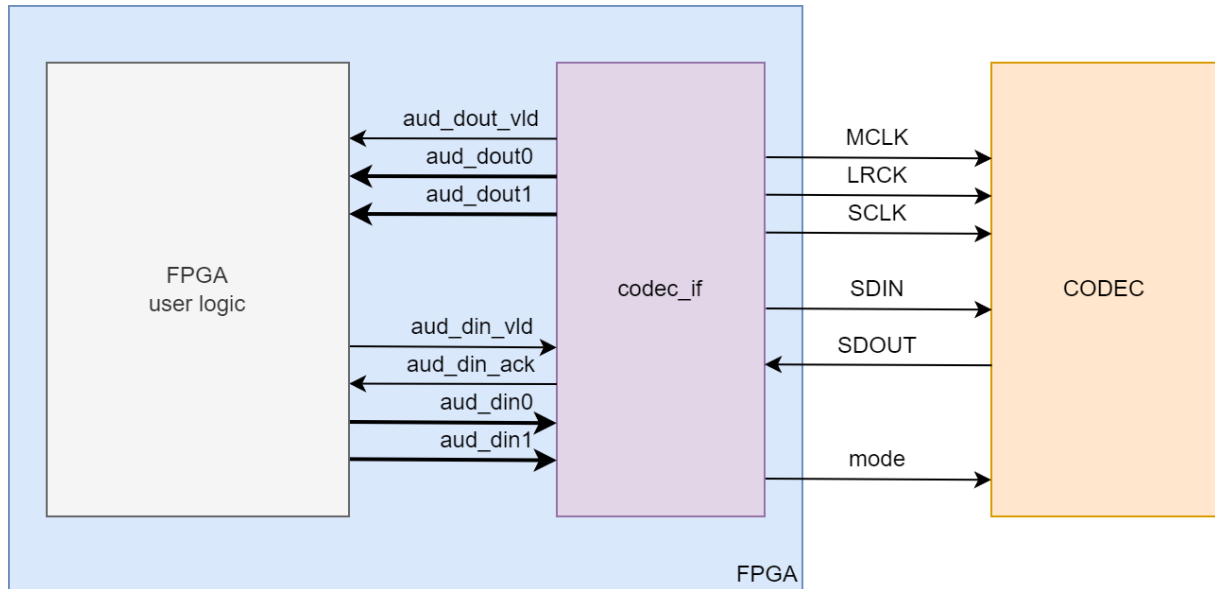
- A DAC az SCLK felfutó élére mintavételezi a bemeneti soros adatot, ezt legalább ezen él előtt 16 ns-mal ki kell adni (setup time), illetve 20 ns-ig még ott kell tartani (hold time). Az SCLK periódusidejének fele ennél jóval nagyobb, így SCLK lefutó éle megfelelő időpont az adatkiadásra.
- A CODEC adatkimenete 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 biteinek kivezetésével generálhatók. Konkrétan:
 - $LRCK = CLK / 512 \rightarrow \text{bit}[8]$
 - $SCLK = CLK / 8 \rightarrow \text{bit}[2]$
 - $MCLK = CLK / 2 \rightarrow \text{bit}[0]$
- A bemeneti soros \rightarrow párhuzamos, illetve a kimeneti párhuzamos \rightarrow 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 $\rightarrow \text{bit}[7:3]$.
 - 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_dout0: 24 bites kimenet; a CODEC-től fogadott párhuzamos adat. Értéke akkor érvényes, ha aud_dout_vld[0] jel 1 értékű.

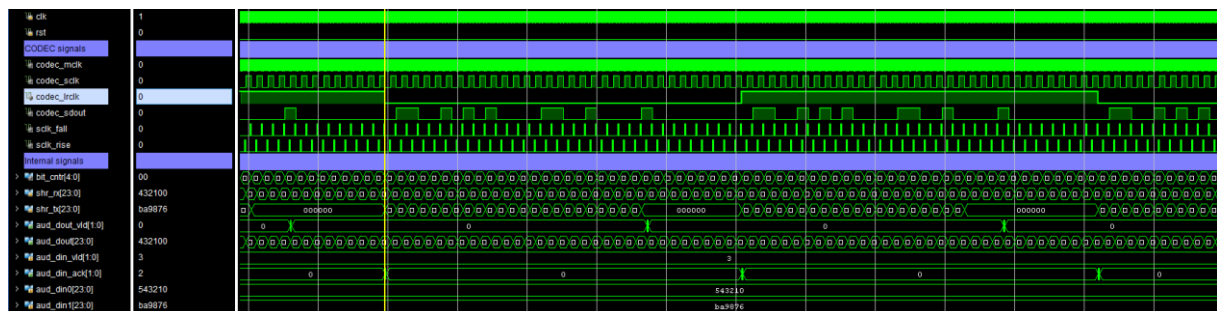
- aud_dout1: 24 bites kimenet; a CODEC-től fogadott párhuzamos adat. Értéke akkor érvényes, ha aud_dout_vld[1] jel értékű.
- aud_din_vld: 2 bites bemenet; DAC bemeneti adat (aud_din0, illetve aud_din1) érvényes.
- aud_din_ack: 2 bites kimenet; azt jelzi, hogy az I2S interfész a megfelelő (0. vagy 1. 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.

