0.1 Arbejdsblad

Hej Søren

Lidt respons på det tilsendte ville være rart.

AALBORG UNIVERSITET
STUDENTERRAPPORT

Andet Studieår v/ Det Teknisk-Naturvidenskabelige Fakultet
Elektronik og IT
Frederiks Bajers Vej 7B
9200 Aalborg
http://www.tnb.aau.dk

Titel:

portnoykasse
med minimeret effektforbrug

Synopsis:

Tema:

Analoge elektroniske kredsløb og systemer

Projektperiode:

Februar 2017 - Maj 2017

Projektgruppe:

Gr. 411

Deltagere:

Mathias L. Kjeldgaard
Christiano Rosnaldo
Rasmus B. Bertelsen
Thomas D. Amgaard
Troels S. Andersen

Vejleder:

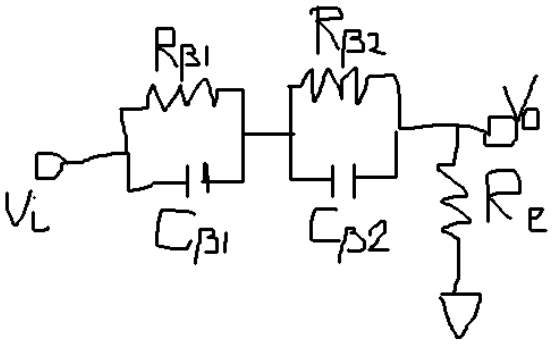
Søren Krarup Olesen

Sidetal: 79
Appendiks: 38
Afsluttet 21. December 2016

Indholdsfortegnelse

0.2 Sequencer

Som nævnt tidligere, er det nødvendigt for den endelige trommemaskine, at kunne holde en rytme og afspille lyde på et bestemt tidspunkt. For at styrer dette er der brug for nogle input og output. Med hensyn til input, skal dette delmodul styres af en række knapper, med forskellige funktioner. For det første skal det være muligt at vælge mellem forskellige sekvenser, som man gerne vil afspille. Derudover skal det være muligt at vælge mellem forskellige trommelyde, da det ellers vil være en forholdsvis kedelig trommemaskine. Yderligere skal der være knapper til at styre hastigheden, som en sekvens afspilles med. Til sidst skal der være knapper for de enkelte dele af sekvensen, så det kan bestemmes hvornår trommelyde skal afspilles.



Figur 0.1: Placeholder for blokdiagram

0.2.1 Kontrol af hastighed

Da der sidder en intern clock i Papilio'en bruges denne til at holde tempoet. Denne clock har en frekvens på 32 MHz og skal derfor sænkes i hastighed for at holde rytmen for trommemaskinen. Til at starte med tages der udgangspunkt i 100 BPM. Da det ønskes at sequenceren kan opfange lyd i realtid er der blevet undersøgt hvor præcist et signal skal værre, for at det passer med præcisionen for en trommeslager. Det er her blevet besluttet at 1 ms er passende præcist for dette. Det er derfor nødvendigt at beregne hvor lang tid det tager at afspille en sekvens.

$$\frac{60 \cdot 4}{100 \text{ min}} = 2{,}4\,\text{s} \tag{0.2.1}$$

Da det ønskes at trommemaskinen skal kunne gå ned til en hastighed på omkring 50 BPM udføres de følgende beregninger på 4,8 s. For at gøre det nemmere at programmere i VHDL, er der taget udgangspunkt i den nærmeste to'er potens, som i dette tilfælde er 4096. Derved kan præcisionen for et signal men 50 BPM beregnes:

$$\frac{4,8 \text{ s}}{4096} = 1,17 \text{ ms} \tag{0.2.2}$$

Det vil sige at når hastigheden for sequenceren er 50 BPM vil der være 1,17 ms mellem sekvensens dele, og dette antages at være tilstrækkeligt. For 60 BPM er dette krav opfyldt da der er 0,97 ms mellem de enkelte dele i sekvensen, hvilket medføre at højere hastigheder også vil opfylde kravet hertil. Da der bruges den samme opløsning for højere hastigheder, vil dette blot øge præcisionen.

