NTNU
Det skapende universitet

Evolution in Materio:

En Kaotisk Tilnærming

Eirik Lund Flogard

Master i datateknologi
Innlevert: november 2015
Hovedveileder: Gunnar Tufte, IDI
Medveileder: Magnus Jahre, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Problembeskrivelse

This project work will be part of the EU-funded Future Emerging Thecnologies initiative. In this research project we try to exploit computational properties of unconventional materials (materials usually not considered as a computational substrate). Such materials may offer computation at extreme low cost and may also enable us to do computation that is hard (or impossible) on a von Neumann stored program machine. Currently we explore possible computational properties of carbon nano tubes.

In 2010 a first version of a platform was made. This system consists of a PCB, including an Atmel microcontroller and a Xilinx FPGA, that acts as an interface between a PC and a micro electrode array. The array interfaces the material under investigation.

In this project initial investigation of evolutionary processes toward control problems is the main target.

"Debating creationists on the topic of evolution is rather like trying to play chess with a pigeon — it knocks the pieces over, craps on the board, and flies back to its flock to claim victory."

Scott Weitzenhoffer, 2005

Sammendrag

Denne avhandlingen omhandler et konsept kalt Evolution in Materio, der man gjennom datakontrollert evolusjon forsøker å utnytte et materies naturlige egenskaper for å løse oppgaver eller utføre beregninger. Motivasjonen for forskningen innen dette område er at dette på sikt kan være et reelt alternativ til digitalteknikk og integrerte kretser. Konseptet er forøvrig relativt nytt og det er derfor mange problemer som må løses. Det største problemet er knyttet til begrensninger i forhold til hvilken grad materiets oppførsel kan påvirkes og kontrolleres. Dette forårsaker ustabilitet, lav forutsigbarhet og begrenset mulighet for reproduksjon av resultater. Tidligere arbeid innenfor liknende områder har indikert at et system må kunne veksle mellom ordnede og kaotiske tilstander for å kunne utføre beregninger. I et prosjekt kalt Nascence er derfor målet å forstå oppførselen til systemer, samt å utforske nye metoder for utnyttelse av disse.

Før et materie kan utføre beregninger må forholdene legges til rette for at reaksjonene assosiert med disse spontant kan oppstå. I Evolution in Materio gjøres dette ved å påføre signaler som påvirker lokale kvantetilstander i materiet. I den forbindelse var målet for denne oppgaven å utforske hvordan de periodiske funksjonene sinus, sagtann og triangel, påvirker mulighetene for å optimalisere en klump med nano-karbon til å utføre beregninger gjennom en genetisk algoritme. I tillegg var målet å se om kaotisk oppførsel på signalet kan utnyttes av en slik algoritme. For å generere signalene samt ulike variasjoner av kaotisk oppførsel for disse ble 3 nye rammeverk utviklet. Den eksperimentelle metoden gikk ut på å kvantisere den "kaotiske" oppførselen i de forskjellige signalene gjennom ordinære metoder for måling av støy og forvrengning. Et eksperiment med en genetisk algoritme ble deretter gjennomført som en objektiv evaluering av de ulike signalene. Det ble også gjennomført en analyse av hvordan de ulike signalene påvirker algoritmens søkerom. Resultatene ga indikasjoner på at en genetisk algoritme er i stand til å utnytte kaotisk oppførsel dersom dette til en viss grad kan kontrolleres. I tillegg viste resultatene at spesielt sagtann funksjoner er velegnet som konfigurasjonssignal for den genetiske algoritmen som ble brukt.

Abstract

This thesis deals with a concept called Evolution in Materio, where we through computer controlled evolution(CCE) attempt to exploit a material's natural abilities to solve problems, puzzles or perform computation. The motivation for research in this area is the possibillity that the concept in long term, can serve as a viable alternative to digital technology and integrated circuits. The concept is however relatively new and thus many problems need to be solved. The biggest problem is related to the limitations in relation to what extent non-human designed physical systems' behavior can be influenced and controlled. This causes instability, low predictability and limited ability for reproduction of results. Previous work in similar areas has indicated that a system must be able to switch between states of order and chaos in a relatively controlable manner in order to perform computation. A project called Nascence aims to achieve a greater understanding of the behavior of such systems, as well as to explore new methods of exploitation of these.

Before a material can perform computation, the physical conditions must be ideal in order for the neccesary associated reactions to spontaneously occur. For Evolution in Materio optimizing these conditions is usually done by applying signals, which affects the quantum states of a given material. In order to achieve a greater understanding of this, the aim of this thesis was to explore the ability of the periodic functions sine, sawtooth and triangle to optimize a lump of nano-carbon material to perform calculations through a genetic algorithm. Also another goal was to investigate the possibility that induced chaotic behavior on these signals can be utilized by such an algorithm. To generate signals as well as different variations of chaotic behavior 3 different frameworks were developed. The experimental method was to quantify the "chaotic" behavior of the various signals through ordinary methods for measuring noise and distortion. Additionally experiments with a genetic algorithm was then performed as an objective evaluation of the various signals. Following the experiments an analysis of how the different signals influence the algorithm's search space were performed. The results showed clear indications that a genetic algorithm is capable of utilizing chaotic behavior if this can be controlled to a certain extent. In addition, the results showed that especially sawtooth functions are suitable as configuration signals for the genetic algorithm was used.

Anerkjennelser

Jeg ønsker å takke min veileder Gunnar Tufte for assistanse med oppgaven og fleksibilitet med hensyn på veiledningstimer. Hans erfaring og bidrag med kunnskap innenfor evolusjonære datamaskiner, vitenskapelige metoder og råd om rapportskriving har vært til stor hjelp både læringsmessig og for gjennomføring av selve oppgaven. Jeg vil også takke Odd Rune Lykkebø for lån av plattformen Mecobo samt hjelp til å løse problemer med hardware.

Jeg vil også uttrykke min takknemlighet til mine foreldre Tom Lund og Kristin Flogard, samt mine besteforeldre Randi Lund, Oddvar Lund, Else Marie Flogard og Jan Sverre Flogard for moralsk og økonomisk støtte. Til slutt ønsker jeg å takke Ellen Beate Hove, Anders Baklund og Britt Sandaune som sørget for at det var mulig for meg å gjennomføre prosjektet.

Innholdsfortegnelse

Figurer

Tabeller

Forkortelser

DDS	Direct Digital Synthesizer.
EiM	Evolution in Materio.
EM	Evolutionary Motherboard.
GA	Genetisk algoritme.
NCO	Numerical Controlled Oscillator.
THD	Total Harmonic Distortion.
RMS	Root Mean Squared.
SNR	Signal to Noise Ration.
SWCNT	Single Walled Carbon Nano-Tubes.

Ordliste

Design1	Selvstendig DDS med variabel samplingsresolusjon.
Design2	Selvstendig DDS med NCO arkitektur og fase-dithering.
Design3	Selvstendig DDS der signalet utsettes for "clipping".
Faserom	En representasjon av alle tilstandene for et system.
Fitness	En estimert sannsynlighet for reproduksjon.
Fitnesslandskap	Fitness angitt som en funksjon av et faserom.
Material	Et materie som potenselt kan utføre beregninger.
Mecobo	Rammeverk designet for utforskning av material.
Tensor	"Jakobiansk" relasjon mellom forskjellige basiser.

Symboler

f	frekvens	Hz
THD	Total harmonisk forvrengning	dB
SNR	Signal til støynivå	dB
$SFDR$	Spurious Free Dynamic Range	dB
L^p	Lesbegue funksjonsrom	
l^p	Lesbegue sekvensrom	
p	p-distansenorm	
RMS	Kvadratisk gjennomsnitt	
δ	Dirak-deltafunksjon	
ω	angular frequency	rads^{-1}

Chapter 1

Introduksjon

Hva er en datamaskin? For mange er svaret på spørsmålet ganske enkelt "en innretning vi kommuniserer med og gjennom". Datamaskinen kan muligens anses å være den viktigste oppfinnelsen i det 20. århundret, ettersom vi kan kommunisere med hele verden på utallige mange måter uten å bevege oss i det hele tatt. For å realisere dette må datamaskinen kunne utføre *beregninger*. Mer konkret kan vi som et minstekrav si at en datamaskin er en maskin som løser et veldefinert problem. Enhver maskin som kan emulere en datamaskin er dermed en *universell maskin*[10][11].

I 1945 presenterte Von Neumann en skisse av en universell maskin kalt *EDVAC* [12]. Rapporten beskriver hvordan EDVAC realiseres gjennom "E-elementer". Et E-element er en hypotetisk entitet som representerer en tidsuavhengig funksjon og er i stand til å opprettholde diskrete ekvilibria over ubegrenset tid. Von Neumann viser at egenskapene til hvert enkelt E-element bevares når de settes sammen med andre E-elementer. Det ble også vist hvordan matematiske operasjoner kan deduseres til komplekse nettverk av E-elementer gjennom *abstraksjonslag*.

I moderne datamaskiner er slike nettverk realisert gjennom transistorer i integrerte kretser. Et problem i forbindelse med dette er at å utnytte hver transistors potensiale til å utføre beregninger stadig blir vanskeligere. I løpet av de siste årene har varmeutvikling og strømlekasjer fra transistorene ført til at kun et begrenset antall transistorer kan være aktive samtidig. Dette fenomenet kalles "Dark Silicon" [13] [14]. Samtidig blir også produksjonsprosessene stadig mer kostbare og vanskeligere å gjennomføre.

Selv om universelle maskiner er i stand til å løse alle beregnbare problemer finnes det enkelte problemer som deterministiske maskiner er dårlige til å løse. Enkelte av disse problemene kan løses på polynomisk tid for ikke-deterministiske maskiner. De vanskeligste av disse problemene er definert som *NP-komplette*[11]. En annen svakhet med universelle maskiner er at de har problemer med å etterlikne ikke-deterministiske og komplekse fysiske prosesser. Generering av tilfeldige tall og sanntidsrendering av lys innen dataspill-grafikk er kjente eksempler på problemer der man forsøker å etterlikne "virkeligheten" [15]. Problemet er at virkeligheten er kompleks og sammensatt, noe som gjør tilnærming gjennom deterministiske beregninger vanskelig[16].

1.1 Prosjektbeskrivelse

Evolution in Materio(EiM) er et konsept som kan ses på som en implementasjon av en ikke-deterministisk maskin. Konseptet går ut på å optimalisere *beregningspotensialet* i et eksisterende objekt, heller enn å deduktivt definere selve objektet via abstraksjonslag slik det gjøres i tradisjonelt kretsdesign. Denne "optimaliseringen" utføres ved å sende signaler mot en klump av uorganisert materie gjennom datakontrollert evolusjon (CCE) [17]. Denne prosessen kan formelt deles inn i et fysisk og digitalt domene slik figur 1.1 viser. Materiet representerer det *fysiske domenet* og kan beskrives som et *kvantemekanisk ensemble*[18] og kan for eksempel være nano-karbonrør, LCD-krystaller eller ødelagte transistorer. Materiets kvantetilstand kan uttrykkes gjennom en tetthetsmatrise for ensemblet. Den differensielle endringen i tettheten over tid kan dermed beskrives gjennom Liouville-Von Neumanns formel[18]. Når materiet er under påvirkning av signalene vil det oppstå endringer i materialets kvantetilstander i henhold til en (ukjent) tidsavhengig Hamilitonian-funksjon.

Dersom endringene i kvantetilstandene gir en målbar påvirkning på materialets egenskaper, kan dette optimaliseres og utnyttes av en genetisk algoritme i det *digitale domenet*. Figur 1.1 viser en generell beskrivelse av hvordan den genetiske algoritmen fungerer. Et veldefinert problem kan uttrykkes gjennom en objektiv funksjon kalt en *fitnessfunksjon*. Målet til den objektive funksjonen kan for eksempel være den matematiske funksjonen $3 = x+y$. Konfigurasjonssignalene på figuren

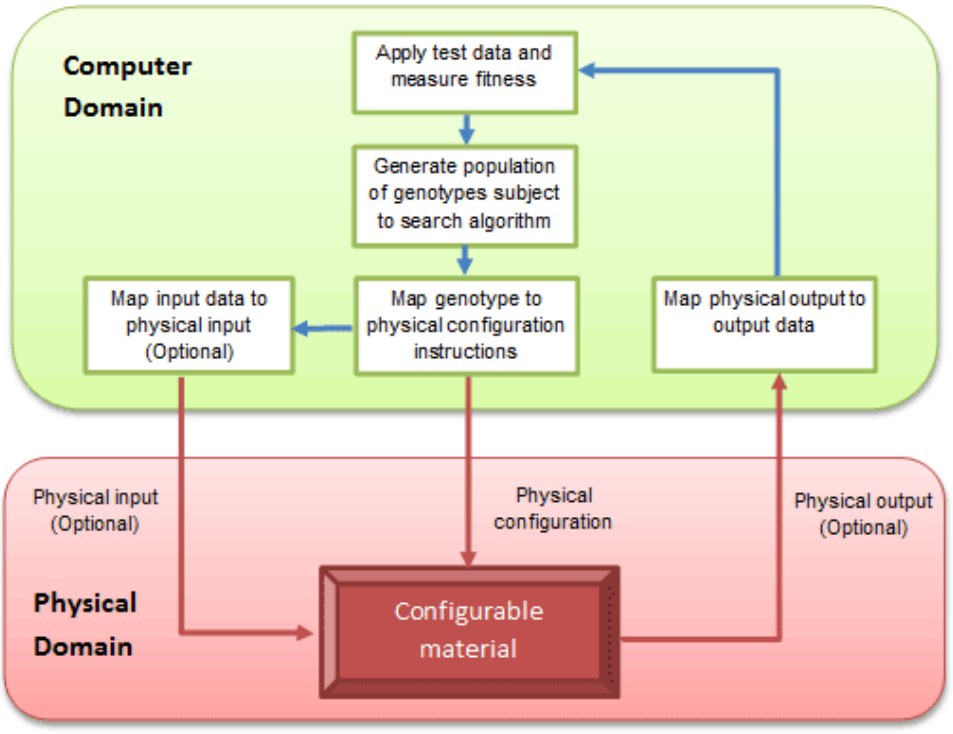

FIGURE 1.1: Figuren er hentet fra [1] og viser en konseptuell beskrivelse av hvordan "evolution in materio" fungerer.

vil da sammen med materialet representere funksjonen, mens inputsignalene representerer funksjonens domene. Algoritmen må finne en riktig kombinasjon av konfigurasjonssignaler som gjør denne funksjonen sann for et gitt funksjonsdomene. Den genetiske algoritmen assosierer en gruppe med konfigurasjonssignaler med et abstrakt digitalt objekt kalt et *individ*, som tilhører en *genetisk populasjon*[19].

I forbindelse med EiM er eksperimentell plattform kalt Mecobo blitt utviklet. Plattformen tilbyr et generisk grensesnitt for generell utførelse av eksperimenter, og gjør det mulig å generere opptil 18 signaler samtidig. Plattformen støtter Firkant bølger samt statisk DC-spenning som kan uttrykkes som en form for *basis* for inputsignaler og konfigurasjonssignaler. Signalene påføres materialet gjennom et elektrode array (ofte kalt "configuration pins"). Motivasjonen er at hver pin kan "omprogrammeres" til å generere forskjellige signaler eller utføre forskjellige operasjoner. Dette tillater den genetiske algoritmen å variere den fysiske plasseringen av signaler på materialet, noe som tidligere ikke har vært mulig.

1.1.1 Mål

For å oppnå gode resultater er den genetiske algoritmen sannsynligvis avhengig mange faktorer. En av disse er konfigurasjonssignalene. Ettersom den genetiske algoritmen optimaliserer materialets *beregningspotensiale* gjennom variasjon av konfigurasjonssignalene, er det heller ikke usannsynlig at algoritmens måloppnåelse er avhengig av basisen for konfigurasjonssignalene. **Forskningsmålet for denne oppgaven er derfor å studere hvordan basisen for konfigurasjonssignaler påvirker den genetiske algoritmen**. Motivasjonen er at dette kan avdekke flere og mer velegnede metoder for generering av konfigurasjonssignaler enn det som brukes i dag. Dette kan isåfall gjøre det enklere å løse komplekse problemer og beregninger gjennom EiM.

1.1.2 Arbeidshypoteser

I [20] ble det gjort en sammenlikning mellom statisk spenning og firkantbølger. Resultatet indikerte at firkantbølger er spesielt egnet som konfigurasjonssignal (se seksjon 2.4.3). En vesentlig forskjell mellom disse basisene er at firkantbølgene er en periodisk funksjon. Dersom det antas at resultatet ikke oppsto tilfeldig kan følgende forskningsspørsmål framstilles: Er signaler med periodiske egenskaper velegnede basiser for konfigurasjonssignaler?

For å besvare spørsmålet er det derfor valgt 3 periodiske funksjoner: sinus, sagtann og triangel. Tanken er at disse kan vise seg å være velegnede kandidater som konfigurasjonssignaler. Basert på dette framsettes følgende arbeidshypoteser:

1. En sinusfunksjon er en egnet basis for konfigurasjonssignaler.
2. En triangelfunksjon er en egnet basis for konfigurasjonssignaler.
3. En sagtannfunksjon er en egnet basis for konfigurasjonssignaler.

Det bør nevnes at i forbindelse med generering av konfigurasjonssignalene, vil det være uungåelig at det oppstår støy på funksjonene når det genereres i det digitale domenet. Spørsmålet er hvordan dette påvirker materialet og hvorvidt dette påvirker den genetiske algoritmen på en positiv eller negativ måte. I forbindelse med forskningsspørsmålet kan det være spesielt interessant å undersøke om *periodiske artefakter* kan utnyttes av den genetiske algoritmen. Basert på dette

framsettes følgende arbeidshypotese: **Periodiske artefakter på konfigurasjonssignalene kan med fordel utnyttes av en genetisk algoritme**.

1.2 Rapportstruktur

Resten av rapporten har til hensikt å besvare spørsmålet og arbeidshypotesene som ble framsatt i denne seksjonen.

Kapittel 2 er ment å gi en innføring i Evolution in materio, samt en oversikt og kritisk analyse av eksisterende faglitteratur og arbeid. I tilegg presenteres en oversikt over plattformen Mecobo. I Kapittel 3 presenteres 3 oppdaterte versjoner av Mecobo som er implementert for å realisere de periodiske funksjonene, samt periodiske artefakter på disse. I Kapittel 4 presenteres en metode for å kvantisere de periodiske artefaktene på signalene. I tillegg presenteres en genetisk algoritme i den hensikt å evaluere signalene generert av de oppdaterte versjonene av Mecobo. I Kapittel 5 presenteres et utvalg av de viktigste metodene og observasjonene som ble gjort i forbindelse med testing. I Kapittel 6 presenteres et utvalg av de mest relevante resultatene som ble funnet etter gjennomførelsen av metodene, samt en tolkning av disse. En analyse av resultatene samt evaluering av arbeidshypotesene gjøres i kapittel 7. I tilegg gjøres det en vurdering av hva som bør gjøres med hensyn på framtidig arbeid.

Chapter 2

Bakgrunn

Hvordan kan vi konfigurere et fysisk system til å utføre beregninger? Det samme spørsmålet var aktuelt på 40-tallet da ledende forskere forsøkte å finne et egnet fysisk system for å implementere en *universell maskin*. Svaret kom i form av Von-Neumans EDVAC som ble implementert gjennom en deduktiv *topdown* metode[12]. Selv om det idag brukes moderne implementasjonsteknikker som digitale simuleringer og HDL språk, benyttes fremdeles den samme deduktive prosessen. Spørsmålet er om vi faktisk utnytter det fysiske *beregningspotensialet* på en effektiv måte?

Problemet er at den universelle maskinen er definert med diskrete symboler, en egenskap i prinsippet er uforenlig med fysiske systemer. I en integrert krets løses dette ved å bruke transistorer som overføringsfunksjoner. Ved å gjøre dette ekskluderes andre potensielle løsninger som kan gi fysiske systemer med bedre beregningspotensialer. I følge Conrad er dette prisen vi må betale for programmerbarhet [21]. En løsning er å lete etter alternative måter for å utføre beregninger.

Resten av denne seksjon er dedikert for å gi en kort innføring i EiM, samt en presentasjon av tidligere arbeid som er mest relevant i forbindelse med denne oppgaven. I seksjon 2.5 presenteres også en oversikt over plattformen Mecobo.

2.1 Komplekse systemer

I forbindelse med utforskning av fysiske systemer, er det viktig å bruke de riktige "verktøyene" for å studere dem. Weaver påpeker at kompleksitet gjerne er en egenskap som er tilstede i "levende systemer" [22] . Komplekse systemer er generelt fysiske systemer med oppførsel og egenskaper som ikke kan forklares utifra systemets elementer eller byggesteiner. Et eksempel på en slik egenskap kan være tettheten til en beholder med gass. Vi sier ofte at slike egenskaper er *emergente*.

I følge Heylighen er komplekse systemer ofte ikke-lineære[16]. Det er derfor ofte vanskelig å observere sammenhenger mellom årsak og virkning for slike systemer. En tilsynelatende liten endring eller forstyrrelse i en del av systemet kan for eksempel i visse tilfeller forårsake en markant økning i perturbasjon. Dette kalles positiv feedback. På den andre siden kan det store endringer i systemet ved et annet tilfelle gi en *negativ feedback* som har en "dempnings-effekt" på systemet ved at systemet går i en mer ordnet tilstand. Disse *sommerfugleffektene* observeres ofte som følger av at komplekse systemer er sensitive til *initielle forhold*.

Weaver påpeker også at komplekse systemer generelt er vanskelig å etterlikne eller modellere. Grunnen er at en tilstrekkelig deduktiv modell av systemet ikke er gjennomførbart på grunn av det nødvendige antall variabler som kreves for å gjennomføre dette. Istedet mener Weaver at systemene heller kan forklares gjennom **dets egenskaper og oppførsel**: Hvor stor sannsynlighet er det for at systemet befinner seg i en gitt tilstand? Hvordan endrer temperaturen(e) i systemet seg over tid? Weaver påpeker derfor at bruk av statistiske metoder er en mer realistisk løsning med hensyn på studier av slike systemer. Slike metoder kan karakteriseres som *induktiv* eller *abduktiv*.

2.1.1 Statistiske ensembler

En velkjent metode som benyttes for å studere kompliserte fysiske systemer er statistiske ensembler[18]. Ensembler ble introdusert av John Gibbs for å beskrive "de mest sannsynlige tilstandene" i mikroskopiske systemer utifra makroskopiske variabler[23]. Mer generelt kan man si at modellen beskriver sannsynlighetsfordelingen av tilstandene eller *faserommet* til et opptil uendelig høyt antall kopier av et vilkårlig system. Faserommet kan beskrives gjennom tetthetsmatrisen $\hat{p}$.

Da er sannsynlighetsfordelingen for faserommet gitt av tetthetsoperatoren $\hat{p} = \sum_i p_i|\psi_i\rangle \langle\psi_i|$. Hvor $p_i|\psi_i\rangle$ er sannsynligheten for tilstandsvektoren $|\psi_i\rangle$. En fordeling er gitt for en Hamiliton opperator $\hat{H}$ som tilsvarer den totale energien som kan måles fra systemet.

Som følger av statistisk entropi og Liouvilles teorem vil tetthetpunkter i modellen kalt *attraktorer* dannes over tid. Dette kan beskrives som et sett med punkter som modellen, gitt en initiell tilstand, har tendens til å bevege seg mot[16].