A CODEC – FPGA összeköttetés és az FPGA-ban megvalósított interfész (codec_if) blokkvázlata:

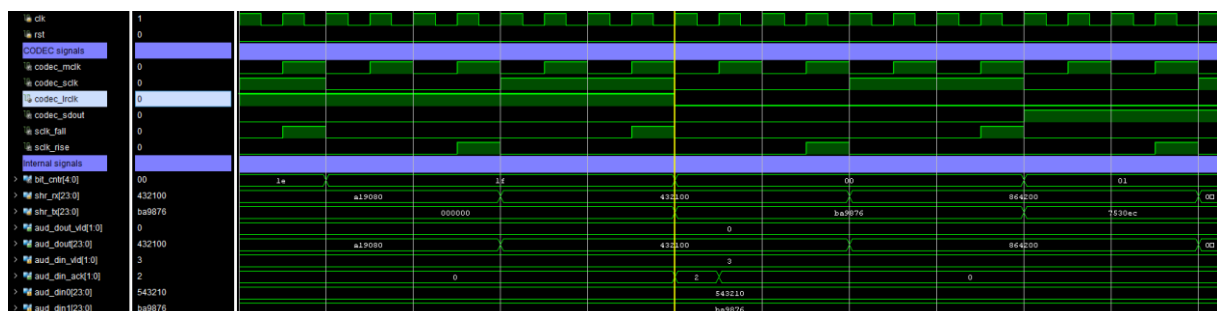


Hullámformák:

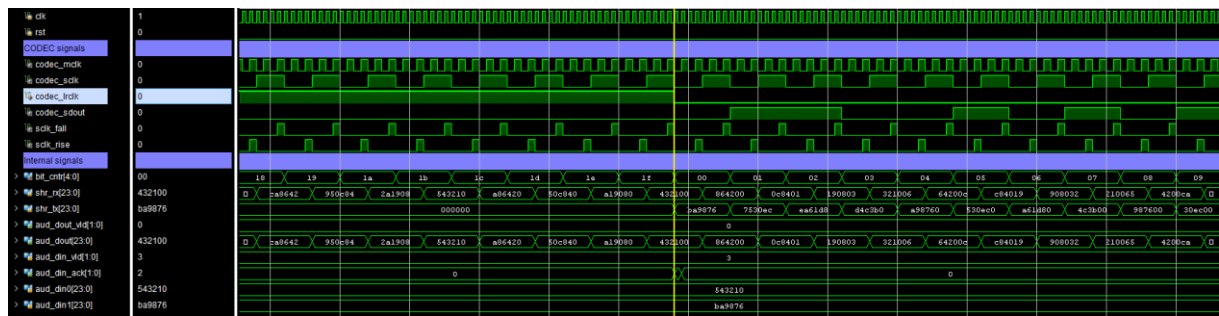
Egy teljes LRC periódus:



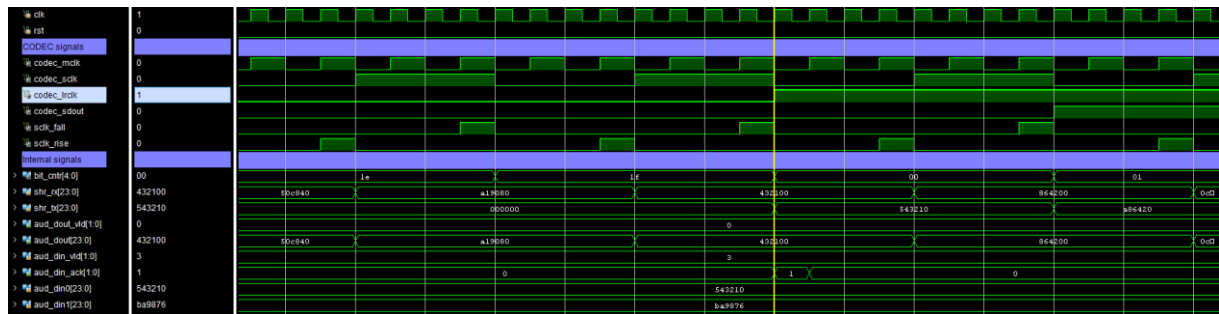
LRC lefutó éle:



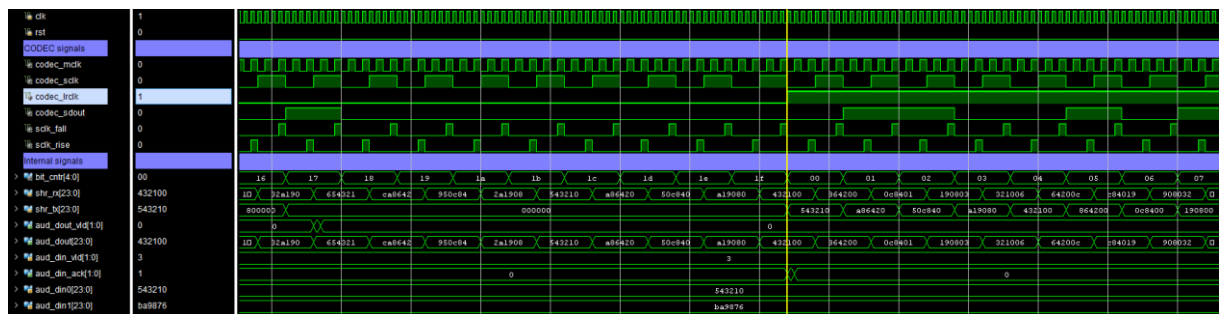
Ugyanez kissé messzebről nézve:



LRC felfutó éle:



És messzebről:

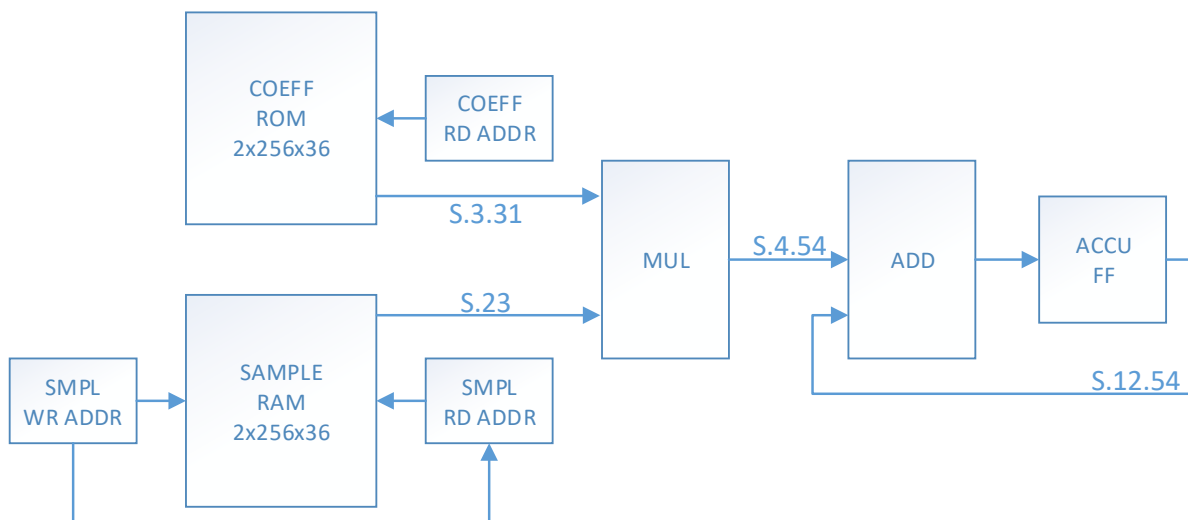


3. Audio FIR szűrő

A 4. gyakorlaton a 3. gyakorlaton megvalósított, loopback módban működött 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 takarékoság 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 címezése minden kimeneti minta előállításánál $N-1 \rightarrow 0$ értékeket jár be, míg a mintatár címezését az aktuális minta címétől kell kezdeni és dekrementálni. Tehát [A, A-1, 0, N-1 ... A+1] a címezés.