For et få en clock til at passe med de ønskede BPM, laves en tæller, som skifter et signal når der nås til at bestemt tal. Dette signal bruges så som clock i en anden process. Ud fra de tidligere oplysninger kan det nu beregnes hvad der skal tælles til, ved en hastighed på 100 BPM. Dette medføre naturligvis at der skal tælles til det dobbelte for 50 BPM og det halve for 200 BPM. For at beregne hvad der skal tælles til tages følgende formel i brug:

$$\frac{32\ \text{MHz} \cdot 2,4\ \text{s}}{4096} = 18750 \tag{0.2.3}$$

For at opsummere strukturen for sequenceren tages der først udgangspunkt i den interne clock på 32 MHz, som tæller til 18750 hvorefter der stiftes et signal. Dette signal bruges som clock'en for en process, der skal hold styr på realtid, hvis man gerne vil holde rytmen selv. Processen der holder styr på realtid har igen en tæller, til at holde styr på hvor langt i processen man er. Når denne tæller når 4096 vil der være gennemgået en fuld sekvens og tælleren nulstilles. Hvis der ikke ønskes at bruge realtid kan tælleren for realtid stadig tages i brug, men der ses kun på de 4 MSB, da disse vil være en 16. del af sekvensen.

0.2.2 Datastruktur

Sequenceren har også til opgave at holde styr på de enkelte sekvenser for de forskellige trommelyde. For at holde styr på dette skal der brugen en række input fra kontrolmodulet. For det første skal der bruges et input der fortæller hvilken sekvens og hvilken tromme der er valgt. Derudover skal der bruges input fra de knapper, som holder styr på realtid og de knapper, som holder styr på trommelydene ved sekstendedele af en sekvens. For at tage højde for alle disse input er der brug for en datastruktur som set i eksemplet herunder:

Med denne datastruktur er det muligt at have et signal for den valgt sekvens og den valgt tromme. Ud fra disse signaler kan selve dataet så tilgås, både så den kan læses eller ændres. Da sekvensen, som sequneceren skal holde rytmen for, skal deles i sekstendedele, tages der blot udgangspunkt i en sekstendedel af 4096. Det vil sige at inputtet fra sequenceres rytmeknapper skal opdatere hver 256'ende bit i dataet. Her vil de enkelte knapper have hver deres tilhørende plads i bit array'et. Hvis der spilles i realtid skal der bruges et input der fortæller hvornår der bliver trykket på en trommeknap, og et tidspunkt hvor der bliver trykket på knappen på. Dette styres med den tidligere nævnte clock.

0.3 Brugerinterface

Hallo guys and welcome to user-interface description.txt - aka introduktion

Brugerinterfacet bliver delt op i x: step-sequencer indikator, step-sequencer knapper, takt indikator, tromme knapper, sequence behandling, mode select og BMP kontrol.

Step-sequencer indikator er en række af 16 LED'er, som lyser op når man trykker på tilsvarende step-sequencer knap. Disse LED'er har til formål at vise brugeren, når man har valgt at afspille en sample, på en specifik takt.

Step-sequencer knapper består af en række af 16 tryk-knapper, som sammen med step-sequencer LED'erne, bruges til at vælge og se hvornår man vælger at spille en sample på et specifikt tidspunkt i de 16 dele af en takt.

