Jelgenerátor

1. Bevezetés

Az újra konfigurálható digitális áramkörök tantárgyon belül az FPGA áramkörről tanultunk.

Az FPGA konfigurálható logikai komponenseket és programozható összeköttetéseket tartalmaz és sokféle alkalmazásban használják. Például: videó és képprocesszálás, repülésvédelem, orvosi elektronika, biztonsági rendszerek stb.

A Vivado egy rendszerközpontú tervezési környezet, amely felgyorsítja a tervezés folyamatot újra konfigurálható digitális áramkörök esetén. A rendszert a XILINX fejlesztette ki. Lehetőség van

magas szintű rendszer-integrációra, rendszerszintű IP csomagolásra és telepítésre és még sok más hasonló tevékenységre. A Vivado segítségével tudjuk a rendszereket implementálni és szimulálni is. A tantárgyon belül a projekt egy jelgenerátor megalkotása volt, a jeleket SPI vezérlőn keresztül egy PmodAD2 digitál-analóg átalakítón keresztül küldtük ki.

A projekt elérhető: https://github.com/hammasattila/Sapientia-FPGA2020-Project GitHub oldalon.

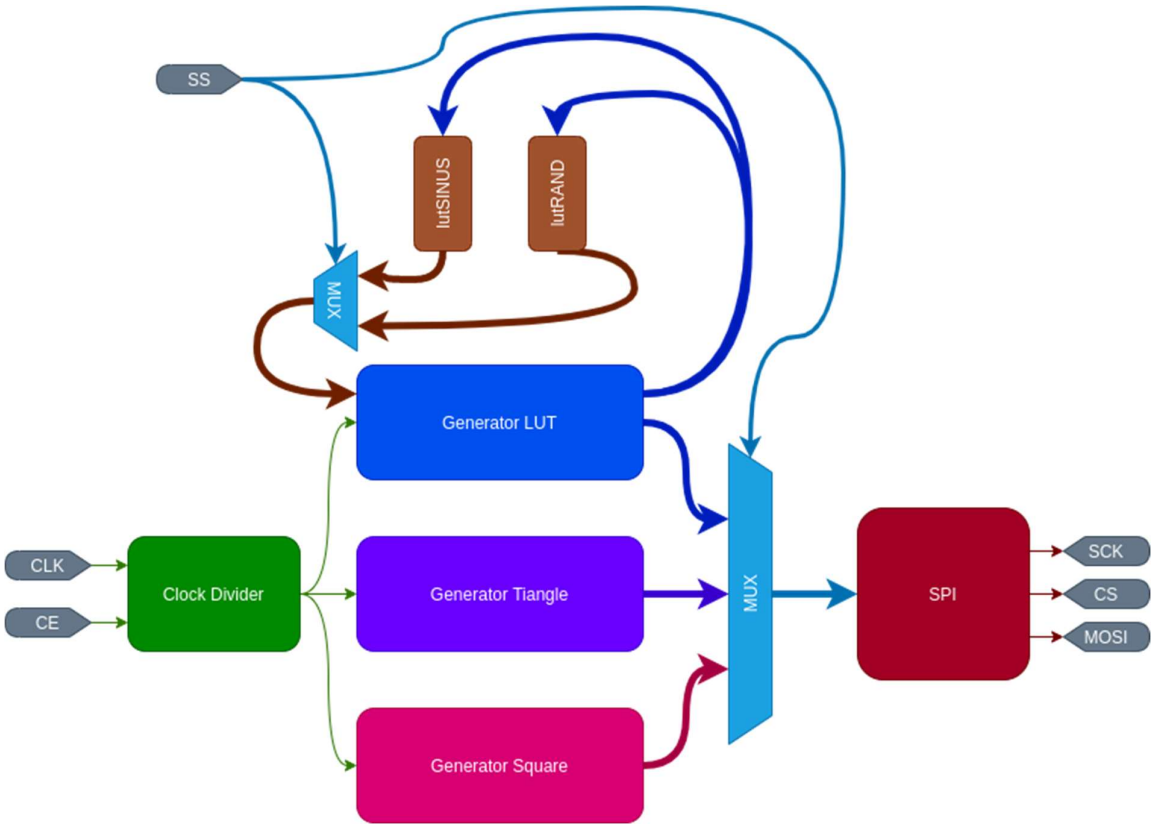
2. Követelmények

A projekt követelményei a következők:

1. A rendszer több típusú jelt tudjon generálni (sinus, háromszög, fűrészfog, négyszög, random)
2. A jelek paramétereit könnyeden tudjuk vétóztatni (frekvencia/periódus, amplitúdó, eltolás, sebesség stb.).

