Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Elektronikus Eszközök Tanszék

Szakdolgozat

DIGITÁLIS HULLÁMFORMA-GENERÁTOR MEGVALÓSÍTÁSA ÉS MÉRÉSE FPGA KÖRNYEZETBEN

Készítette: Mózer Viktor

Konzulens: Dr. Horváth Péter

BUDAPEST, 2023

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Mózer Viktor*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2023. június 1.

Mózer Viktor
hallgató

Kivonat

Az elektronikában gyakran használnak perdiodikus jeleket különböző mérésekre, kommunikációra, jelfeldolgozásra stb. Ezeket a periodikus jeleket különféle analóg vagy digitális jelgenerátorokkal állítják elő. Ebben a dolgozatban egy digitális elven működő hullámforma generátor tervezését és megvalósítását mutatom be FPGA (Field Programmable Gate Arrays) környezetben. Ezen eszköz segítségével lehetővé válik különböző típusú -és frekvenciájú jelek előállítása. A tervezett rendszer a Direct Digital Synthesizer (DDS) módszeren alapul, amelynek lényege, hogy egy numerikus vezérlő segítségével állíthatjuk elő a kívánt jeleket. A DDS módszerrel könnyen és rugalmasan generálhatók a különböző típusú jelek, például szinusz, négyszög vagy háromszög hullámformák, valamint azok frekvenciái is könnyen, gyorsan és pontosan állíthatók. A tervezett rendszert számítógépen keresztül lehet vezérelni és beállítani a megjeleníteni kívánt hullámforma adatait. A dolgozatban bemutatom a tervezési folyamatot, a rendszer architektúráját, a hardver- és szoftver megvalósítását, valamint a rendszer működésének szimulációját is. A tervezett digitális hullámforma generátor nagy pontossággal és stabilan képes generálni a különböző típusú és frekvenciájú jeleket, így széles körű alkalmazási lehetőséget kínál a digitális jelszintézis területén.

Abstract

Periodic signals are often used in electronics for various measurements, communications, signal processing, and so on. These periodic signals are generated using different analog or digital function generators. This paper presents the design and implementation of a digital waveform generator based on FPGA (Field Programmable Gate Arrays) technology. This device enables the production of various types and frequencies of signals. The designed system is based on the Direct Digital Synthesizer (DDS) method, which uses a numerical controller to generate the desired signals. The DDS method allows for easy and flexible generation of various types of signals, such as sine, square or triangular waveforms, and their frequencies can also be freely adjusted. The designed system can be controlled and configured through a computer to set the parameters of the waveform to be displayed. In the paper, I will present the design process, system architecture, hardware, and software implementation, as well as the simulation of the system's operation. The planned digital waveform generator can generate signals of various types and frequencies with high accuracy and stability, thus offering a wide range of applications in the field of digital signal synthesis.

1. Bevezetés

1.1. Jelgenerátorok

A hullámforma generátorok olyan elektronikai eszközök, amelyek periodikus jelek előállítására szolgálnak. Ezek az eszközök fontos szerepet játszanak a jelgenerálásban és a tesztelésben számos iparágban, beleértve az elektronikát, a kommunikációt, a hang- és videótechnikát, az optikát, az orvosi képalkotást, a tudományos kutatást és még sok más területet. A függvénygenerátorok segítségével lehetőség van a különböző hullámformák jellemzőinek finomhangolására is, például a frekvencia, amplitúdó és a fáziskésleltetés. Az ilyen generátorok általában két fő részből állnak, egy oszcillátor– és egy adó áramkörből. Az oszcillátor áramkör felelős a stabil és pontos alapjel előállításáért, míg az adó áramkör a szükséges jelalak előállítását végzi a kívánt frekvencia, amplitúdó és fázis beállítása mellett. A jelgenerátorok megvalósíthatóak analóg- és digitális áramkörrel egyaránt, valamint ezen módszereknek együttes használatával is. Utóbbihoz tartozik például a PLL (Phase Locked Loop), vagyis a fáziszárt hurok elvén működő generátor.

A digitális megvalósítások közül az egyik széles körben használt megoldás az úgynevezett Direct Digital Synthesis (röviden DDS) elven működő generátor, melynek működése, tervezése és tesztelése jelen dolgozat témája is. A digitális hullámformák előállítása számos alkalmazási területen fontos szerepet játszik. Analóg szűrők és erősítők tesztelésekor szükséges számos fajta bemenő jelet előállítani, mint például szinusz, négyszög, háromszög hullámformákat. Emellett a digitális jelgenerátorok használhatók műsorszóráshoz, diagnosztikai alkalmazásokhoz, hangfrekvenciás vizsgálatokhoz és egyéb mérési feladatokhoz is. A DDS generátor előállítására többféle megoldás létezik, például használhatunk erre a célra tervezett integrált áramköröket, mikrovezérlőket vagy programozható logikai áramköröket. Ezek közül egy nagyon elterjedt típus az FPGA (Field Programmable Gate Array), mely digitális logikai áramkörök tervezésére, megvalósítására és tesztelésére is alkalmas. Az FPGA rugalmassága és nagy sebessége miatt a digitális jelgenerátorok tervezése és megvalósítása kiválóan megoldható vele.

1.2. Történelmi áttekintés[1]

Joseph A. Webb 1970-ben nyújtotta be a digitális jelgenerátor szintetizátorának szabadalmát, amely a DDS (Direct Digital Synthesis) mechanizmusán alapul. A szabadalomban leírta a különböző típusú analóg hullámformák előállítását, beleértve a szinusz hullámokat is, a digitális logikai modulok segítségével. Ezután, 1971 elején, Tierney és munkatársai publikáltak egy gyakran idézett tanulmányt a közvetlen digitális frekvenciagenerálásról, amely kibővítette a DDS működését a kvadratúra generálására, valamint bemutatta a mintavételezett rendszerek elméletével kapcsolatos korlátokat. Gyakorlati megvalósítások jelentek meg, főként diszkrét szabványos logikai IC-k (integrated circuit) felhasználásával, mint például a TTL74xx vagy az ECL 10K családok. Körülbelül 10 évvel később teljesen integrált megoldások jelentek meg a piacon, amelyeket olyan vállalatok fejlesztettek ki, mint a Stanford Telecom, a Qualcomm, a Plessey és az Analog Devices az AD9950 és az AD9955 bevezetésével. A legjobb sebesség, teljesítmény és költségarány elérése érdekében a logikai IC-k architektúrái egy keresőtáblára (LUT) épültek, amely korlátozott fázis-, frekvencia- és amplitúdófelbontással biztosította a fázis-szinusz amplitúdó átalakítását. Ma az Analog Devices a legnagyobb és talán legkülönlegesebb DDS önálló integrált áramköröket szállító vállalat, míg a jelenlegi numerikusan vezérelt oszcillátorok (NCO-k) általában nagy számban vannak integrálva RF DAC-okba, mint például az AD9164 vagy az AD9174. Bár ezek az eszközök lenyűgöző zaj- és linearitás-teljesítménnyel rendelkeznek több GHz-es sávszélességen, nem alkalmasak mérsékelt sebességű, nagy felbontású ADC-k (Analog-to-Digital Converter), mint például az LTC2378-20, az AD4020 vagy az AD7768 tesztelésére. A digitális hullámforma generátorok története folyamatosan bővül új fejlesztésekkel és technológiákkal. Az új technológiák előretörése és a fejlődő alkalmazások pedig újabb lehetőségeket nyújtanak a digitális hullámforma generátorok számára. Fontos megemlíteni, hogy ezek a generátorok jelentős szerepet játszanak a kutatásban, az oktatásban, az iparban és a mindennapi életünkben is. A további fejlesztések és kutatások pedig elősegíthetik az alkalmazási területek még szélesebb körű kiterjesztését.

1.3. A feladat célja

A feladatom célja egy DDS (Direct Digital Synthesis) elven működő digitális hullámforma generátor tervezése volt, amely kellő rugalmasságot biztosít a megfelelő jelek előállításához. Az eszköz 20 Hz és 20 kHz közötti frekvenciatartományban (hangfrekvenciás tartomány) képes a megfelelő jelek előállítására, amelyek alkalmazási területek széles spektrumában használhatóak, mint például a hangfeldolgozás, akusztikai mérések vagy akár az orvosi diagnosztika. A tervezést FPGA technológiával valósítom meg. A DDS elvű jelgenerátor az egyik legelterjedtebb módszer a hullámformák digitális előállítására. Az elv alapja egy digitális hullámforma előállítása, melyet egy Digitális-Analóg konverterrel alakítunk át analóg jellé. A jelgenerálási folyamat során a frekvenciát és a fázist digitálisan állítjuk be, ami nagy pontosságot és stabilitást biztosít a generált jelben. Az előnyei közé tartozik az egyszerűsége, a nagy pontossága, a nagy dinamikatartománya és a rugalmassága. A DDS technológia számos alkalmazásban hasznos, különösen olyan területeken, ahol stabil frekvencia és gyors frekvenciaváltás szükséges. A szakdolgozatom célja a felhasználó által megadott periodikus jelek előállítása, a tervezett digitális hullámforma generátor megfelelő működésének biztosítása, amely megfelel az előre meghatározott követelményeknek. Az elkészült rendszert tesztelem, ezáltal biztosítom a megfelelő működést a helyes használat mellett. A szakdolgozat meghatározza a rendszer megfelelő használatát, részletesen bemutatom a tervezett generátor működését és felépítését, a tervezési folyamatot, az FPGA technológiát és az áramkörök tervezési lépéseit. Emellett részletesen beszámolok a tesztek/mérések elvégzésének módszereiről és a kapott eredményekről.

A szakdolgozat első részében megvizsgáltam, hogy hogyan épül fel, hogyan működik és hogy milyen feladatokra, milyen körülmények között használható egy DDS jelgenerátor. Továbbá megismerkedtem azzal az eszköztárral, amelyek segítségével garantálni tudom a megfelelő működést, gondolok itt az FPGA-ra tervezés technológiájáról, a szimulációra stb. A következő fejezetben meghatároztam a rendszer működésének feltételeit és hogy a generált hullámforma milyen jellemzőkkel rendelkezzen, például frekvenciatartomány, felbontás, dinamikatartomány stb. Ezután a rendszertervezés fázisa következett, ahol a mind a hardvert, mind a szoftvert megterveztem és megvalósítottam. A hardvert HDL (Hardware Description Language), vagyis hardverleíró nyelv segítségével modelleztem és

optimalizáltam FPGA technológiára. A szoftveres részt pedig magas szintű programozási nyelven készítettem el (Python). A két egységet a különálló tesztek lefuttatása és kiértékelése után összekapcsoltam.

2. Irodalomkutatás [2]

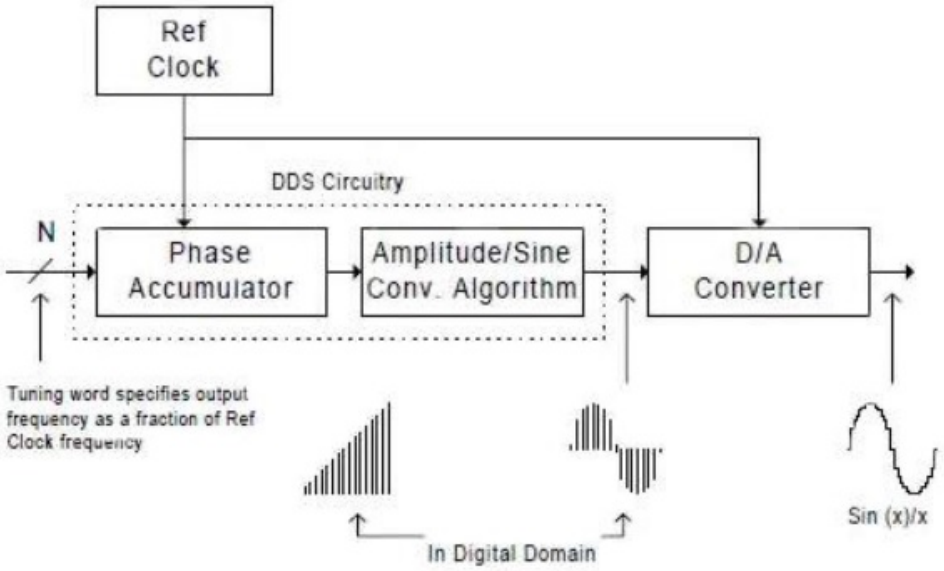
2.1. Közvetlen Digitális Szintézis fogalma

A Direct Digital Synthesis (röviden DDS), magyarul Közvetlen Digitális Szintézis egy módszer, amely digitális adatokat és vegyes/analóg jelfeldolgozási blokkokat használ a valóságban ismétlődő hullámformák generálására. Előállítható vele például szinuszjel vagy akár négyszög- és háromszögjel is. Teszi mindezt úgy, hogy először egy időben változó digitális jelet állít elő, majd ezt egy digitális-analóg átalakítóval analóg jellé alakítja. Ezen technika ma már széles körben használt és a technológia fejlődésével alacsony fogyasztású, nagy megbízhatóságú és gyors működésű jelgenerátorokat képesek előállítani.

2.2. Közvetlen Digitális Szintézis működési elve

Egy ilyen elven működő hullámforma generátor jellemzően három fő részből áll. Egy fázisakkumulátor vagy fázis-összeadó áramkörből, egy fázis-amplitúdókonverziós részből, mely jellemzően egy memória, ami egy 1 periódusnyi függvény mintavételezett amplitúdóértékeivel van fetöltve. Ezeket együttesen nevezzük numerikusan vezérelt oszcillátornak, angolul Numerically Controlled Oscillator-nak, röviden NCO-nak. Ez gyakorlatilag egy digitális periodikus jelet állít elő, innen tehát a neve. A rendszer továbbá áll egy digitális-analóg átalakítóból, ami az NCO által előállított digitális hullámformát alakítja át analóg jellé, ennek kimenetén sokszor egy aluláteresztő szűrő található, ami kiszűri a magas frekvenciájú felharmonikusokat, ezáltal javítva a jel minőségét. Továbbá az áramkör tartalmaz egy olyan modult, mely a belső referencia órajelet állítja elő, ami megadja hogy milyen időközönként veszünk mintát a tárolt adatokból. A fázisakkumulátor egy úgynevezett “adatszót” (tunning word) kap, mely megadja, hogy a fázisakkumulátor mekkora “lépésekkel” menjen végig a szinusz táblán. Gyakorlatilag ez határozza meg a kimeneti frekvenciát. Erről részletesen a következő alfejezetben írok.