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:

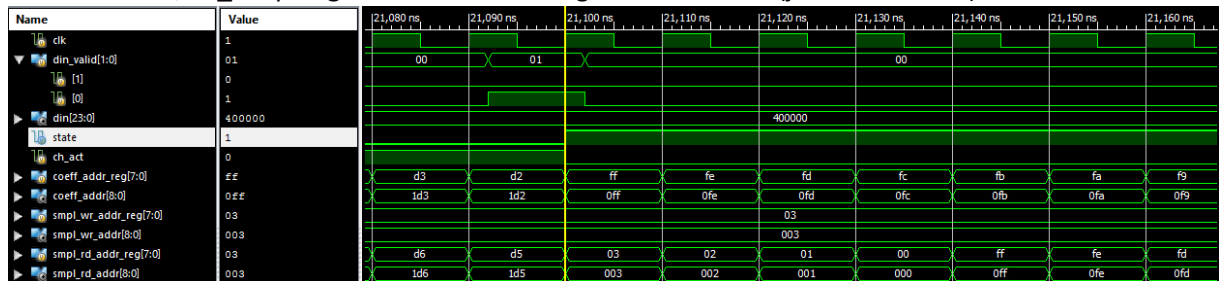
- A bemeneti minták 24 bites előjeles adatok, ezeket előjeles, csak törtrészt tartalmazó fixpontos számokként értelmezzük: azaz 23 bitnyi törtrész van, a formátum tehát s.23
- Az együtthatók (1-es DC erősítést feltételezve) jóval kisebbek, mint 1, így alapvetően ezeket is fixpontosként ábrázoljuk. Részben önkényesen, részben az FPGA tulajdonságait figyelembe véve 35 bites, s.3.31 formátumú értékeket használunk.
- A minta és az együttható szorzata: $s.23 * s.3.31 \rightarrow s.4.54$
- Annak érdekében, hogy a 256 szorzat akkumulálásánál ne léphessen fel túlcsondulás 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:

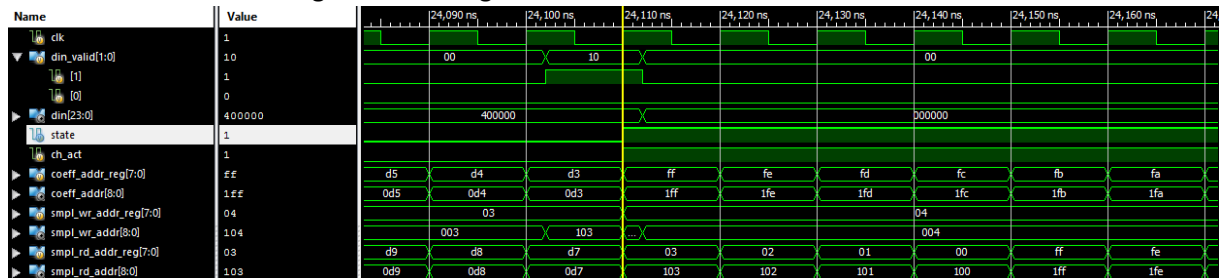
- Mind az együttható, mind pedig a mintatár két csatorna adatát tárolja. Az első 256 (0....255) cím tartozik a 0. csatornához, a második 256 (256....511) pedig az 1. csatornához.
- A minták írását az ADC interfésztől kapott `adc_valid` jel bitjeinek vagy kapcsolata engedélyezi.
- Az írási címszámláló 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 címszámlálóba, 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 (\rightarrow shift regiszter).
- Az akkumulátort minden egyes konvolúció megkezdése előtt reset-elni kell. Erre minden olyan időpont megfelelő, ami megelőzi az első érvényes részsorzat 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

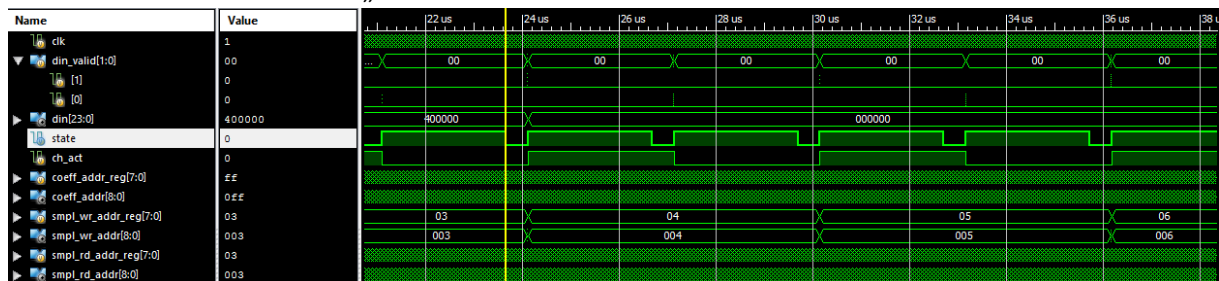
0. csatorna feldolgozásának megkezdése → írási címszámláló (smpl_rd_addr_reg) nem nő; együtttható címszámláló (coeff_addr_reg) 255-ről indul; minta olvasási címszámláló (smpl_rd_addr_reg) az írási címről – 0x3 – indul. state=1 jelenti, hogy érvényesek az olvasási címek, ch_act pedig az aktuálisan feldolgozott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