3. Megvalósítás

Amint már említettem a feladat egy jelgenerátor megalkotása volt. Az alábbi látható a feladatot megalkotó egyszerű tömbvázlat.

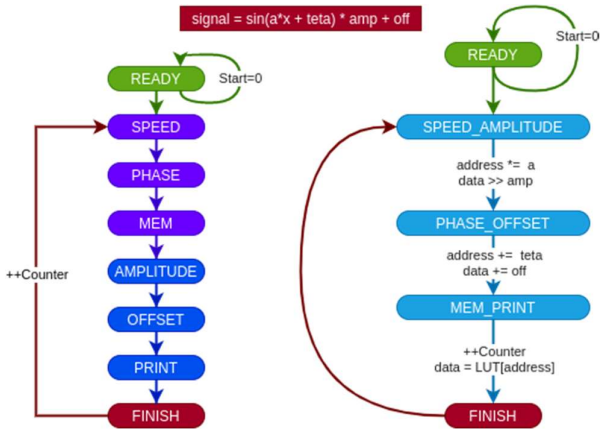


a. Jelgenerátorok

Összesen 5 típusú jelet sikerült generálni: sinus/cosinus, random, háromszög, fűrészfog, négyszög. Ezt 3 jelgenerátor modul segítségével valósítottam meg, amint a tömbvázlatban is látható volt.

I. Kereső táblás generátor

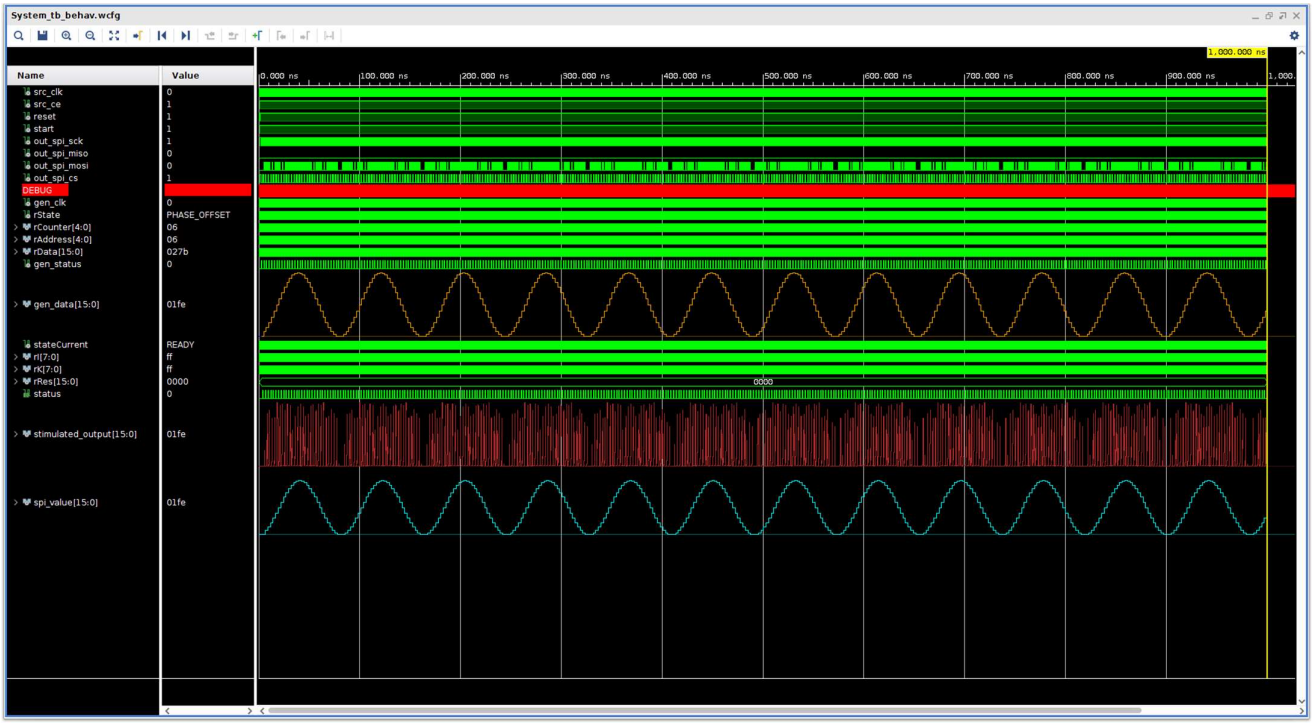
Ez a modul a sin/cos valamint a random jelt képes generálni. Illetve, ha egy másik kereső táblát generálunk, akkor bármilyen más jelet is képes generálni. A következő ábrán látható a véges állapot automata. Észrevehető hogy van egy címmel dolgozó rész és egy adattal dolgozó rész ezért az automata állapotait lecsökkenthessük a jobboldali automatára, ezzel egy hibrid architektúrát kapunk ahol van egy „pipeline", amibe két szekvenciális automata van egymás után.