2.1.2 Evolusjon og Fitnesslandskap

Vi har tidligere uttrykt hvordan vi kan beskrive alle tilstandene til et system som et ensemble, men hva med systemer som er under påvirkning av en evolusjonær prosess? Vi kunne for eksempel antatt at påvirkning fra omgivelsene er en del av systemet vi studerer. En ulempe med dette er at det fort fører til at systemets *grenser* blir uklare ettersom det da ikke er mulig å skille mellom systemet vi ønsker å studere og dets omgivelser. En løsning kan være å bruke en modell som for systemet som inkluderer den evolusjonære prosessen. Denne ideen kom opprinnelig fra biologen Sevall Wright i 1932. I følge Wright kan genene til en "høyere biologisk organisme" representeres gjennom en *adaptiv verdi*[2]. Den adaptive verdien representerer en objektiv beskrivelse av en fenotype, og kan fra et biologisk ståsted beskrives som *et potensial for reproduksjon*. Det er viktig å bemerke at det også er underforstått at dette også er nært knyttet til organismenes robusthet mot ytre påvirkning som kan true individenes og populasjonens eksistens [24]. Gjennom fitnesslandskap er Wright i stand til å vise hvordan "dårlig" naturlig utvalg i evolusjon skaper fenotypiske feil hos dyr.

Et fitnesslandskap kan generelt beskrives som en funksjon som assosierer alle mulige kombinasjoner av gener med en fitnessverdi. Wright beskriver disse kombinasjonene gjennom en hyperkube som består av det genotypiske faserommet. Figur 2.1 viser hvordan faserommet av genotyper henger sammen gjennom "mutasjonsakser". Hver node på grafen kan assosieres med en estimert fenotype og fitnessverdi. Wright påpeker også at de 5 dimensjonene på grafen i figur 2.1 er et dårlig estimat. I den forbindelse må presiseres at fitnesslandskap kun er en abstraksjon, eller mer presist en matematisk modell basert på målinger og antakelser for et fysisk system. Man står derfor mer eller mindre fritt i forhold til hvordan man ønsker å estimere en modell for fitnesslandskapet.

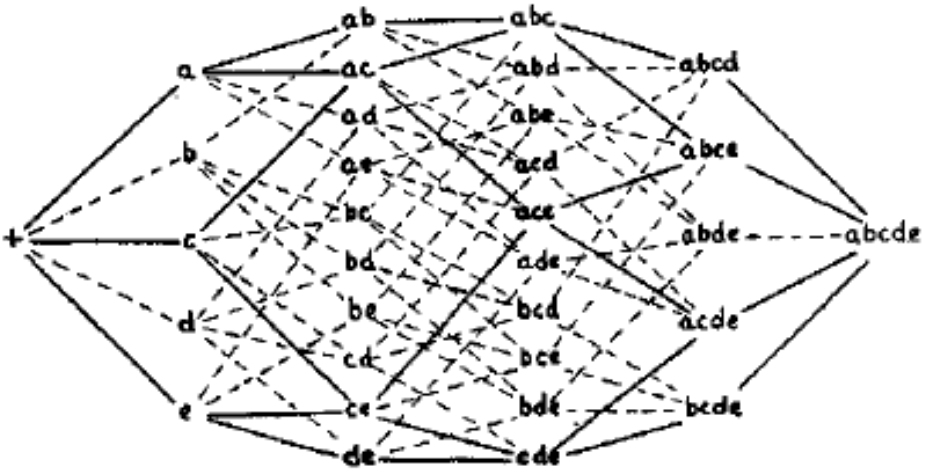

FIGURE 2.1: Wrights beskrivelse av et 5-dimensjonalt "hyperkubisk" faserom[2]

2.1.2.1 NK-modellen

En svakhet med Wrights metode er at den er *deduktiv*. Den nødvendige mengden av informasjon som ofte kreves for å realisere et fitnesslandskap øker eksponentielt med antall dimensjoner som kreves for å representere landskapets faserom. Å gjøre dette utifra Wrights modell kan innebære at en fitness-verdi må måles for alle tilstandene av systemet. Dersom fitness-verdien for en tilstand er tidsavhengig eller systemets faserom består av et høyt antall tilstander, vil det være svært vanskelig å estimere en god, nøyaktig modell for landskapet.

En mulig løsning er å representere systemet som et statistisk ensemble, slik at fitnesslandskapet kan estimeres gjennom et *justerbart NK-landskap*. I følge Kauffman et al er fitnesslandskapenes "høyde" og form avhengig av relasjoner mellom fundamentale enheter i et ensemble[25][26]. De fundamentale enhetene er ofte representert som gener, og deles inn i spesifikke sekvenser som kalles lokii. Modellen kan beskrives utifra formel 2.1 for et funksjonsrom som utspennes av en serie med gener definert gjennom S. $Fitness(S)$ kan i visse tilfeller beskrives som en funksjon gitt for en sigma-måling $\sum$, som gjennom funksjonen $f()_i$ (gitt for en instans av i) beskriver hvordan hvert lokus S_i påvirker systemets fitnessverdi gjennom en relasjon med K andre *lokii*. Et definert system må også ha en "ytterkant". Formelt vil det si at variabelen i i formel 2.1 må oppfylle betingelsen $|i| < \infty$. I følge Barnett er fitnessverdier gitt for et reelt fysisk system "bråkete"[27]. Funksjonen $f()$ er derfor i mange tilfeller assosiert med en varians. Det bør også nevnes at formelen er definert utifra en kvalitativ beskrivelse av modellen.

Et fitnesslandskap kan ofte beskrives som dynamisk, eller med andre ord at det endrer seg over tid. Landskapet har også et naturlig metrisk system definert gjennom Hamming distanse[27].

$$Fitness(S) = \sum_i f(S_i, S^i_{k=1}, S^i_{k=2}...S^i_{k=K})_i \tag{2.1}$$

I følge Kauffman et al representerer NK-modellen en sammenheng mellom den adaptive verdien som assosieres med fenotypen for et lokus i ensemblet, og dens avhengighet av K andre lokii. Disse komplekse relasjonene fører til at endringer i ulike gener vil påvirke det fysiske domenet på forskjellige måter. Dette kalles *epistase* og kan forekomme både innenfor et gen, samt mellom ulike gener. Et lokus eller en allele som i stor grad påvirker den adaptive verdien og fenotypen hos en organisme kalles *dominant*. Alleler som under tilsvarende endringer, påvirker disse i liten grad kalles *recessive*[27]. Dette kan ses som en analogi til positiv og negativ feedback.

Fordelen med denne modellen er at det er mulig å estimere en verdi for K utifra formen på landskapet. Ettersom K representerer et gjennomsnitt for antall relasjoner hver fundamentale enhet inngår i, vil K derfor ofte gi god kvantitativ beskrivelse av systemet landskapet representerer. Kauffman et al påpeker også hvordan ulike variasjoner av K påvirker muligheten for at en evolusjonær prosess finner løsninger.

2.1.3 Beregninger og kaos

NK-modellen som ble formalisert gjennom Kauffman et als arbeid viser at det er mulig å gi en kvantitativ beskrivelse av hvordan evolusjon påvirkes av ulike fysiske betingelser i komplekse systemer. Spørsmålet er hvilke betingelser kan anses som ideelle for at et fysisk system skal kunne utføre beregninger? I følge Langton må et system kunne "lagre, overføre og modifisere informasjon for å utføre beregninger"[28]. For å gjøre dette må systemet "vise en tilfeldig korrelasjonslengde i rom og tid. Lengden må potensielt være uendelig, men trenger ikke nødvendigvis å være det". Interessant nok ga Von Neumann et al det samme kriteriet for et E-element. Langton viser til at en hypotetisk parameter λ som representerer forholdet mellom systemets totalt antall tilstander og antall tilstander som leder til en *slutt-tilstand*. Variabelen λ defineres gjennom formel 2.2. Ved å

variere parameteren K for en cellulær automat blir det vist at tiden det i gjennomsnitt tar før systemet havner i slutt-tilstanden øker eksponentielt med K for $\lambda < 0.5$. Slutt-tilstanden representerer en faseovergang mellom orden og kaos, og i følge Langton er det systemer som befinner seg ved grensen mellom disse overgangene som er mest velegnede til å utføre kompliserte beregninger.

Det bør påpekes at en cellulær automat per definisjon representerer et generelt statistisk ensemble[18] og modellen kan derfor anses som en god generell analog for fysiske systemer.

$$\lambda = \frac{K^n - n}{K^n} \tag{2.2}$$

2.2 Gordon Pask

En av de første som utnyttet ideen med å bruke materier for å utføre beregninger var Gordon Pask. Hans hensikt var forøvrig å reprodusere oppførselen til neuroner, inkludert *genesis*[3]. Figur 2.2 viser Pasks eksperimentelle oppsett. Oppsettet består av 2 metallplater, der den ene platen senkes ned i en løsning av et egnet materiale. Elektriske felt påføres platen gjennom elektroder som er fysisk koblet til forsterkere og topplaten slik figuren viser.

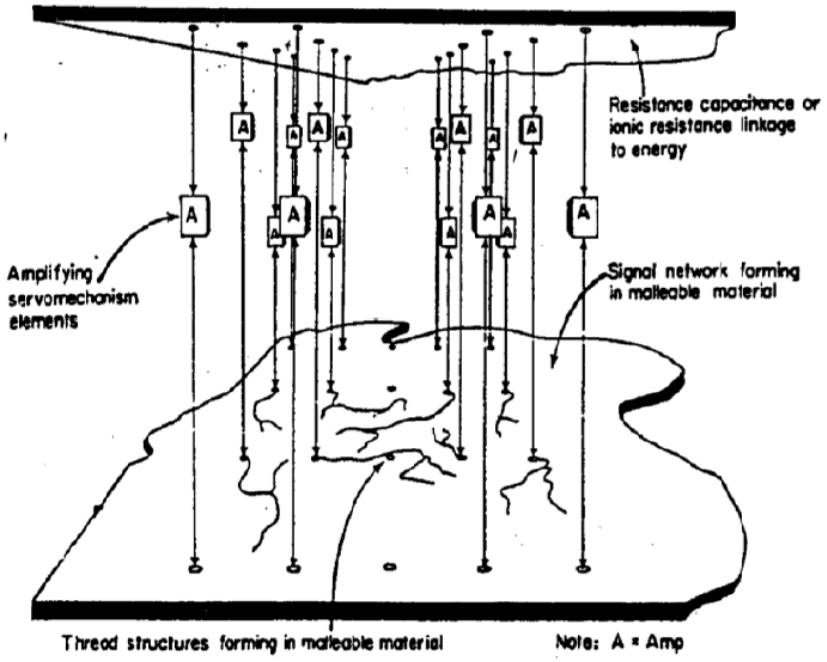

FIGURE 2.2: Oppsett for Pasks eksperiment[3]

Pask utførte samme forsøk med flere ulike typer materialer, og konkluderte med at materialer med gode strukturelle muligheter til å danne spontane reaksjoner var mest egnet. Etter flere forsøk falt valget på en kjemisk løsning bestående av jern og salt. Pask brukte en "treningsteknikk" på systemet der han prøvde seg fram med ulike forandringer på et sett med sannsynlighetskontrollerte resistorer. Han innså at ved å variere signalene kunne han få systemet til å spontant generere et nettverk av tråder slik det er vist i bunnplaten på figur 2.2. Gjennom disse *selvorganiserende* nettverkene utviklet Pask en tonediskriminator som kunne skille mellom 2 ulike lydfrekvenser.

En av de viktigste implikasjonene fra Pasks eksperiment er det faktum at tonediskriminatoren oppsto spontant. Drivkraften for prosessen er i realiteten *frie energi* i systemet. Dette oppstår som følger av endringer materialets kjemiske og elektromagnetiske potensial til å utføre et *arbeid*[23]. Pasks rolle i forsøket var kun å sørge for at *forholdene* eller systemets premisser og omgivelser var korrekt tilrettelagt. Det kan bemerkes at naturlig evolusjon utnytter spontane reaksjoner på en liknende måte. En vesentlig forskjell er at naturlig, *open-ended* evolusjon foregår uten spesiell tilrettelegging av systemet. Det fører til at forandringer skjer uten underliggende mål i systemer påvirket naturlig evolusjon. Det faktum at mennesker eksisterer som et produkt av naturlig evolusjon er en indikasjon på hvilke fantastiske muligheter som kan utnyttes ved å bruke et systems egne *referansekriterier* for å løse problemer.

2.3 Thompsons eksperiment

Selv om evolusjonære teknikker for datamaskiner har eksistert siden 50-tallet, har dette forøvrig kun foregått i det digitale domenet. Framveksten av rekonfigurerbare integrertekretser på 80 og 90-tallet førte til framveksten av *intrinsisk hardware-evolusjon*.

I 1996 viste Adrian Thompson at det var mulig å skape integrerte kretser i en FPGA (Field Programmable Gate Array) ved hjelp av en genetisk algoritme[4]. Figur 2.3 viser Thompsons eksperimentelle oppsett. En tonediskrimminator ble utviklet ved konfigurere en Xilinx XC6216 FPGA gjennom en genetisk algoritme

kjørt på en ordinær datamaskin. Algoritmen utførte evolusjon på en digital representasjon av FPGAens konfigurasjon. Hver av disse ble evaluert ved å måle responsen fra FPGAen, for inputsignaler på 10 og 1 khz. Inputsignalene ble generert av en tonegenerator slik figuren viser.

Thompson påpeker at den genetiske algoritmen fjerner begrensningene som ellers ville vært tilstede når en tilsvarende krets designes etter tradisjonelle prinsipper. På den måten reduseres også den fysiske størrelsen på kretsen betraktelig. En ulempe med FPGAen er forøvrig at det er et menneskelig design. Systemet påfører derfor begrensninger med hensyn på hva den genetiske algoritmen kan gjøre med systemet. I den forbindelse påpekte Kauffman at når kompleksiteten øker for entitetene i et ensemble under påvirknkning av en adaptiv prosess, "vil oppnåelige lokale optima være relativt nært ensemblets gjennomsnittlige egenskaper"[25]. I dette tilfellet kan det tolkes som at FPGAen i Thompsons forsøk begrenser mulighetene for å oppnå løsninger gjennom en genetisk algoritme.

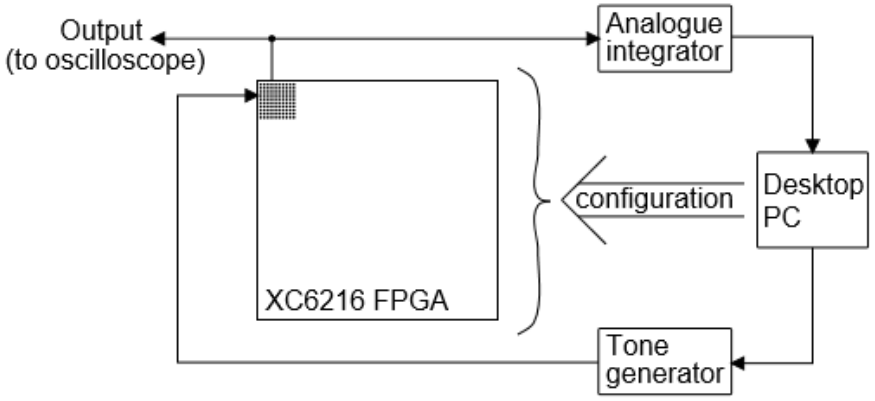

FIGURE 2.3: Thompsons eksperimentelle oppsett[4]

2.4 Evolution in materio

Både Gordon og Thompsons eksperiment kan anses å være viktig ettersom de sammen utgjør en ny tilnærming for utforming av datamaskiner. Evolution in materio er derfor på mange måter en videreutvikling av Pask og Thompsons arbeid.

Nascence er et samarbeidsprosjekt innenfor EiM og startet i 2012 med en varighet på tre år. Målet med prosjektet er å "Modellere, forstå og utnytte oppførselen

til utviklende nanosystemer, og over lengre tid lage informasjonsprosesserende enheter som utnytter slike arkitekturer uten å reprodusere individuelle komponenter"[1]. Vi er med andre ord interessert i å finne ut hvordan vi kan løse problemer og utføre beregninger med et materialet som CAP. Målet er å systematisk finne nye og bedre metoder for å utnytte det naturlige beregningspotensialet i materialet.Resten av denne seksjonen inneholder en kort gjennomgang av tidligere arbeid som er relevant i forbindelse med utforskning av konfigurasjonssignaler for EiM.

2.4.1 Evolusjonært hovedkort

En av grunnene til at FPGA raskt ble populært i forbindelse med studier av intrinsisk hardware evolusjon var sannsynligvis det faktum at FPGAen opprinnelig var utviklet som et rekonfigurerbart, adaptivt medium. Konfigurasjonsinstillingene for FPGAen kunne dermed representeres som manipulerbare genotyper eller objekter i det digitale domenet. Disse kunne derretter evalueres **uten behov for menneskelig innblanding** ved å implementere innstillingene på FPGAen. Formelt sett kan dette ses på som en automatisering av Gordon Pasks "trial and error"-framgangsmåte. I følge Paul Layzell er forøvrig ulempen med å bruke FPGA at uønskede egenskaper som høy varme og dårlig portabilitet [5]. Med sistnevnte egenskap menes det at en ferdig framstilt krets kun fungerer i miljøet det ble framstilt i.

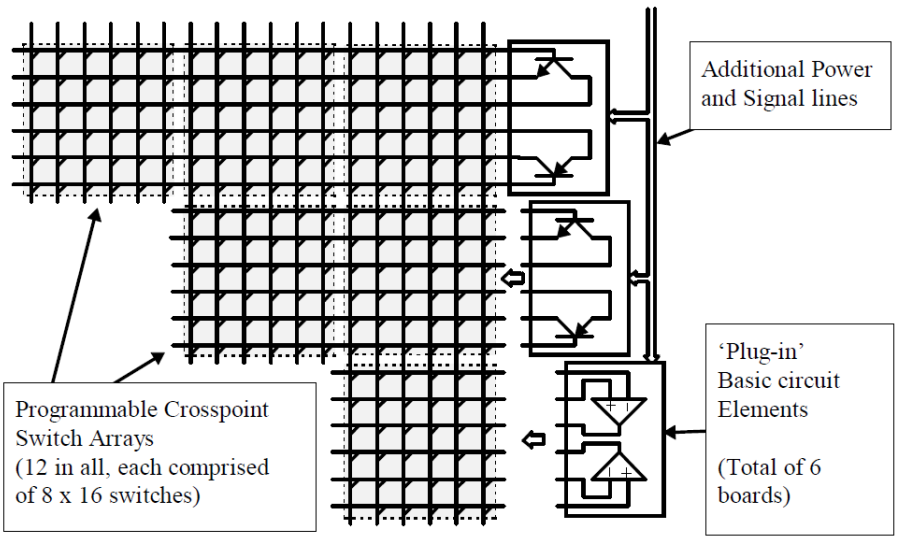

FIGURE 2.4: Layzells EM-plattform[5]

For å løse dette utviklet Layzell et "evolusjonært hovedkort" (EM). Hensikten var å tilby et rammeverk med samme fleksibilitet som FPGA men med muligheter for å måle og generere signaler integrert i plattformen. Figur 2.4 viser en forenklet modell av Layzells plattform. Som figuren viser er plattformen i stor grad påvirket av designprinsippene fra FPGA. Plattformen består av et nettverk av 1500 "switcher" som i følge Layzell gir en genetisk algoritme et søkerom på 10^{420} kretser. Plattformen tillater også at switch-matrisene deles inn i undernettverk, slik at kretsene lettere kan aksesseres og analyseres. Layzell demonstrerer den nye plattformen ved å gjennomføre 3 ulike eksperimenter for å synliggjøre dens potensiale innenfor videre forskning av intrinsisk hardware evolusjon.

2.4.2 Beregninger i Karbon-nanotuber

I følge Langton må et generelt fysisk system være "rik på muligheter" for å kunne utføre beregninger. Denne egenskapen vil derfor være avhengig av hvilke materialer som velges. I den forbindelse har Nichele et al undersøkt hvordan et lovende materiale bestående av karbon-nanotuber (SWCNT) oppfører seg under påvirkning av forskjellige signalformer[6]. Det defineres 3 parametere: intrinsisk, miljø og konstruksjon. Disse antas å ha påvirkning på materialets evne til å utføre beregninger. For å undersøke disse, ble 12 forskjellige signaler av ulike frekvenser generert gjennom plattformen Mecobo. I disse forsøkene representerer plattformen en tonegenerator for systemet. Det eksperimentelle oppsettet er vist på figur 2.5(Mecobo er ikke vist). Gjennom tidsserieanalyser av de genererte signalene ble det konkludert: "with a single square wave input it is possible to observe a rich variety of behaviors while the frequency spectrum is traversed"[6]. I følge Nichele et al kunne ikke-lineære relasjoner observeres i fasebanen mellom inngang og utgang på materialet.

Et spørsmål som forøvrig forblir åpent er **hvilken** oppførsel og hvilken form og grad av "rik variasjon" er optimal? Intuitivt vil et slikt system ha en høy entropi slik Langton beskrev. Det ironiske med dette er at i fysiske systemer representerer entropi den *totale energien* i systemet som ikke kan utføre arbeid[29]. Dette implisserer at systemer med rik oppførsel også er betydelig vanskeligere å **kontrollere**. Den samme tendensen finner vi også i moderne kretsdesign uttrykt som Pollacks regel[30]. En naturlig implikasjon fra Langtons beskrivelser er derfor at det ikke bare er nødvendig å observere en rik variasjon i oppførsel i et velegnet system,

men det må også i stor grad være mulig å kontrollere faseovergangene mellom orden og uorden. Denne egenskapen ble også bemerket av Von Neumann[12], og er essensiell for digitale kretser.

Fra et termodynamisk eller kvantefysisk perspektiv kan oppførselen til fysiske systemer kontrolleres ved å overføre energi fra omgivelsene til systemet[23]. For EiM er denne energien representert i signalene som påføres materialet.

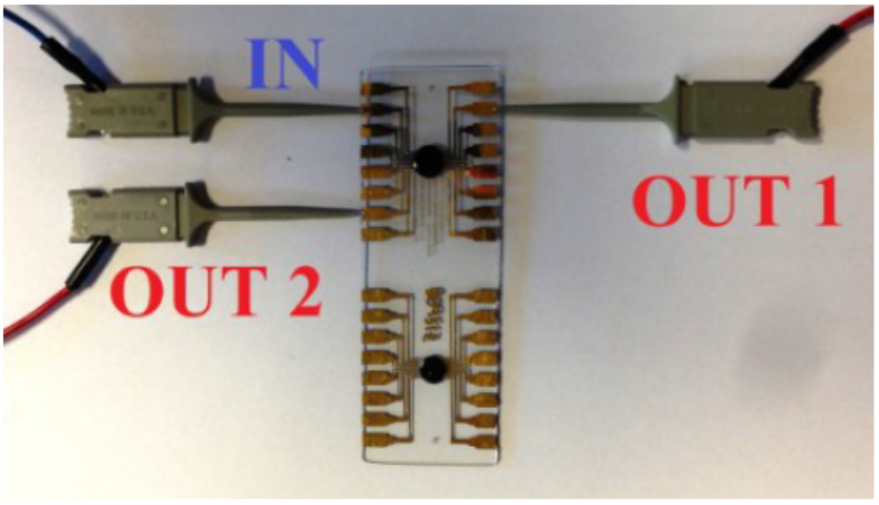

FIGURE 2.5: Nichele et als eksperimentelle oppsett[6]

2.4.3 Evaluering av signalrepresentasjoner

Lykkebø et al[20] evaluerer effektiviteten av firkantbølger og statisk spenning som konfigurasjonssignaler når de påføres karbon-nanorør for å løse et klassisk node-fargingsproblem for fire noder slik at ingen av nabo-nodene har samme farge. En input på 1 bit brukes for å representere de to løsningene som er vist på figuren, og relasjonen (mapping) er direkte mellom genotypen og fenotypen for hvert individ i populasjonen til en genetisk algoritme. For dette forsøket ble derfor konfigurasjonsignalene representert direkte i genotypene til den genetiske algoritmen, og algoritmen ble kjørt for 100 generasjoner.