Az áramkör még tartalmazhat úgynevezett “profil regisztereket”, melyek előre beprogramozott regiszterek, ezekben be lehet állítani a frekvenciát, fázist, interpolációt és még sok más értéket [3].

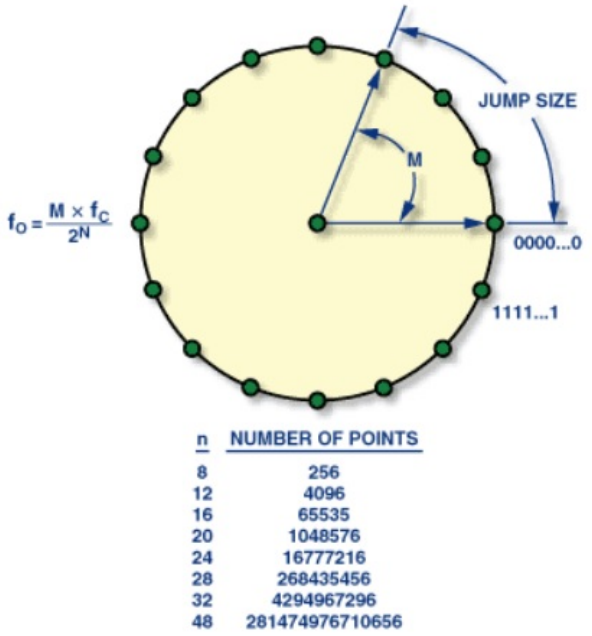


2.1. ábra. A DDS generátor működését szemléltető ábra [3]

2.2.1. Fázisakkumulálás

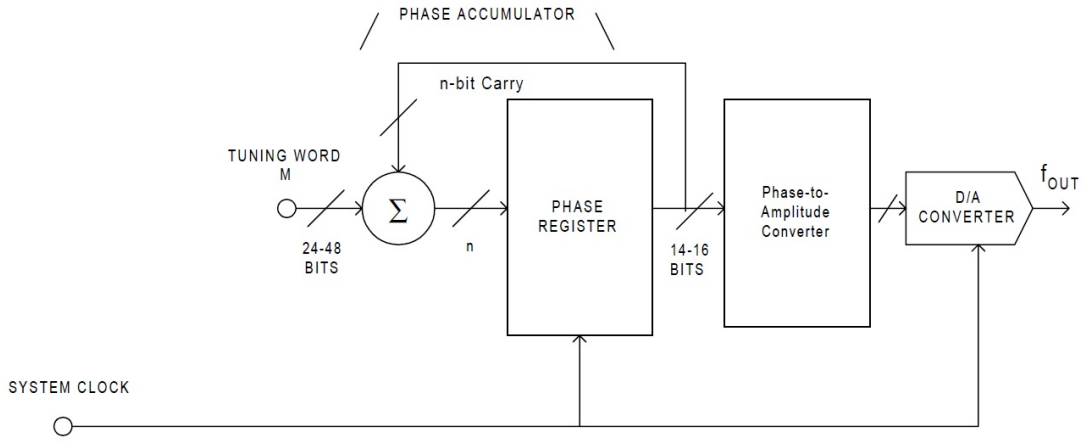
Adott egy folytonos idejű hullámforma, legyen ez most egy szinuszhulllám. Ezt a következőképpen fogalmazzuk meg matematikailag: $u(t) = \sin(\theta)$ Ennek fázisát (θ) egy 0-tól 2π-ig terjedő szöggel lehet leírni. A digitális jelet szintén le lehet írni így viszont ilyenkor csak diszkrét értékeket vehetnek fel ezek a szögek. Ezt a 2.2 ábrán látható digitális fáziskerékkel tudjuk reprezentálni.

Adott egy vektor melynek kezdőpontja a kör középpontja és végig a kör mentén mozog adott sebességgel, amint véget ért a körön, vagyis 0-tól 2π-ig, akkor indul előről. Ez tehát egy folyamatosan futó szinuszhullámot jelent.



2.2. ábra. Fáziskerék [2]

Ezen digitális szinusz adatait egy memóriába tároljuk el és ezen a memórián egy számlálóval tudunk végigmenni, mely ha túlcsordul újrakezdődik a számlálás. Ez a számláló a bemenetére kap egy már az előbbiekben említett adatszót (tunning word), ezt legtöbbször M betűvel jelöljük, ez a szó egy bináris számnak felel meg. Ezzel állítjuk a frekvenciát mégpedig úgy, hogy a számlálóhoz mindig ezt a számot adjuk hozzá, ez azt eredményezi, hogy minden egyes belső órajelre (mintavételi frekvencia) ennyit lépünk a memóriában. Így gyorsabban végig érünk az adatokon, vagyis a frekvenciánk nagyobb lesz. Azonban mivel kevesebb adatból veszünk mintát a felbontásunk csökken. Ezt a következő ábra szemlélteti, amin látszik ez az adatszó (itt M betűvel jelöli az ábra készítője) és maga a számláló, amit itt “phase register” névvel szerepel. Ez folyamatosan címzi a memóriát, itt “Phase-to-Amplitude Converter”.



2.3. ábra. Fázisakkumulátor [3]

A kimeneti frekvencia a következő összefüggés segítségével számítható:

$$f_{out} = \frac{M \times f_c}{2^n}$$

Ahol:

- f_{out} a kimeneti frekvencia
- M a bináris frekvencialépés
- f_c belső órajel
- n a fázisakkumulátor bitjeinek száma

Tegyük fel, hogy adott egy $n = 16$ bites fázisakkumulátorunk, vagyis $2^n = 2^{16} = 65\ 535$ értéket tárolunk az akkumulátorban. A minimális lépésköz '1' értéket vehet fel, ezáltal végigmegyünk minden egyes adaton. Ez a minimális frekvenciát és a maximális felbontást eredményezi. Ha M '32768' értéket veheti fel, akkor két órajel ciklus után túl is csordul a számlálónk. Azonban a mintavételi törvény kimondja, hogy egy mintavett jel akkor és csak akkor rekonstruálható a mintáiból, ha a jel sávszélessége (maximális frekvencia) nem nagyobb a mintavételezési frekvencia felénél.

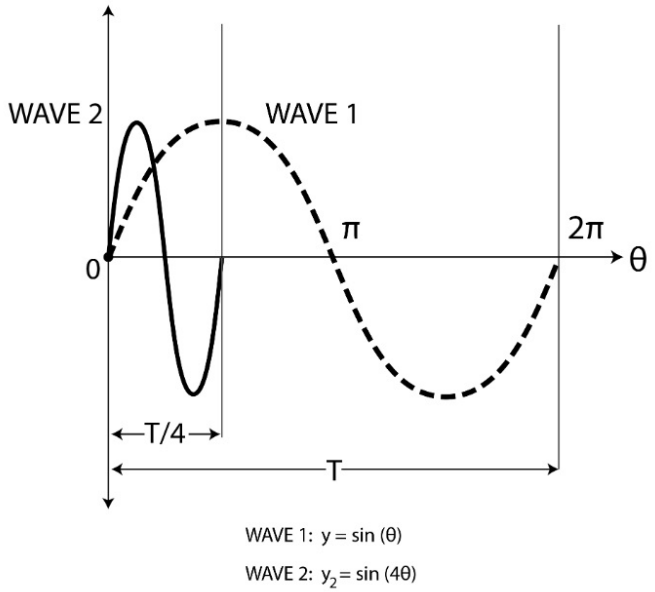
Képlettel felírva ez annyit tesz, hogy:

$$f \leq \frac{f_s}{2}$$

Jelen példánkra kiszámítható, hogy a maximálisan megengedhető lépésköz:

$$\frac{f_s}{2} = \frac{M \times f_s}{2^{16}} \Rightarrow M = 32\ 768$$

A következő ábrán jól szemléltethető a frekvencia változtatása. Analóg jelnél ezt úgy tesszük, hogy a szinusz függvény szögét minél nagyobbra választjuk. Analóg jelnél ezt úgy



2.4. ábra. Frekvenciaváltás [6]

tesszük, hogy a szinusz függvény szögét minél nagyobbra választjuk. Az ábrán látható, hogy ha 4-szer akkora a szög, akkor a periódus idő negyede lesz az eredeti periódusidőnek. Digitális jelnél ugyanez a jelenség figyelhető meg, négyszer gyorsabban végig tud lépkedni a fázisakkumulátor az adatokon, mivel a lépésköz a négyszerese az eredetinek, ezért minden negyedik adaton megy csak végig. Ez szintén azt eredményezi, hogy T idő alatt kiadódik négy egész periódus szinuszjel.

2.3. DDS generátorok főbb tulajdonságai

Az egyik ilyen fontos tulajdonság a **frekvenciahangoló szó** vagy más néven tunning word. Erről az előző fejezetben szó volt, ennek segítségével beállíthatjuk a kimeneti frekvenciát. Fontos tulajdonság továbbá a **felbontás**, amely a generált hullámforma frekvenciájának finomságát jelenti, vagyis a minimális frekvencia lépésközt, amit meg tudunk jeleníteni. A felbontás meghatározza, hogy milyen pontosan állítható be a generált hullámforma frekvenciája. A felbontást általában a fázisakkumulátor bithossza határozza meg, ami a digitális jelfeldolgozás során használt legkisebb egységeket jelenti. Például legyen 32 bites a memória mérete, tehát $n = 32$, ekkor $2^{32} = 4,2 \times 10^9$ biten tároljuk 1 periódusnyi jel adatait. A felbontást a következőképpen tudjuk számolni, ha 96kHz a mintavételi frekvenciánk:

$$f_{res} = \frac{f_s}{2^n} = \frac{96kHz}{2^{32}} = 2,235 \times 10^{-5} Hz = 22,35 \mu Hz$$

Vagyis μHz nagyságú felbontás is elérhető a DDS jelgenerátoral, nyilván itt figyelembe kell venni más értékeket is, például az órajel zaját, a DA átalakító mintavételi frekvenciáját, annak zaját, a kimeneti szűrőt stb. Abban az esetben, ha a rendszer órajele (SCLK) 50MHz és ebből szeretnék előállítani 96kHz mintavételi frekvenciájú (REFCLK) jelet, a következő összefüggéssel lehet kiszámítani, hogy milyen szorzószámmal lehet létrehozni ezt a frekvenciát egy úgynevezett “strobe” áramkörrel:

$$\text{Szorzó} = \frac{SCLK}{REFCLK} = \frac{50MHz}{96kH} \approx 520$$

A kimeneti frekvencia sávszélességét a rendszer számos tulajdonsága befolyásolja. A mintavételi törvény értelmében a kimenti frekvencia a mintavételi frekvencia legfeljebb fele lehet. Azonban itt figyelembe kell venni, hogy ahogy növeljük a kimeneti frekvencia értékét (a frekvenciaszó segítségével) úgy csökken a felbontása is a jelünknek. Továbbá a frekvencia sávszélességét meghatározza a Digitális-Analóg átalakító mintavételi frekvenciája.

A digitális hullámforma generátor által előállított jeleknél fontos figyelembe venni és mérni a következő tulajdonságokat, mivel ezek meghatározzák hol és miképpen lehet haszbálni az adott generátort:

- Dinamikatartomány (DR és SFDR)
- Fázisstabilitás (fáziszaj és jitter)
- Glitch

2.3.1. Dinamikatartomány [3]

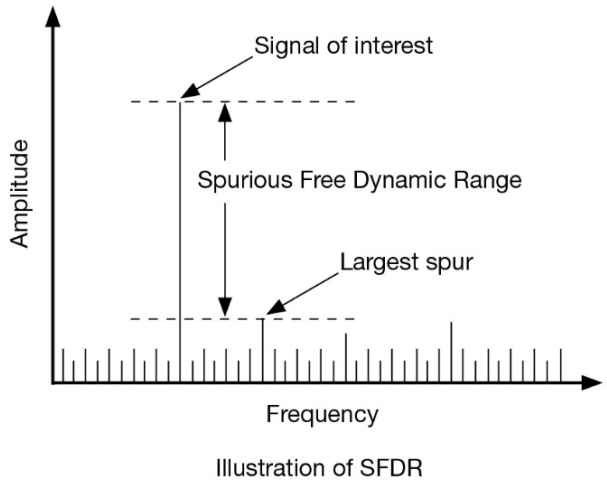
A DDS (Direct Digital Synthesis) kapcsán fontos megemlíteni a dinamikatartomány fogalmát. A dinamikatartomány az az amplitúdótartomány, amelyen belül a DDS generátor képes kimeneti jelet generálni. A dinamikatartományt általában decibelben (dB) fejezzük ki, és azt mutatja, hogy a generált jel legnagyobb- és legkisebb amplitúdója közötti arány mennyire nagy. Minél nagyobb a dinamikatartomány, annál nagyobb a jel amplitúdókülönbsége, és jobb a generált jel minősége. A dinamikatartományt befolyásoló tényezők közé tartozik a DDS generátor bitmélysége, az NCO (Numerically Controlled Oscillator) felbontása, a referenciaóra stabilitása és a DAC (Digital-to-Analog Converter) jel-zaj viszonya. Mindezeknek a tényezőknek az optimalizálása lehetővé teszi a nagyobb dinamikatartomány elérését.

Azonban sokszor használják a dinamikatartomány helyett az úgynevezett **SFDR mérőszámot**. Az SFDR, vagyis "Spurious-Free Dynamic Range" azt az értéket méri, hogy milyen nagy amplitúdójú lehet a legnagyobb hasznos jel, amelyet az adott rendszer elő tud állítani anélkül, hogy a torzítások (vagyis a nemkívánatos komponensek) elérnék a hasznos jel amplitúdóját. Ez megadja, hogy mekkora a legnagyobb (maximális szintű) jel és a legmagasabb zajszint közötti különbség. Az SFDR mértékegysége decibel (dB) és a nagyobb értéke jelent jobb minőséget.

A Spurious-Free Dynamic Range (SFDR) képlete a következő:

$$SFDR^{\mathrm{dB}} = 20\log_{10}\left(\frac{A}{B}\right)$$

, ahol A jelöli a jel maximális amplitúdója, és B a legmagasabb szintű "spur"-t vagyis a szennyező jel amplitúdója.