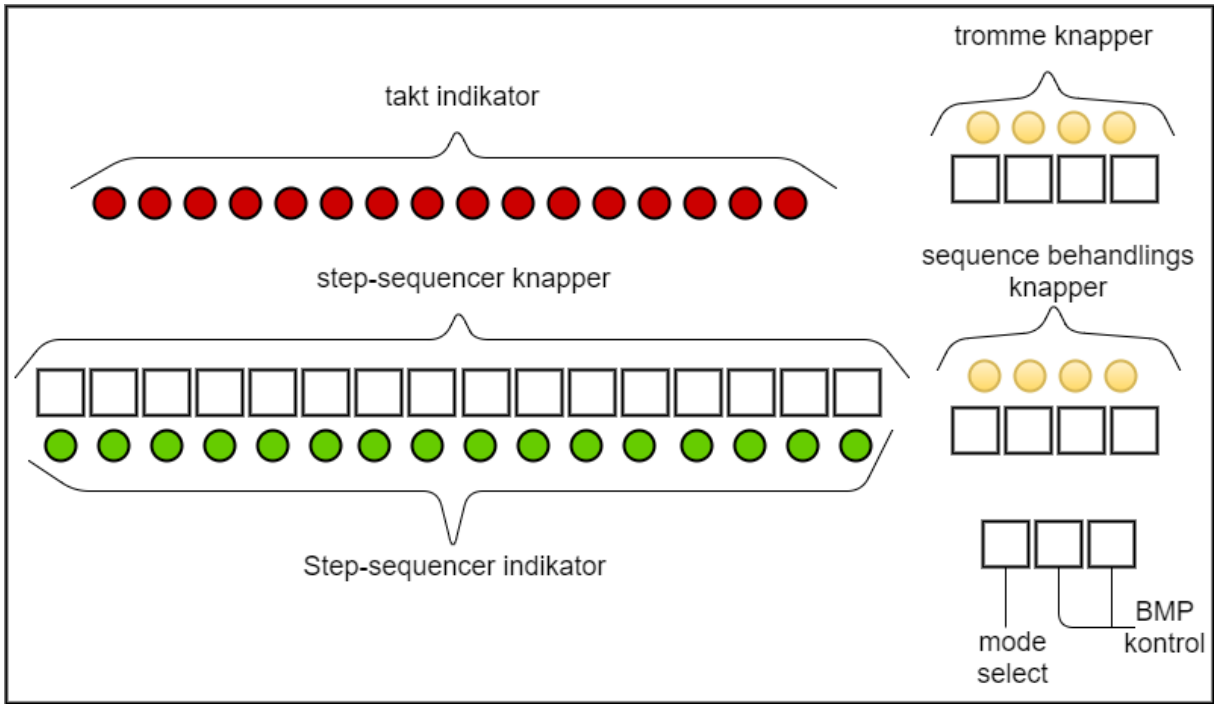
Takt indikator består af 16 LED'er som tænder og slukker en efter en, som viser i hvilken af de 16 dele af en takt man er på. Altså vil lyd eksempelvis spille når der er lys i både den specifikke takt indikator LED samt den tilsvarende step-sequencer LED.

Tromme knapper er fire tryk-knapper, som enten bruges til at tromme real-time når man trykker på dem. Eller også bruges de til at vælge hvilket sample man ønsker at ligge ind i step-sequenceren (vha. af step-sequencer knapperne).

Sequence behandling består af fire tryk-knapper, som skal gøre det muligt at gemme, hente, slette og opstille sekvenser i en ønsket rækkefølge disse bruges sammen med step-sequencer knapperne til at bestemme hvilken sekvens man ønsker gemme, hente osv. - måske?.

Mode select består af én enkelt tryk-knap, som skal bruges til at vælge om man vil bruge step-sequence-knapperne, om man vil bruge tromme knapperne real-time eller en kombination af begge.

BMP kontrol består af to tryk-knapper, som kan skrue op eller ned for BMP i et interval mellem 50 og 200.



Figur 0.2: Umiddelbar billede af brugerinterfacet - high ress pl0x

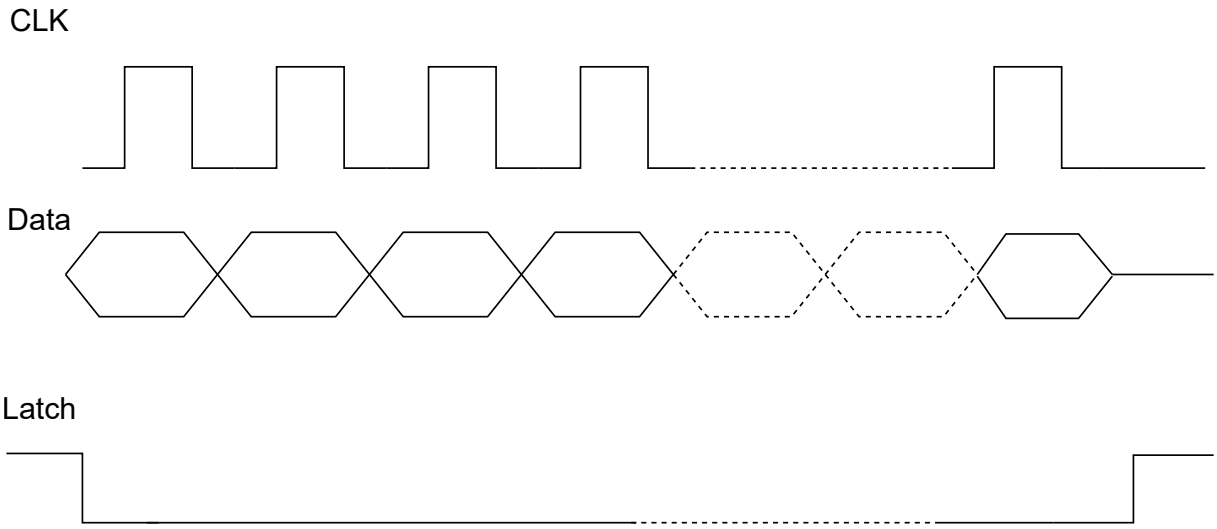
0.3.1 ved ikke hvad dette afsnit skal hedde men det handler om teknikken bag bit-boardet