Resultatet viser at begge representasjonene av konfigurasjonsignal var effektive når det gjaldt å løse problemet. For firkantbølger som signal fant den genetiske algoritmen en fungerende løsning etter omtrent 50 generasjoner med totalt 11 fungerende løsninger etter 100 generasjoner. Signalrepresentasjon gjennom statisk

spenning ga færre løsninger. I dette tilfellet fant algoritmen 1 fungerende løsning etter 100 generasjoner. Da man lot algoritmen variere mellom de to signaltypene ble den første løsningen funnet etter 65 generasjoner, men fitnessverdiene som ble målt indikerte at det var mer effektivt å bruke firkantfunksjoner.

Et viktig moment ved den eksperimentelle metoden som benyttes av Lykkebø et al er at den i prinsippet evaluerer konfigurasjonssignalene der de objektive målene bestemmes av en adaptiv prosess. I prinsippet er dette en induktiv metode som med i prinsippet gjør det mulig å evaluere signalene uten å gjøre antakelser i forbindelse med evalueringskriterier.

2.5 Systemoversikt for Mecobo

Mecobo er en eksperimentell EM-plattform som opprinnelig ble utviklet av Odd Rune Lykkebø[31][32] og er videreutviklet i forbindelse med Nascence-prosjektet[1]. På mange måter kan den ses på som en mer generell utgave av Layzells EM. Plattformen ble utviklet primært for å studere oppførselen til nano-karbon materiale (SWCNT) i forbindelse med EiM, men kan i prinsippet modifiseres til å fungere med ethvert materiale. Figur 2.6 viser en blokkdiagram av arkitekturen for hele systemet. En genetisk algoritme som kjører på klient-maskinen utfører evolusjon på en gruppe med individer, og hvert enkelt individ har en form for genotype. Genotypen til hvert enkelt individ er unik og representerer 1 eller flere kommandoer. Når genotypen skal evalueres, må den først prøves ut på materialet. Kommandoene sendes som pakker fra klient til EM Host PC på figuren. "Host PC" bearbeider pakken og sender kommandoen videre til "Evolvable Motherboard"-modulen. Hver kommando legges deretter i en sortert kø slik det er vist på figur 2.7. Når dataoverføringen er fullført, sender klientmaskinen en siste kommando til "Evolvable Motherboard"-modulen som starter kjøringen av kommandoene i køen.

Resten av denne seksjonen er ment å gi en innføring av hardware for den eksisterende versjonen av Mecobo. Dette er gjort ettersom innholdet av kapittel 3 bygger på denne informasjonen, og fordi det ikke lyktes å finne eksisterende dokumentasjon. Plattformen forklares utifra modellen på figur 2.6.

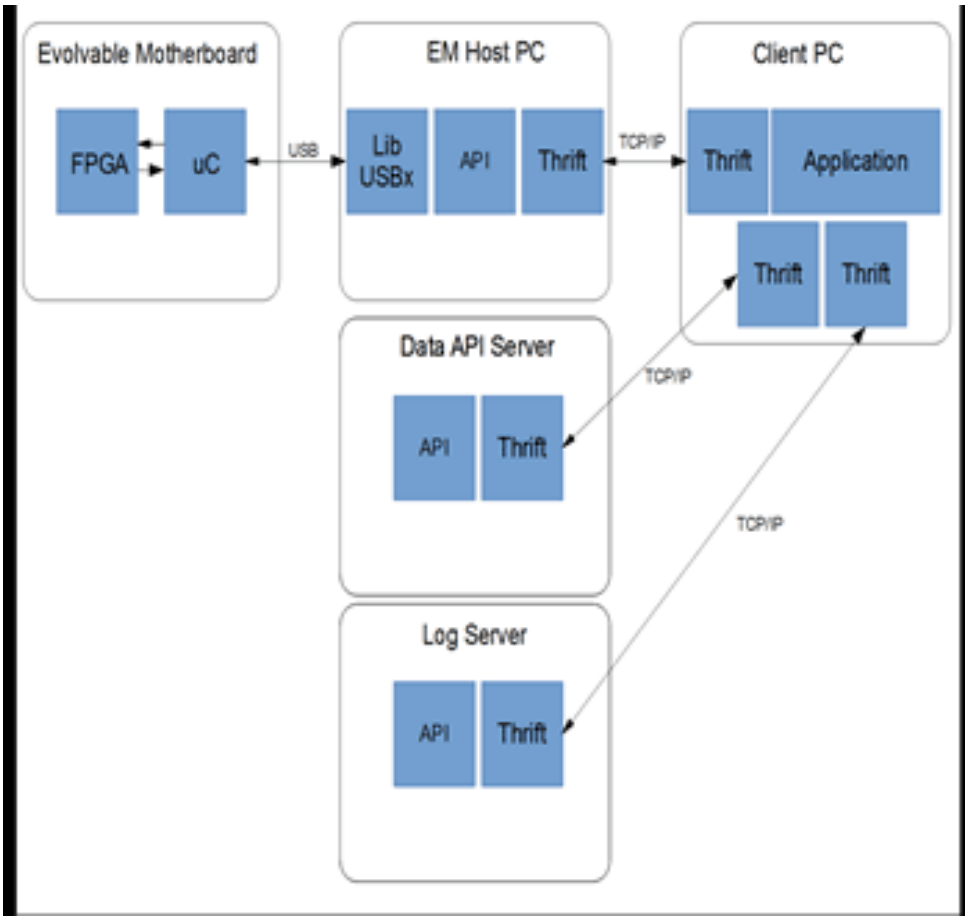

FIGURE 2.6: Figuren[1] viser en oversikt over plattformen Mecobo som brukes å utforske muligheter for ulike typer materialer innenfor EiM.

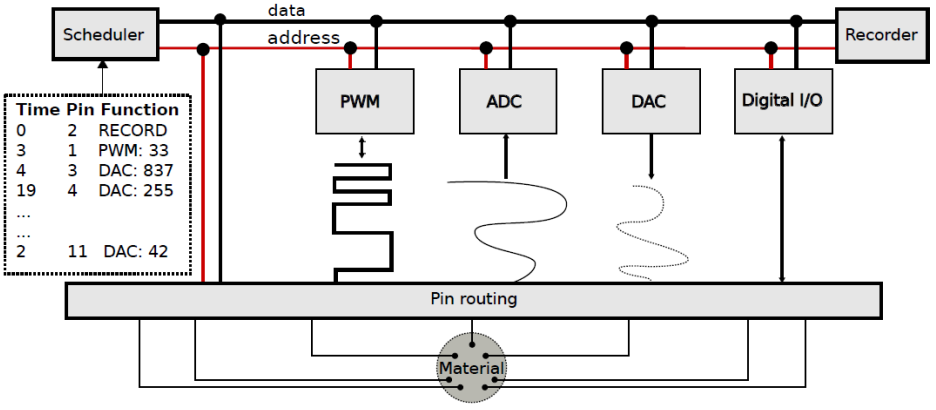

FIGURE 2.7: Figuren[1] viser hvordan Evolvable Motherboard modulen ser ut for en programmerers perspektiv

2.5.1 Mikrokontroller

Mikrokontrolleren fungerer som et grensesitt mellom FPGA modulen og Vertsmaskinen. Med hensyn på å forenkle forklaringen kan Mikrokontrollerens programflyt hovedsakelig deles inn i 2 faser. I den første fasen mottar Mikrokontrolleren et usortert sett med sekvenser fra Vertsmaskinen gjennom USB. For hver sekvens medfølger et tidsstempel som angir start-tid og stopp-tid. Disse sekvensene organiseres i en sekvenskø på samme måte som i figur 2.7. Fasen slutter når alle sekvensene er mottatt og organisert.

Den andre fasen starter ved at Mikrokontrolleren mottar kommandoen $USB\_CMD\_RUN\_SEQ$ fra USB. Denne kommandoen starter en scheduler som sørger for at alle sekvensene i køen itereres og eksekveres etter round-robin. Scheduleren måler hvor lang tid det har gått med en klokkevariabel. Scheduleren gjennomløper både start og stopp-tid i hver av sekvensene. Dersom Schedulerens tid har passert start-tiden i en sekvens, kalles metoden "execute()" som starter eksekvering av sekvensen. Dersom schedulerens tid har passert stop-tiden for en sekvens, kalles metoden "killItem()" som avbryter eller stopper eksekvering av sekvensen. Den siste fasen når mikrokontrolleren har stanset eksekvering av den siste sekvensen.

2.5.2 DAC-kontroller

Kontrollmodulen for DAC er implementert som et "Serial Peripheral Interface"(SPI) som fungerer som et kommunikasjonsgrensesnitt mot DACen. Modulen drives av en klokke med frekvens på 30 MHz. SPI-bussen har en forsinkelse på 19 sykluser mellom inngang og utgang. Dette gir en effektiv samplingsfrekvens på $\frac{30}{19}$ MHz som kan avrundes til 1.57 MHz.

Bitdybden på DACen er effektivt på 8 bit og har en analog spenningsrekkevidde på $5V$ (gitt som absoulttverdi mot en referansespenning). DACen har støtte for inntil 8 forskjellige kanaler gjennom samme SPI-buss. Måten dette gjennomføres på, kan best beskrives som "Time Dimension Multiplexing"(TDM) hvor hver kanal får en "time-share" på bussen[30]. Foreløpig brukes DACen kun for å generere statisk spenning.

DAC-kontrolleren er avbildet i addresserommet fra og med addresse 0x80006400. Mikrokontrolleren benytter metoden "setVoltage()" lokalisert i filen "dac.h" til å

skrive et sample til DAC-kontrolleren via den nevnte addressen. Metoden kalles når en sekvens av typen "DAC_CONST" eksekveres fra sekvenskøen.

2.5.3 ADC-kontroller

ADCen har en bitdybde på 12 bit og en spenningsrekkevidde med en absoluttverdi på $||10V||$. I likhet med DACen har ADCen støtte for inntil 8 ulike kanaler som benytter TDM for deling av bussen. Ved et fast intervall angitt av samplingsraten, skrives et sample ut fra ADCen til kontrolleren. Hvert sample legges deretter i et lokalt buffer på FPGA. Når sekvensene i sekvenskøen er terminert, overføres innholdet av bufferet til Vertsmaskinen via Mikrokontrolleren.

ADC-kontrolleren er avbildet i addresserommet på EBI-bussen på samme måte som DAC-kontrolleren og samplingsregistrene for hver kanal er lokalisert i addresserommet i stigende rekkefølge fra og med 0x800C800 til og med 0x80000D600. Avstanden mellom addressene er 0x200 for hver kanal. Alle kanalene i ADC-kontrolleren forhåndsaktiveres før eksekvering av sekvenskøen, men samples lagres ikke i bufferene før eksekvering av sekvenskøen. Forhåndsaktivering skjer gjennom metoden "executeCurrentPack()" i "main.c". Bufferene aktiveres på globalt basis når en sekvens i sekvenskøen starter eksekvering.

2.5.4 I/O-kontrollere

Kontrollmodulene for I/O styrer en OP-AMP som er avbildet mot en pin mot det fysiske materialet gjennom crossbarene. Som navnet indikerer kommer modulen med støtte for både digital sampling og output. I denne versjonen av Mecobo er det 16 I/O-kontrollere tilgjengelig. Addressen for hver kontrollmodul er lokalisert fra 0x80000000 til 0x80001400.

2.5.5 Crossbar-kontroller

Crossbar-kontrolleren bruker SPI for å kommunisere med 2 crossbar-enheter på brettet (plassert på datterbrettet). Den første crossbarenheten avbilder(mapping) 16 pins mot opptil 16 I/O kontrollere. Hver pin er koblet til en elektrode som er festet på et bestemt sted i materialet. Dette gjør det mulig å programmere hver

I/O kontroller til ett eller flere steder i materialet uten å skifte rekkefølge på de fysiske ledningene.

Den andre crossbar-enheten deler de samme 16 pins som den første crossbar-enheten, og avbilder disse mot opptil 8 DAC-kanaler og 8 ADC-kanaler. I den forbindelse bør det nevnes at 1 pin kan avbildes mot maksimalt 1 IO,DAC-kanal eller ADC-kanal. Kanal 1 i DAC-en kan for eksempel tildeles Pin 1 og Pin 3, men Kanal 1 og Kanal 3 kan ikke tildeles Pin 1. Av samme grunn kan ikke flere enn 16 kontrollere og kanaler sammenlagt, være aktive samtidig.

Avbildning mellom pins og enheter foregår på Vertsmaskinen gjennom klassen "channelmap.h". En avbildning vil finne sted hver gang en ny sekvens legges i sekvens-køen. En representasjon av avbildningen sendes som en pakke over USB med headeren "USB_CMD_PROGRAM_XBAR". Crossbarene konfigureres for en gitt sekvens før kjøring, samtidig som sekvensen mottas i mikrokontrolleren. I koden for mikrokontrolleren skjer dette i metoden "execCurrentPack()".

Chapter 3

Implementasjon

Dette kapittlet beskriver implementasjon av rammeverk og forberedelser som er gjennomført i forbindelse med denne oppgaven. Det presenteres 3 ulike oppdaterte versjoner av Mecobo-rammeverket fra seksjon 2.5. Fra et funksjonelt perspektiv kan de nye oppdateringene for Mecobo generaliseres som Direkte Digitale Synthesizere. Begrepet beskriver et generisk rammeverk som brukes for å skape tilfeldige bølgeformer utifra en referanse-klokke av konstant frekvens[33]. For hver versjon er det implementert en signalgenerator med støtte for generering av sinus, triangel og sagtannfunksjoner. I tillegg er det gjort nødvendige tilpasninger for å bevare eksisterende funksjonalitet. For hvert av designene vil det med hensikt forekomme forvrengning av forskjellig distribusjon og intensitet på signalene som genereres.

3.1 Krav til implementasjonene

For å generere de nye signalene ble følgende metoder vurdert:

- Software/firmware på mikrokontrolleren.
- Egen modul i FPGA som påbygning til DAC-kontrolleren.

Fordelen med den førstnevnte metoden er at den er relativt enkel å implementere. Ulempen er at det kan gå på bekostning av kontroll med hensyn på generering av signalene. Ettersom mikrokontrolleren må håndtere flere andre prosesser kan dette på en ukontrollerbar måte gå utover nøyaktigheten og forutsigbarheten på

Navn	Arkitektur	Maksimal oppløsning	Særtrekk
Design 1	Egendefinert	8 bit	Justerbar oppløsning
Design 2	NCO	8 bit	Gaussisk dithering
Design 3	NCO	5 bit	Svak clipping

TABLE 3.1: Tabellen gir en kort beskrivelse av 3 ulike bølgegenerator-arkitekturer for FPGA. Intensjonen er å se hvordan små periodiske forskjeller på signalene de genererer påvirker materialet.

signalene som genereres. Av hensyn til dette ble derfor de nye DDS-versjonene implementert gjennom FPGA som påbygning til DAC-kontrolleren. I den forbindelse er en fullstendig kravspesifikasjon av implementasjonene utformet i henhold til standard praksis. Følgende funksjonelle krav er satt for alle rammeverkene:

- Generering av sinus, sagtann og triangel-bølgefunksjoner.
- Generering av statisk spenning.
- Mulighet til å veksle mellom ulike frekvenser.
- Mulighet til å veksle mellom forskjellig amplitude.

Implementasjonene må også være robust og kunne håndtere mottak av ugyldig eller dårlig input.

3.2 FPGA

Som tidligere nevnt er det implementert 3 nye versjoner av Mecobo. De nye oppdateringene av FPGA-delen for Mecobo er skrevet i VHDL og Verilog og kan beskrives gjennom figur 3.1. Som figuren viser inkluderer er EM-plattformen oppdatert med en bølgegenerator som er direkte tilknyttet kontroll-modulen for DA-Cen i det eksisterende rammeverket. De viktigste funksjonelle forskjellene mellom de nye implementasjonene er kort oppsummert i tabell 3.1. "Design 2" og "Design 3" har en arkitektur som er basert på "Numerisk kontrollert oscillator" (NCO)[33]. "Design 1" er basert på en egendefinert arkitektur som ble utviklet i forbindelse med denne oppgaven.

Hver bølgeform er kvantisert med en presisjon på 8 bit for i alle designene. Det bør presiseres at begrepet "presisjon" antyder til den faktiske bitdybden for hvert

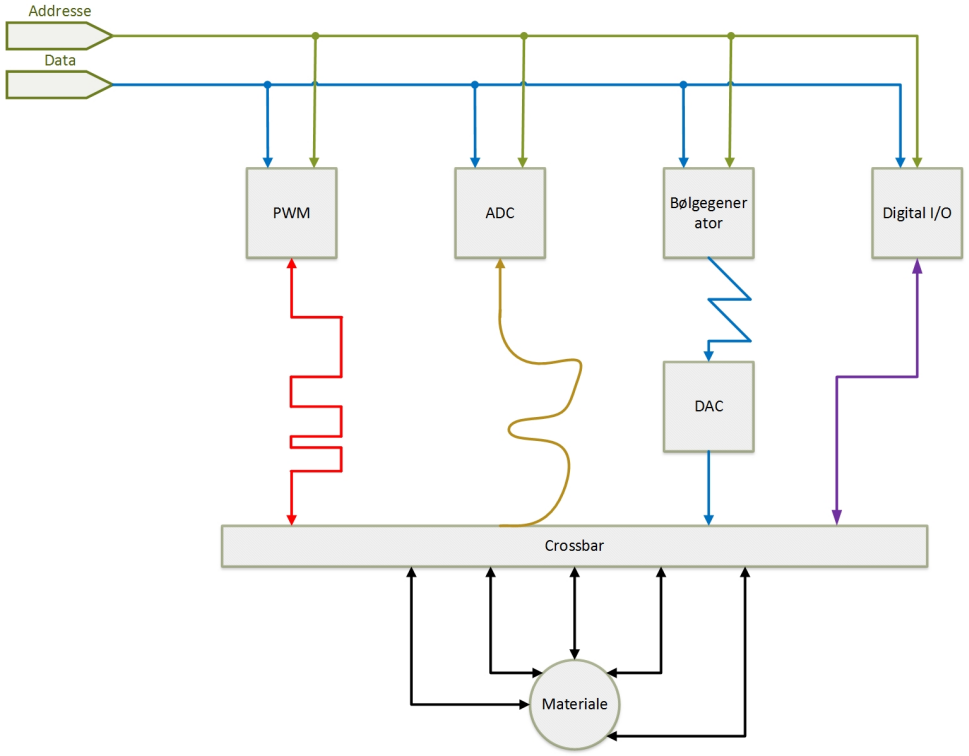

FIGURE 3.1: Figuren viser FPGA-arkitekturen for de nye versjonene av Mecobo. Det eksisterer en tilsvarende kontrollmodul på FPGA for hver av modulene på figuren.

sample. "Maksimal oppløsning" på tabell 3.1 refererer til antall tilstander som effektivt brukes ved generering eller reproduksjon av et signal. Hvis oppløsningen er lavere enn presisjonen er signalet "ikke-uniformt kvantisert" [34]. Under kollonnen "Særtrekk" på tabellen er de funksjonelle forskjellene mellom designene listet. Særtrekkene er unike for hvert DDS-design.

Figur 3.2 viser en generell oversikt over et DDS-design sett fra FPGA. Grensesnittet består av en amplitudekontroller, en scheduler og et grensesnitt (I/O) mot mikrokontroller. Med unntak av små detaljer er implementasjonen av dette grensesnittet lik over alle designene. De funksjonelle forskjellene mellom de 3 designene som vises av tabell 3.1, ligger hovedsakelig i "bølgegenerator"-modulen på figuren.

3.2.1 I/O

I likhet med de eksisterende kontrollmodulene på FPGAen kommuniserer det nye bølgegeneratorgrensesnittet via EBI-bussen. Figur 3.3 viser layout av addresserommet. Hver kjerne i de nye bølgegeneratorene er tilknyttet et register

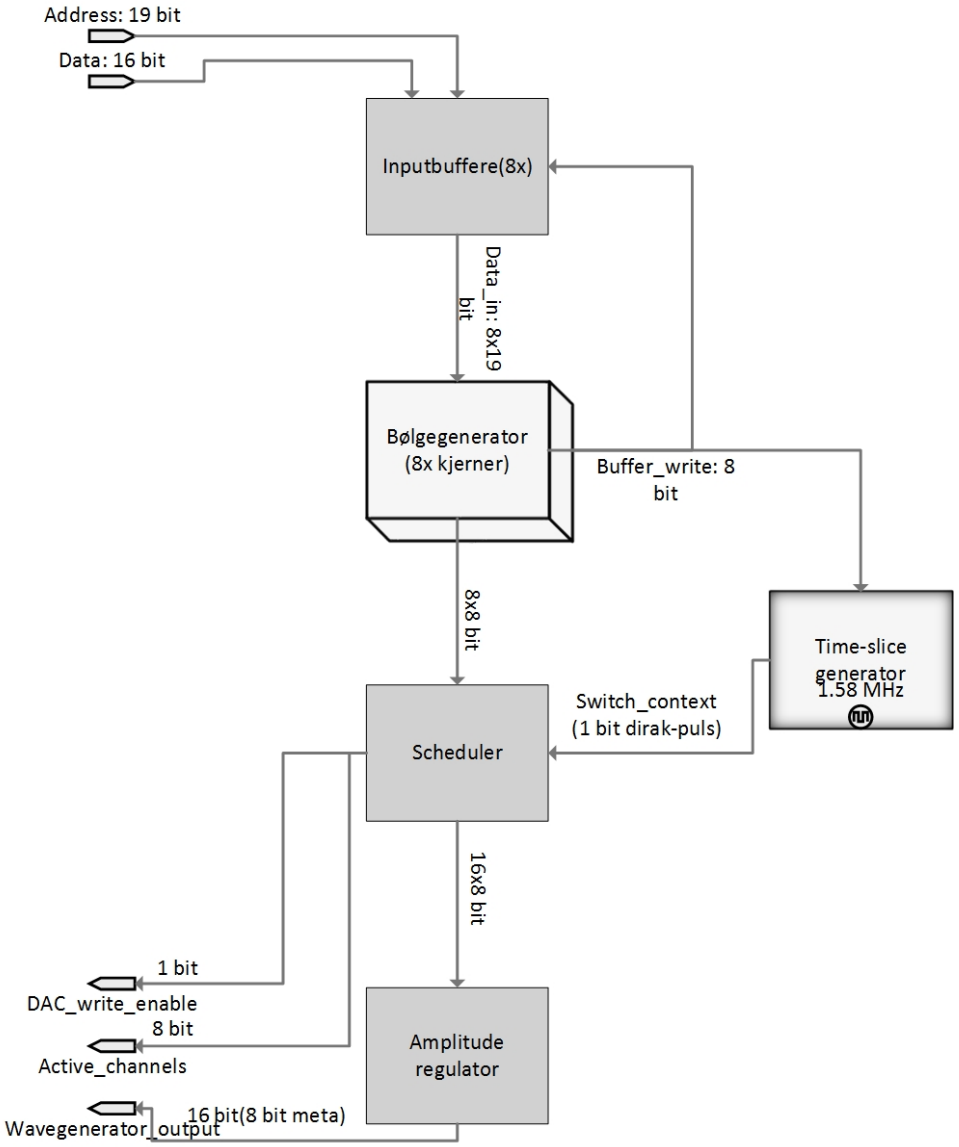

FIGURE 3.2: En generell oversikt over DDS grensesnitt(ene).

som aksesseres gjennom EBI-bussen. Utifra dette registret er det implementert en abstraksjon av 3 databuffere (frekvens, bølgeform og amplitude). Uttrykt som referanse av EBI-bussen kan dette beskrives gjennom figur 3.4.