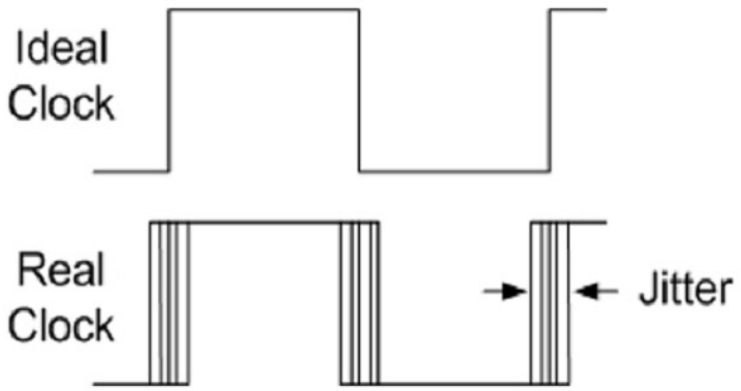
2.5. ábra. SFDR szemléltetése a jel spektrumán [7]

2.3.2. Fázisstabilitás

A DDS jelgenerátorok kiváló fázisstabilitással rendelkeznek. A fázisstabilitás azt jelenti, hogy a generált jel fázisa időben minimális változást mutat. Ez rendkívül fontos olyan alkalmazásokban, ahol a fázis kritikus tényező, például kommunikációs rendszerekben és radarrendszerekben. A fázisstabilitást a pontos órajelforrások, a hőmérséklet kompenzáció, a digitális fázisakkumuláció algoritmusok és a megfelelő szűrők alkalmazása biztosítja. A fázisstabilitást főként két tulajdonság befolyásolja, az egyik a fáziszaj, a másik pedig a jitter. A phase noise (fáziszaj) az időzítési jitter okozta frekvenciaspektrum zaj. Ez a jellemző határozza meg, hogy milyen mértékben jelennek meg a spektrumban az alapfrekvencia körül zajkomponensek. A fáziszajt dBc/Hz-ben szokták megadni, ami azt jelenti, hogy az adott frekvencián belüli egy adott sávszélességre jutó teljesítmény az alapfrekvencia teljesítményének dB-ben kifejezett értékeivel van arányban [2].

A **jitter** a digitális jel élek dinamikus elmozdulása az átlagos pozíciójuktól, hosszú távú átlagpozícióktól mért, rms fokban kifejezett érték. Egy tökéletes oszcillátorban a

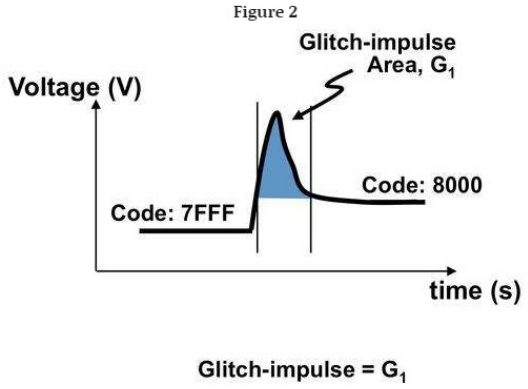
jel emelkedő és lecsengő élei pontosan rendszeres időpillanatokban történnek és soha nem változnak. Ez természetesen lehetetlen, mivel még a legjobb oszcillátorok valós alkatrészekből készülnek, amelyek zajforrásokat és más tökéletlenségeket tartalmaznak. Egy magas minőségű, alacsony fázisú zajú kristályoszcillátor jitter-e kevesebb mint 35 pikoszekundum (ps) periódusjitter, amely sok millió órajeléből összegződik. Az oszcillátorokban a jitter-t a hőzaj, az oszcillátor elektronikai instabilitásai, külső interferencia az energiaellátás, a földelés és akár a kimeneti csatlakozások révén okozzák. Még egy egyszerű erősítő, inverter vagy buffer is hozzájárulhat a jel jitter-éhez. Ezért egy DDS eszköz kimenete bizonyos mértékű jitter-t ad hozzá. Mivel minden órajelnek már van egy belső szintű jittere, kritikus szerepet játszik egy alacsony jitterű oszcillátor kiválasztása kezdetben. A nagyfrekvenciás órajel frekvenciájának leosztása az egyik módja a jitter csökkentésének. A frekvencia leosztásával ugyanaz a jitter egy hosszabb időtartamon belül fordul elő, csökkentve a rendszeridő százalékos arányát. Általánosságban elmondható, hogy a jitter lényeges forrásainak csökkentése és további források elkerülése érdekében stabil referenciaóra használata ajánlott, kerülje a lassan változó jelek és áramkörök használatát, és a lehető legmagasabb referenciafrekvenciát használja a túlmintavételezés növeléséhez [2].



2.6. ábra. Jitter [8]

A jitter-t általában szűrőkkel és időzítő áramkörökkel szűrik ki. A szűrőkkel a kimeneti jelet simábbá lehet tenni, míg az időzítő áramkörökkel biztosítani lehet a pontos időzítést,ennek érdekében gyakran használnak fázisszabályzókat és PLL áramköröket is.

A **glitch** azon pillanatokra utal, amikor a generátor kimeneti jelében rövid ideig megjelenik egy nem kívánt, átmeneti impulzus vagy zavaró jelenség. A glitch-ek általában akkor jelentkezhetnek, amikor a DDS generátor frekvenciát vagy fázist vált, az új értékek beállításakor rövid időre előfordulhatnak fázisugrások vagy frekvenciaugrások, amelyek glitch-eket okozhatnak a kimeneti jelben. A DDS generátorok működéséhez digitális algoritmusokat alkalmaznak, például fázisakkumulációs algoritmusokat. Az algoritmusok hibái vagy nem megfelelő implementáció esetén előfordulhatnak glitch-ek a generált jelben. A következő ábrán látható egy példa a glitch-re, itt egy h7FFF adat után egy h8000 adat következik. A problémát az okozza, hogy minden egyes bit-nek megváltozik az állapota, így a két érték között keletkezhetnek köztes adatok és ezek rövid időre megjelenhetnek a kimeneten.



2.7. ábra. Glitch [9]

A glitch-eknek különféle következményei lehetnek a jelgenerátor alkalmazásában, például a mérési pontosság csökkenése, a kommunikációs hibák vagy a torzított hang- vagy videojelek. Ezért a glitch-ek minimalizálása fontos a DDS generátorok tervezésénél. A glitch-ek minimalizálására és csökkentésére számos tervezési technika létezik: Időzítés és szinkronizáció: A generátor vezérlőbitjeinek időzítése és szinkronizálása segíthet minimalizálni a glitch-eket. Pontos időzítési mechanizmusok és szinkronizációs technikák alkalmazása javíthatja a generátor teljesítményét. A jó tervezési gyakorlatok, a pontos időzítés, a megfelelő szűrés és az algoritmusok optimalizálása mind hozzájárulhatnak a glitch-ek minimalizálásához és a kiváló minőségű jelgenerációhoz DDS generátorok esetén.

2.4. A DDS generátorok előnyei és alkalmazási példák

Különféle frekvenciák és profilok pontos előállításának és vezérlésének képessége ma már kulcsfontosságú követelmény számos iparágban. Sok lehetőség áll rendelkezésre a frekvenciageneráláshoz, a tervezők számára, a nagyfrekvenciás szintézishez alkalmazott fáziszárt hurok (PLL) technikáktól kezdve, az alacsonyabb frekvenciájú tetszőleges hullámformák előállításához a digitális-analóg átalakító (DAC) kimeneteinek dinamikus programozásáig. Azonban a DDS technika gyorsan elfogadottá válik a frekvencia (vagy hullámforma) generálási követelmények megoldására mind a kommunikációs, mind az ipari alkalmazásokban, mert ezek egyetlen kis chip-ben megvalósítható áramkörök és ezek az IC-eszközök egyszerűen és magas felbontással és pontossággal képesek programozható stabil analóg kimeneti hullámformákat előállítani. Továbbá, mind a technológiai folyamatok, mind a tervezés folyamatos fejlesztése eredményezte olyan költség- és energiafogyasztási szinteket, amelyek korábban elképzelhetetlenek voltak. Mára már elérhetőek olyan DDS eszközök is, amelyek frekvenciát tudnak generálni kevesebb mint 1 Hz-től egészen 400 MHz-ig (1 GHz-es órajel alapján) [2].

Az DDS generátoroknak azonban vannak bizonyos hátrányai is. Az egyik legfontosabb hátránya az, hogy a nagyobb frekvenciák előállításához szükséges nagyobb mintavételezési frekvencia miatt nagyobb sávszélességű áramkörökre van szükség. Ezen generátoroknak a spektrum tisztaságuk sem olyan, ráadásul numerikus torzítások is lehetségesek [10]. Emellett az DDS generátorok nem tudnak olyan nagy teljesítményt előállítani, mint az analóg társaik.

2.5. D/A átalakító [4]

A digitális-analóg átalakító a bemeneti digitális adatoknak megfelelő analóg kimeneti jelet állít elő. A modern DA (Digitális-Analóg) átalakítók általában integrált áramköri formában állnak rendelkezésünkre, és számos fontos komponenst tartalmaznak, melyek a következők:

- Analóg kimenetet
- Analóg referencia bemenetet
- Digitális adatbemenet
- DA átalakító
- Kommunikációs interfész
- Kimeneti erősítő

Az átalakításhoz szükség van egy fizikai referencia jelre, amely a digitális bemenetnek megfelelően átskálázva eredményez egy analóg kimeneti jelet. Ez általában egy referenciafeszültség, amelyet a digitális bemenetnek megfelelően átskálázva eredményez egy analóg kimeneti jelet referenciajel fontos szerepet játszik a dinamikatartományban, ami a DA átalakító teljes kimeneti amplitúdóját jelenti. A dinamikatartomány mértéke azt mutatja meg, hogy az átalakító milyen nagy amplitúdót képes kezelni a kimeneti jelekben. Minél nagyobb a dinamikatartomány, annál több információt képes átvinni az átalakító, és annál nagyobb a jel-zaj aránya. A jel-zaj arány az átalakító kimeneti jele és a benne lévő zaj mennyisége közötti arányt jelenti. Az átalakító dinamikatartományát a referenciajel határozza meg, ennek a jelnek a stabil volta közvetlen hatással van a dinamikatartományra. Ha a referenciajel stabilitása magas, akkor a dinamikatartomány is nagy lesz. Azaz az átalakító képes lesz kezelni nagyobb amplitúdókat, és több információt adni át a kimeneti jelekben. Ez különösen fontos audio alkalmazásokban, ahol a nagyobb dinamikatartomány magasabb hangerőt és részletgazdagságot eredményez. Ha a referenciajel instabil vagy zajos, akkor a dinamikatartomány csökkenhet, mivel a referenciajel ingadozásai és a zaj is beleszámítódnak a kimeneti jelbe. Ezért fontos, hogy a referenciajel minél stabilabb legyen, és a zajszintet a lehető legalacsonyabbra csökkentsék.

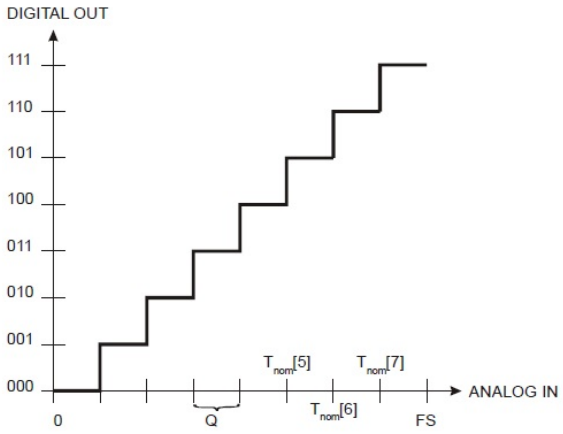
A DA átalakítók még egy fontos jellemzője a bitmélység, ez meghatározza, hogy a kimeneti amplitúdó mennyi lépésben érhető el a nulla kimenettől a teljes skáláig. A bitmélység azt mutatja, hogy hány bitet használ az átalakító a digitális bemeneti adatok reprezentálásához. Minden egyes bit egyenlő lépésközt jelent a kimeneti amplitúdó skáláján. Az átalakító bitmélységének növelése több lépéssel bővíti az amplitúdó tartományt, és így finomabb felbontást eredményez. A nagyobb bitmélységű átalakítók több információt képesek átvinni a kimeneti jelekben, ami kisebb kvantálási hibát eredményez. Például ha van egy 10 bites átalakítónk, 2^{10} (vagyis 1024) különböző amplitúdóértéket tudunk beállítani a referenciajel segítségével. Tehát a kimeneti amplitúdó 1024 lépésben változhat a nulla kimenettől a maximális értékig. Általánosan elmondható, hogy az audio alkalmazásokban a 16-24 bitmélység széles körben használt, mivel ezek a tartományok már kielégítő hangminőséget és dinamikatartományt biztosítanak. A mintavételezés során mintákat veszünk az adott jelből, jelen esetben ez digitális jel, a mintavételi frekvencia azt adja meg, hogy ezt milyen időközönként tesszük meg, vagyis hány mintát veszünk másodpercenként. Az audio alkalmazásokban általában 44,1 kHz (44 100 mintavétel/-másodperc) vagy 48 kHz (48 000 mintavétel/másodperc) mintavételezési frekvenciákat használnak. Ez azt jelenti, hogy a bemeneti jelből ezeken a frekvenciákon 44 100 vagy 48 000 mintát veszünk másodpercenként [4].

A megfelelő mintavételezési frekvencia kiválasztása fontos a hangminőség és a jelintegritás szempontjából. A Nyquist-Shannon mintavételezési tétel alapján a mintavételezési frekvencia legalább kétszerese kell legyen a reprodukálni kívánt jel legmagasabb frekvenciájának. Ez biztosítja, hogy a digitális átalakító pontosan rekonstruálhassa a jelünket. A Digitális-analóg átalakítóknak sajnos zajuk is van, amely jelentős szerepet játszik a dinamikatartomány méretében és hatással van a jelminőségre. A zaj forrásai lehetnek például a rendszer elemeiben lévő elektromos zajok, mint a kvantálási zaj vagy az áramkörök belső zajai. A zajcsökkentés kulcsfontosságú a kívánt hangminőség eléréséhez. Ennek érdekében gyakran alkalmaznak szűrőket és egyéb zajcsökkentő módszereket.

A szűrők képesek kiszűrni a nem kívánt frekvenciákat vagy csökkenteni azok amplitúdóját, ezáltal minimalizálva a zaj hatását. További fontos tulajdonságuk ezeknek az átalakítóknak az, hogy milyen bemenettel rendelkeznek és milyen formán fogadják a bemeneti digitális adatokat. Általában soros kommunikációt használnak, ami az esetek nagy

részében SPI (Serial Peripheral Interface), de gyakran használnak I2S (Inter-IC Sound) protokollt, amely kifejezetten audió felhasználásra tervezett kommunikációs protokoll. Ezeknél az átalakítóknál a bemenő adat formátuma is sokféle lehet, például hogy egy- vagy kétcsatornás hangrendszerről beszélünk.