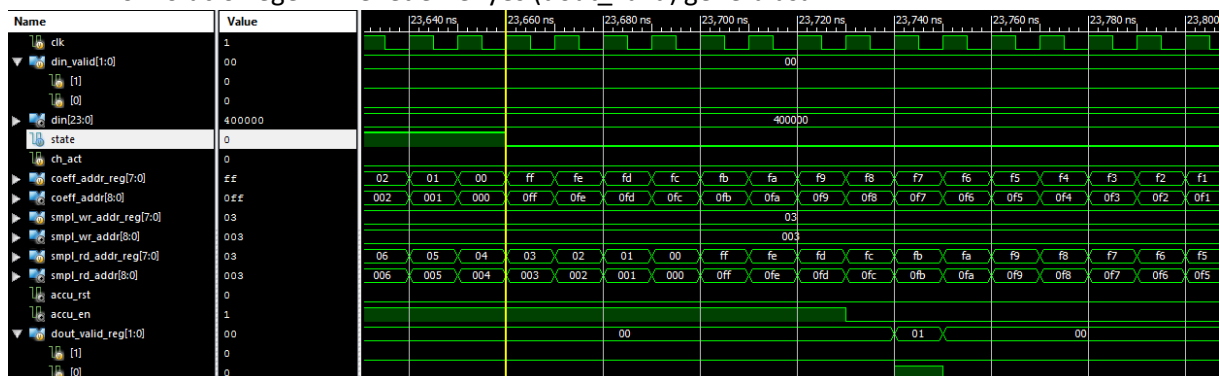
1. csatorna feldolgozásának megkezdése → nő az írási címszámláló



- Működési szekvencia „távolról” nézve



- Konvolúció vége: kimenet érvényes (dout_valid) generálása.