Databufferene er direkte tilknyttet til platformens EBI-buss slik det er vist på figur 3.5. Det bør nevnes at modellen er noe forenklet ettersom arkitekturen er pipelinet. Inputbuffrenes arkitektur er implementert etter master-slave prinsippet. Slave registrene er minnemappet direkte mot EBI-bussen, mens master-registrene grenser mot bølgegeneratorkjernene og resten av grensesnittet. På denne måten blir innholdet i slave-registrene er usynlig for resten av grensesnittet. Master-registrene tar inn data fra slaveregistrene når det mottar et "write-enable" signal

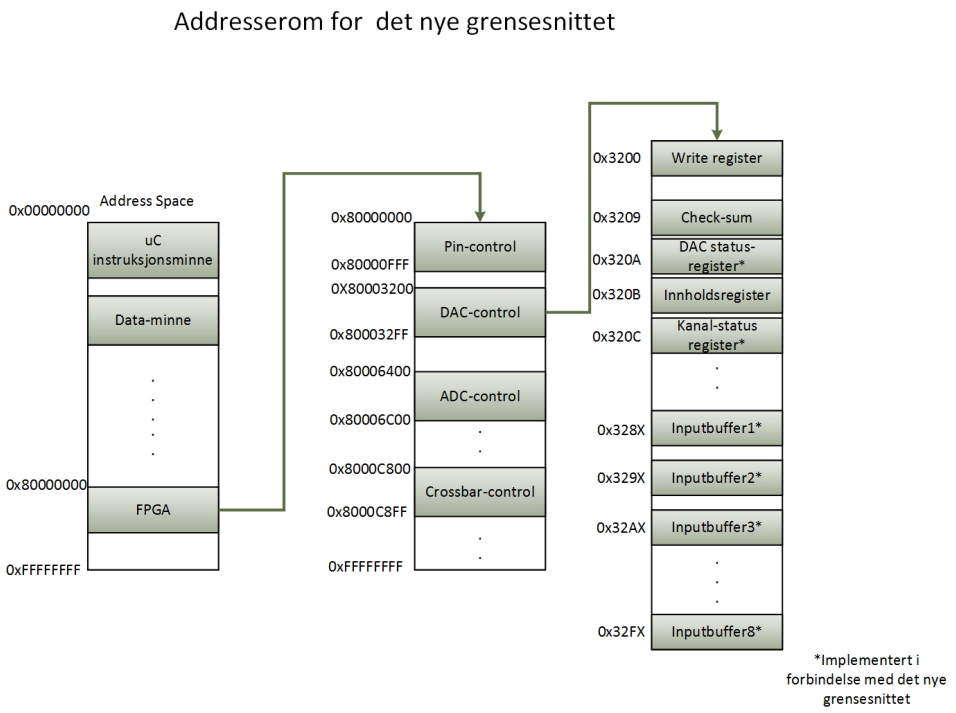

FIGURE 3.3: En oversikt over addresserommet for FPGA på de oppdaterte designene

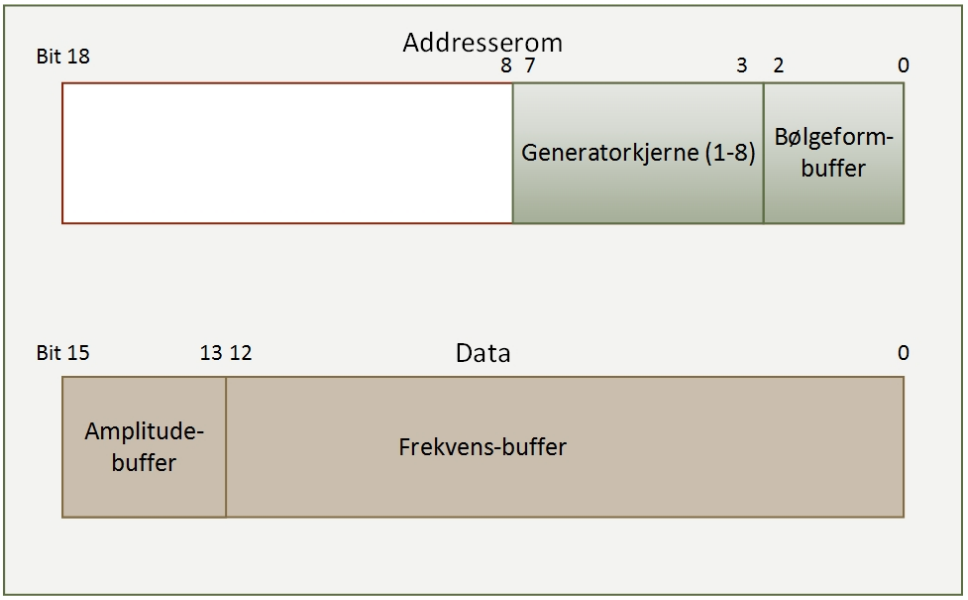

FIGURE 3.4: En visualisering av plasseringen for bufferene

fra generatorkjernen.

I tillegg til bufferene er det implementert et 8 bits "kanal-status"-register. Hvert

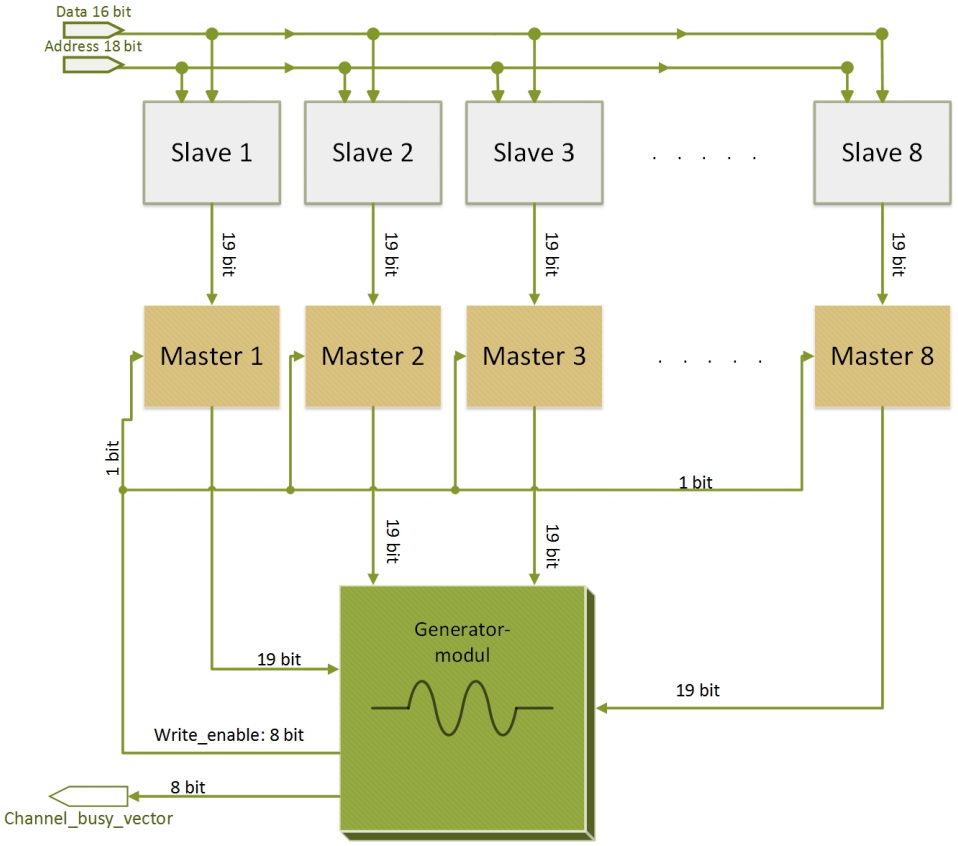

FIGURE 3.5: En beskrivelse av arkitekturen for I/O grensesnittet

bit indikerer om en generatorkjerne er aktiv. Registret benyttes internt på FPGAen av andre moduler i grensesnittet. En avlesbar kopi av registret er også implementert i kontrollmodulen for DACen, og kan aksesseres gjennom EBI-bussen ved å lese av offset 0xC i DAC-kontrollmodulens addresserom (se figur 3.3). Registret representerer generatorkjernene og DAC-kanalene etter "Little Endian" konvensjonen.

3.2.2 Scheduler

Som nevnt i seksjon 2.5 er DACen i stand til å håndtere intil 8 kanaler. For å skrive til flere enn 1 kanal samtidig via SPI-grensesnittet brukes tidsdimensjonsmultiplexing (TDM)[30]. Av hensyn til dette ble en round robin scheduler implementert. Scheduleren veksler mellom output fra intil 8 aktive kjerner samtidig slik figur 3.6 viser. Med "aktiv kjerne" menes at kjernen aktivt opptar en kanal i DACen.

Scheduleren gjenkjenner en aktiv kjerne gjennom "kanal-status"-registeret, som signaliserer hvilke kjerner som er aktive. Nyquistfrekvensen for hver kanal er gitt utifra formel 3.1. Variablen $kanaler_{robin}$ representerer antall aktive generatorkjerner.

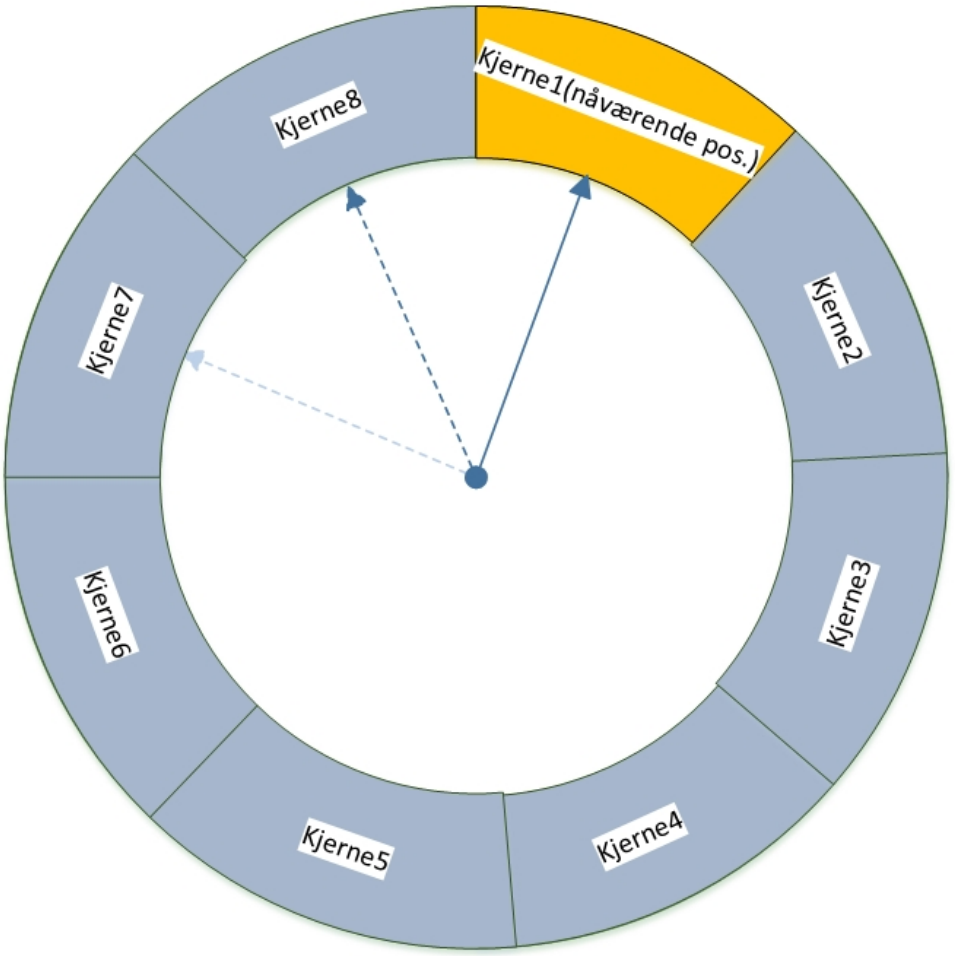

FIGURE 3.6: En visualisering av round-robin scheduleren

$$f_{nyquist} = \frac{f_{sampling}}{2 \cdot kanaler_{robin}} \tag{3.1}$$

3.2.3 Amplitudekontroller

Før hvert sample overføres til DAC-kontrolleren reguleres amplituden for bølgeformen. Amplituden reguleres ved å dividere hvert sample med en konstant. Amplituden som lagres i hvert sample er gitt av formel 3.2. $|\cdot|$ i formelen representerer

innholdet av hvert sample, og a er en periodevis konstant verdi angitt av amplitudebufferet som ble presentert tidligere. Variabelen a er definert for det diskrete intervallet $[0, 6]$.

$$Amplitude = \frac{|sample_{input}|}{2^a} \tag{3.2}$$

3.2.4 Design 1: Digital faselåst løkke

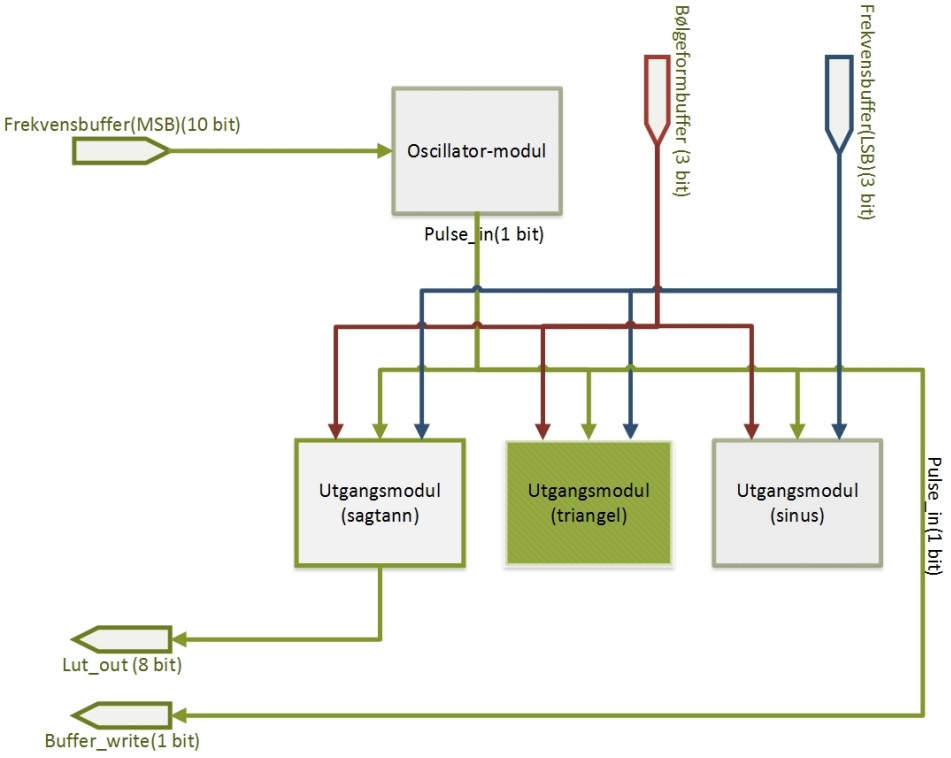

FIGURE 3.7: Arkitektur for Design 1, vist for 1 kjerne

Det første designet er basert på en egendefinert arkitektur, men kan klassifiseres som en digital form for Voltage Controlled Oscillator (VCO)[33]. For ordens skyld (med noe misbruk av begrep) kan vi klassifisere Design 1 som digital kontrollert oscillator (DCO). Designet består totalt av 8 kjerner, der hver kjerne driver én enkelt DAC-kanal. Blokkdiagrammet på figur 3.7 viser en oversikt over en enkelt kjerne. De ulike komponentene fra figuren er forklart nedenfor.

3.2.4.1 Oscillator

For dette designet benyttes en digital oscillator for å kontrollere frekvensen for bølgeformene som genereres av utgangsmodulene. Oscillatoren drives av en klokke på 75 mHz og tar de 10 mest signifikante bitsene av frekvensbufferet (se figur 3.2) som input. Output er et Dirac pulstog og kan beskrives gjennom formel 3.3. Variabelen t representerer klokken som driver oscillatoren, mens fw representerer innholdet av frekvensbufferet. Variablen "T" representerer tidsrommet oscillatoren er aktiv for en glatt t og en gitt fw.

$$\delta_{fw}[t] = \sum_{k=0}^{k=T} \delta(t - k \cdot fw),\ 0 < fw < 1024 \tag{3.3}$$

Alternativt kan oscillatoren også uttrykkes som Mealy tilstandsmaskin. Et diagram for denne er vist på figur 3.8. Når kjernen er aktiv veksler maskinen mellom tilstandene "countdown" og "reset". I Reset-moduset generes en aktiv puls som også fungerer som et signal for inputbufferene.

3.2.4.2 Utgangsmodulene

$$Sample(t) = x(t) \cdot \delta_{fw}[t] = x(t) \sum_{k=0}^{k=T} \delta(t - k \cdot fw) \tag{3.4}$$

For å støtte et utvalg av 3 bølgeformer har hver generatorkjerne 3 utgangsmoduler. Når en bølgeform er valgt, oppdateres den respektive utgangsmodulen i en frekvens angitt av pulstoget fra oscillatormodulen. Vi kan utvide formel 3.3 slik at samples generert fra utgangsmodulen er uttrykt gjennom formel 3.4. I hver modul er det et tilstandsregister, der dets innhold er uttrykt som funksjonen x(t) i formlen. Tilstandsregistrene har en lengde på 8 bits. Det bør også nevnes at formelen gjelder for det digitale domenet, og tar derfor ikke hensyn til rekonstruksjon.

Utgangsmodulen for sagtann-bølgeformen består kun av tilstandsregistret. Dersom vi tar i betraktning at lengden på tilstandsregistret er 8 bit kan sagtannfunksjonen uttrykkes ved å utvide formel 3.4 med formel 3.5. Variabelen n påvirker både signalets frekvens og oppløsning og representerer de 3 minst signifikante bitsene i frekvensbufferet.

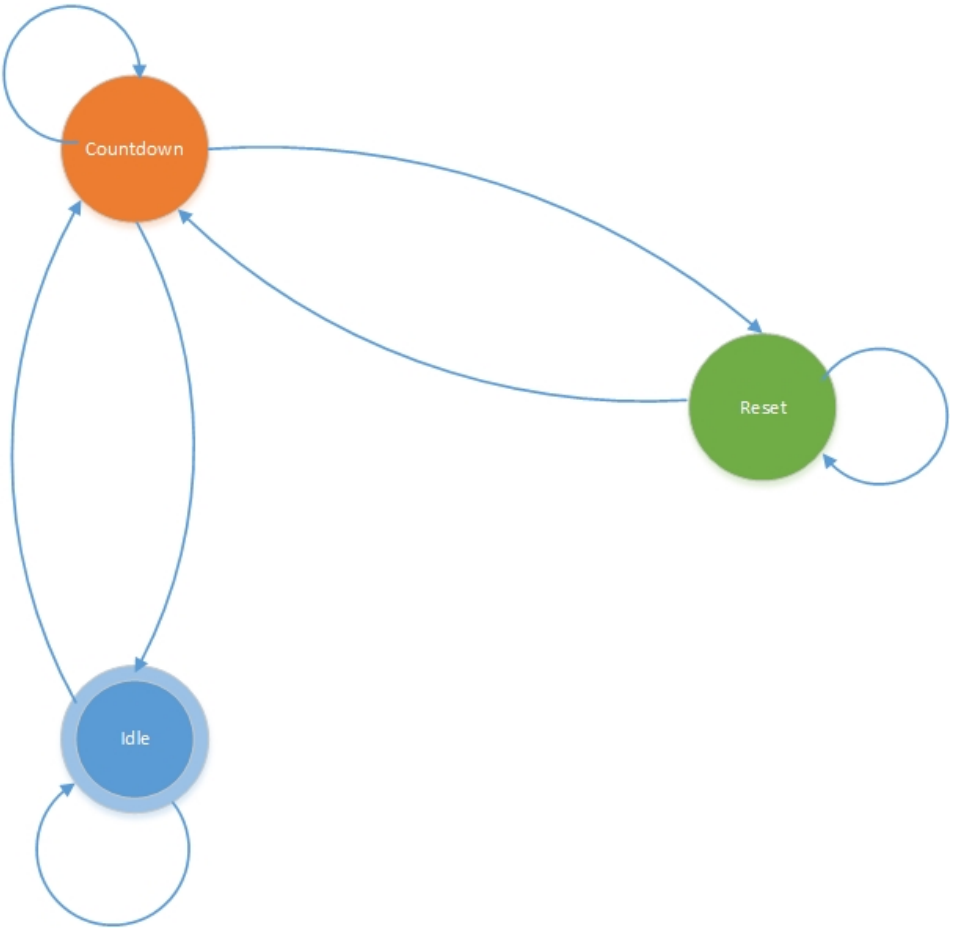

FIGURE 3.8: Tilstandsmaskin for oscillatormodulen

$$x(t) = t \cdot 2^n \ mod\ 256,\ 0 \leq n \leq 6 \tag{3.5}$$

Utgangsmodulen for triangel-bølgeformen består av et tilstandsregister samt en tilstandsmaskin som triggres ved overflyt av tilstandsregistret. Oppførselen til utgangsmodulen kan beskrives ved å utvide formel 3.4 med formel 3.6. Ettersom triangelfunksjonen er har 2 knekkpunkter representerer formelen 2 ulike tilstander. Alternativt kan oppførselen til modulen forklares utifra figur 3.9, der triangelbølgeformen genereres gjennom 1 ordinær og 1 omvendt sagtannfunksjon.

$$x(t) = \begin{cases} t \cdot 2^n \ mod\ 256 & \text{gitt } t \cdot 2^n \ mod\ 512\ < 256 \\ 255 - (t \cdot 2^n \ mod\ 256) & \text{gitt } t \cdot 2^n \ mod\ 512\ \geq 256 \end{cases} \tag{3.6}$$