De mange knapper og LED'er vil kræve mange I/O, hvis du skulle forbindes direkte til FPGA'en. Så for at spare på I/O's bruges der shift-register. Med disse er det muligt med sekventielt input til shift-registerne. shift-registerne skriver derefter parallelt ud til hvor mange ben den nu engang har. Ligeledes gør shift-registerne også det muligt at læse parallelt fra mange ben, som så kan sendes sekventielt til FPGA'en. I dette projekt bruges der 8-bit shift register. Vi bruger tre inputs til at styre dem, en clk, en parallel-load "latch"og et data ben.

Der skrives til og læses fra shift-registerne på samme måde. Når latch benet er højt, vil shift-registerne til knapperne læse om der er trykket på knappern eller ej. Når det i stedet er lavt, er det muligt at læse knappernes tilstand ind i FPGA'en.

shift-registerne til LED'erne vil når latch benet er lavt er det muligt at skrive om LED'erne skal være tændte eller ej. Når latch benet er højt, vil shift-registerne sætte de forbundne LED'er høj eller lav, alt efter hvad der var skrevet førhen.

Data bliver læst eller skrevet til shift-registerne, under overgangen fra lav til høj på clk benet.



Figur 0.3: Umiddelbar billede af signal til shift-registerne - if (ugly and shit) remake;

Som tidligere nævnt består brugerinterfacet af knapper og LED'er. Step-sequence knapperne og alle LED'er er forbundet vha. shift-register. Til takt-indikatoren bruges der to 8-bits 74HC595 shift-registre i daisy-chain. Step-sequencer LED'erne og LED'er til indikation af sequence behandling og tromme knapper, består ligeledes af tre shift-register i daisy-chain, dermed skal der bruges hhv. 16-bit eller 24-bit når der skrives til dem. De 16 sequence knapperne bruger også to 8-bits HEF4021B shift-register, på samme måde som ved LED'erne, hvor der læses fra dem i stedet.

resterende knapper (tromme, sequence knapper, mode select og BMP kontrol) er forbundet direkte til FPGA'ens I/O, dette fordi disse er langt mere tids følsomme ikke dem alle sammen meennnn?, og dermed ønskes det ikke, at der spildes tid på at shift-registrene skal behandle dem.

Ordliste

BPM Beats per minute. generelt andvent måleenhed for musiktempo. Antallet af taktslag per minut. Eksemplelvis antallet af fjerdedele på et minut i taktarten $\frac{4}{4}$.. *se* takt

KCPSM6 Processer til en picoBlaze lavet specifikt til Spartan-6 og lign. FPGA'er. *se* picoBlaze

picoBlaze 8-bits soft mikrocontroller designet til FPGA'er fra Xilinx. *se* FPGA

SRAM Static Random Access Memory. Statisk RAM. . *se* RAM

TDA1541A 16 bits stereo D/A-konverter designet til brug i Hi-Fi-aparater. Dette er D/A-konverteren anvendt i dette projekt.. *se* D/A-konverter

Bibliografi

[1] Philips. *TDA1541A datasheet*. Sidst set: 14/04/2017. 1991. URL: http://pdf.datasheetcatalog.com/datasheet/philips/TDA1541A.pdf.

Test af D/A-konverter og I2S 1

1.1 Formål

Formålet med dette forsøg er at undersøge hvorvidt Papilio DUOen kan kommunikere med den valgt D/A-konverter via I2S, og om D/A-konverteren kan repræsentere alle værdier i et 16 bits ord som en spænding.

1.2 Udstyr

På tabel 1.1 ses en tabel over det anvendte forsøgsudstyr.

Instrument	Model	AAU-nummer
FPGA	Papilio-duo 2MB	
Logik analysator	Digilent Analog Discovery 2	2179-04
D/A-konverter	TD1541A	
Strømforsygning	HAMEG HM7042	B1-101-N-7 B1-101-O-6

Tabel 1.1: Tabel over anvendt forsøgsudstyr.