Az automata értéktáblázata:

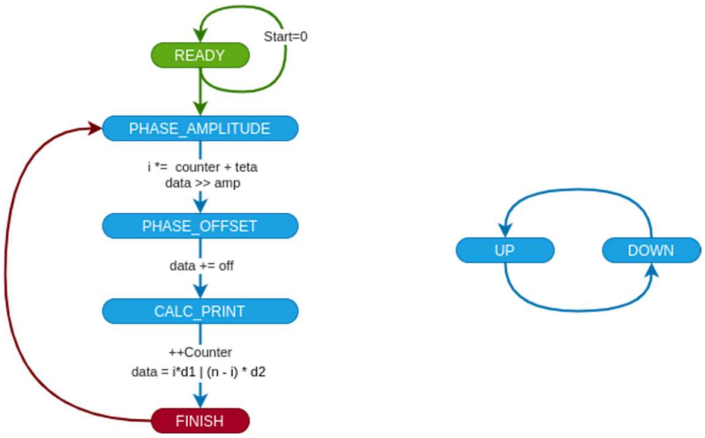
	counter	i	data	addr	status	dout
READY	0	0	0	0	0	X
SPEED_AMPLITUDE	counter	counter * a	data >> amp	0	0	X
PHASE_OFFSET	counter	i + teta	data + off	0	0	X
MEM_PRINT	counter + 1 | 0	i	val	i	0	data
FINISH	counter	i	data	0	1	data

A következő ábrán látható a egy teszteset, ahol a sárga jel az automata kimenete.