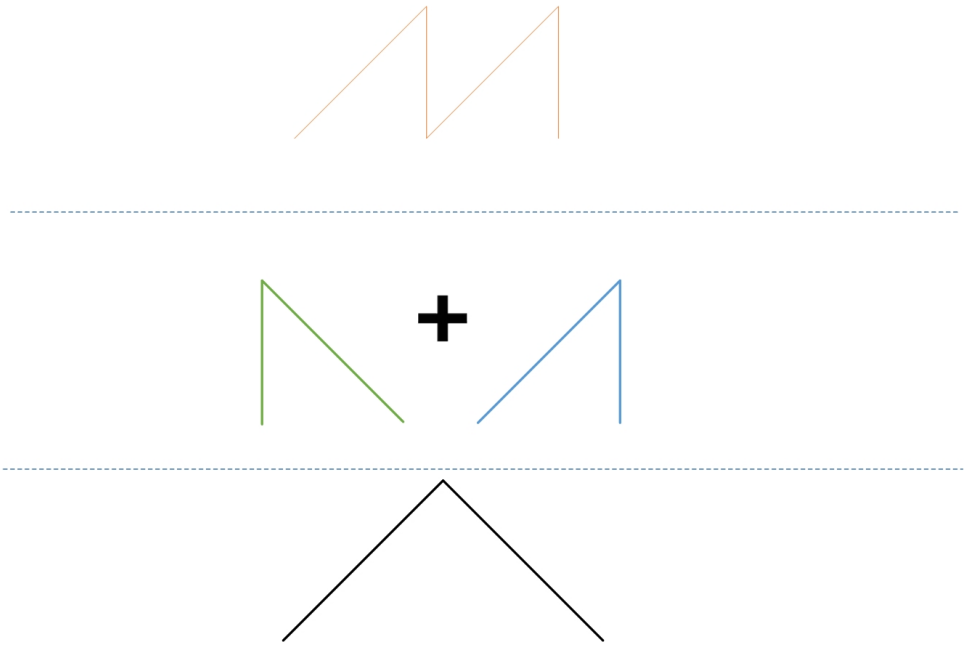

FIGURE 3.9

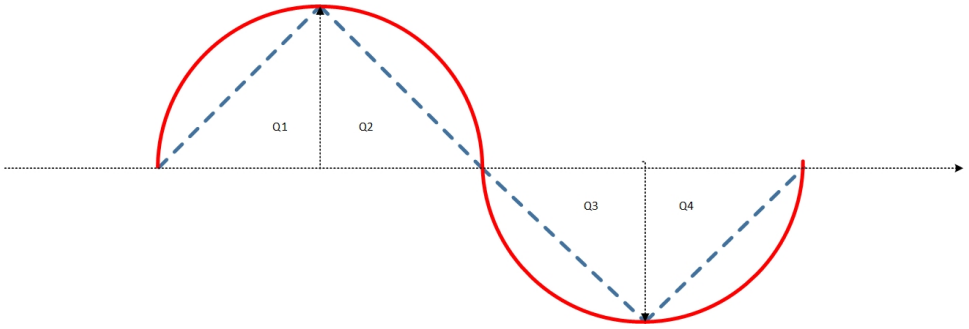

FIGURE 3.10

Utgangsmodulen for sinus er implementert slik formel 3.7 viser. Variabelen $\hat{x}(t)$ representerer funksjonen $x(t)$ fra formel 3.6, mens funksjonen $Tabell()$ representerer en sinustabell med et 7 bits addresserom. Tabellen har 4 "kvadranter" og sammenhengen mellom $\hat{x}(t)$ og $x(t)$ fra formel 3.7 er illustrert gjennom figur 3.10. Denne teknikken gir signalet samme kvalitet som om tabellen skulle hatt et addresserom på 9 bit, hvilket reduserer den nødvendige plassen tabellene opptar på FPGA med en fjerdedel.

$$x(t) = \begin{cases} Tabell(\hat{x}(t))_{sinus} & \text{gitt } \hat{x}(t) \ mod \ 1024 \ < 512 \\ -Tabell(\hat{x}(t))_{sinus} & \text{gitt } \hat{x}(t) \ mod \ 1024 \ \geq 512 \end{cases} \tag{3.7}$$

3.2.5 Design 2: NCO med dithering

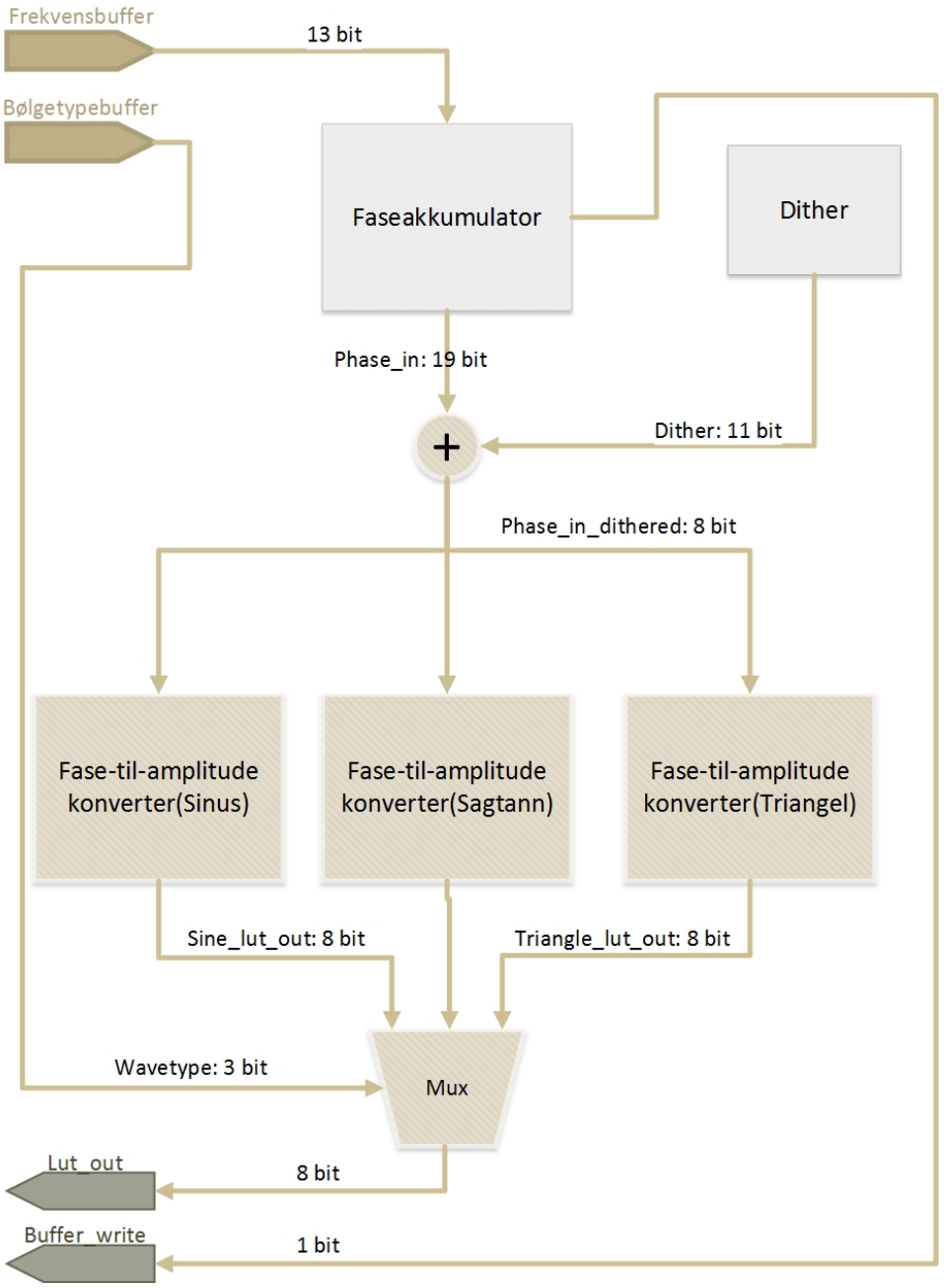

FIGURE 3.11: Figuren viser arkitekturen for 1 av de 8 kjernene. Alle kjernene har identisk arkitektur.

"Design 2" er implementert som en numerisk kontrollert oscillator (NCO) [33]. En framstilling av arkitekturen er vist på figur 3.11. Den resulterende bølgeformen får en frekvens som avhenger av verdien i frekvensbufferet, frekvensen på klokken som driver rammeverket samt perioden for den valgte bølgeformen. Dette er uttrykt gjennom formel 3.8.

$$f_{NCO} = \frac{\|f_{frekvensbuffer}\| \cdot f_{clk}}{2^{19} \cdot P_{periode}} \tag{3.8}$$

3.2.5.1 Faseakkumulator

$$f_{PA} = \frac{\|f_{frekvensbuffer}\|}{2^{19}} \cdot f_{clk} \tag{3.9}$$

Faseakkumulatoren består hovedsakelig av en 19 bits "adder", et 19 bits register samt en tilstandsmaskin. Innholdet i registret endres over tid i en sagtann-form slik det er vist på figur 3.12. Ved hver klokkehendelse (rising edge) summeres innholdet av frekvensbufferet og faseakkumulator-registret. Etter et visst antall klokkesykluser vil verdien i registret bli større enn $2^{19} - 1$, noe som forårsaker overflyt (overflow). Gjennomsnittlig frekvens for inntreffelse av overflyt er angitt i formel 3.9. f_{clk} er i dette tilfellet frekvensen på klokken som driver designet.

$$f_{GRR} = \frac{2^{19}}{GCD(|frekvensbuffer|, 2^{19})} \tag{3.10}$$

Når overflyt inntreffer i faseakkumulatoren oppstår en restverdi. Denne restverdien blir gjenværende i registret slik at den nye perioden starter fra en verdi som er større enn 0. Etter en viss tid ved sample S vil faseakkumulatoren-registret innholde den samme verdien som den inneholdt i begynnelsen. Dette intervallet kalles "Grand Repetition Rate" og er angitt gjennom formel 3.10.

3.2.5.2 Dithering

Som figur 3.12 illustrerer blir innholdet av faseakkumulator-registret kvantisert. Dette skjer fordi innholdet i registret blir avkortet (truncation) før det konverteres til amplitude. Som følger av dette oppstår *phase truncation spurs*[33]. I følge [35] kan harmoniske avhengigheter som oppstår i støyspektret som følger av dette, reduseres ved å addere gaussisk hvit støy i de $W + 1$ minst signifikante bits av faseregistret før avkortning. Variablen "W" representerer totalt antall bits som avkortes mellom faseakkumulatoren og fase-til-amplitude konverteren. Det finnes forøvrig mange sofistikerte måter å gjøre dette på, men målet i dette tilfellet er kun å redusere forvrengning med en **viss** grad.

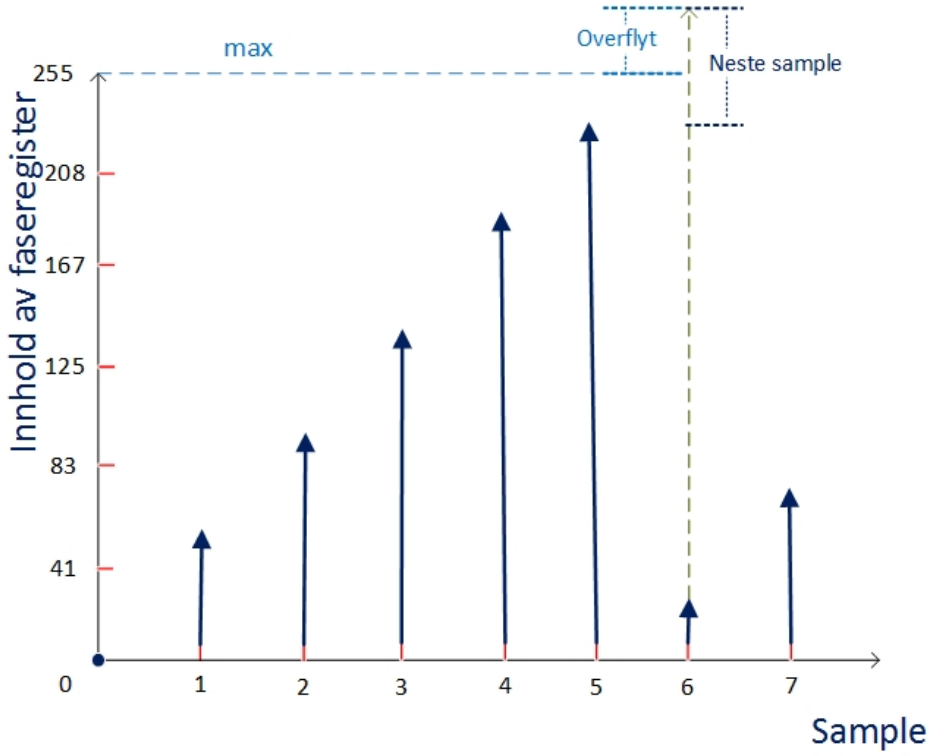

FIGURE 3.12: Visualisering av faseakkumulatorens oppførsel

I følge Sentralgrenseteoremet presentert i formel 3.11 vil det aritmetiske gjennomsnittet av en stokastisk prosess av "n" uavhengige stokastiske variabler være tilnærmet normalfordelt når "n" går mot uendelig. I følge [36] er det tilstrekkelig å anta at gjennomsnittet er tilnærmet normalfordelt for n=20 dersom variablene er uniformt fordelt.

$$\frac{X_1 + X_2 ... + X_n}{n} \rightarrow N(\mu, \frac{\sigma^2}{n}) \tag{3.11}$$

For denne arkitekturen ble dithering implementert som en stokastisk prosess bestående av 16 uavhengige variabler (n=16), ettersom dette var mest fornuftig med hensyn på kompleksitet og resursbruk på FPGA. Et blokkdiagram av arkitekturen er vist på figur 3.13. Dithering genereres gjennom en array bestående av 16 moduler, hvor hver modul genererer et tall med en størrelse på 10 bit fra en uniformfordeling. Hver modul består i realiteten av et shift-register kombinert med en xor-funksjon, og initialiseres med forskjellige startverdier (seeds).

Det bør også nevnes at et fenomen oppstår når samplene overføres til DAC-en, ettersom DAC-modulen drives av en annen klokke. Dette gjelder forøvrig for alle

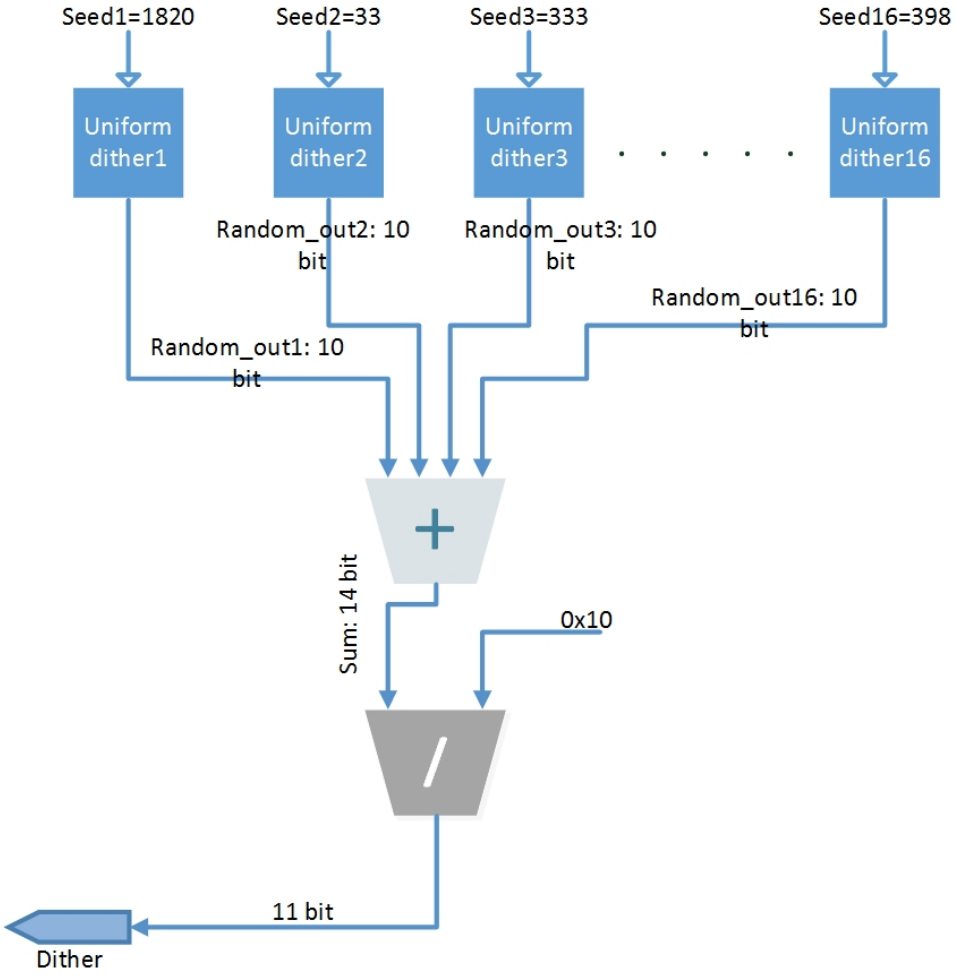

FIGURE 3.13: Ditherarkitekturen for Design 2

DDS-designene.

3.2.5.3 Fase-til-amplitude konverter

Som tidligere nevnt tar Fase-til-amplitude konverteren det lineæriserte signalet fra faseakkumulatoren og konverterer det til en amplitude. I en generatorkjerne er det èn fase-til-amplitude konverter for hver bølgeform.

I Fase-til-amplitude modulen for sagtann gjøres det i hovedsakelig ingen funksjonelle endringer på signalet ettersom signalet allerede er en sagtannfunksjon (se figur 3.12). Triangelfunksjonen består av 2 sammensatte sagtannfunksjoner på samme måte som i Design 1. I dette designet generes de nødvendige sagtannfunksjonene i faseakkumulatoren, der funksjonen inverteres for en av periodene. Modulen tar en vektor bestående av de 9 mest signifikante bits fra faseakkumulatoren. Av denne

vektoren representerer de 8 minst signifikante bitsene output, mens det mest signifikante bitet er et "overflow"-bit. Verdien i overflow bitet inverterer verdien fra faseakkumulatoren slik figur 3.14 viser.

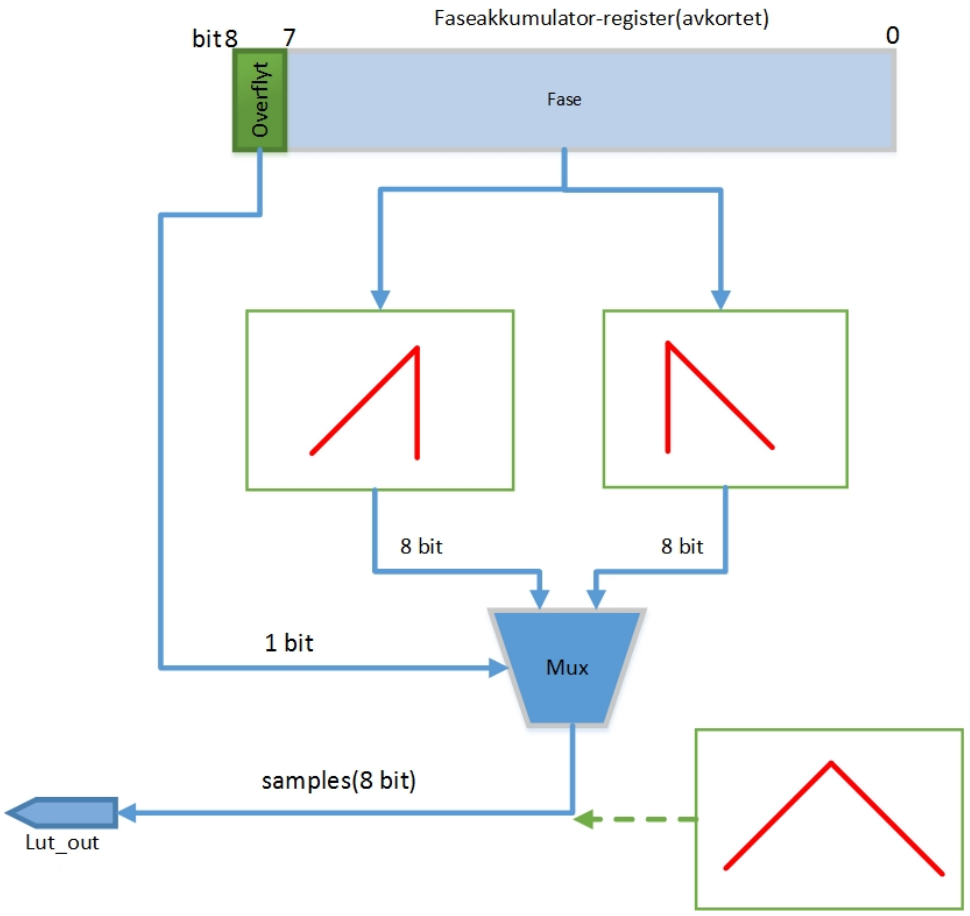

FIGURE 3.14: Abstraksjon av fase-til-amplitude konverteren for triangel i den implementert NCO arkitekturen

Sinus funksjonen genereres gjennom en lookup-tabell med 256 verdier (8 bits addresse) hvor hver verdi har en maksimal størrelse på 7 bit. Tabellen inneholder en fjerdedel av sinusfunksjonen som skal genereres. Modulen tar en vektor bestående av de 10 mest signifikante bits fra faseakkumulatoren. De 2 mest signifikante bitsene representerer de 4 nødvendige tilstandene som kreves for å generere en hel periode, mens resten av vektoren brukes som addresse for tabellen. Figur 3.15 viser hvordan signalet genereres. Bit 8 inverterer verdien på de 8 minst signifikante bitsene slik innholdet av tabellen speiles om Y-aksen på figuren. Mens bit 9 speiler de 2 første kvadrantene fra lookup-tabellen om X-aksen ved å invertere amplituden. Hver av kvadrantene har en maksimumsresolusjon på 7 bit. Når disse legges sammen får det resulterende signalet en bitdybde på tilsammen 8 bit.