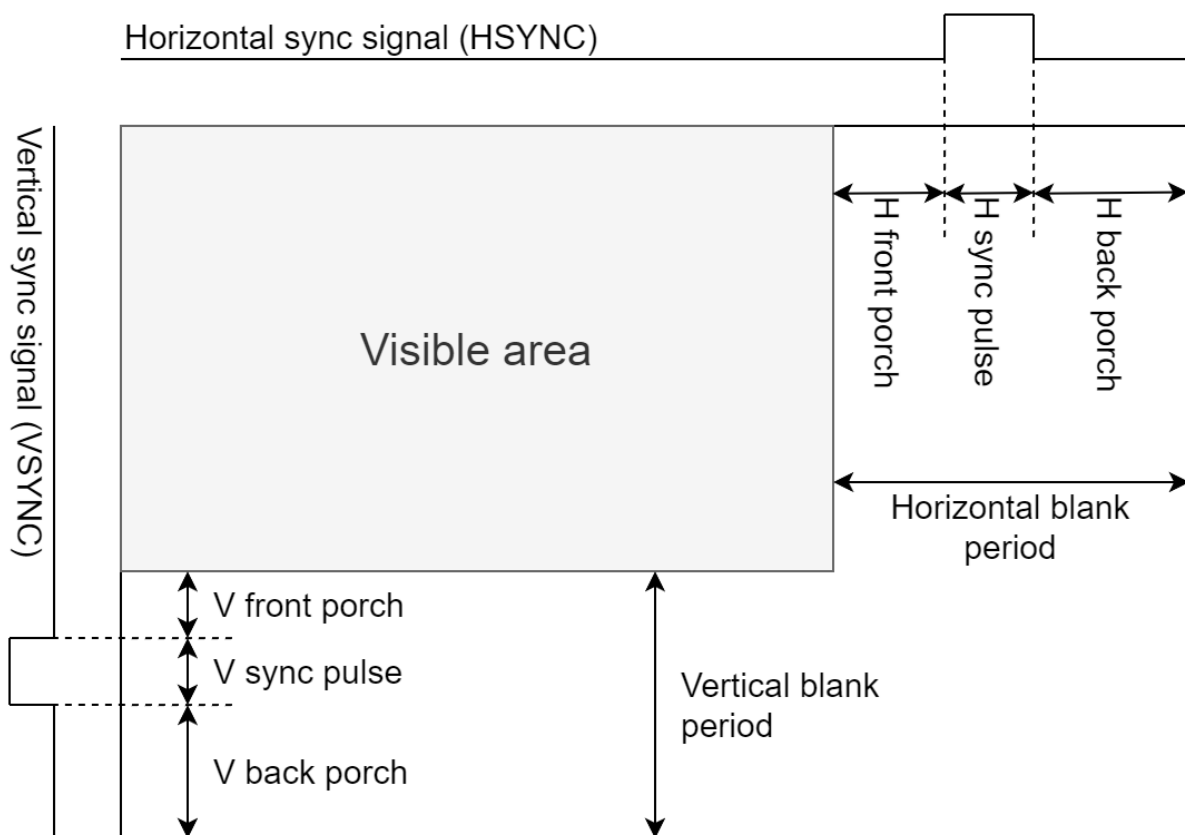


4. RGB → Y átalakítás + logikai analízátor

A gyakorlat során egy (részleges) szintér konvertert valósítunk meg, amely a HDMI bemeneten érkező RGB adatokból világosság (Y) komponenst számol, majd a HDMI kimenetre szürkeárnyaltos képet továbbít, azaz a világosság komponens kerül az R, G és B komponensek helyére. (Színterekről röviden: https://en.wikipedia.org/wiki/Color_space)

Videó formátum

A HDMI vevő minden órajelben egy pixel értékét, valamint a 3 vezérlőjelet szolgáltat, melyek megegyeznek a VGA interfész jeleivel. A teljes továbbított kép mind horizontális, mind pedig vertikális irányban látható tartományból és kioltási (blank) intervallumokból áll. A horizontális kioltási idő alatt (azaz minden egyes sorban) található a horizontális szinkron pulzus (HSYNC), míg a vertikális kéпкиoltási idő alatt (tehát képenként egyszer) a vertikális szinkron pulzus. A pulzusok polaritása felbontástól függően lehet ponált vagy negált, az alábbi ábra ponált esetet mutat.



A HDMI vevő által szolgáltatott jelek:

- rx_red, rx_green, rx_blue: a 3 színkomponens 8-8 biten
- rx_hsync: horizontális szinkronjel
- rx_vsync: vertikális szinkronjel
- rx_dv: a látható pixelek alatt 1, a blank periódusok alatt 0

RGB → Y átalakítás

Az RGB → YCbCr/YUV átalakításra számos szabvány létezik, amelyek más-más együtthatót használnak, a világosság komponens számításának általános képlete:

$$Y = K_R * R + (1 - K_R - K_B) * G + K_B * B$$

A gyakorlat során a HD televíziózásban használt, az ITU-R BT.709-6 szabványban rögzített együtthatókat fogjuk használni (https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-E.pdf):

$$Y = 0,2126 * R - 0,7152 * G - 0,0722 * B$$

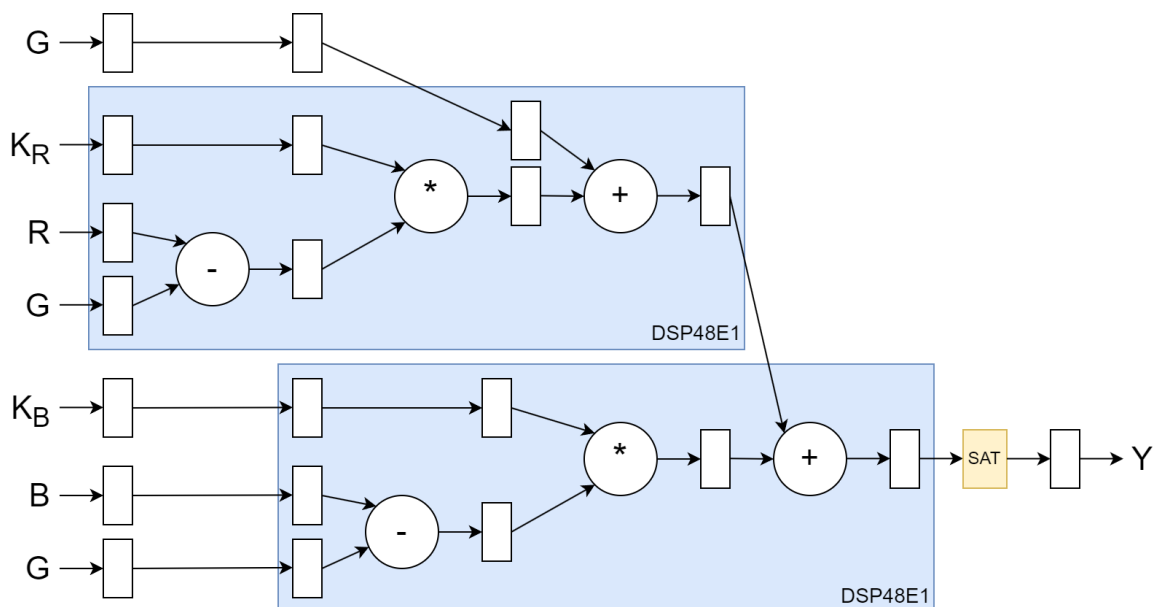
A 3 szorzás és összeadás triviálisan megvalósítható 3 DSP blokkal:

- A 25 bites bemeneteket használva az együtthatókra, ezek fix pontos számként igen pontosan ábrázolhatók (ez a pontosság már felesleges, hiszen a kimenetünk 8 bites).
- A 18 bites bemenetekre a 8 bites R, G, B értékek kerülnek.

Megspórolhatunk egy DSP blokkot, ha átrendezzük az általános képletet és felhasználjuk a DSP48E1 elő-összeadóját:

$$Y = K_R * (R - G) + K_B * (B - G) + G$$

Ebben az esetben az RGB értékek kerülnek a 25 bites bemenetekre, az együtthatók pedig a 18 bites bemenetekre. Az így adódó blokkvázlat:

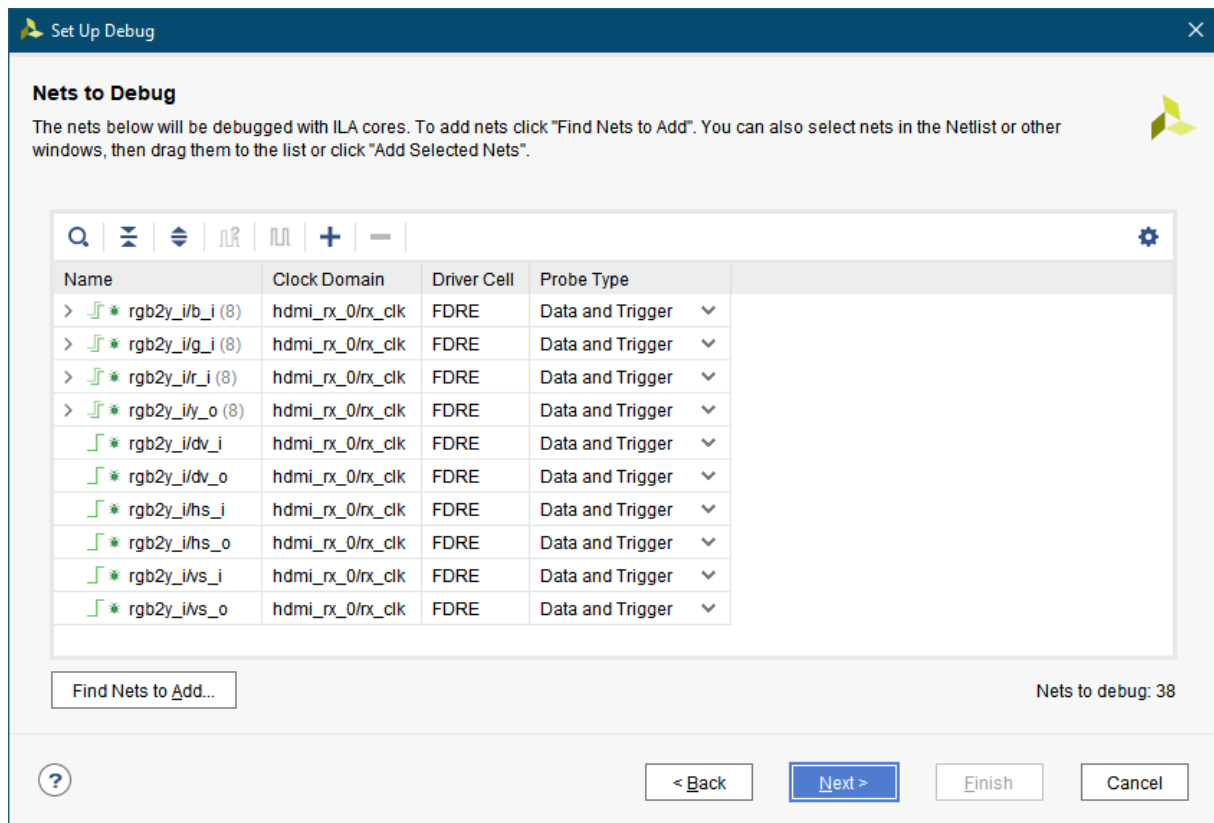


Mivel az együtthatók 1-nél kisebb számok, a 18 bites előjeles bemenet esetén 17 törtész bitet használunk. Az együtthatók összege 1, az R/G/B komponensek pedig előjel nélküli 8 bites értékek, így túlsordulás nem fordulhat elő. Ugyanakkor az eredmény lehet negatív, amit szaturációval kezelni kell.

Annak érdekében, hogy a kimeneti pixel értékek (Y) és a vezérlőjelek (valid, hsync, vsync) szinkronban maradjanak, a vezérlőjeleket ugyanannyival kell késleltetni, mint amennyi az adatút késleltetése – ez jelen esetben – a szaturációt is beleszámítva – 6 órajel.

Logikai analízátor (ChipScope)

Az FPGA-ban megvalósított logikai analízátor az FPGA belső BRAM-jait használja mintatárként, a megfigyelt jelek számát és a mintatár nagyságát az elérhető BRAM-ok száma limitálja. HDL kód vizsgálatának legegyszerűbb módja, hogy a vizsgálni kívánt jelek deklarációját kiegészítjük a MARK_DEBUG szintézis direktívával. Ezen jeleket az analízátor konfigurációjakor a Vivado automatikusan hozzáadja a megfigyelt jelekhez. Minden jelre megadhatjuk, hogy azt csak trigger jelként szeretnénk használni; csak a hullámformáját szeretnénk megnézni; vagy mindkét opcióra igényt tartunk (Trigger/Data/Data and Trigger).



A konfiguráció során ezen kívül megadhatjuk:

- A mintatár mélysége. A memória igény a megfigyelt jelek (bitek) száma szorozva a mintatár mélységével.