II. Lépcsőzetes generátor

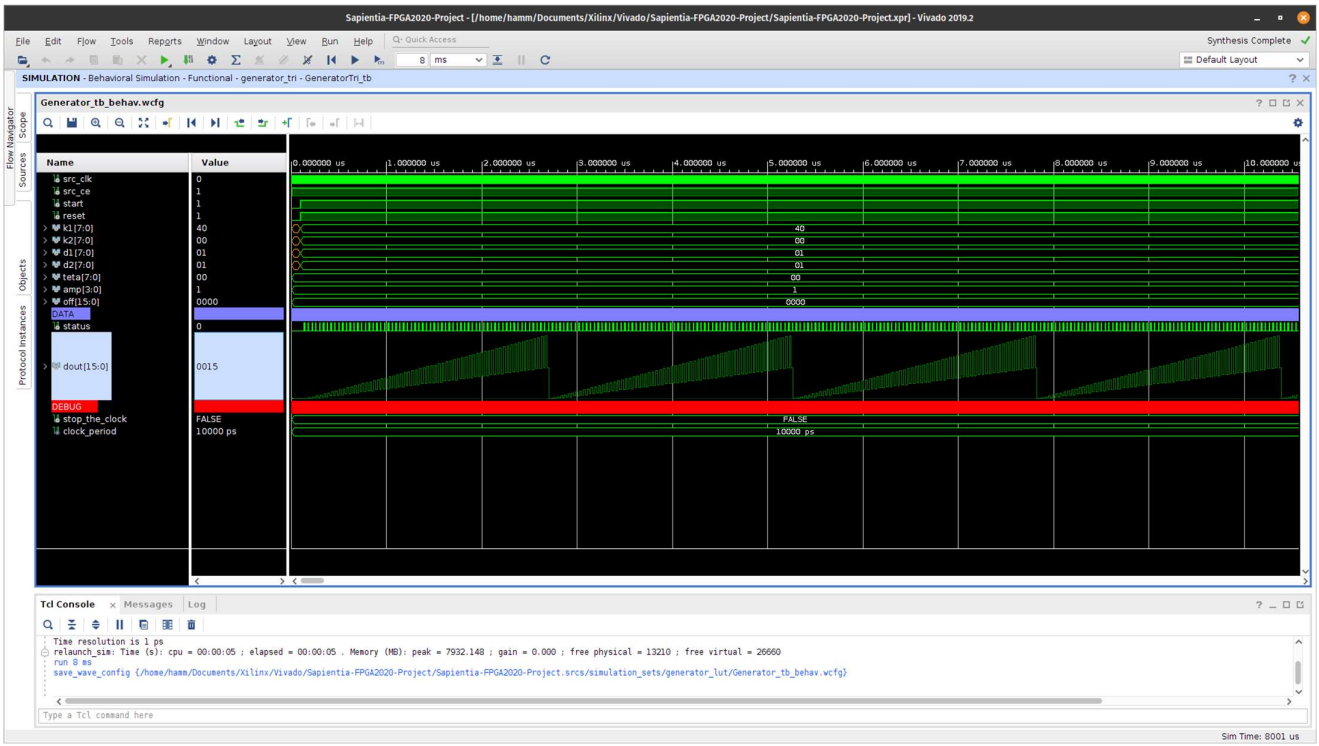
Ez a modul a háromszög és a fűrészfog jelért felelős. Itt is a már fent említett módszer szerint az állapotok száma le van csökkentve és egy hasónló hibrid architektúrát kapunk. Két automatát használunk egyik a folyamat fázisát jelképezi a másik a maga az automata.



Egyes pontokon a folyamat fázisa szerint elágazik az út. Lényegében a fázis automata a fő automata bemenete.

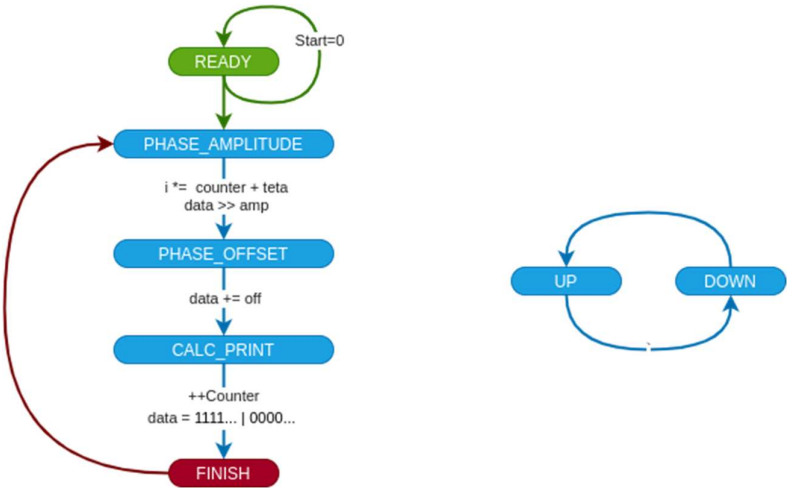
	counter	i	data	status	dout
READY	0	0	0	0	X
PHASE_AMPLITUDE	counter	counter + teta	data >> amp	0	X
PHASE_OFFSET	counter	i	data + off	0	X
CALC_PRINT	counter + 1 | 0	i	i*d1 | (n - i) * d2	0	data
FINISH	counter	i	data	1	data

A következő ábrán látható egy teszteset, ahol a modul egy fűrészfog jelet generál.