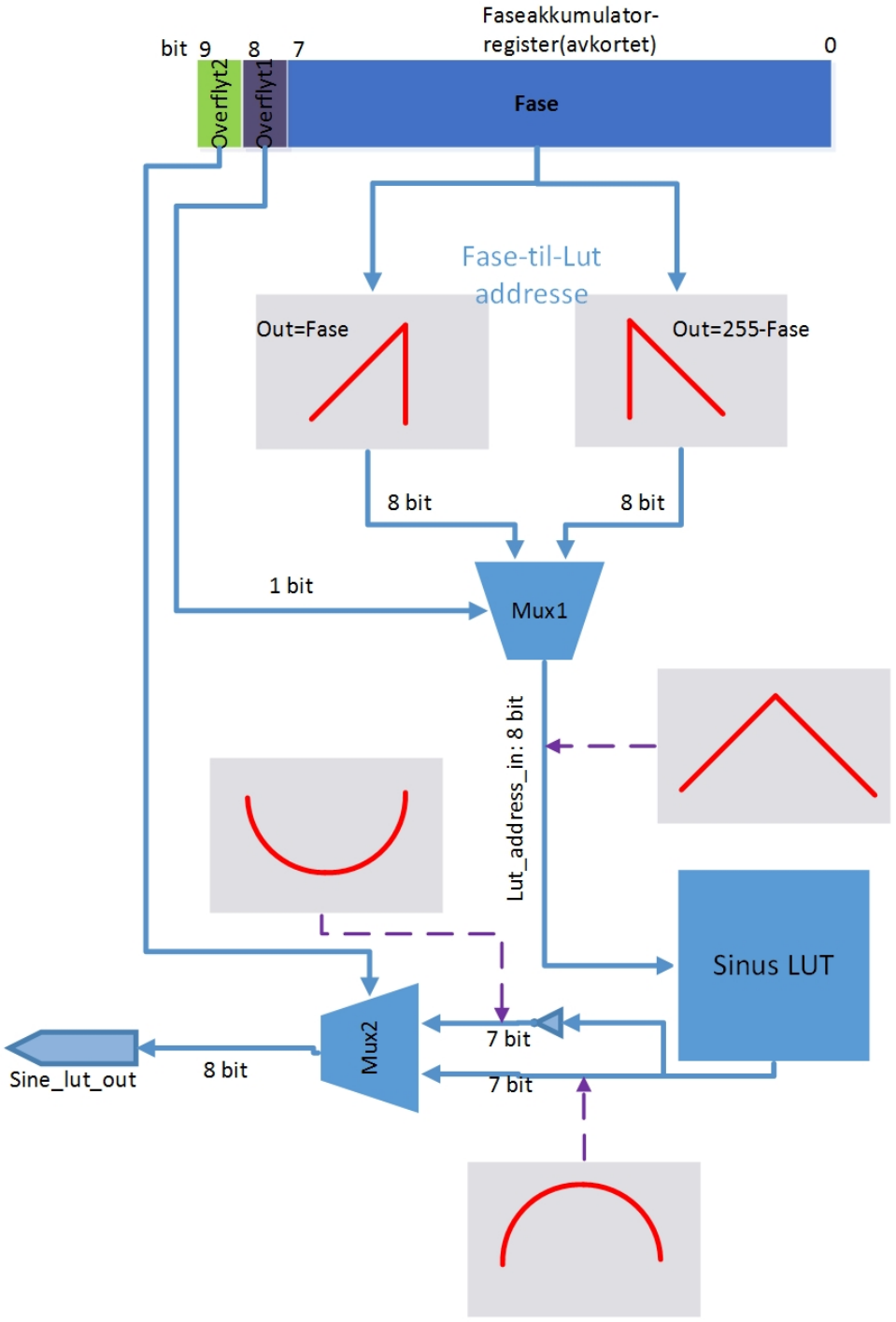

FIGURE 3.15: Abstraksjon av fase-til-amplitude konverteren for sinus i den implementert NCO arkitekturen

3.2.6 Design 3: NCO med clipping

En oversikt over arkitekturen på Design 3 er vist på figur 3.16. Som figuren antyder har denne implementasjonen mange likhetstrekk med design 2. Dette er fordi

Design 3 implementert med samme arkitektur (NCO) og grensesnitt. Forskjellene på implementasjonene er listet opp under.

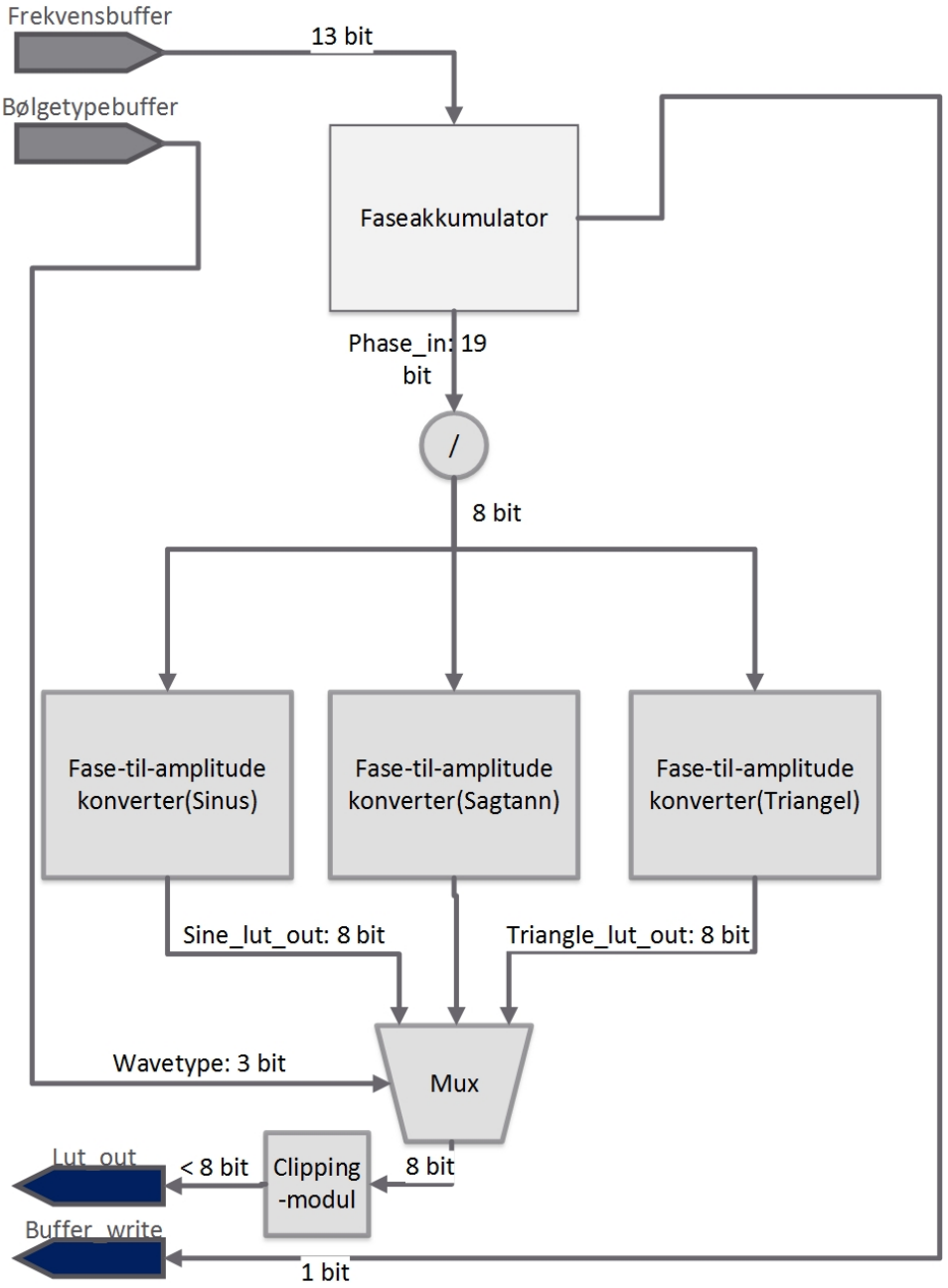

FIGURE 3.16: Oversikt over arkitekturen for 1 kjerne i Design 3

- Ingen dithering mellom faseakkumulator og fase-til-amplitude konverter.
- Avkortning av utgangsvektoren fra faseakkumulatoren er økt fra 11 til 15 bit.

- Størrelsen på lookuptabellen i fase-til-amplitude konverteren for sinusfunksjonen er redusert fra 256 til 16 verdier (8 bits kvantisering).
- Begrensning av dynamisk rekkevidde på de genererte bølgeformene.

I dette designet reduseres bredden for faseakkumulator-registret gjennom avkortning av de minst 15 signifikante bitsene. Dette betyr at maksimal oppløsning i fase-til-amplitude konverteren i de 8 generatorkjernene er redusert sammenliknet med Design 2. Maksimal oppløsning gitt i antall samples for hver periode er 2^{19-15} for sagtann. På grunn av overflyt-bitet for triangelfunksjonen (se figur 3.14) er maksimal oppløsning $2^{(19-15)+1}$ samples for hver periode. Maksimal oppløsning for sinus er av samme grunnlag $2^{(19-15)+2}$ samples for hver periode (se figur 3.15).

I tillegg begrenses den dynamiske rekkevidden for de genererte bølgeformene slik det er uttrykt i formel 3.12. Variablen A representerer amplituden fra fase-til-amplitude konverteren, og $A_{clipped}$ representerer amplituden etter den påførte begrensningen. Dette er implementert som et digitalt filter og er plassert etter multiplekseren slik det er vist på figur 3.16. Begrensning av amplituden eller "clipping" utføres for å påføre harmonisk forvrengning på signalet.

$$A_{clipped} = \begin{cases} A & \text{for } 12 < A < 244, \\ 13 & \text{for } A \leq 12, \\ 243 & \text{for } A \geq 244 \end{cases} \tag{3.12}$$

3.2.7 Thrift

For kommunikasjon mellom mikrokontroller og vertsmaskin bruker det eksisterende designet et grensesnitt kalt basert på Apache Thrift [37]. Thrift er et generisk grensesnitt som blant annet gjør det mulig å generere samme kode på forskjellige programmeringsspråk. For dette tilfellet benyttes Thrift for å generere en felles kommunikasjonsprotokoll over USB. Protokollen er lagret i filen "mecoprot.h" og filen i sin helhet fungerer på samme måte som transportklasser i andre kodespråk. I forbindelse med implementasjon av den oppdaterte versjonen av Mecobo er protokollen utvidet med en ny konstant for hver nye bølgeform. Konstantene er markert i rødt på figur 3.17. Disse gjør det mulig for mikrokontrolleren å rekonstruere pinkonfigurasjoner som mottas gjennom USB dersom de er assosiert med de nye bølgeformene.

```
#define LED_MODE 0
#define LED_SELECT 1

#define PINCONFIG_DATA_TYPE_DIGITAL_OUT 10
#define PINCONFIG_DATA_TYPE_PREDEFINED_PWM 11
#define PINCONFIG_DATA_TYPE_DAC_CONST 12
#define PINCONFIG_DATA_TYPE_RECORD 13
#define PINCONFIG_DATA_TYPE_RECORD_ANALOGUE 14
#define PINCONFIG_DATA_TYPE_PREDEFINED_SINE 15
#define PINCONFIG_DATA_TYPE_CONSTANT_FROM_REGISTER 16
#define PINCONFIG_DATA_TYPE_DIGITAL_CONST 17
#define PINCONFIG_DATA_TYPE_PREDEFINED_TRIANGLE 18
#define PINCONFIG_DATA_TYPE_PREDEFINED_SAWTOOTH 19
```

FIGURE 3.17

Funksjoner:
activate_wavegenerator_channel()
clear_wavegenerator()
clear_wavegenerator_channel()
deactivate_wavegenerator_channel()
reset_to_constant_voltage()
setup_wavegen()

TABLE 3.2: Tabellen viser en oversikt over nye funksjoner som er implementert på mikrokontrolleren.

3.2.8 Mikrokontroller

Mikrokontrolleren på plattformen er utvidet med ny funksjonalitet for å støtte det nye bølgegenerator-grensesnittet. Utenom å fungere som et grensesnitt mellom FPGA og "Host" er mikrokontrollerens formål å sørge for at hver generatorkjerne starter og stopper i korrekt tidsrom. Tabell 3.2 viser en oversikt over nye funksjoner som er implementert. Tabellen inkluderer forøvrig ikke eksisterende funksjoner som har blitt endret eller tilpasset de nye designene.

Den funksjonelle rollen til mikrokontrolleren på Mecobo en scheduler som fordeler de ulike ressursene på FPGAen.

Den viktigste oppgaven til mikrokontrolleren sett fra et funksjonellt perspektiv er å sørge for at sekvenser starter og avsluttes i henhold til definert tid. I den forbindelse er det svært viktig å skille mellom sanntid og brukertid[]. Brukertid er i dette tilfellet den tiden som brukes på å kjøre en gitt sekvens. Ideelt sett er dette nøyaktig det samme tidsvinduet som er angitt for sekvensen. Sanntid er den totale tiden systemet bruker på å kjøre sekvensen, og inkluderer for eksempel ventetid for håndtering av andre sekvenser eller ventetid som følger av interruptrutiner.

Den viktige faktoren er brukertiden. Dette kan forklares gjennom følgende eksempel. En sinus-sekvens er angitt med en starttid på 100 ms og en slutttid på 200 ms. Dette tilsvarer en forventet brukertid på 100 ms. Så lenge starttid og sluttid overholdes er det uproblematisk fra et funksjonelt ståsted at sekvensen kjøres med en sanntid på 500 ms. Det riktignok ta lengere tid enn 100 ms å kjøre sekvensen, men dette påvirker i prinsippet ikke materialet på noe som helst vis. Det er derimot et problem dersom forventet brukertid er på 100 ms, mens faktisk brukertid er på 200 ms. For sanntidssystemer kalles dette en deadline-miss. Problemet i denne situasjonen er at sekvensen kjører i det dobbelte av den angitte tiden. I denne situasjonen er det heller ønskelig at den overskytende brukertiden reduseres så mye som mulig. For å oppnå dette for den nye funksjonaliteten er kjøring av sekvenser med de nye bølgeformene delt inn i 3 faser: Oppsett, Kjøring og Deaktivering. Funksjonene i tabell 3.2 er fordelt mellom de ulike fasene.

3.2.8.1 Oppsettsfasen

Oppsettfasen var implementert i implementert i den opprinnelige versjonen ved at alle sekvensene som skal kjøres mottas gjennom usb fra vertsmaskinen før kjøringen begynner. Ved å gjøre dette påvirkes ikke brukertiden til hver av sekvensene. For ny funksjonalitet benyttes denne fasen i tillegg denne muligheten til å forhåndsberegne data og addresse.

Når en ny pakke med data mottas gjennom USB, startes en interrupt rutine. Hva denne gjør avhenger av headeren på usb-pakken. Hvis denne pakken er en sequence-item av sagtann, triangel eller sinus kalles funksjonen *Setup_wavegen*. *Setup_wavegen* tar valg av DAC-kanal regner ut hvilken kjerne som skal brukes for kanalen. Funksjonen tar bit-representasjonen av kjerne og bølgeform og regner ut addressen til inputbufferen som skal aktiveres. Addressen lagres i en array av pekere i minnet uten at metoden aktiverer bølgegenerator kjernen. Indeksen til arrayen lagres i et ubrukt datafelt (noccounter) i objektrepresentasjonen av sekvensen.

3.2.8.2 Kjøringsfasen

Som tidligere nevnt initieres kjøringsfasen av vertsmaskinen gjennom kommandoen *Run_items*() etterfulgt av at mikrokontrolleren har mottatt og bearbeidet

alle sekvensene. Kommandoen starter opp en klokke. Scheduleren går igjennom alle sekvensene som ble utstedt og håndtert i oppstartsfasen etter Round-robin prinsippet. Dersom klokken har passert start-tiden kalles metoden *execute*(), som tar objektrepresentasjonen av sekvensen som argument. Dersom sekvensen er en bølgeform, kalles metoden *activate_wavegenerator_channel*(). Metoden dereferererer den forhåndskalkulerte addressen og skriver data til inputbufferet. Etter latenstiden for generatorkjernen er over, vil kjernen rapportere at den er aktiv gjennom F1-lampen på brettet. Som nevnt i seksjon 3.2.1 er det mulig å se hvilke kjerner som er aktive ved å lese av "Aktiv-kanal"-registret som er lokalisert på addresse 0x320C.

3.2.8.3 Deaktiveringsfasen

Deaktiveringsfasen foregår individuelt for hver sekvens. Fasen initieres når scheduleren finner enn sekvens der schedulerens klokke har passert sluttiden for sekvensen. Dersom sekvensen er en bølgeform kalles metoden *Deactivate_wavegenerator_channel*(). Metoden tar nummeret på bølgegenerator-kjernen som skal deaktiveres som argument. Metoden tar denne informasjonen og regner ut hvilken addresse inputbufferet for kjernen som skal deaktiveres, er lokalisert på. I tillegg beregnes en bitmaske utifra nummeret på kjernen som skal deaktiveres. Dersom vi for eksempel ønsker å deaktivere generatorkjerne1 anordnes bitmasken verdien 0x01.

Før deaktivering nullstilles DAC-kanalen ved at kjernen genererer en konstant verdi på 128. For Design 1 gjøres dette ved å skrive 0x0001 til inputbufferet. For Design 2 og Design 3 gjøres dette ved å skrive 0x8000 til inputbufferet (dette er i hovedsak den eneste forskjellen mellom koden for de ulike designene). Generator kjernen deaktiveres derretter ved å skrive 0x0000 til innputbufferen. Metoden går til slutt inn i en spinlås i påvente av indikasjon fra aktiv-kjerne vektoren (se seksjon 3.2.1) om at kjernen er deaktivert. Dersom resulatet av en logisk "and" opperasjon mellom bitmasken og Aktiv-kjerne vektoren er 0, indikerer dette at kjernen er inaktiv.

3.2.8.4 Nullstilling av mikrokontroller

Dersom en feil skulle oppstå slik at for eksempel en av kjernene ikke deaktiveres riktig, vil det ikke lenger være mulig å skrive til DACen. Ettersom generator-modulen har høyere prioritet til å skrive til DACen. Selv om det ikke finnes noen kjente feil i koden som vil forårsake dette, kan for eksempel en uheldig kombinasjon av minneopperasjoner (metastabilitet) føre til at slike problemer oppstår i sjeldne tilfeller.

Får å unngå "snøball-effekter" når feil oppstår er det implementert en funksjon for å nullstille mikrokontrolleren både før og etter kjøring av sekvensene. Dette er implementert ved å utvide usb-kommandoen $USB\_CMD\_RESET\_ALL$ som opprinnelig ble brukt i den eksisterende versjonen til å nullstille alle kontrollmodulene. Denne funksjonen er nå utvidet med nullstilling av alle bølgegeneratorkjernene. Metoden $clear\_wavegenerator()$ deaktiverer alle generatorkjernene ved å skrive 0x0000 til alle input-bufferene, og derretter nullstille DAC-kanalene ved å aksessere dem direkte gjennom mikrokontrolleren.

3.2.8.5 Oppstart av mikrokontroller

I oppstarts-rutinen på mikrokontrolleren kalles metoden $clear\_wavegenerator()$ før "power on self test" (POST) og konfigurering av DACen utføres. Skrivetilgang til DACen er essensielt for gjennomføring av oppstartrutinen ettersom det benyttes en spin-lås som sørger for at rutinen fullføres på korrekt måte. I tillegg brukes oppstartsrutinen til å initialisere datastrukturene som brukes for midlertidig lagring av de forhåndskalkulerte addressepekerne og data for inputbufferene.

3.2.9 Vertsmaskin

I likhet med mikrokontrolleren er det også gjort utvidelser av funksjonaliteten på Vertsmaskin-koden (Host). Det kan nevnes at programkoden for vertsmaskinen er identisk over alle designene. Oppstartskommandoen vil forøvrig være forskjellig avhengig av hvilket design som benyttes. Nærmere detaljer rundt dette forklares i Appendix B.

Som tidligere nevnt brukes Vertsmaskinen hovesakelig som grensesnitt mot mikrokontrolleren. Tabell 3.3 viser en oversikt over nye funksjoner som er implementert.

Funksjoner:
scheduleSine()
scheduleTriangle()
scheduleSawtooth()
getChannelForPins()

TABLE 3.3

Metodene *scheduleSine*(), *scheduleTriangle*() og *scheduleSawtooth*() legger inn sekvenser med de nye bølgeformene på mikrokontrolleren. Hver av metodene tar pin-nummer, start-tid(ms), slutt-tid(ms), frekvens(hz), amplitude og generatortype som argument. Ettersom frekvens-resolusjonen i DDS-grensesnittet er begrenset, rundes frekvensen først av til nærmeste gyldige frekvens. Den avrundede frekvensen samt amplituden formateres til ett enkelt ord på 16 bit med samme format som data-delen på inputbufferet.

Før dataene overføres til mikrokontrolleren kaller hver av metodene funksjonen *getChannelForPins*() og sender pin-nummeret som argument. Metoden tar pin-nummeret og oppretter en avbildning til en ledig DAC-kanal. Avbildningen sendes deretter videre til mikrokontrolleren (via metoden *setXbar*()) som konfigurerer crossbarene for den nye avbildningen. Metoden returnerer deretter nummeret på den utvalgte DAC-kanalen. Nummeret på DAC-kanalen, start-tid, slutt-tid samt det kombinerte datafeltet for frekvens og amplitude sendes til slutt til mikrokontrolleren gjennom USB-grensesnittet.

Chapter 4

Metode

Dette kapittelet omhandler og beskriver metodene som benyttes for gjennomføring av oppgaven. Hensikten er å besvare forskningsspørsmålene og arbeidshypotesene som er framsatt i forbindelse med oppgaven. Dette gjøres gjennom praktiske eksperimenter og målinger. Målet er at resultatet som et minimum skal indikere om gjennomføring av videre forskning er hensiktsmessig. Basert på litteraturen som ble presentert i kapittel 2 kan forskningsspørsmålene enten besvares deduktivt eller induktivt.

En deduktiv metode i dette tilfellet vil innebære å separere signalene og/eller materialet og analysere dette gjennom en liknende metode som ble brukt av Nichele et al. Problemet er at det er vanskelig å med sikkerhet avgjøre nøyaktig hva som bør observeres for å besvare spørsmålene.

Istedet for å analysere de forskjellige delene av systemet separat kan vi heller se på hvordan en adaptiv prosess reagerer når vi påfører endringer i systemet. Dette kan gjennomføres med utgangspunkt i de samme metodene som ble brukt av Kaufman et al [26], Lykkebø et al [20] og Barnett[27]. Metodevalget er også forenlig med Weavers anbefalinger med hensyn på analysemetoder for komplekse systemer[22].

På bakgrunn av dette bør forskjellene mellom de nye signalformene og DDS-designene kvantiseres. Dette kan uttrykkes som et mål for støy/forvrengning og beskrives i seksjon 4.1. I seksjon 4.2 presenteres en genetisk algoritme som representerer den adaptive prosessen vi ønsker å studere.

Frekvenser:
71.595/143.19/286.38 Hz (sinus/triangel/sagtann)
572.20 Hz
1444.4 Hz
2930.0 Hz
36621 Hz
73242 Hz
146500 Hz
195300 Hz

TABLE 4.1: Tabellen viser en liste av frekvenser som skal testes og analyseres.

4.1 Estimat av forvrengning på signalene

Denne seksjonen forklarer framgangsmåten som blir brukt til måling av forvrengning for de nye designene. Hensikten er at målingene skal fungere som referanse for støy og periodiske artefakter som forekommer ved de ulike signalene og DDS-designene. Framgangsmåten kan oppsummeres etter følgende liste:

1. Valg av 8 frekvenser som skal fungere som referanse for hver signalform.
2. Generering av sinus, triangel og sagtann funksjoner for hver referansefrekvens og hvert DDS-design.
3. Bølgeformene samples og skrives til fil gjennom simulering av DDS-designene.
4. Basert på DFT-analyse av samplene estimeres Total Harmonisk Forvrengning(THD), Signal Til Støy-forhold (SNR) samt forvrengningsfritt dynamisk rekkevidde (SFDR) for alle signalene.

4.1.1 Valg av referansefrekvenser

Et felles sett bestående av 8 frekvenser er valgt som referanse for hver signalform. Utvalget er representert i tabell 4.1. Det bør nevnes at det første elementet i tabellen representerer den laveste tilgjengelige frekvensen for en gitt bølgeform. Frekvensene er valgt ut logaritmisk med hensyn på frekvensspektret, med den hensikt at estimatene på best mulig måte skal representere frekvensspektret fra 0 til 200 kHz.