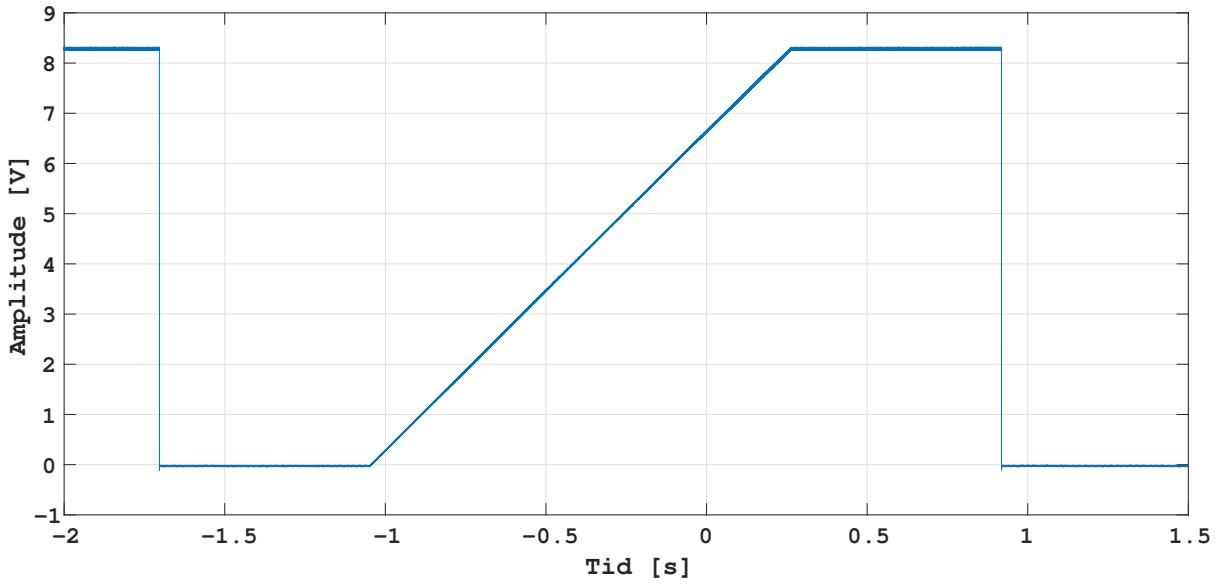
1.3 Metode

Der er kodet et VHDL-modul der skal kommunikere med D/A-konverteren via I2S. Dataet der sendes til D/A-konverteren skal falder inden for grænserne for et 16 bits ord i 2's kompliment. I decimaltal svare dette til at den sendte serielle data skal gå fra -32768 til 32767. Der er i kravspecifikation vedtaget at outputtet skal gå i klipning hvis værdien overskriver disse grænser, og for at teste hvorvidt dette overholdes tælles en variable op fra -33000 til 33000 hver gang der skiftes fra venstre til højre lydkanal. Dette medfører at hver gang der er blevet sendt en specifik værdi på hver kanal, vil den næste værdi være én størrer. Når variablen når sin øvre grænser nulstilles den.

D/A-konverteren opkobles jævnfor dens datablad[1], hvor dens input bliver danne af Papilioen. I databladet ses det at D/A-konverterens output, skal direkte i en operationsforstærker med predefineret tilbagekobling. Forstærkningsgraden, og derved den maksimale udgangsspænding, er derfor ukendt. Der benyttes intet DC-ofset i operationsforstærkeren, så negative spændinger forventes ikke. Operationsforstærkeren medtages som en del af D/A-konverteren, og dette udgangssignal måles og gemmes af logikanalysatoren. Hvis outputdatavariablen overskider grænserne for et 16 bits ord forventes det at ouputtet klipper, og hvis variablen ligger inden for grænsen forvæntes en lineær stignen i D/A-konverterens output.

1.4 Resultater

På figur 1.1 ses forsøgets måledata. Der ses her at udgangsspændingen klipper ved 0 V og lige over 8 V. Det ses også at stignen i spændingen en lineær mellem den øvre og nedre grænse for klipning.



Figur 1.1: Måledata over spændingsniveauerne fra D/A-konverteren.

1.5 Konklusion

Det konkluderes at Papilioen kan kommunikere med D/A-konverteren via I2S, og at alle værdier for et 16 bit ord kan repræsenteres som en spænding. Hvis ikke de alle var repræsenteret ville stigningen på figur 1.1 have små ujævnhedere hvor værdien ikke kunne repræsenteres. Det konkluderes også at klipning håndteres på den forventede måde.