III. Négyszög generátor

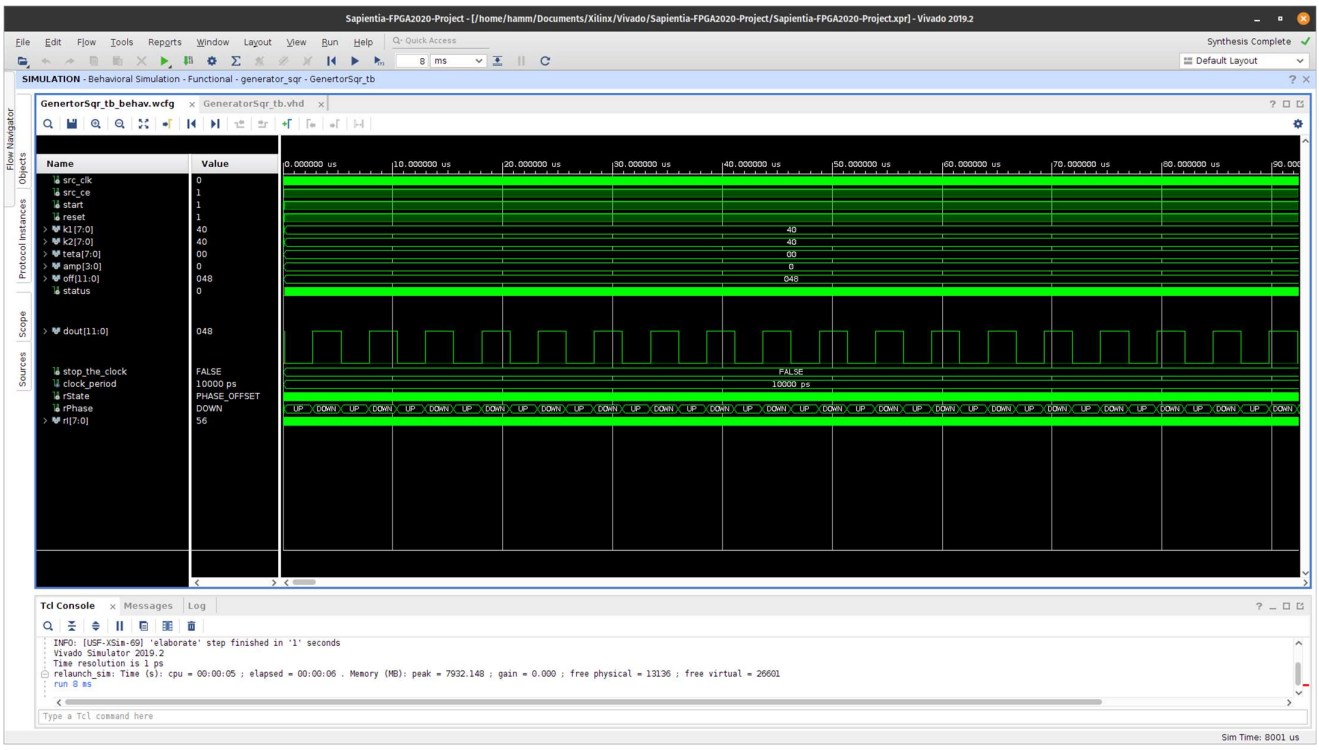
Ez a modul a háromszög és a fűrészfog jelért felelős. Itt is a már fent említett módszer szerint az állapotok száma le van csökkentve és egy hasónló hibrid architektúrát kapunk. Két automatát használunk egyik a folyamat fázisát jelképezi a másik a maga az automata.



Egyes pontokon a folyamat fázisa szerint elágazik az út. Lényegében a fázis automata a fő automata bemenete.

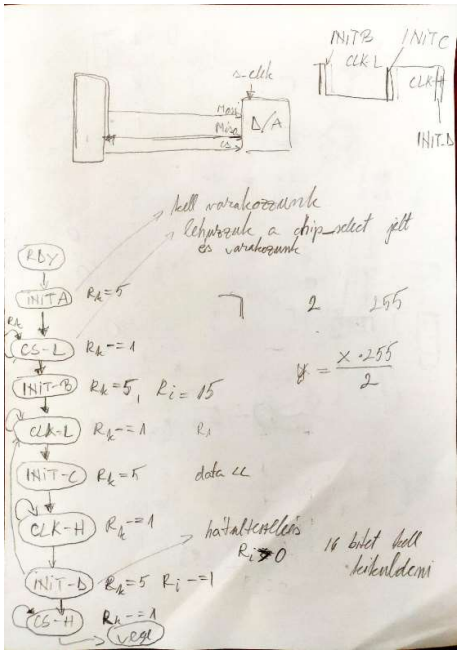
	counter	i	data	status	dout
READY	0	0	0	0	X
PHASE_AMPLITUDE	counter	counter + teta	data >> amp	0	X
PHASE_OFFSET	counter	i	data + off	0	X
CALC_PRINT	counter + 1 | 0	i	1111… | 0000…	0	data
FINISH	counter	i	data	1	data

Az alábbi teszteset az egy általános négyszögnél.