4.1.2 Genering og sampling av signalene

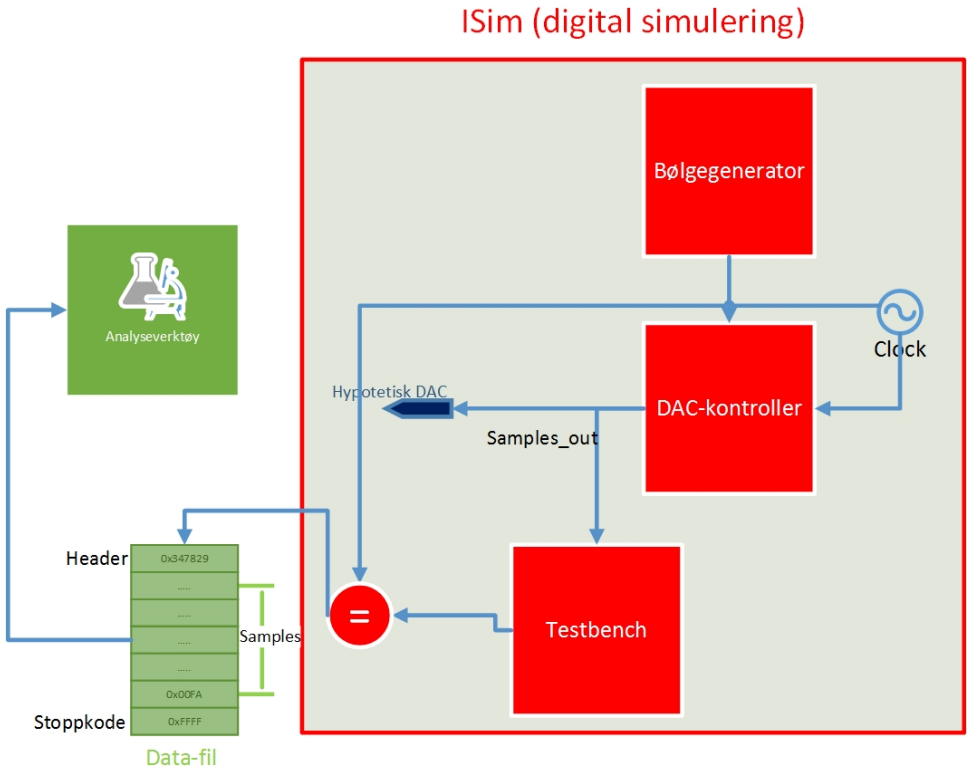

FIGURE 4.1: En oversikt over det implementerte oppsettet for utførelse av sampling

For å analysere forvrengning for de valgte frekvensene, må signalet genereres og samples på en hensiktsmessig måte. Et alternativ som ble overveid er å måle AC-spenningen fra DACen gjennom et oscilloskop mens DACen. Ettersom det ikke er mulig å få tilgang på et oscilloskop som var egnet for å gjøre dette på en tilstrekkelig måte, gjøres samplingen istedet gjennom simuleringer.

Simuleringerene gjennomføres på en forenklet, digitalt syntesisert form av hardwaren slik figur 4.1 viser. Det syntesiserte digitale designet drives av stimulus som genereres av en "test-bench". For dette oppsettet kjøres simuleringene gjennom simuleringsverktøyet Isim. For hvert av designene som ble presentert i seksjon 3, ble 3 "test-benches" implementert. Det tilsvarer èn test-bench for hver bølgeform, og hver test-bench genererer stimulii for frekvensene presentert i tabell 4.1.

Samplingen gjøres direkte fra "last_executed_command"-registeret i DAC-kontrolleren (se seksjon 3.2.1). Hvert sample lagres lokalt i en fil på harddisk som en datastrøm

der samples for alle frekvensene lagres. For å skille hvilke frekvenser de forskjellige samplene tilhører på en fornuftig måte, er det utformet en felles protokoll bestående av en header samt en stoppkode. Headeren er et flyt-tall i et egendefinert format. Slutten på segmentet indikeres av stoppkoden 0xFFFF. Dette gjør det mulig å lagre data med ulik lengde for de forskjellige frekvensene i den samme filen.

4.1.3 Analyse av signalene

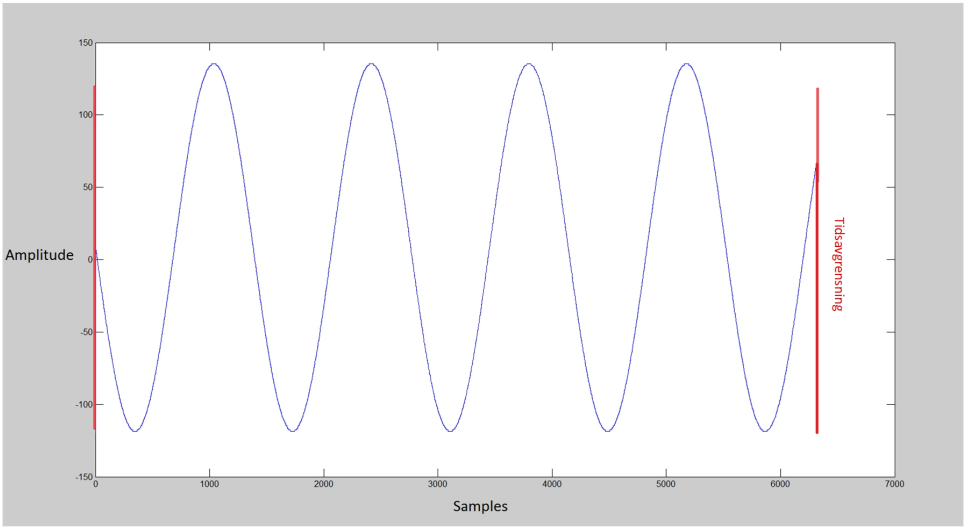

FIGURE 4.2: En visualisering av tidsavgrensningsproblemet for PSD

Forvrengning for de ulike designene gjennom analyse av styrkespektrumet (PSD) for de ulike bølgeformene. Styrkespektrumet for et generelt signal kan uttrykkes gjennom formel 4.1[38]. Der $x(t)$ er en parameterisering som en funksjon av tid for signalet som skal analyseres, og T er tidsvinduet for signalet. P er dermed et normert gjennomsnitt av styrken på signalet x(t). Et problem er at samplingen av signalet er i det digitale domenet, og dessuten tidsavgrenset. På grunn av dette brukes en Diskret Fourier Transformasjon (DFT) som avkortes, slik at transformasjonen kan gjennomføres over et begrenset tidsintervall. Dette gjøres gjennom formel 4.2. Alternativt kan PSD beskrives som en metode som vektoriserer variansen i signalet over frekvensdomenet. PSD er derfor uttrykt gjennom ortonormale basiser i et *Hilbert-rom*.

En ulempe med styrkespektrumsanalysen er at det begrensede tidsintervallet medfører artefakter og uønsket støy[]. Dette kan visualiseres gjennom Figur 4.2. For

dette eksemplet kan vi anta at Figur 4.2 viser en tidsavgrenset sinusfunksjon i det digitale domenet på 70 Hz. Videre kan det antas at bølgeformen som er vist på figuren er en perfekt kvantisert sinusfunksjon. Ideelt sett skal styrkespektrumet da gi utslag for kun 70 hz. I virkeligheten vil styrkespektrumet gi utslag for 70Hz og for flere av de nærliggende frekvensene slik figur 4.3 viser. Utslagene for de nærliggende frekvensene i spektret på figuren forårsakes av tidsavbruddene på signalet, og kan karakteriseres som en form for aliasing [39].

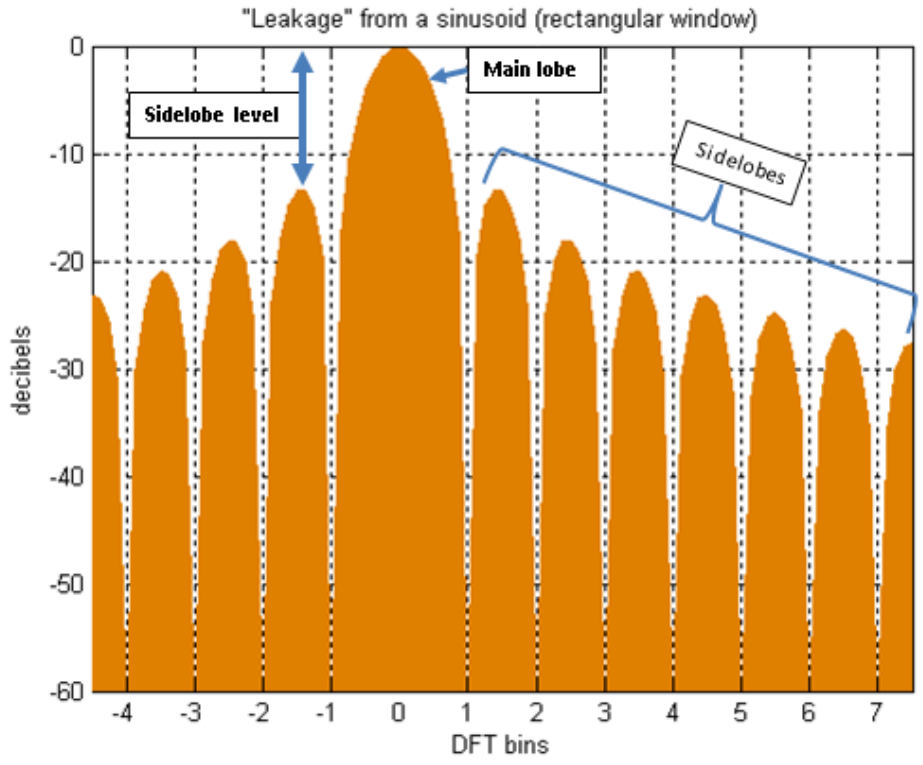

FIGURE 4.3: Figuren er hentet fra Wikipedia[7] og viser lekasje mellom "DFT bins".

Denne *spektrallekasjen* er et problem fordi det påvirker målinger av nærliggende frekvenser. En vanlig metode for å redusere lekasjen er gjennom "vekting" av signalet i tidsdomenet, slik at *variansen* på signalet blir vesentlig mindre i nærheten av bruddpunktene. Dette gjøres ved å multiplisere signalet med en vindufunksjon før en diskret fouriertransformasjon finner sted. Det enkleste vinduet er rektangulært kan defineres gjennom funksjonen $W(x) = 1$ i kontekst med formel 4.1. Rektangulære vinduer har et lavt dynamisk rekkevidde ettersom vinduet ikke begrenser lekasje.

$$P(x(t)) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)^2 dt \tag{4.1}$$

$$S_{xx} = \frac{\Delta t}{N} \left| \sum_{n=1}^{N} x_n e^{-i\omega n} \right|^2 \tag{4.2}$$

Et annet type vindu er Blackman-vinduer. Som vist på figur 4.4, er vinduet formet som en tynn bell-kurve i tidsdomenet. Når det multipliseres med signalet fra figur 4.2 resulterer dette i vesentlig redusert lekasje ettersom variansen i amplituden blir vesentlig svakere i området nært avbruddspunktene. Ulempen er at dette introduserer en frekvensmodulasjon av signalet, som fører til at "Loben" illustrert på figur 4.3 vil oppta plass i nærliggende frekvenser. Etter dette ressonomentet har Blackman-vinduer derfor høy dynamisk rekkevidde, men til gjengjeld lav sensitivitet. Siden vi i dette tilfelle ønsker å måle forvrengning, er det ønskelig å bruke en vindusfunksjon med høyt dynamisk rekkevidde slik at nevneverdig påvirkning på støygulvet kan unngås.

For analyse av signalene brukes et Hamming-vindu. Hammingvinduet har samme bell-kurveform som Blackman-vindu men med en vesentlig høyere "spredning". Det gir derfor høyere sensitivitet men lavere dynamisk rekkevidde sammenliknet med et Blackman-vindu. Testing viste at Blackman-vinduet var uegnet til målinger av SFDR og THD. Grunnen var at lav sensitivitet førte til vesentlige avvik av resultatene sammenliknet med hva som bør forventes. En visuell inspeksjon av PSD med Blackman-vinduet bekreftet dette.

For å gjennomføre analysen av styrkespektrumet brukes Matlab. Matlab anses som et naturlig valg ettersom standardutgaven leveres med introduksjon og støtte for bølgeanalyse og håndtering av grafer. Etter filene genereres av I-sim, åpnes de i matlab og de digitale samplene leses av som vektorer. Vektorene multipliseres deretter med et hammingvindu av samme lengde gjennom Matlabfunksjonen "hamming()". Et styrkespektrum opprettes for hver av vektorene gjennom Matlabfunksjonen "periodogram()". Fordi PSD er definert (og komplett) for Lesbeque-rommet L^2 benyttes den ekleudianske distansenormen $\|\cdot\|_2$ for alle videre utregninger, med mindre annet ikke er spesifisert.

For å beregne estimatene benyttes THD via funksjonen "thd()" i Matlab. THD er uttrykt gjennom Formel 4.3. Formelen beskriver forholdet mellom grunnkomponenten og summen av amplitudene til de harmoniske komponentene for et signal. "Grunnkomponenten" tilsvarer den høyeste registrerte verdien i styrkespektrumet. Antall harmoniske komponenter som inkluderes i beregningene er begrenset til 6. Dette tallet er noe lavere for frekvensene $f = 146.5hz$ og $f = 193.3hz$.

SNR er et estimat beskriver forholdet mellom grunnfrekvensen og summen av amplituden for alle frekvensene i styrkespektrumet. Vi kan estimere grunnfrekvensen ved å måle den totale metriske distansen for signalet gjennom normen $p = infty$, og deretter finne størrelsen på alle komponentene i spektret. Summen av distansene vil da konvergere mot verdien av den største komponenten som er inkludert i beregningen. Et estimat for SNR kan dermed uttrykkes gjennom formel 4.4. Estimatet beregnes gjennom funksjonen "snr()" i Matlab. SNR inkluderer både harmoniske og ikke-harmoniske komponenter av grunnfrekvensen.

SFDR estimerer forholdet mellom grunnfrekvensen og frekvensen med høyest styrke blant frekvensene i støygulvet. SFDR beregnes i Matlab gjennom funksjonen $sfdr()$ og kan gjennom de samme argumentene som for SNR beskrives gjennom formel 4.5. Funksjonen $\|stoy\|_\infty$ er et estimat for frekvensen med høyest styrke, blant alle frekvensene i støygulvet.

$$THD = \frac{\sqrt{\sum_{n=1}^{n=6}(H_n)^2}}{\sqrt{(H_0)^2}}dt \tag{4.3}$$

$$SNR = \frac{\|signal\|_\infty}{\|stoy\|_2} \tag{4.4}$$

$$SFDR = \frac{\|signal\|_\infty}{\|stoy\|_\infty} \tag{4.5}$$

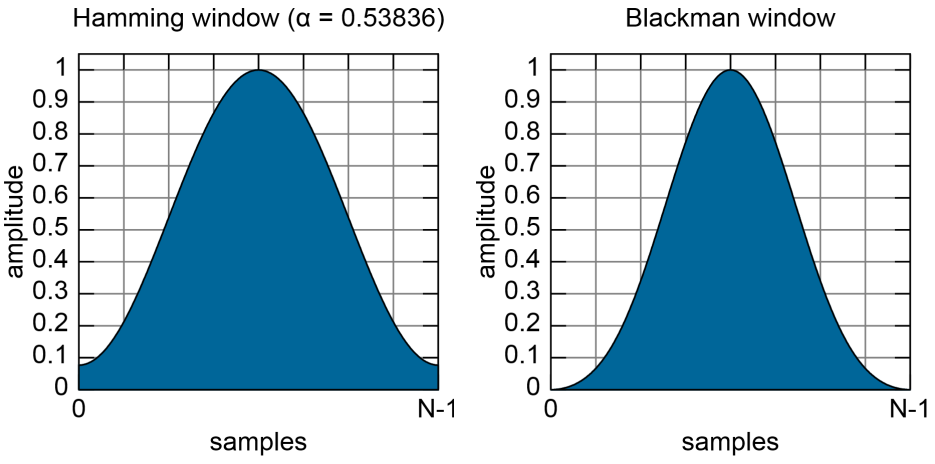

FIGURE 4.4: Figuren er satt sammen av illustrasjoner fra Wikipedia[8][9]. Figuren viser Blackmanvinduet (t.v) og Hammingvinduet (t.h) i tidsdomenet.

4.1.3.1 Normaliserte estimater for forvrengning

Et problem med de vanlige estimatene for forvrengning er at de bruker en ren sinusfunksjon som referanse. Dette fører til at estimatene konvergerer forskjellig for de ulike bølgefunksjonene. Dette gjør det vanskelig å sammenlikne den objektive signalkvaliteten mellom for eksempel sinus og sagtann. For å løse dette beregnes en "normalisert" form av estimatene (for triangel og sagtann) utifra en estimert referanse.

$$SNR_{norm}(dB) = SNR - exSNR \tag{4.6}$$

$$SNR_{norm}(dB) = SFDR - exSFDR \tag{4.7}$$

Referansesignalene genereres gjennom Matlab-funksjonen *sawtooth*() og påføres et hammingvindu. De transformeres derretter til et PSD-estimat. Resultatene fra disse presenteres som "forventet SNR", "forventet THD" og "forventet SFDR". Disse brukes til å beregne et *normalisert* estimat for SFDR og SNR etter formel 4.6 og 4.7.

Ulempen med disse formlene er at estimatene blir unøyaktige fordi de ikke inkluderer styrkevariasjonene for de individuelle frekvenskomponentene i PSD. Det optimale ville forøvrig vært å beregne residualene mellom alle støykomponentene istedet. For SFDR og SNR er dette potensielt vanskelig og tidkrevende å gjennomføre i praksis på en god måte ettersom begge estimatene inkluderer et høyt antall komponenter. Dette medfører at det er nødvendig med et tilnærmet forventningsrett estimat for hver komponent, noe som stiller svært høye krav til PSD-estimatene. På den andre siden er denne metoden velegnet for THD ettersom estimatet bruker et begrenset antall SPD-komponenter.

På bakgrunn av dette beregnes estimatet "Normalisert THD" derfor utifra differansen mellom komponentene for referanse-signalet og det målte signalet. Dette er beskrevet gjennom formel 4.8. Der H representerer de harmoniske komponentene fra en måling og exH representerer forventningsverdien. Både exH og H representerer en ortonormal basis for sin respektive PSD. Ettersom de harmoniske komponentene parvis utspennes for de samme frekvensene i spektret, er det enkelt å se utifra Cauchy-Schwarz ulikheten at $\langle exH_n, H_n \rangle = \|exH_n\| \cdot \|H_n\|$[40]. I dette

tilfellet betyr det at residualen mellom hver av de harmoniske komponentene kan beregnes ved subtraksjon mellom basisene slik det er gjort i formel 4.8.

$$THD_{norm} = \frac{\left\| \sum_{n=1}^{7} H_n - exH_n \right\|_2}{\|H_0\|_2} \tag{4.8}$$

4.2 Genetisk algoritme

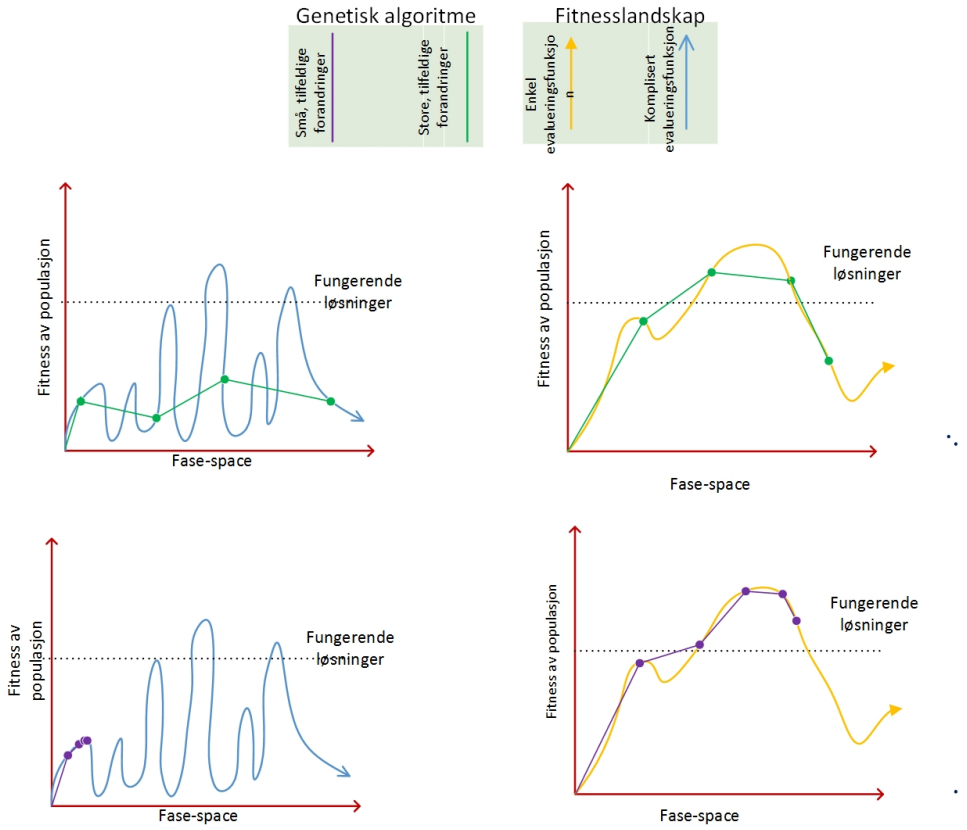

FIGURE 4.5: Variasjon i fitnesslandskap for ulike genetiske algoritmer og evalueringsfunksjoner.

En genetisk algoritme kan beskrives som en ukjent tidsvarierende funksjon og uttrykkes på mange forskjellige måter[27]. En forenklet modell av et fitnesslandskap for genetiske algoritmer og andre *adaptive prosesser* kan visualiseres gjennom pilene på figur 4.5. Med "forenklet" menes at landskapet i selv enklest mulig form har minimum $n(+1)$ dimensjoner, hvor hver dimensjon er representert som en kompleks tidsvarierende funksjon av hvordan n bestanddeler i en genotypisk representasjon påvirker materialet (se seksjon 2.1.2). De stykkvis lineære linjene på figuren

viser hvordan fitness endrer seg for populasjonen når den beveger seg gjennom faserommet. Som figuren indikerer er det en gjensidig og sannsynligvis tidsvarierende avhengighet mellom faserommet og fitness til populasjonen når systemet og materialet påvirkes av den genetiske algoritmen. Øverst til venstre på figuren er det avbildet et fitnesslandskap uttrykt gjennom en komplisert evalueringsfunksjon med 6 lokale maksima. Hvert maksimum representerer konvergenspunkter eller potensielle attraktorer. I dette tilfellet har konvergenspunktene kun en svak effekt på fitness fordi den genetiske sammensetningen i populasjonen endrer seg for raskt. En annen genetisk algoritme med identisk fitnesslandskap er vist nederst til venstre på figuren. Her utfører den genetiske algoritmen kun små inkrementelle endringer på populasjonen. I dette fitnesslandskapet fører dette til at populasjonen konvergerer mot et lokalt maksima, og forhindrer algoritmen å finne en komplett løsning innen et rimelig tidsrom. Fitnesslandskapene plassert til venstre på figuren viser en annen evalueringsfunksjon der begge evalueringsfunksjonene er effektive. Målet med en slik modell er at vi kan si noe om hvordan landskapet oppfører seg ved å studere hvordan regelmessige eller endringer i genotyper påvirker materialet.