Az átalakítás folyamata során a Digitális-Analóg átalakító (DA átalakító) megőrzi a mintavett adatokat a következő mintavételi időpontig. Ennek a folyamatnak a célja, hogy a digitális adatokból analóg jel képződjön. A mintavételi időpontok közötti adatok tárolása és visszaállítása kulcsfontosságú a helyes átalakítás érdekében. A leggyakoribb megoldás a nulladrendű tartó alkalmazása, ami azt jelenti, hogy a mintavételi időpontok között a bemeneti érték állandó marad. Ez a megoldás egyszerű és hatékony, és általában elegendő a megfelelő átalakításhoz. Azonban bizonyos esetekben, amikor nagyobb pontosság és simább analóg jel előállítása a cél, más tartó megoldások is alkalmazhatók. Például az elsőrendű tartó lehetőséget nyújt a pontosabb átalakításra. Az elsőrendű tartó a két mintavételi időpont közötti adatot lineárisan közelíti, így a kimeneti jel simább és kevésbé torzított lesz. Ez látható 2.8 ábrán is.

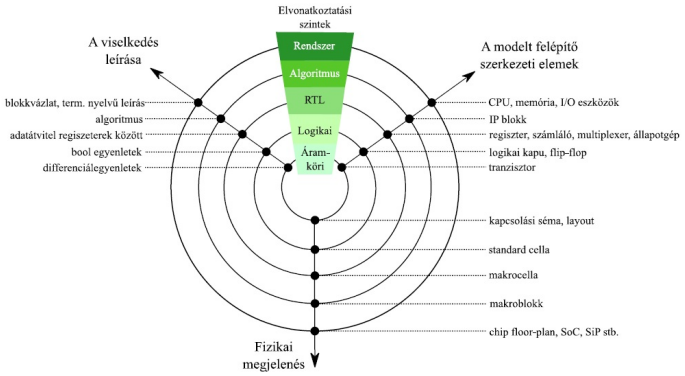


2.8. ábra. Digitális jelek tartása és átalakítása analóg jellé [11]

2.6. FPGA-ra tervezés

A digitális rendszer tervezésének különböző elvonatkoztatási szintjeit az alábbi Gajski-Kuhn Y-diagram szemlélteti:

Először meg kell határoznunk a specifikáció alapján a rendszer interfészeit, működési tulajdonságait, stb. Itt megfigyelhető, hogy először a viselkedési modellt állítjuk elő



2.9. ábra. Gajski-Kuhn Y-diagram [12]

(algoritmus szint), jelen esetben generálunk egy megfelelő hullámformát. Ezután az RTL (Regsiter Transfer Level) tervezése következik, melyben megtervezésre kerül az RTL modell egyes almoduljainak működése és interfészei. Majd ezeket a modulokat integráljuk egy top level modulba, melyben az almodulok az interfészeiken keresztül kommunikálnak egymással. A helyes működést ellenőrizni is kell, ezt az úgynevezett verifikációval tesszük meg.

Különböző verifikációs formákból, eljárásokból válogathatunk attól függően, hogy milyen alkalmazásról van szó, vagy milyen mélységben kell tesztelni az adott rendszert.

Verifikáció lépései: Először a teszt környezetet építjük fel, amely tartalmazza az általunk tervezett rendszer modulját, melyet DUT-nak (Design Under Test) nevezünk. Ezután tesztelési eseteket és eljárásokat készítünk, amelyek előállítják a bemeneti adatokat, amelyeket a DUT-nak adunk át, ezek ellenőrzik a kimenetet, hogy megfelel-e a specifikációnak. Végül a teszt indítása és az eredmények ellenőrzésével zárjuk a folyamatot. Ezután következik az RTL szintézise, melyben a szintézis szoftver kialakítja az áramkör fizikai reprezentációját, melynek segítségével az FPGA-t konfigurálhatjuk fel. Itt figyelni kell arra, hogy a működés eltérhet a verifikáció során tapasztalt működéstől, mivel a fizikai megvalósításban az időzítési tulajdonságok is fontos szerepet játszanak, ezért ennek megfelelően kell a rendszert megtervezni és további vizsgálatokat kell végrehajtani a helyes működés érdekében. Ilyen fontos vizsgálat a statikus időzítésvizsgálat (STA-Static Time Analysis) is.

A digitális hullámforma generátor tervezése és megvalósítása az FPGA (Field Programmable Gate Array) technológia segítségével számos előnnyel jár. Az FPGA egy olyan

speciális integrált áramkör, amely lehetővé teszi a felhasználók számára, hogy testre szabják az áramkört az alkalmazásukhoz. Az FPGA egy rekonfigurálható logikai eszköz, amely nagy számú egyéni logikai áramkör és digitális jelfeldolgozó (DSP-Digital Signal Processor) blokkokból áll. Az FPGA-k programozása Verilog vagy VHDL, úgynevezett HDL (Hardware Description Language) nyelvekkel történhet, amelyek mindegyike lehetővé teszi a rendszer egyszerű és hatékony tervezését. Az FPGA-k a digitális jelfeldolgozás területén is nagy szerepet játszanak, és számos digitális jelfeldolgozó algoritmus implementálására alkalmasak. A digitális hullámforma generátor tervezése és megvalósítása az FPGA technológiával kiváló lehetőséget kínál a digitális rendszerek fejlesztői számára. A dolgozatban bemutatott tervezési folyamat és az FPGA-k használata segítséget nyújt a rendszer hatékony és gyors tervezéséhez, implementálására és tesztelésére.

3. Célkitűzések és felhasznált eszközök

Feladatom célja egy olyan digitális hullámforma generátor tervezése, mely könnyen kezelhető, gyorsan változtatható frekvenciájú általános hullámformát biztosít hangfrekvenciás tartományban (20Hz és 20kHz közötti frekvenciák). Ezt FPGA-ra optimalizálom. Jelen FGPA áramköröm egy ALTERA DE2 boardon megtalálható Cyclone II EP2C35 típusú FPGA. Ez a chip rendelkezik egy 50MHz-es órajelet előállító modullal. Továbbá található benne PLL (Phase Locked Loop) is, mely az alap órajelből képes előállítani egy vagy több kívánt frekvenciájú órajelet. Az eszköz tartalmaz block ram részeket is, melyek 4kbyte tárolási kapacitással bírnak. Az alkalmazásomhoz szükség van egy Digitális-Analóg átalakítóra, ami egy Wolfson WM8731 24-bites sigma-delta audio CODEC chip-ben található meg. Ez 16 és 24 bites átalakításra képes, a mintavételi frekvenciája 44kHz-re vagy 96kHz-re állítható. Ezen kívül lehetőség van külső Digitális-Analóg átalakítót használni és akkor ez a mintavételi frekvencia nem jelent szűk keresztmetszetet és nagyobb frekvenciájú jeleket is képes előállítani a generátor. Én az előbbi megoldást választottam, mivel ez állt rendelkezésemre és a chip hiány miatt a diszkrét DA átalakító beszerzése is nehézkes volt. A board-on megtalálható chippel ugyan korlátozódik a frekvenciatartományunk, viszont mivel hangfrekvenciás tartományban szeretnénk előállítani jeleket ez nem befolyásolja az általunk elvárt működést. Az áramkör megfelelő működését I2C (Inter- Integrated Circuit) interfészen keresztül állíthatjuk be. Az amplitúdóadatokat pedig egy I2S (Inter-IC Sound) interfészen keresztül küldhetjük az átalakítónak, ami sorosan fogadja a 16 vagy 24 bites digitális amplitúdó adatokat. A számítógép által generált hullámformát az FPGA block ram-jába szeretném beletölteni. Mivel ez a rendszer egy általános hullámforma generátor, így a tárolni kívánt hullámforma a szoftver használója által tetszőlegesen generált lehet, valamint az amplitúdóadatok száma, vagyis a memória mérete is testreszabható.

3.1. Rendszerkövetelmények

A hullámformák megjelenítése hangfrekvenciás tartományban történik, 20 Hz és 20 kHz közötti frekvenciákon. A rendszer mintavételi frekvenciáját 96 kHz-re választottam, ami megfelel a mintavételi törvénynek, és meghaladja a Nyquist-kritériumban megfogalmazott minimális mintavételi frekvenciát, ami legalább kétszerese az általam megjeleníteni kívánt maximális frekvenciának. Jelen alkalmazásban az FPGA áramkörömön a hullámforma amplitúdó adatainak száma 2^{12}-re korlátozódik, ez pedig 1024 darab 8-bites amplitúdó adatot jelent. Azonban ez az érték testreszabható, és szintézis paraméterként megadható, továbbá más nagyobb memóriával rendelkező eszközre is szintetizálható annak érdekében, hogy elegendő mennyiségű adatot töltsünk be. Ennek alapján kiszámítható a kimeneti frekvencia felbontása és a minimális lépésköze a következőképpen:

$$f_{\text{out}} = \frac{M \times f_C}{2^n} = \frac{96\text{kHz}}{2^{12}} = 23,43 Hz\text{Hz}$$

Tehát 23,43Hz-es felbontást jelent és a maximálisan használható frekvencialépés pedig így alakul, ha 20kHz-es tartományban szeretném megjeleníteni az hullámformát.

$$M = \frac{f_{\text{out}} \times 2^n}{f_C} = \frac{20\text{kHz} \times 2^{12}}{96\text{kHz}} \approx 853$$

Mivel a kimeneti adatok 8 bitesek, de az átalakító 16 vagy 24 bites adatokat képes fogadni és átalakítani, szükséges az adatokat 16 bitesre átalakítani. Erre a célra a 8 bitet különböző módon csatlakoztatjuk a 16 bites bemenetre, és a többi bemenetre nullát kötünk. Számos lehetőség van a csatlakoztatás módjára, azonban a nagyobb jel- és dinamikatartomány kihasználása érdekében úgy döntöttem, hogy a felső 8 bitet kötöm a kimeneti vezetékekhez, míg az alsó 8 bitre konstans 0 értéket adok.

Ennek számos előnye van számunkra:

- Amikor a felső 8 bitet használjuk az adatokhoz, a kimeneti feszültség nagyobb lesz, ami előnyös lehet, ha nagyobb jel- és dinamikatartományra van szükségünk. Azonban ez a megoldás érzékenyebb lehet a zajra és a kvantálási hibákra.

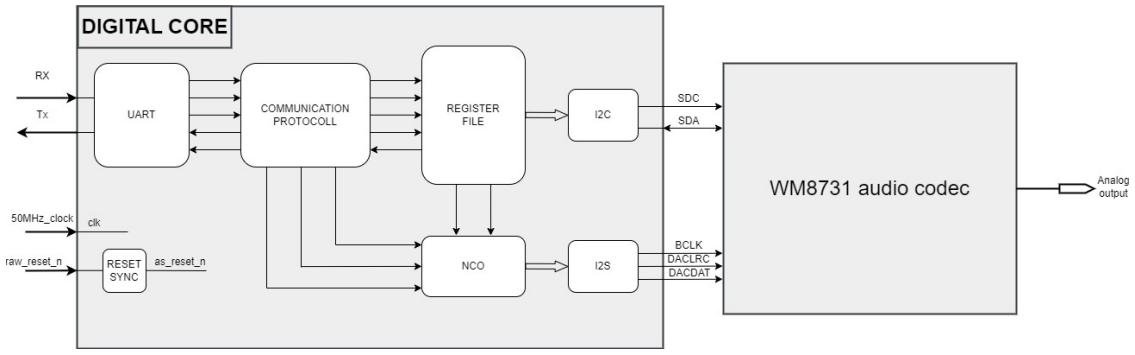
- Amikor az alsó 8 bitet használjuk az adatokhoz, a kimeneti feszültség kisebb lesz, ami előnyös lehet, ha kisebb feszültségszintekkel dolgozunk, vagy ha csökkenteni szeretnénk a zaj és a kvantálási hibák hatását. Ugyanakkor a jel- és dinamikatartomány kisebb lesz.

Mindkét megoldásnak vannak előnyei, és a választás a konkrét alkalmazástól, költségkerettől és előírásoktól függ. Például egy R-2R hálózatos DA átalakítónál nincs különbség abban, hogy melyik 8 bitre kötjük az adatokat, mert mindkét esetben ugyanazt a kimeneti feszültséget kapjuk. Egy string vagy multiplying DA átalakítónál azonban fontos lehet a bitrend, mert az befolyásolja a kimeneti feszültség skálázását.

A rendszer RTL (Register Transfer Level), vagyis regiszter szintű tervezését a Quartus II tervezői környezetben végezem, és a block ram modul konfigurálását is ebben a fejlesztői környezetben hajtom végre. A szimulációhoz a ModelSim szimulációs szoftvert használom.

4. Rendszerterv

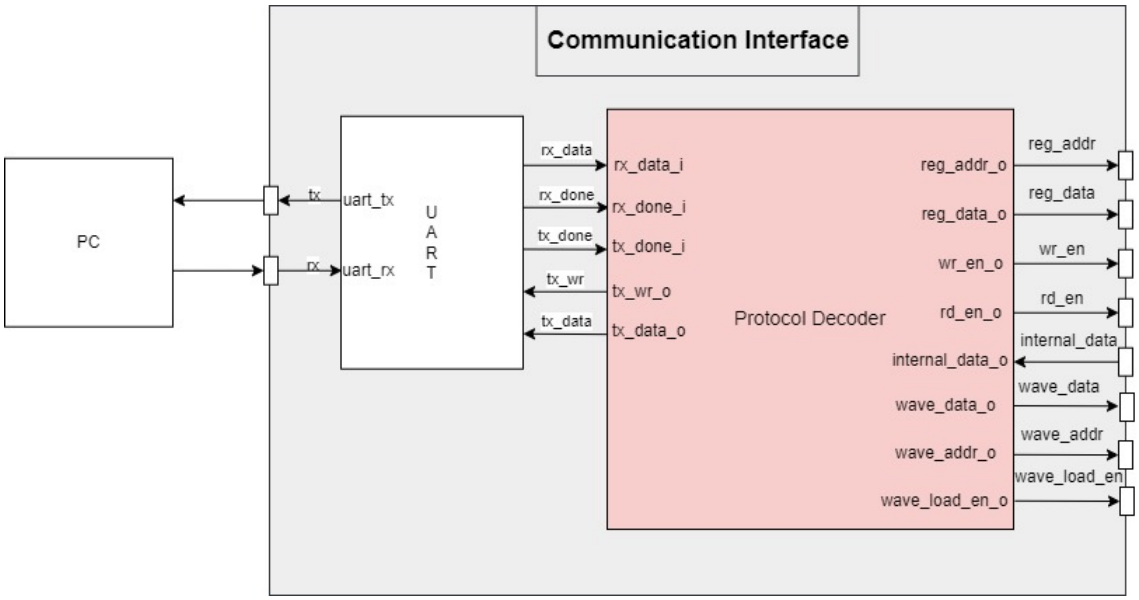
Az általam tervezett DDS generátor rendelkezik egy UART bemeneti interfésszel, mely a számítógépből fogadja az adatokat az *'uart_rx'* soros bemeneten keresztül. Ez az interfész főleg adatok fogadására szolgál, ugyanakkor ha kell tudunk adatokat küldeni, mely akkor szükséges, ha a rendszer belső regisztereiből szeretnénk olvasni a rendszer beállítására szolgáló adatokat. Az elküldött adatokat egy *'protocol_decoder'* modul dolgozza fel és továbbítja a regiszterfájl felé. A protokoll dekóder az adatátviteli protokoll dekódolására használatos, mely szétválasztja a bejövő üzeneteket a kidolgozott protokoll alapján. A rendszer ezután eltárolja ezeket a bejövő adatokat a megfelelő regiszterbe majd ezek szolgálnak bemenetként az NCO modulunknak, amely ezen adatok alapján beállítja a kimeneti hullámforma tulajdonságait. A regiszterben tároljuk még a WM8731 konfigurálásához szükséges 2 byte-os adatokat, melyeket szintén a szoftverben határozunk meg, az adatlap alapján a hullámforma küldése előtt. Továbbá az NCO memóriájába (block ram) történő amplitúdó értékek írása is UART-on keresztül történik és a *'protocol_decoder'* modulban egy külön állapot végzi ezt a műveletet.