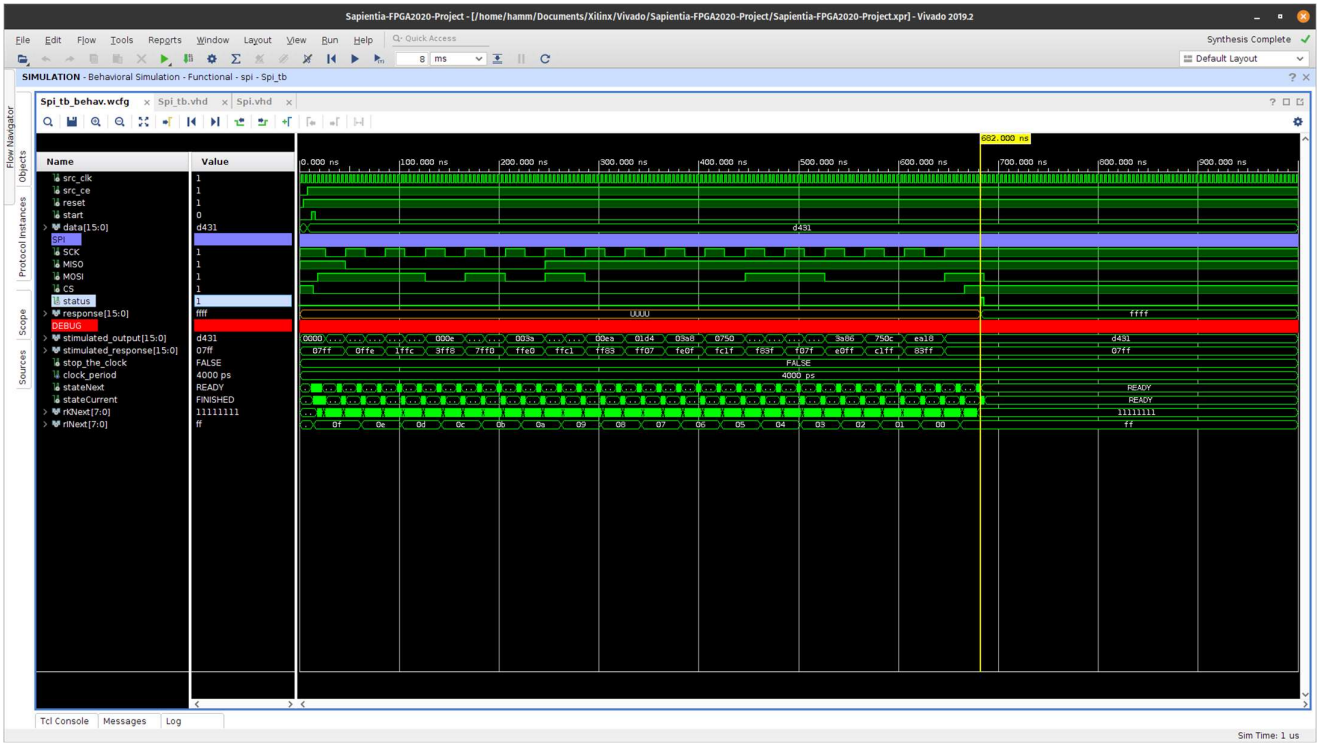


b. SPI

Az SPI modul azért felelős, hogy az adatot az SPI protokollon keresztül átadja a PmodDA2 digitál-analog konverternek. Az alábbi képen látható egy tömbvázlat, az állapotautomata és egy példa diagram.