- Capture control. Lehetővé teszi, hogy a bemeneti jelek alapján megfogalmazott feltétellel engedélyezzük a mintavételt. Amennyiben nem használjuk, úgy az analizátor minden órajelben mintát vesz a megfigyelt jelekből).
- Advanced trigger. Bonyolultabb trigger feltételek megfogalmazását teszi lehetővé kb HDL szintaxissal. Használata nélkül is lehetőségünk van több trigger esemény logikai kapcsolatát vizsgálni, de például trigger szekvenciák beállításához már szükség van az opcióra.

A működési idejű analízishez az analizátor felületén a következő beállítási lehetőségeink vannak:

- Trigger mode. Egyszerű vagy összetett trigger mód.
- Capture Mode Settings
 - o Always: Minden órajelben mintát vesz.
 - o Basic: Csak a Capture Setup-ban megadott feltétel teljesülésekor vesz mintát.
- Number of windows. A teljes mintatárat hány darab független részre szeretnénk osztani. Az egyes ablakok egymás utáni trigger események környezetét mutatják.
- Window data depth. Egy minta-ablak mérete. A teljes mintatár mérete = ablakok száma * ablak méret.
- Trigger position. A trigger esemény helye a mintatárban.
- Trigger Setup. A trigger feltétel beállítása. Több feltétel megadása esetén az egyes feltételek egyszerű logikai kapcsolatba hozhatók.
- Capture Setup. A mintavételt engedélyező feltétel megadása.

hw_ila_1

Waveform - hw_ila_1

ILA Status: Idle

Name	Value
> rgb2y_i/g_i[7:0]	14
> rgb2y_i/y_o[7:0]	14
> rgb2y_i/r_i[7:0]	14
> rgb2y_i/b_i[7:0]	14
rgb2y_i/dv_i	1
rgb2y_i/dv_o	1
rgb2y_i/bc_i	0

Updated at: 2023-Apr-12 16:50:53

Settings - hw_ila_1

Trigger Mode settings

Trigger mode: BASIC_ONLY

Capture Mode Settings

Capture mode: ALWAYS

Number of windows: 1 [1 - 4096]

Window data depth: 4096 [1 - 4096]

Trigger position in window: 2,048 [0 - 4095]

Trigger Setup - hw_ila_1

Name	Operator	Radix	Value	Port
rgb2y_i/dv_i	==	[B]	R	probe4

Capture Setup - hw_ila_1