4.1. ábra. Rendszerterv

4.1. Adatátviteli protokoll

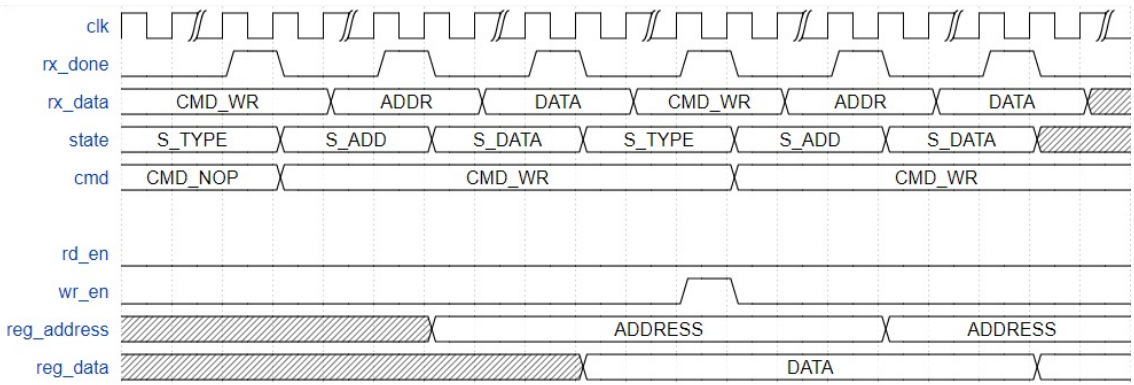
Mint korábban említésre került a számítógép által UART interfészen keresztül küldünk adatokat. Az UART modul RTL modellje egy nyílt forráskódú Verilog projektből származik [13]. Itt a baudrate beállítását érdemes megemlíteni, mely az adatátvitel sebességét állítjaű be. Hogy az adatok megfelelően átjussanak a számítógépből az FPGA rendszerünkre ugyanakkora baudrate-et kell beállítani mind az RTL modellben, mind pedig a szoftverünkben. Ez a paraméter futásidőben állítható. Ez nálam a gyakran használt 9600 Baud/s. Miután az adatok átküldésre kerültek és sikeresen megérkeztek az $'rx\_data'$ regiszteren keresztül a már előbb említett protokoll dekóderbe kerülnek, amely gyakorlatilag egy állapotgép, mely a különböző adatokat szétválogatja az adatátviteli protokoll alapján. Ez a két modul az adatátvitelt valósítja meg és a rendszer szintű blokkdiagramja a 4.2 ábrán látható.



4.2. ábra. Kommunikációs interfész

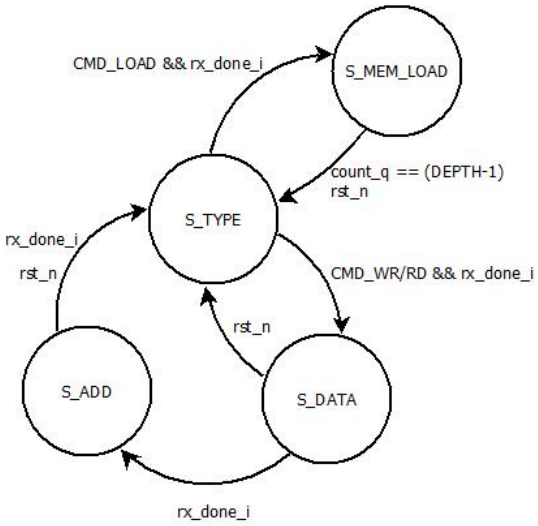
A szoftverben összeállítunk egy három, egyenként 8 bitből álló csomagot. Ennek a csomagnak az első bájtja a parancs (command), ezt követi a cím (address), végül az adat (data). Ezeket mindig egy csomagban küldjük el. A command parancsunk háromféle lehet, írás (CMD_WR), olvasás (CMD_RD), ez dönti el, hogy írni vagy éppen olvasni szeretnénk a regiszterfájlokat/regiszterfájlokból. Az cím adatunk a regiszter fájlok címét jelzi, amibe írni szeretnénk a soron következő adatot. Továbbá load (CMD_LOAD),

melyet akkor használunk, ha új hullámformát szeretnénk betölteni a memóriába. A protokoll hullámformája a következőképpen néz ki, azonban a 4.3 ábra nem tartalmazza az adatok betöltését, csak a regiszterekbe történő írást:



4.3. ábra. Kommunikációs protokoll hullámformája.

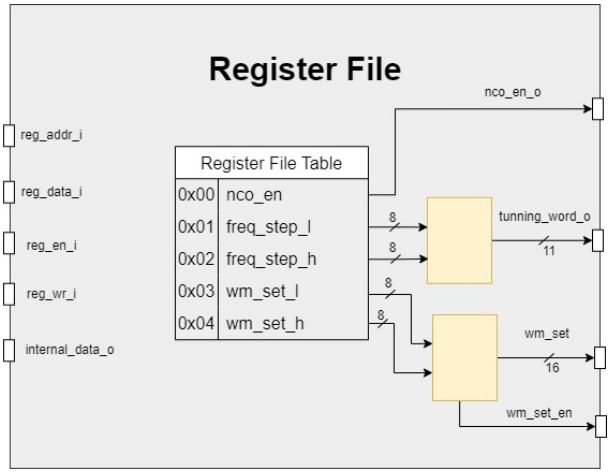
A modul folyamatosan várja az adatokat és egy kezdő állapotból (S_TYPE) indul ki, ami vizsgálja, hogy milyen parancs érkezett a bemeneten. Akkor vizsgálja meg, hogy írás, olvasás vagy betöltés, mikor az rx_done_i jelet megkapta, ami az UART felől jön és akkor ad egy egy órajel szélességű impulzust mikor végzett az UART az aktuális adat olvasásával. Ha az előbbi kettő parancs valamelyikét kaptuk (írás/olvasás), akkor a következő órajelre a következő állapotba lépünk, ami az S_ADD állapot, ez következő rx_done_i hatására átadja az aktuális adatot az address regiszternek, melyben a megcímzendő regiszter címét tároljuk. Ezután átlépünk az S_DATA állapotban. Itt megvizsgáljuk, hogy írás vagy olvasás parancsot kaptunk, ha olvasás, akkor az UART-nak átadjuk a belső regiszterből jövő adatot, ha írás akkor az UART felőli adatot adjuk át a regiszter fájlnak. Az address hatására kiválasztódik a regiszter, amibe ezt az adatot tudjuk tárolni. Majd az adat betöltése után újra a kezdő állapotba ugrunk (S_TYPE). Ha a betöltés parancsot kapta a rendszer, akkor az állapotgépünk belép egy S_LOAD állapotba és ezután a soros portról csak a hullámforma adatait várja sorban, amíg az utolsó adat be nem érkezik. Ezt egy számláló biztosítja, mind hardver, mind szoftver oldalon. Ha az adatoka írása befejeződött, akkor visszaküldünk egy nyugtázó adatot a szoftvernek (8'h05) érték. Ilyenkor kilépünk az állapotból és az alap állapotba térünk vissza. Ennek az állapotgépnek az ún. állapotgráfját a következő ábrán láthatjuk.



4.4. ábra. Protokoll dekóder állapotgráf.

4.2. Regiszterfájl

A regiszterfájl az NCO-t működtető adatok tárolására szolgál. Ennek kettős szerepe van. Az egyik ilyen, hogy eltárolja és továbbítsa a szükséges beállításokat az NCO felé, a másik pedig az, hogy leválasztja az fő funkciót, vagyis az NCO-t a kommunikációs protokollról, tehát ha a kommunikációs protokoll megváltoztatjuk valami miatt, vagy több nco modult szeretnénk a továbbiakban hozzáadni a rendszerhez, akár különböző modulációk miatt, akkor a rendszer többi részén nem kell változtatni csak példányosítani az nco-t.



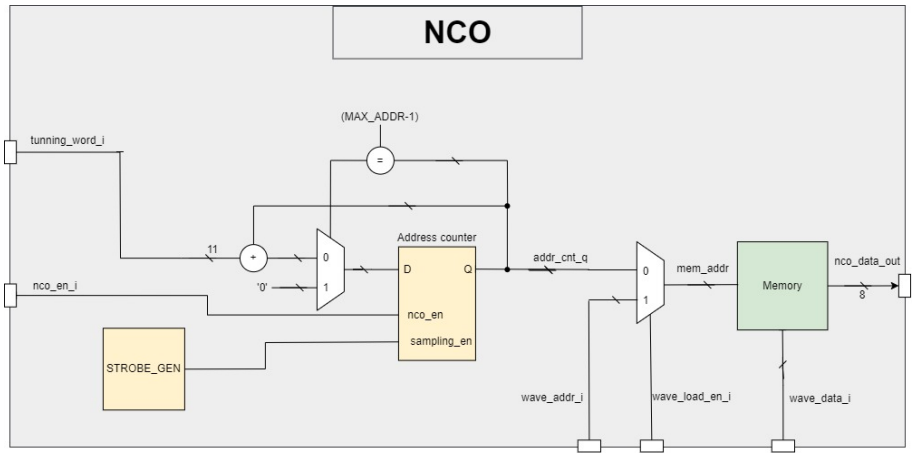
4.5. ábra. Regiszter fájl interfésze és a benne tárolt adatok.

Ebben a modulban három adatot tárolunk:

- az nco engedélyezése, ezt 1 bites regiszterben tároljuk
- dac adatainak beállítása, ez egy 16 bites regiszterben történik. Ebbe a regiszterbe az audió kodek beállításaihoz használt digitális szót írjuk, melyet I2C kommunikáción keresztül küldünk el az áramkörhöz. Ide csak két részletben tudunk írni. Ezért olyan megoldáshoz kellett folyamodni, ahol szoftveresen két 8 bites adatra választjuk szét ezt a szót, majd eküldjük az UART-on, fogadjuk és két 8 bites regiszterbe írjuk őket, miután ezzel végeztünk összefűzzök ezeket egyetlen 16 bites regiszterbe.
- tunning word, vagyis a frekvencia beállítására szolgáló adatot, pedig egy 10 bites regiszterben tároljuk, az előzőhöz hasonlóan itt is két részletben tudjuk elküldeni és fogadni az adatot.

4.3. Numerikusan Vezérelt Oszcillátor

Ez az a modul, ami a digitális hullámformát állítja elő. A bemenetére kap egy tunning word-öt, amit a számlálóhoz mindig hozzáad. Ennek elvéről az Irodalomkutatásban esett szó. Itt a megvalósításáról szeretnék pár mondatot írni. A modell blokkvázlata a következő:

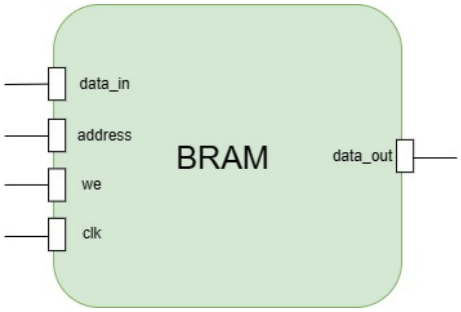


4.6. ábra. NCO blokkdiagram.

Ez egyrészt áll egy strobe generátorból, mely egy 1 órajel hosszúságú impulzust állít elő minden 520-adik órajelre, ez onnan jön, hogy 96kHz mintavételi frekvenciát szeretnénk és 50MHz-es órajel áll rendelkezésünkre, így 50MHz/520 = 96kHz. Ez engedélyezi mindig a számlálót, vagyis a fázis akkumulátort (phase accumulator), amihez minden egyes számláláskor hozzáadjuk a lépésközt. Így előáll a kimeneti cím, amit az ebben a modulban létrehozott memóriára kötünk, melyet egy blockram valósít meg, aminek az eltárolt adatait megcímezve és kiolvasva rákötünk a kimenetre és előáll a szinusz jelünk. Az NCO bemenetén kap egy engedélyező jelet és egy frekvencialépést. Amikor az engedélyező jel '1' állapotban van, akkor a rendszer kilép az alap (S_IDLE) állapotba és átmegy a működési állapotába (S_RUN). Itt megvizsgálja, hogy van-e a frekvencialépés bemenetén adat, vagyis nem nulla ez a bemenet. Ha ez a bemenet nulla, akkor átlép egy várakozó (S_WAIT) állapotba és addig ott marad, amíg nem kapja meg a bemenetére ezt a frekvencialépést vagy resetet nem kap. Ezt folyamatosan, minden órajelre megvizsgálja.

4.4. Block Ram modul

A ram modult adat táblaként használjuk, melyben eltároljuk a szoftveresen előállított hullámforma adatait. Ezek az adatok 8 bitesek. A tábla maga 12-bit mélységű, vagyis 4096 adatot tudunk benne tárolni. A memóriát UART-on keresztül töltjük fel egy szinusz hullámforma adataival, ezzel szimuláljuk a működését. A szintézis során a memóriát inicializáljuk egy szinusz hullám adataival, hogy alapértelmezetten legyen benne egy szinuszos jel. Az inicializálásról a szintézis részben lesz szó, lásd 5. fejezet.