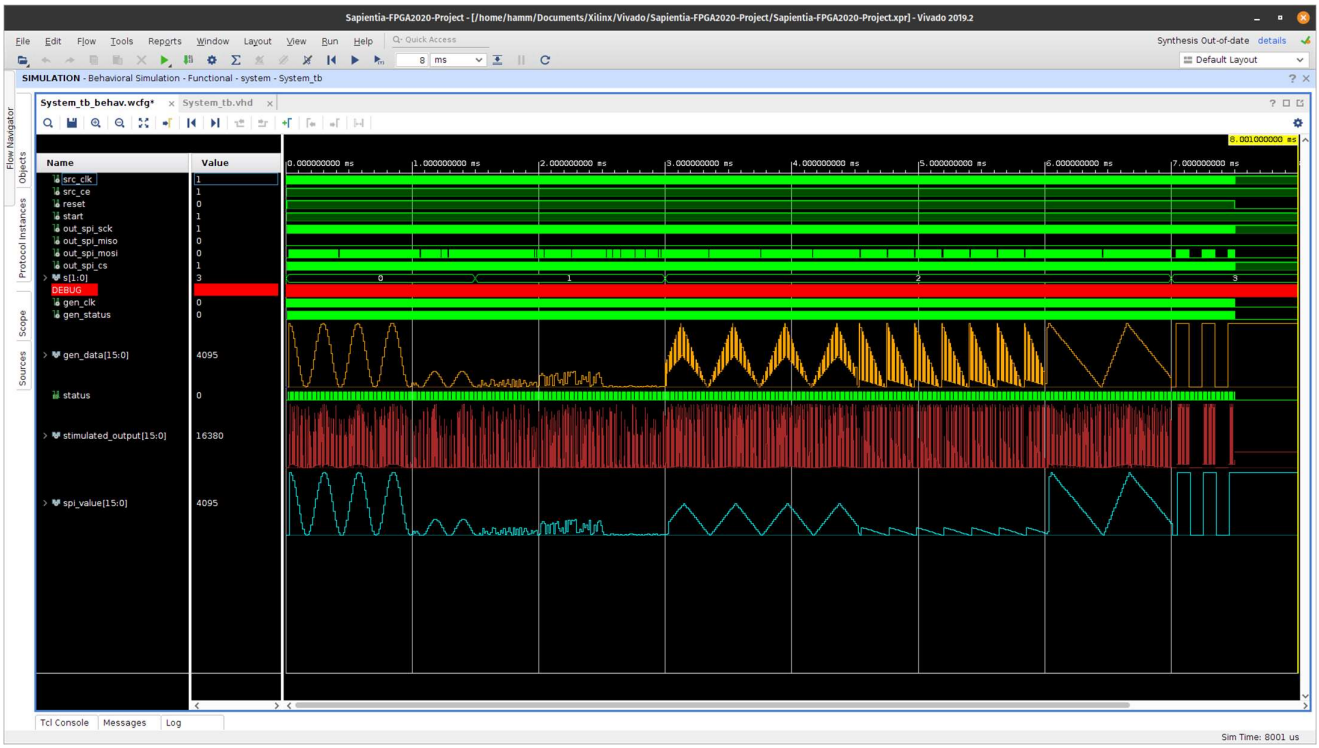


Az alábbi tesztesetben látható egy példa kimenet. Megfigyelhető hogy a „data” bemenet megegyezik az SPI kimenetével amit dekódoltunk a simulated_output jelen, és mindkettő „d431”.



c. Rendszer

A rendszer moduljait sikeresen össze lehetet kötni és szinkronizálni. Az alábbi szimuláción látszik, hogy megfelel a követelményeknek, tudja az elvárt jeleket generálni, az eltolást, sebességet, amplitúdót és a fázist futás közben tudjuk módosítani probléma nélkül. Valamint a jelek között is tudunk váltani az „ss” bemenet segítségével.



A dekódolt kimeneten látható, hogy a jel az tiszta és folytonos.

4. Következtetések

A rendszer az tartalmaz néhány potenciális hiányosságot, valamint továbbfejlesztési lehetőséget:

- Nincs túlcsordulás védelem ezért, ha túl csordul az adat a paraméterek miatt egy hibás jelet kaphatunk a kimeneten.
- A négyszögjel és a háromszögjel generátor nagyon hasonló ezért össze lehetne őket vonni, mint a sin és a random jelgenerátort ezzel helyet spórolnánk az FPGA-n.
- A háromszög jelen lehetne matematikailag optimalizálni.
- A sin keresőtáblát lehetne csak 1 negyedre felépíteni így egy sokkal kisseb táblát kapnánk, vagy sokkal pontosabb táblát ugyanakkora méretben.