4.7. ábra. Block RAM modul.

A RAM modul RTL modelljének a következő módon kell kinéznie, hogy azt a szintézis tool felismerje és block ram-ként szintetizálja.

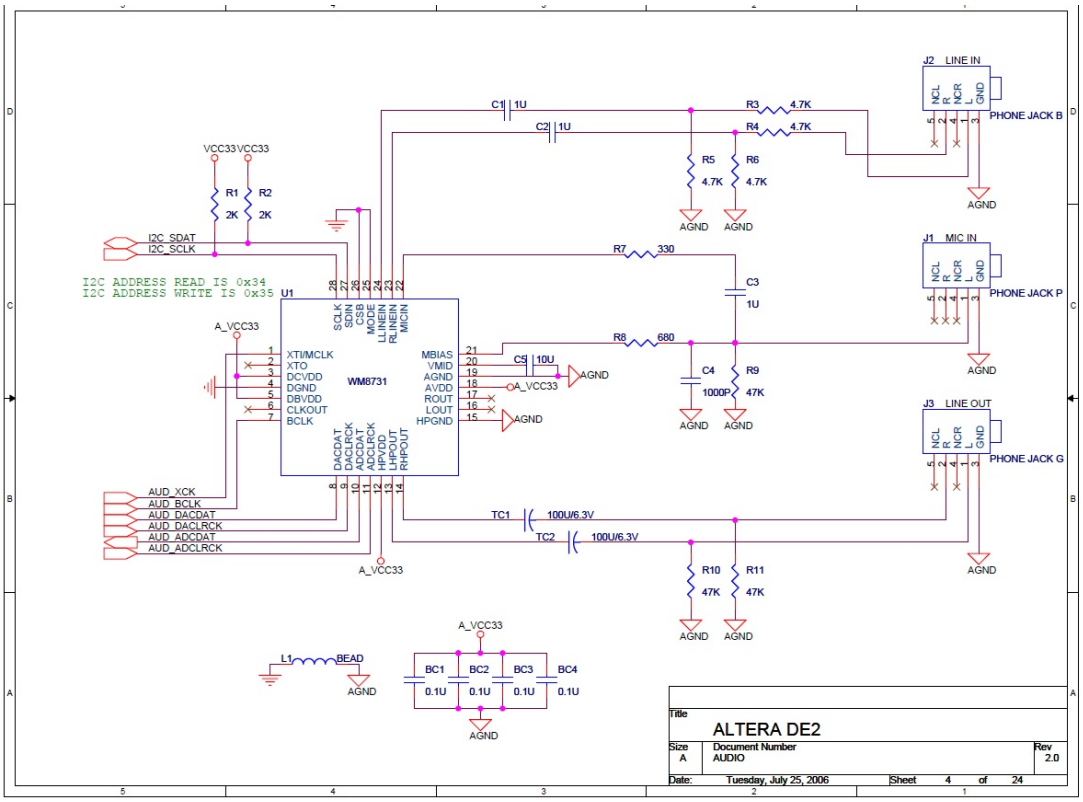
```
module sinus_lut #(
        parameter ADDR_WIDTH = 11,
        parameter DATA_WIDTH = 8,
        parameter DEPTH = 4096
)(
        input   wire clk,
        input   wire we,
        input   wire  [(ADDR_WIDTH - 1) : 0] addr,
        input   wire  [(DATA_WIDTH - 1) : 0] data_in,
        output  logic [(DATA_WIDTH - 1) : 0] data_out
);
logic [(DATA_WIDTH - 1) : 0] content [0 : (DEPTH-1)];

always_ff @ (posedge clk) begin
        if ( we ) content[addr] <= data_in;
end
assign data_out = content[addr];
endmodule
```

Az ADD_WIDHT a címbusz bitjeinek száma, a $DATA\_WIDTH$ az adatbiteké, a $DEPTH$ pedig a memória mélysége. Gyakorlatilag ezzel a kódsorral hozzuk létre a memóriát, amely 8 4096 (2^{12}) darab 8 bites regiszterből áll. A továbbiakban pedig a memóriaregiszterek írásának és olvasásának működését adjuk meg, ha a $'we'$ jel aktív, akkor az órajel felfutó élére az $'addr'$ bemeneten megjelenő címre írjuk be a $data\_in$ bemeneten megjelenő adatot. A modul paramétereit szintézis paraméterként adhatjuk meg és írhatjuk felül a szintetizálás előtt a top level modulban, ezáltal a felhasználó tetszőleges méretű memóriát használhat, ezt csak a fizikai eszközünk memóriamérete korlátozza. Az egész rendszerre vonatkozik az, hogy a paramétereket szintézis paraméterként implementáljuk, mivel FPGA technológiáról van szó, így a felhasználó a szintézis előtt könnyen meg tudja változtatni ezeket a tulajdonságokat.

4.5. Digitál-Analóg átalakító illesztése az FPGA-hoz

Ahhoz hogy a digitális jelünket analóggá alakítsuk át szükségünk van egy Digitál-Analóg átalakító áramkörre, ami az Altera DE2 board-on megtalálható. Ez egy WM8731 24-bites

sigma-delta audio CODEC. Ennek az áramkörnek a következő ábrán látszik a kapcsolási rajza, amit az altera DE2 adatlapjáról vettem át.



4.8. ábra. WM8731 kapcsolási rajza. [14]

4.5.1. Kommunikáció a WM8731 Audió codec-el

A WM8731 beállításait az I2C interfészen keresztül tudja kommunikálni. Az I2C (Inter-Integrated Circuit) egy szinkron soros kommunikációs protokoll. Ez alapvetően két vezetéket használ: SDA (Serial Data) és SCL (Serial Clock). Ezen vezetékek úgynevezett tri-state meghajtásúak, vagyis három állapotuk lehetséges, '0' vagy '1' vagy 'z', azaz nagy impedanciás állapot. Az SDA a kétirányú adatvonal, ahol az adatok továbbítása történik. Az SCL a szinkronizációs órajelvonal, amely szabályozza az adatok átvitelének sebességét és időzítését. Az I2C rendszerben általában van egy vagy több központi eszköz, amelyet vagy amelyeket mesternek (master) nevezünk, és egy vagy több perifériás eszköz, amelyet vagy amelyeket szolgának (slave) hívunk. A mester irányítja az adatátvitelt a szolgákkal, és az eszközök címzése segítségével kommunikálnak egymással. Az I2C protokoll lehetővé teszi a soros adatok továbbítását, a multimaszter működést (több mester), az eszközök

címzését, adatok olvasását és írását, valamint az eszközökkel való szinkronizációt. Jelen alkalmazásban a mester az FPGA-ban megírt I2C modul ($'wm\_i2c\_master'$) és egy szolga van, ami a WM8731 codec. A modellt a [15] forrásban megjelölt rtl modell segítségével írtam. A WM8731 adatlapjában megtalálhatók azok a regiszterek és bitek, amelyeket be kell állítani a megfelelő működés érdekében. A következő lépésekkel lehet I2C segítségével kommunikálni a WM8731 kodekkal [16].

1 Először egy Start bitet küldünk át. Az adatátviteli vonal magas állapotban van, vagyis egy lefutó él jelzi a startot.

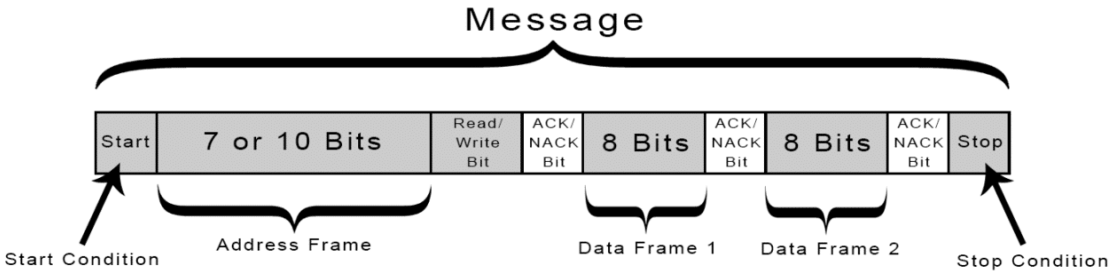
2 Címzés: Az I2C kommunikáció során a kodeket a címével kell azonosítani. A WM8731 címe előre definiált, és a datasheet-ben megtalálható (0x35). Az FPGA elküldi a kodek címét, jelezve, hogy a további kommunikáció a WM8731 kodekkel történik.

3 Ezután beállítjuk, hogy írást vagy olvasást szeretnénk végrehajtani.

4 Majd egy ACK (acknowledgment) jelet küld a slave a master-nek

5 Regiszterek beállítása: ezután írhatunk két 8 bites adatot, melyek között szintén egy ACK jelet kapunk vissza. Ezek az adatok címzik a megfelelő regisztert, melybe beállítjuk a megfelelő biteket. Ez lehetővé teszi, hogy konfigurálhassuk a kodekünket.

6 Végül egy Stop bitet küldünk, ami jelzi az üzenet végét.



4.9. ábra. I2C kommunikációs protokoll [16]

4.5.2. WM8731 Audió codec felkonfigurálása [5]

Először beállítjuk a kikapcsolási vezérlést, amit a '0000110' című regiszterben tehetünk meg. Ez a beállítás lehetővé teszi a kodek energiafogyasztásának szabályozását és az egyes áramkörök inaktív állapotba helyezését. Itt engedélyezzük a vonalbemenet, a microphone, az AD átalakító kikapcsolását. Viszont nem engedélyezzük a DAC, a kimenet, az oszcillátor, a kimenti frekvencia és a POWEROFF mód kikapcsolását.

Register		DATA set	
Name	Address	Label	Description
POWER DOWN CONTROL	7'b0000110	LINEINPD=1	Enable Input Power Down
		MICPD=1	Enable Mic Power Down
		ADCPD=1	Enable ADC Power Down
		DACPD=0	Disable DAC Power Down
		OUTPD=0	Disable Output Power Down
		OSCPD=0	Disable Oscillator Power Down
		POWEROFF=0	Power Off mode off

1. táblázat. Energiafogyasztás szabályozását beállító regiszeter

A “0000111” című regiszterben beállítjuk az adatformátumot (FORMAT[1:0]), amit szeretnénk fogadni a bemeneten. Itt az MSB adatot küldjük először és az adatkeret balra igazított. A harmadik és negyedik bit a bementi adatok hosszúságának megadására szolgál, itt beállítjuk a 16 bites adathosszt. Továbbá beállíthatjuk az adatok feldolgozását végző formátumot, itt a ha a beállítás 0 (Right Channel DAC data when DACLRC low), akkor a jobb csatorna (right channel) DAC adatok akkor lesznek érvényesek, amikor a DACLRC jelzés alacsony szinten van. Továbbá a codec slave módban van (MS=0) és a BCLK jelet nem invertáljuk.

Register		DATA set	
Name	Address	Label	Description
DIGITAL AUDIO INTERFACE FORMAT	7’b0000111	FORMAT[1:0] = 01	MSB-First, left justified
		IWL[1:0]=00	input data 16 bits
		LRP=1	Right Channel DAC data when DACLRC high
		LRSWAP=0	Right Channel DAC Data Right
		MS=0	slave mode
		BCLKINV=0	don’t invert bclk

2. táblázat. Digitális adatformátumot beállító regiszter

Fontos még megemlíteni milyen mintavételezési módban használjuk az áramkört. Ezt a “0000110” regiszterben tudjuk beállítani. USB módra állítottam az eszközt, mivel fs=96kHz-es a mintavételi frekvenciám és ilyenkor az eszköznek az MCLK jelének 12MHz-et szükséges előállítani, tehát 12MHz/96kHz = 125, azaz 125fs-szer vesz mintát a bementen. Ezeket a BOSR és az SR[3:0] bitekkel lehet állítani.

Register		DATA set	
Name	Address	Label	Description
SAMPLING CONTROL	7'b0001000	USB/NORMAL=1	USB mode 256/384 fs
		BOSR=0	
		SR[3:0]=0111	DAC sample rate control, 12MHz órajel mellett
		CLKIDIV2=0	clk in is MCLK
		CLKODIV2=0	clk out is Core clock

3. táblázat. Mintavételezés tulajdonságait beállító regiszter

Ahhoz, hogy a DAC-ot használhassuk be kell állítani az ANALOGUE AUDIO PATH CONTROL nevű, '0000100' című regiszter értékeit is. Itt kiválasztjuk a DAC-ot, kikapcsoljuk az ADC-t, a mikrofont stb. Pontos beállítások a következő táblázatban láthatók, ahol az első oszlopban a regiszter neve, másodikban a címe látható. Következő oszlopban a beállítandó adatbitek, végül pedig a leírása, hogy mire szolgál vagy mit állítottunk be ezzel a bittel.

Register		DATA set	
Name	Address	Label	Description
ANALOGUE AUDIO PATH CONTROL	7'b0000100	MICBOOST=0	Microphone Input Level Boost
		MUTEMIC=1	Mic Input Mute to ADC
		INSEL=0	Microphone/Line Input Select to ADC
		BYPASS=0	Bypass Switch
		DACSEL=1	DAC Select
		SIDETONE=0	Side Tone Switch
		SIDEATT[1:0]=00	Side Tone Attenuation=-6dB

4. táblázat. Analóg be- és kimeneti tulajdonságok beállítását végző regiszter

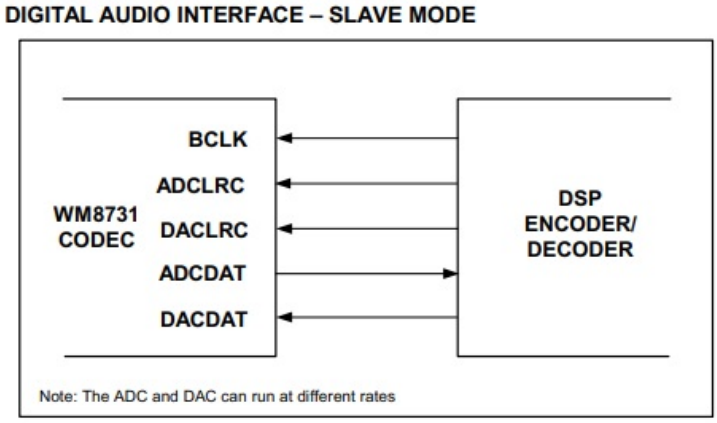
Ezen kívül lehetőségünk van az egyes regisztereket resetelni, ehhez annyit kell tenni, hogy a '0001111' címre elküljük a resetelni kívánt regiszter címét és még néhány más beállítás, mint például az aktív vezérlés és a kimenet beállítása.

Register		DATA set	
Name	Address	Label	Description
ACTIVE CONTROL	7'b0001001	active=1	
DIGITAL AUDIO PATH CONTROL	7'b0000101	data=00000	
RIGHT HEADPHONE OUT	7'b0000011	data=101111001	
RESET REGISTERS	7'b0001111	Megadott című regisztert resetelem	

5. táblázat. Egyéb beállításokat végző regsizterek

4.5.3. Amplitúdóadatok küldése a WM8731 Audió codec-nek

Miután az átalakító beállíása megtörtént az adatok küldése egy I2S kommunikációs buszon történik. Ezeket az adatokat egymás után elküldjük, amelyeket átalakít a DAC analóg jellé. A következő ábrán az I2S interfésze látható.



4.10. ábra. WM8731 codec I2S interfésze [5]

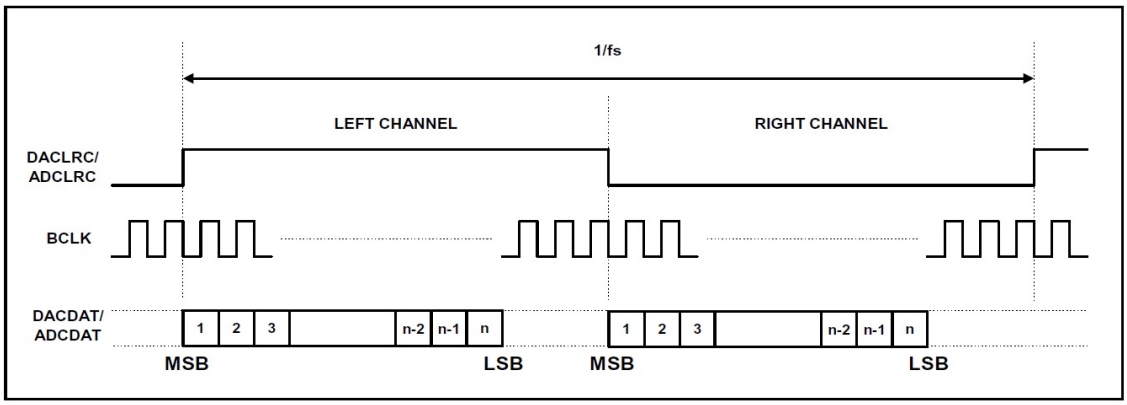
A pontos adatformátum és a küldési mód a kodek beállításától függ. A DAC megfelelő kimeneti jelet szolgáltat egy jack csatlakozón keresztül. Ezután nincs más dolgunk csak mérni az adott csatlakozón a kimeneti jelet.

A BLCK jel az egyes adatbitek átviteléért felelős. Ez egy 12MHz-es órajel, amit a bementi 50MHz-es órajeléből állítottam elő.

A DACDAT értelemszerűen az átalakítani kívánt jelünk egyes adatainak egyes bitjei.

A DACLRC jel a mintavételi frekvenciát jelöli, tehát ez a jel jelzi nekünk, hogy a kívánt adatot milyen idősávban vihetjük át. Egy 1/fs hosszúságú időrés alatt van lehetőségünk a hullámforma egyes adatainak átvitelére. Mivel ez az audió kodek használható stereo adatok átviteléré és átalakítására, ezért a konfigurációtól függ, hogy mono vagy stereo hangot viszünk-e át. Egy stereo hangnál egy mintavételi periódusban egy bal és jobb csatornás adatot is átviszünk, itt a DACLRC jel változása jelzi az egyes csatornák átvitelét. Egy mono hangnál mindkét csatornán ugyanazt az adatot visszük át és a beállítástól függően a codec eldönti mit csinál az adattal. Azonban lehetőség van olyan beállításra, hogy csak egyetlen csatornát viszünk át. Az én esetemben ez utóbbi volt kézenfekvő.

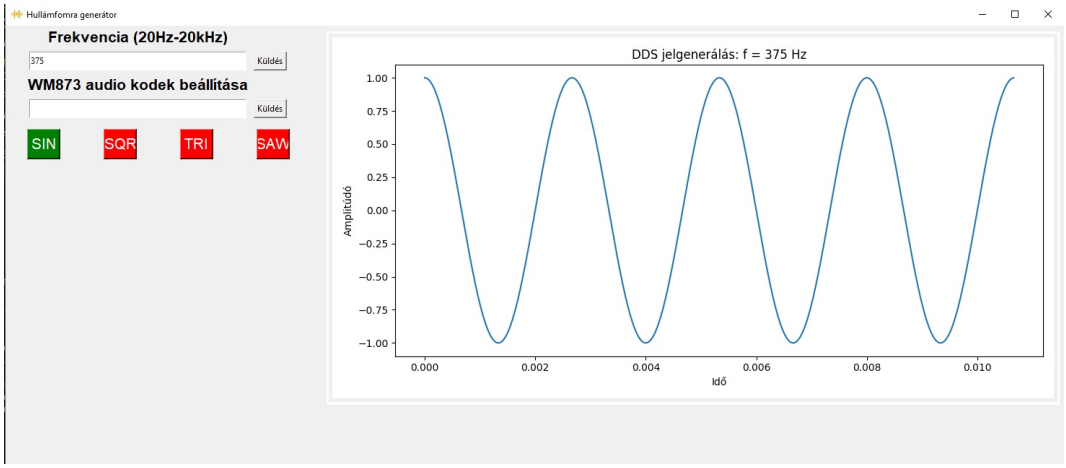
Az adatátviteli protokoll a következő ábrán látható:



4.11. ábra. I2S kétcsatornás (stereo) adatátvitel [5]

4.6. Szoftver

A szoftvert Python nyelven írtam. Ennek egyik része egy GUI (Graphical User Interface) réteg, ahol azokat az adatokat a felhasználó meg tudja adni, amivel szeretné vezérelni az generátort. Ez, mint látható két részből tevődik össze, az ablak bal oldalán megadhatjuk azokat az adatokat, amiket szeretnénk beállítani. Itt megadhatjuk a DDS generátor frekvenciáját, amelyet csak 20Hz és 20kHZ között lehet megadni, ha nem ebben a tartományban adjuk meg, akkor a rendszer hibát jelez. Továbbá a digitális-analóg átalakító beállításait elvégző digitális szó megadása is ebben az ablakban lehetséges. Ez vár egy 16 bites bináris számot és ezt küldi el a megfelelő regiszterfájlba. A rendszer elindítása után a konfigúráció ugyan autómatikusan megtörténik, azonban itt lehetőség van a beállításokat módosítani és resetelni is. A felhasználónak lehetősége van kiválasztani hullámformákat és azt feltölteni a rendszer memóriájába.



4.12. ábra. Felhasználói Interfész

Az ablak jobb oldalán a kimeneti hullámformát láthatjuk, ha a rendszerünk jól működik ugyanez jelenik meg a generátor kimenetén is, amit akár oszcilloszkóppal meg is tudunk mérni. A szoftver soros porton keresztül kommunikál az FPGA-val. A GUI megírása a tkinter könyvtár segítségével történt. A hullámforma adatok generálása során először egy n elemű vektort állítottam elő mely tartalmazza az adatok sorszámát. Ez a vektor 14 bitmélységnek megfelelő értékű, vagyis 4096 adatot jelent. A mintavételezési frekvencia 96kHz, ebből a mintavételezési idő könnyen kiszámítható:

$$dt = 1/fs = 1/96kHz = 10.42\mu s$$

Ez azt jelenti, hogy a szinusz hullámforma $4096 * 10.42\mu s = 42,680ms$ időtartamú. Ezután eltárolom az egyes mintavételezési időpontokat, melyet úgy teszek, hogy ezt a mintavételi időtartamot megszorzom az n elemű vektor egyes vektorelemeivel. Így létrehoztuk az egyes mintavételi időpontok vektorát:

$$xt = nt * dt$$

A következő a szinusz jel létrehozása. Itt gyakorlatilag egy fázisakkumulálás hajtódik végre, mivel a szinuszos függvény:

$$yt = np.cos(2 * np.pi * f * xt), \text{ ahol}$$

- f az alap frekvenciafelbontás
- xt a mintavételezési időpontok vektora
- np olyan függvény, mely hatására egy függvény minden egyes vektorelemre külön hajtódik végre

Ez azonban egy *int* típusú vektor, melyet át kell alakítani, hogy az FPGA számára is értelmezhető legyen. Az FPGA block ram-jába 8 bites kettes komplemens számokat szeretnénk tárolni. Ezért a következőképpen tudjuk a vektort átalakítani:

$$sin\_wave = (sin\_wave\_int * 127).astype(np.int8).$$

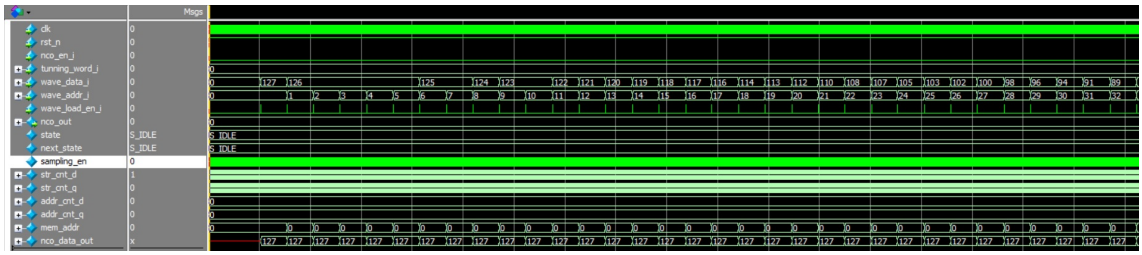
Ez csak egy példa volt egy szinuszos jel generálására, melyen keresztül bemutatom a rendszer működését, azonban a felhasználónak lehetősége van bármilyen általános periodikus hullámformát generálni.

5. Verifikáció és szintézis

Az RTL terv elkészülte után szükséges a rendszer szimulációja, hogy meggyőződjünk a megfelelő működésről mielőtt valós környezetben használnánk az adott áramkört.

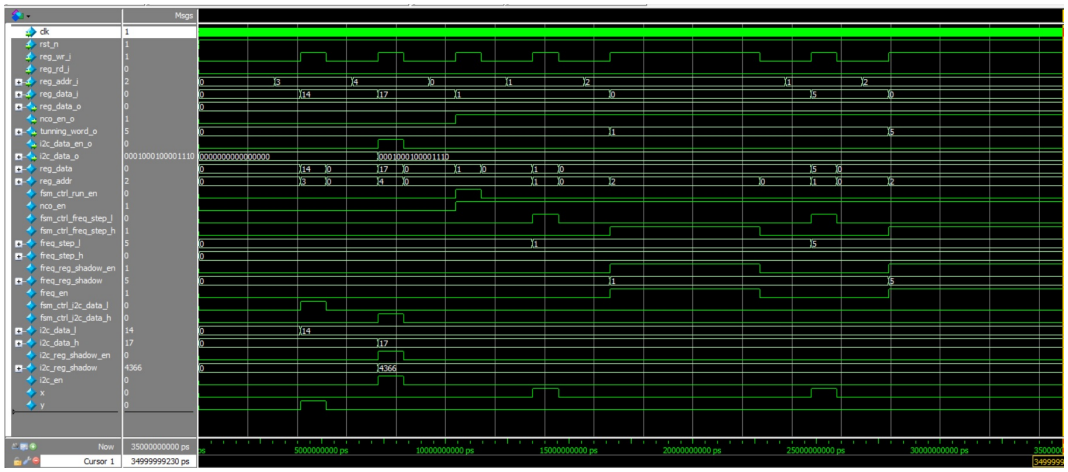
5.1. Szimuláció

A rendszer tesztelésekor az volt a fő szempont, hogy megnézzük az általunk beírt adatok bekerültek e a memóriába, vagy a regiszter fájlokba és hogy ezek alapján a kimeneti hullámformát pontosan létre tudják-e hozni? A szimulációs tesztet Systemverilog nyelven terveztem és a ModelSim programmal szimuláltam. Ennek eredménye a következő:



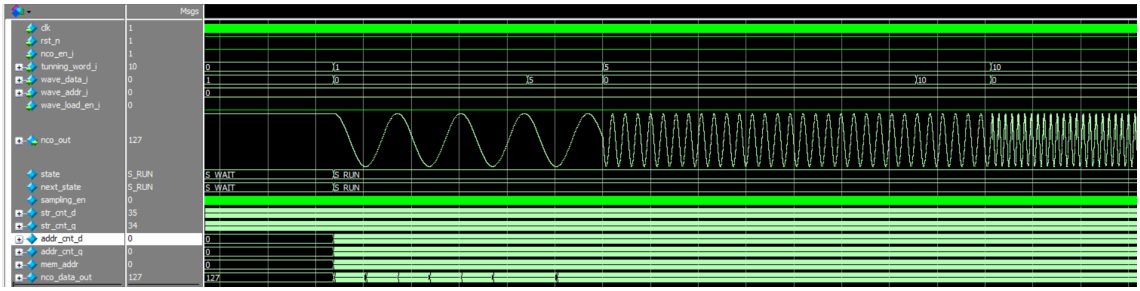
5.1. ábra. ModelSim szimuláció. Adatok betöltése a memóriába.

Itt látható, hogy az adatok betöltése sikeres volt. A szimulációt 256 adaton végeztem el, mivel a szimuláció lassan futott le. Azonban ez nem befolyásolja a valódi működést.



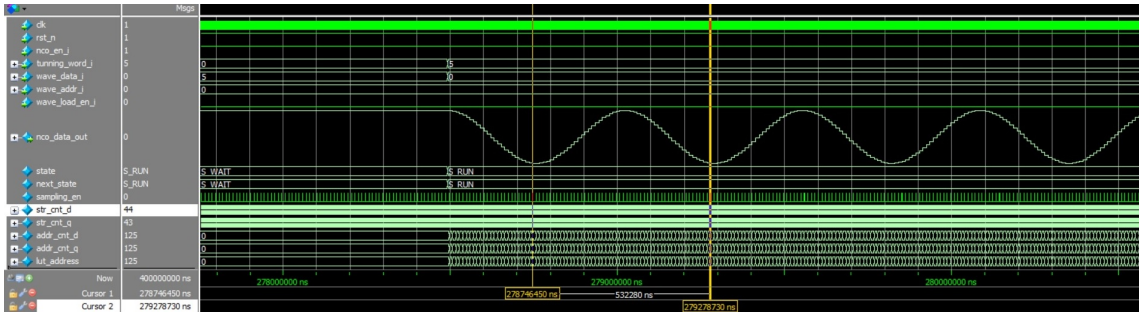
5.2. ábra. ModelSim szimuláció. Adatok írása a regiszter fájlba.

Itt látható, hogy a regiszter fájlokba történő írás is sikeres volt, először az egyes állapotok váltják egymást, amint új adat érkezik be és $'rx\_done\_i'$ jel is kiad egy impulzust, mely azt jelzi az aktuális adat olvasása befejeződött. Továbbá látható, hogy az adott memóriacímű regiszterbe a helyes adat került. A frekvenciák összefésülése is helyesen működött. A szimuláióhoz az "automatic analog" funkciót használtam, mely átalakítja a digitális adatunkat analóg hullámformává. Először az alap Frekvencialépést adtuk meg a rendszernek, ez 1-et jelent, ami a legkisebb frekvenciát eredményezte, vagyis a felbontást. Ez 256 adatra számolva 375Hz frekvenciát jelent.



5.3. ábra. ModelSim szimuláció. A hullámforma szimulációja.

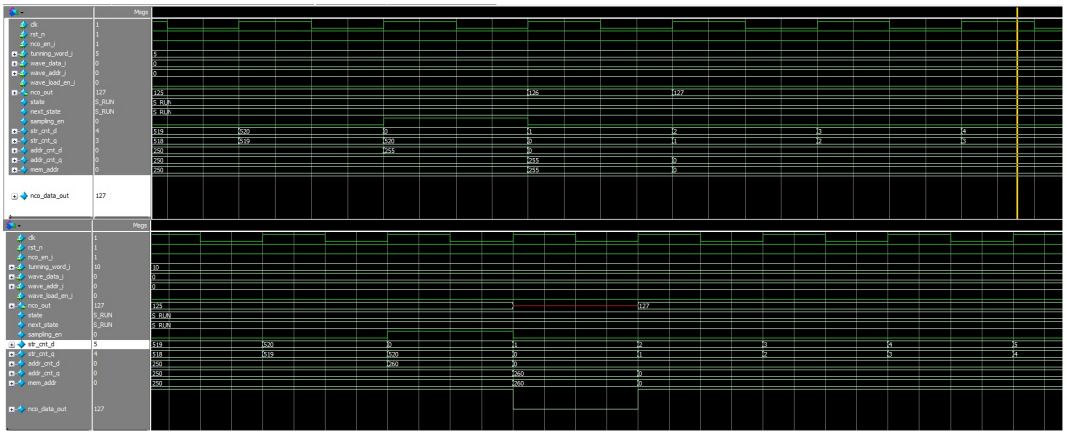
Ezután megváltoztattuk a frekvencialépést 5-ös értékre, ebből könnyen kiszámolható a már ismert összefüggésből, hogy a kívánt frekvencia 1875Hz. A következő ábrán megmérve ez ki is jön.



5.4. ábra. ModelSim szimuláció.Periódusidő mérése.

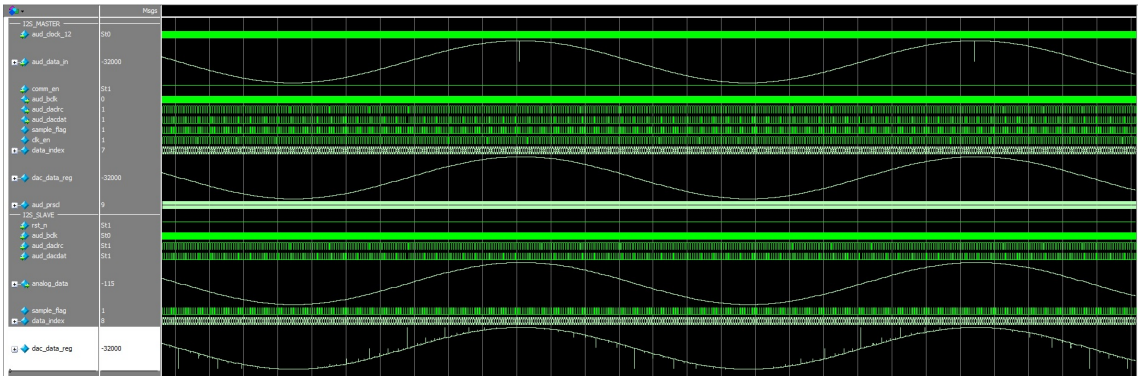
A periódus időt megmérve és átszámítva látható, hogy a hullámforma megfelel az általunk vártnak. Később egy újabb frekvencialépést (10) írunk a regiszterbe, ezáltal megint megváltozik a frekvencia. Itt egy 3750Hz frekvenciájú jelet látunk, ezt megmérve megintcsak mefelel a megfelelő működésnek a rendszerünk.

A 5.5 szimulációs ábrán látható, hogy a jelben glitch-ek jelennek meg bizonyos helyeken. Az is látszik, hogy a glitch helyén a kimeneten határozatlan állapot jelenik meg. Ennek az az oka, hogy a memória túlcsordul, mivel a frekvencialépésünk jelen esetben 10 és az aktuális értékhez, ami 250, ez adódik hozzá. Tehát a 250-es memóriacímet címezzük, viszont ennek nincs értéke, mivel a memórián csak 256 mélységű. Viszont a túlcsordulás vizsgálata során kapunk egy mintavételt engedélyező jelet, tehát rossz címről veszünk mintát, vagyis határozatlan érték jelenik meg a kimeneten. Ezt úgy tudjuk javítani, ha megvizsgáljuk, hogy az adott memóriacím-hez hozzáadva a frekvencialépés értékét nagyobb értéket kapunk e, mint a memória nagysága. A modellt javítva így már glitch mentes digitális jelet kapunk. Azonban fontos megjegyezni, hogy az analóg jelünkben megjelenhetnek glitch-ek, az átalakító hibája, pontatlansága vagy más egyéb hibák és tényezők miatt.



5.5. ábra. ModelSim szimuláció. Glitch és glitch nélküli mintavétel.

A WM8731 I2S protokolljának szimulációjához szükséges volt létrehozni egy 12MHz frekvenciájú órajelet. Ezt a szimuláció során külön hoztam létre a testbench-ben, mivel a PLL kevert jelű (analóg és digitális) így ezt az egyszerűbb megoldást választottam. Azonban a Quartus II fejlesztői környezet lehetővé teszi, hogy létrehozzunk PLL modult és beilleszthessük a szimulációnkba.



5.6. ábra. ModelSim szimuláció. I2S adatátvitel szimulációja.

A szimulációs hullámkformán a felső részben az FPGA I2S kimeneti interfésze látható, alul a WM8731 bemenete és a hullámforma adatainak fogadása. A $'dac\_data\_reg'$ nevű shift regiszterbe gyűjtjük a bemenő adatokat. Ennek a regiszternek a kimenete "szőrös", mivel a jelek itt nem megfelelő módon/időben vannak mintavételezve. Mivel még nem dolgoztuk fel és fűztük össze az egyes amplitúdóadatok összes bitjét. A $'sample\_flag'$ jelzi, hogy az adat teljes egészében beérkezett, ekkor már a kimenetre köthetjük az adatot és így az $'analog\_data'$ regiszter értékeit megjelenítve kiadódik a megfelelő hullámforma. Itt a megfelelő mintavételezésnek hála már a glitch-ek sem jelennek meg a kimeneten.

5.2. Szintézis folyamata

A szintézis során a QUARTUS II szintézis tool-ját használtam. Fontos kihangsúlyozni, hogy az FPGA block ram memóriájaként szerettem volna szintetizálni an nco adattábláját. Az adatokat a táblába a szintézis során szerettem volna betölteni, hogy már a kiindulási (alap) állapotban is tartalmazzon "használható" hullámforma adatot. Ezt egy .mif (memory initialization file) kiterjesztésű fájl segítségével tettem meg. A fájlnak tartalmazni a kell a memória mélységét és szélességét, továbbá, hogy milyen formátumban kívánom beírni az adatokat a memóriába és hogy a modellben mi a neve ennek a memó-

riának (content). Ennek a fájlnak a tartalma a következő módon néz ki:

DEPTH = 4096;

WIDTH = 8;

ADDRESS_RADIX = HEX;

DATA_RADIX = HEX;

CONTENT

BEGIN

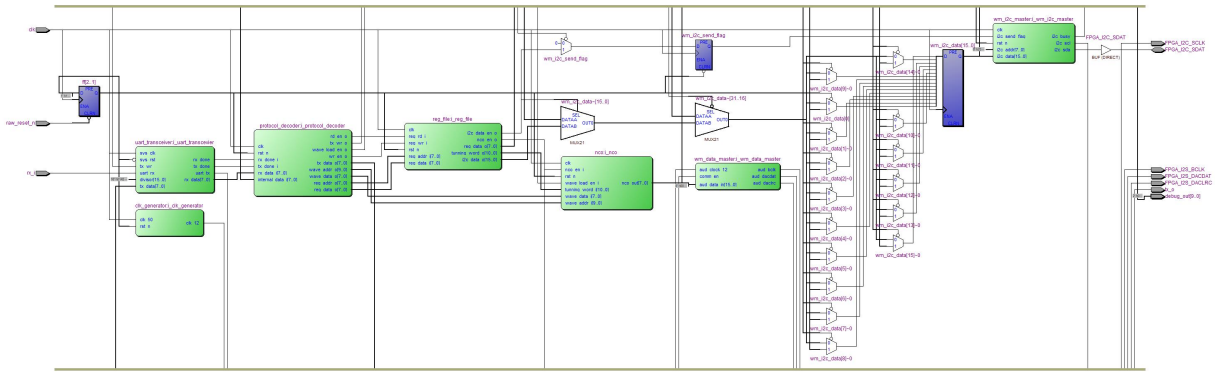
0 : 7F;

1 : 7E;

⋮

FFF : 7E;

Mivel a memória RTL modelljét úgy írtam meg, hogy bram-ként szintetizálja a szintézis tool, ezért ezzel már ne volt külön dolgom. A .mif fájl hozzá kellett adni a projekthez és beállítani, hogy a bram-ba töltse be ezeket az adatokat. Azonban a szintézis sikertelen volt és azt a "170012" hibaüzenetet kaptam, ami annyit jelent, hogy a projekt túl nagy vagy túl bonyolult és nem lehet megvalósítani azon az adott céleszközön. Emiatt leredukáltam a memória méretét, így már csak 1024 adatot tudtam eltárolni a memóriában. Viszont így a szintézis sikeresen zárult.



5.7. ábra. Hullámforma generátor szintézis eredménye.

6. Összefoglalás

A szakdolgozatom során alaposan tanulmányoztam a Közvetlen Digitális Szintézis (DDS) elvét és az ezen elv alapján működő jelgenerátorok jellemzőit. Ezután megterveztem majjd létrehoztam a saját generátorom RTL modelljét. A modellt szimulációval teszteltem, hogy ellenőrizzem a tervezés helyességét és a kívánt funkciók megvalósulását. A sikeres szimuláció alátámasztotta a generátor működőképességét, a modell hibamentesen előállította a tesztelésre szánt szinuszos hullámformát különböző frekvenciákon, és a frekvencia változtatása gyors és folyamatos volt. A következő lépésben elvégeztem a szintézist, amely során az RTL leírást konvertáltam a konkrét FPGA architektúrához illeszkedő konfigurációs bitfájllá. A szintézis eredményeként létrejött a kívánt hullámforma generátor hardverimplementációja.

A bemutatott szakdolgozat részletesen ismerteti a digitális hullámforma generátor tervezési folyamatát, a szimuláció eredményeit és a szintézis lépéseit. Emellett részletesen tárgyaltam a használt DDS elvét és a generátor konfigurálásának lehetőségeit a kívánt periodikus jelek létrehozására és finomhangolására. A szakdolgozatban bemutatott digitális hullámforma generátor FPGA környezetben történő implementációja számos alkalmazásban hasznos lehet, például audió jelgenerálásban, tesztelési célokra vagy adatkommunikációs rendszerekben. A tervezett generátor megfelelően rugalmas és pontos hullámformák előállítására nyújt lehetőséget, amelyek az adott felhasználói igényeket kielégítik.

Irodalomjegyzék

[1] Field Applications Engineer Patrick Butler. An almost pure dds sine wave tone generator. *Analog Devices, Inc*, (TA21719-12/19), 2019.

[2] By Eva Murphy and Colm Slattery. Ask the application engineer—33 all about direct digital synthesis. *Analog Dialogue*, 38-08, August (2004).

[3] Bill Schweber. A technical tutorial on digital signal synthesis. *Analog Devices*, 1999.

[4] Elektronika 2. Analóg-digitális átalakítók. `http://docplayer.hu/26477933-Analog-digitalis-atalakitok-elektronika_2.html`. Accessed: 2023-04-21.

[5] Wolfson Microelectronics plc. *WM8731 / WM8731L*, rev 4.0 edition, February 2005.

[6] Marie Christiano. Everything you need to know about direct digital synthesis. *All About Circuits - Technical Article*, page Figure 3., November 20, 2015.

[7] Denis Smetana. This matters: Spurious free dynamic range (sfdr). *Curtiss-Wright Blog*, April 13, 2015.

[8] ethawicker. What is clock jitter, why does it matter, and what can you do to minimize it? `https://community.silabs.com/s/share/a5U1M000000knrEUAQ/what-is-clock-jitter-why-does-it-matter-and-what-can-you-do-to-minimize-it?language=de`. Accessed: 2023-03-12.

[9] BONNIE BAKER. Dac basics, part 4: The pesky dac output glitch-impulse. *Planet Analog*, APRIL 24, 2015.

[10] Pápay Zsolt. Dds technológia. `http://www.hit.bme.hu/~papay/sci/DDS/start.htm`. 2000.szeptember.

[11] A/d és d/a átalakítók vizsgálata. *BME VIK Laboratórium 2.*, 7.mérés. 7-1. ábra. Unipoláris A/D átalakító statikus karakterisztikája.

[12] Dr.Horváth Péter. *Digitális rendszerek modellezése és szintézise.* L' HARMATTAN KFT., 2021.

[13] Sebastien Bourdeauducq and (2007-2010) Das Labor. Milkymist vj soc. Hozzáférés dátuma: 2021.September.

[14] *TERASIC CYCLONE II EP2C35 Development and Education BOARD Datasheet.*

[15] AntonZero. Wm8731-audio-codec-on-de10standard-fpga-board. `https://github.com/AntonZero/WM8731-Audio-codec-on-DE10Standard-FPGA-board.git`.

[16] FastBitLab. Stm32 i2c lecture 3 : I2c protocol explanation. *Blog*, August 5, 2019.