Figur 4.5 er i realiteten en visualisering med utgangspunkt i Kauffmans NK-modell (se kapittel 2). Figuren er ment å synliggjøre viktigheten av å velge en evalueringsfunksjon som er i stand til nå målene og evaluere arbeidshypotesene. Ettersom vi ønsker å måle effektiviteten til den genetiske algoritmen, bør landskapet analyseres ved å studere alfa-individene eller den *dominante* delen av populasjonen [27]. For at dette skal være oppnåelig må derfor fitness til en viss grad konvergere. Av den grunn er det nødvendig å optimalisere den genetiske algoritmen for de valgte signalene. Mer om dette finnes i seksjon 5.2.

4.2.1 Oppsett

Den genetiske algoritmen er skrevet fra bunnen av i C# i forbindelse med denne oppgaven. Størrelsen på populasjonen for den genetiske algoritmen er 65 individer, og eksperimentet kjøres for nøyaktig 100 generasjoner. For et gitt design og en gitt bølgeform gjennomføres eksperimentet 6 ganger. Dette tilsvarer totalt 54 kjøringer, med 18 kjøringer for hver signalform (uniformt distribuert over de 3 designene). Det bør presiseres at algoritmen er uforandret mellom hver kjøring. Resten av denne seksjonen omhandler oppsett og gjennomførelse av eksperimentet. Kildekoden samt kjørbar fil er vedlagt som i digitalt vedlegg.

4.2.1.1 Genetisk representasjon

De røde, blå og grønne feltene på figur 4.6 viser hvordan genotypen til et individ er satt sammen. Et individs genotype består av tilsammen 16 definerte sekvenser. Sekvens 0-3 inneholder representasjon av input, sekvens 4-11 representerer konfigurasjonsignalene og sekvens 12-15 representerer opptakspinner (recording pins). Sekvens 0-3 er begrenset til statisk spenning på enten 0V eller 3.3V. Siden sekvens 0-3 representerer en 1-bits input, vil alle sekvensene alltid settes til samme spenning. Sekvens 4-11 inneholder en verdi som representerer signaltype, amplitude og frekvens. Sekvens 12-15 har en fast samplingsfrekvens på 10kHz. Før hver kjøring vil alle sekvensene (for hvert individ) initialiseres til et tilfeldig tall som representerer nøyaktig en unik fysisk pinne til materialet. Amplituden på konfigurasjonssignalene initialiseres til maksimal rekkevidde (5V), mens frekvensen initialiseres til en tilfeldig verdi.

4.2.1.2 Genetiske operatorer

Den genetiske algoritmen består av 3 genetiske operatorer: Selection, Crossover og Mutation. I "Selection"-operatoren velges det et individ gjennom et turneringsutvalg fra et sample av den eksisterende populasjonen. Samplets størrelse er tilfeldig valgt fra intervallet $[1, 64]$ og individene trekkes tilfeldig fra populasjonen. Et vilkårlig individ fra samplet velges av operatoren med en betinget sannsynlighet på 80 prosent, gitt at et annet individ med høyere fitness ikke er valgt ut.

"Crossover"-operatoren tar i mot 2 individer som er valgt igjennom turneringsutvalget fra "Selection"-operatoren. De to nye individene representerer foreldre for to nye individer i den nye populasjonen. For å forhindre elitisme i populasjonen, er det et krav om at foreldrene er forskjellige individer. En crossover-sekvens n for $0 < n < 16$ velges tilfeldig og de genetiske sekvensene og kopieres mellom foreldre og barn på samme måte som i Lykkebø et al[20]. Genene for pinkonfigurasjonene passeres forøvrig direkte fra foreldre til barn uten crossover.

"Mutation"-operatoren muterer de nye individene som genereres gjennom Crossover-operatoren. For at utvalgstrykket ikke skal bli for lavt vil kun halvparten av individene i hver nye generasjon muteres. Hvert av de 16 genetiske sekvensene for hvert individ har 97 prosent sannsynelighet for å mutere. Funksjonen består av 2

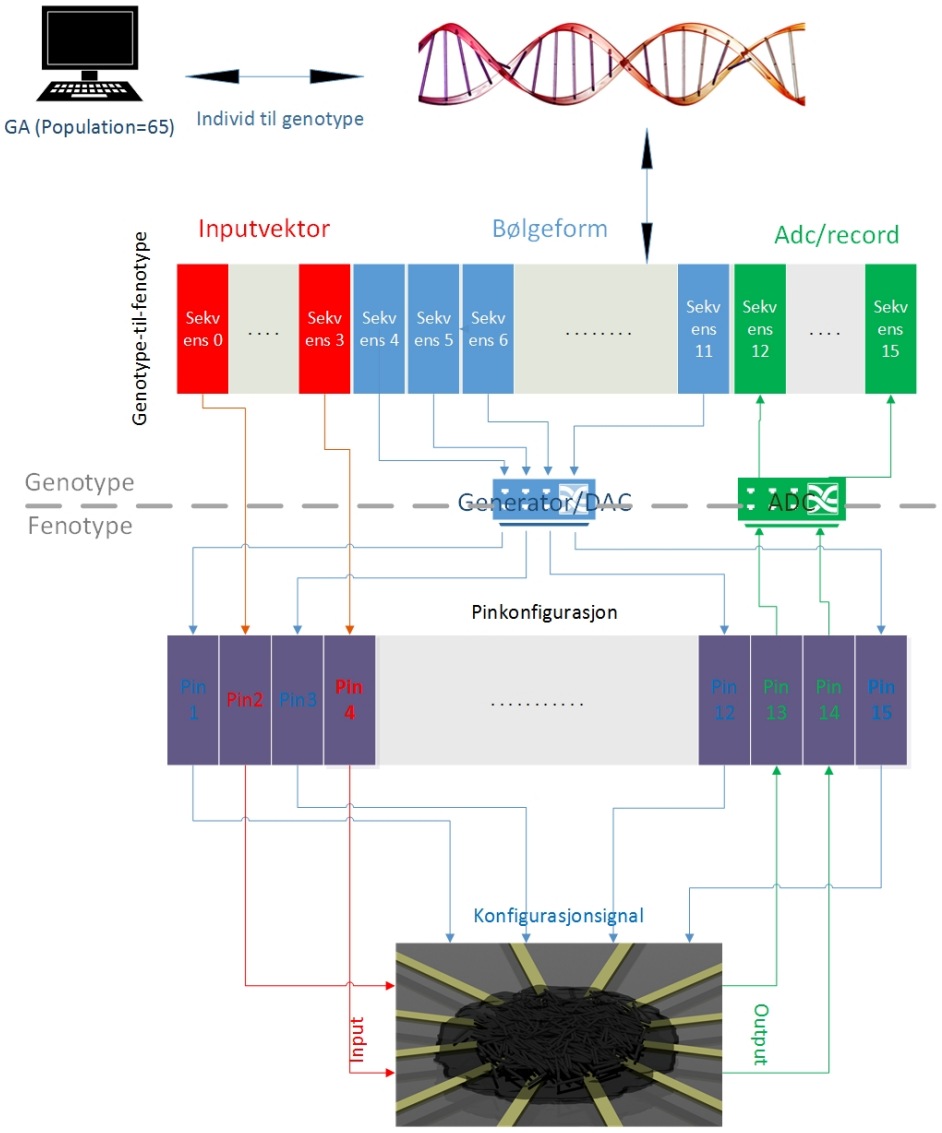

FIGURE 4.6: Relasjon mellom individ, genotype og fenotype.

ulike, uavhengige operatorer. Den første trekker en verdi fra en gaussisk fordeling med $forventing = 100$ og $standardavvik = 66.5$ som legges til frekvensen på konfigurasjonssignalet. Den andre operatoren trekker en verdi fra en uniform distribusjon for intervallet $[-1, 1]$ og legger denne verdien til amplituden.

4.2.1.3 Evaluering av genotypene

Før hvert enkelt individ mottar en fitness-verdi, utføres en fysisk evaluering av individenes genetiske representasjon på Mecobo-plattformen. Dette skjer ved at den genetiske representasjonen eller "genotypen" for hvert individ transformeres

til en fenotype slik Figur 4.6 viser. Den grå stiplede linjen på figuren markerer en tolkning av skillet mellom genotype og fenotype for dette eksperimentet. "Fenotypen" representerer en kompleks sammensetning av spenningspotensialer i materialet. Dette samples via ADCen og evalueres gjennom fitnessfunksjonen til den genetiske algoritmen.

En viktig antagelse som er gjort i figur 4.6 er at ADC, pinkontrollerene, DDS-designet og Crossbarene er en del av fenotypen **og** genotypen for hvert individ. Dette er fordi at komponentene genererer støy når bitstrengene i genotypen transformeres til det fysiske domenet. Denne støyen påvirker signalene og dermed også fenotypen. Samtidig regnes de også som en del av genotypen på grunn av den direkte relasjonen komponentene har til genene.

4.2.1.4 Fitness

På grunn av relasjonen mellom individ, genotype og fenotype (se figur 4.6) kan fitness defineres direkte gjennom målingene av fenotypen. Som tidligere nevnt benyttes problemdefinisjonen og fitnessfunksjonen fra Lykkebø et al [20]. Fitnessfunksjonen er angitt gjennom formel 4.9 for intervallet $[-2048 \leq node \leq 2048]$ når vektorene $node_{green}$, $nodered$ og $nodeblue$ er gitt. Med noe misbruk av definisjon er $||\cdot||$ ekvivalent med $max()$ og $min()$. Hvert element i vektoren(e) $node$ representerer den største registrerte spenningen registrert i et buffer som representerer en av nodene. Som formelen indikerer er det en direkte relasjon mellom fitness og de digitale representasjonene av genene. Genotypenes faserom vil derfor bli analysert gjennom fitness med utgangspunkt i NK-modellen presentert i kapittel 2.

$$Fitness = \|node_{red}\|_{-\infty} - 2 \cdot \|node_{green}\|_{\infty} + 2 \cdot \|node_{green}\|_{-\infty} + \|node_{blue}\|_{\infty} \quad (4.9)$$

I resultatene av eksperimentene vil fitness for enkelhetsskyld bli representert med den dynamiske rekkevidden på ADC-samplene som "referanse verdi". Fra formel 4.9 kan en generell definisjon for maksimalverdien av fitness for et individ angis gjennom supremumet $\|node_{red} - node_{blue}\|_{\infty} = 1.2 \cdot 4096$, gitt $\|node_{green}\|_{\infty} -$

$\|node_{green}\|_{-\infty} = 0$ for funksjonsrommet L^p. Variablene $node_{red}$ og $node_{blue}$ representerer vektorene for verdiene representert for de røde og blå nodene. Minimal verdien for fitnessfunksjonen er -12288.

Som tidligere nevnt er fitnesslandskap ofte *bråkete*[27]. En reell risiko i forbindelse med resultatene er utilsiktet statistisk bias som kan føre til at målene og hypotesene vurderes utifra resultatet på et ukorrekt grunnlag. I den forbindelse er det hensiktsmessig å se på de ulike komponentene i evalueringsfunksjonen over de forskjellige designene. Dersom det kan oppdages at det er vesentlige forskjeller i hvordan fitness optimaliseres over de ulike designene, kan dette være en indikasjon på at evalueringsfunksjonen ikke er tilstrekkelig objektiv.

Det bør nevnes at fitness-verdier alene ikke er representativt for hvor effektivt en bølgeform eller DDS-design er. Slik fitness er definert i formelen over kan den assosieres med en bestemt sannsynlighet for reproduksjon, men hverken mer eller mindre. Som indikert i figur 4.5 er det en nær sammenheng mellom fitness og sannsynligheten for å finne fungerende løsninger. Dette forutsetter at mutasjonsraten og utvalgsfunksjonen er riktig satt opp, og holder seg innenfor en feil-terskel[27]. Ved å optimalisere utvalgsfunksjonen slik som det tidligere er beskrevet, vil resultatet bli analysert ved å studere sammenhengen mellom fitness og fungerende løsninger.

4.2.1.5 Fungerende enheter

I forbindelse med verifisering av arbeidshypotesene presentert tidligere denne seksjonen vil fungerende løsninger være det viktigste kriteriet i vurderingen. En "fungerende løsning" er derfor per definisjon en enhet som representerer 2 løsninger på nodeklassifiseringsproblemet. Av den grunn brukes heller begrepet "fungerende enheter" som i prinsippet er en norsk oversettelse av det samme begrepet brukt i Lykkebø et al. Algoritmen finner en fungerende enhet dersom uttrykkene i formel 4.9 oppfyller $\|node_{red}\|_{-\infty} > \|node_{green}\|_{\infty}$ og $\|node_{green}\|_{-\infty} > \|node_{blue}\|_{\infty}$. Motivasjonen for dette er at fungerende løsninger kan uttrykkes som et kvantitativt og kvalitativt mål for den genetiske algoritmens evne til å oppnå et resultat for en gitt signalform. En annen motivasjon er at en populasjon av fungerende enheter er et objektivt uttrykk for en *dominant* fenotyper. I følge Barnett [27] er studier av dominante fenotyper gi viktig informasjon om formen på et fitnesslandskap.

4.2.1.6 Resulterende datasett

Resultatet vil bestå av datasettene som er presentert i listen nedenfor. Listen inneholder "rådata" som blir automatisk lagret for hvert forsøk. Disse blir vedlagt i det digitale vedlegget. De 3 øverste punktene på listen vil utgjøre et utgangspunkt for resultatet. Gjennomføring og bruk av disse 3 måleenhetene er basert på "fitness-measurement" metoden beskrevet av Barnett [27]. Ettersom vi er spesielt interessert i den genetiske og fenotypiske delen av populasjonen som inneholder fungerende enheter, vil all informasjon om disse enhetene som er tilgjengelig for algoritmen lagres.

- Gjennomsnitts-fitness for alle individer for hver generasjon.
- Maksimal fitness blant alle individer for hver generasjon.
- Antall "fungerende enheter" for hver generasjon.
- Fitness for fungerende enheter.
- Pinkonfigurasjon(avbildning) for fungerende enheter.
- Fitnessfunksjonen for fungerende enheter.
- Frekvens og amplitude for fungerende enheter.

4.2.2 Evaluering av arbeidshypotesene

Som tidligere nevnt er den genetiske algoritmen utformet for hensiktsmessig evaluering av arbeidshypotesene som er framsatt i seksjon 1.1.2. Den første framsatte hypotesen om at signalformene som ble presentert er egnede basiser for konfigurasjonssignaler, vil falsifiseres for en enkelt signalform dersom den genetiske algoritmen ikke finner fungerende enheter i forsøkene der signalformen blir benyttet. For å eliminere antall gjennomførte forsøk som en eventuelt påvirkende faktor, vil derfor hypotesen forkastes dersom algoritmen (i gjennomsnitt av alle forsøkene) finner mindre enn 1 fungerende enhet.

Som tidligere nevnt vil den andre framsatte arbeidshypotesen bli evaluert gjennom forsøk utført med genetiske algoritmen for hver av de nye DDS-designene. Denne hypotesen aksepteres dersom resultatene ensbetydende indikerer flest fungerende

enheter på et av DDS-designene der det ikke er målt lavest forvrengning. Dette er en logisk tilstrekkelig betingelse for at hypotesen er sann. I henhold til *Modus tollendo tollens* bør hypotesen forkastes dersom resultatene ensbetydende viser flest fungerende enheter for DDS-designet med lavest forvrengning.

Det bør presiseres at det også kan være vanskelig å gjennomføre hypotesetester med hensyn på antall forsøk som gjennomføres. Ettersom formålet med begge hypotesene hovedsakelig er å indikere om videre forskning er hensiktsmessig, blir det derfor lagt størst vekt på forkastning med hensyn på evaluering av hypotesene.

Chapter 5

Testing

Dette kapittlet omhandler hvilke metoder som ble brukt i forbindelse med testing av eksperiment og implementasjon. Testingen utføres for å påse at systemene oppfører seg slik som forventet basert på beskrivelsene fra kapittel 3 og 4. Ettersom implementasjonen i seg selv ikke er målet i prosjektet vil dette kapittelet kun være en kort gjennomgang av de viktigste momentene. De viktigste testene som ble gjort er vedlagt i det digitale vedlegget (se B.2).

5.1 Oppsett og utstyr

For gjennomføring av metodene og testing av systemene er følgende utstyr benyttet:

- Mecobo EM m/datterbrett.
- EFM32GG STK3700 (debugging)
- ASUS G73SW bærbar pc med windows 7
- Digital-storage oscilloskop

For testing og utforming av oppsett for eksperimentene og implementasjonene ble en "trial-and-error"-basert metode brukt. Denne metoden var spesielt en viktig del av utformingenfor metodene. Det impliserer også at designprosessen har vært inkrementell. Av den grunn er kun de viktigste testoppsettene og resultatene blitt lagret.

5.2 Mecobo

Denne seksjonen omhandler tester som er utført i forbindelse med plattformen Mecobo, og det evolusjonære hovedkortet (EM).

5.2.1 FPGA

De første funksjonelle testene av DDS-designene ble utført gjennom forskjellige VHDL testbenches i "behavioural simulations" med ISim. Ulempene med denne framgangsmåten er at det i enkelte tilfeller kan være svært tidkrevende. I tillegg tar de heller ikke hensyn til faktorer som "clock-skew" og kombinatorisk forsinkelse.

Løsningen ble derfor å bruke et "debugging-kit". Dette gjør det mulig å lese av tekst gjennom en Serial Wire Output. I tillegg tillater dette bruk av break-points og avlesning av minneinnhold. De funksjonelle testene for DDS-designene ble testet med oppsettet vist på figur 5.1. Et Python-script sørget å aktivere en eller flere kjerner i DDS-designet. Skriptet ble endret mellom hver test for å prøve ut ulike bølgeformer, frekvenser og amplituder. Når DDS-designet er aktivert leser mikrokontrolleren av samplene som skrives til DACen, fra et register i kontrollmodulen. Mikrokontrolleren skriver deretter disse ut til SWO gjennom printf(). Hensikten er å bekrefte at en bølgeform skrives ut til de riktige kanalene i DACen, og at generatormodulen aktiveres og deaktiveres slik den skal. Alle testene som ble utført var velykkede. Testene viste også at avlesning fra registrene i FPGAen tar lengre tid enn å overføre et sample fra FPGA til DAC.

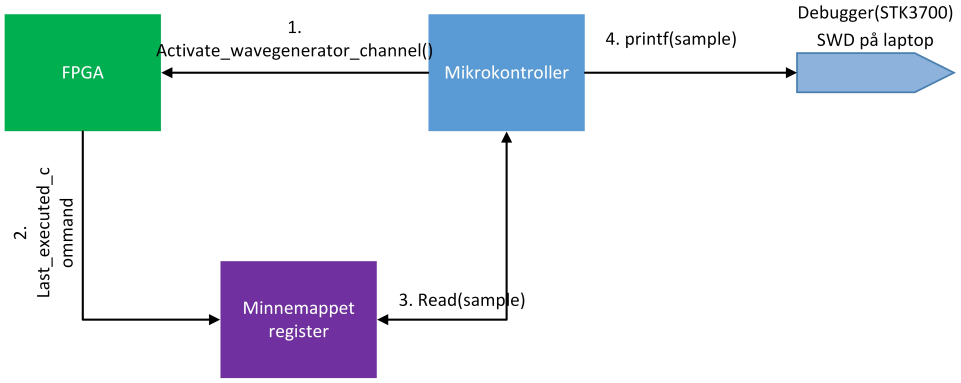

FIGURE 5.1: Implementasjon av test for FPGA implementasjonene av Design 1, Design 2 og Design 3

5.2.2 Bølgefunksjonene

Ettersom dette oppsettet bruker polling gjennom mikrokontrolleren, vil et nytt sample kun registreres hver gang det skjer en endring i registret. Det er derfor ikke mulig å med sikkerhet avgjøre at bølgeformene generes slik de skal. Et digital-lagrings oscilloskop ble derfor koblet direkte opp mot en pin i datterbrettet. Testene ble utført uten materiale. Testene viste at alle DDS-designene fungerte slik de skulle.

5.2.2.1 Dither

I forbindelse med dither som ble generert gjennom Design 2, ble en tidsserie av dette samplet og analysert i matlab som et histogram. Histogrammet vises i figur 5.2. Ettersom signalet genereres gjennom en deterministisk prosess ble det gjennom En visuell inspeksjon av tidsserien bekreftet at det ikke forekommer åpenbare tidsavhengigheter.

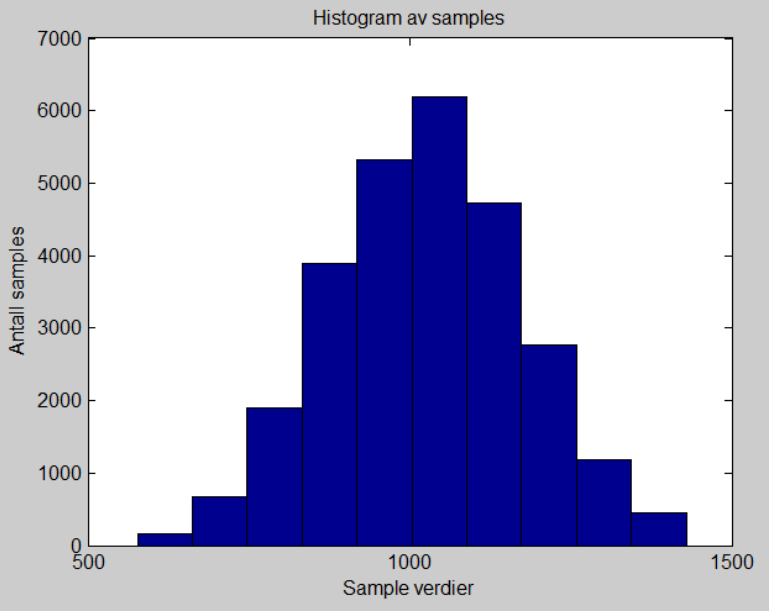

FIGURE 5.2: Histogram for dither-signalet som tilegges fasen i Design 2 **før** det avkortes

5.2.3 Mikrokontroller

Testing av koden for mikrokontrolleren ble hovedsakelig gjort med breakpoints og avlesning av minne gjennom STK3700 debuggeren. Testene som ble gjort med mikrokontrolleren var hovedsakelig for å påse at addresser og kommandoer mot FPGAen ble regnet ut og håndtert riktig. De siste testene som ble gjennomført var "stresstester" med evolusjonær algoritme for å teste ut robusthet.

5.2.3.1 Korreksjon for sanntidsoppførsel

Disse testene avdekket en race-condition i forbindelse med deaktivering av kanaler i de tidligste versjonene av mirokontroller-softwaren. Ettersom mikrokontrolleren ikke kan kommunisere med DACen så lenge en eller flere kjerne i DDS-designet er aktivert, blir deaktivering av kjernene synkronisert gjennom polling av *last_executed_command*-registret i kontrollmodulen for DACen. I sjeldne tilfeller ble dette registret overskrevet av en annen kjerne før mikrokontrolleren kunne registrere at registret ble deaktivert, og førte til en falsk deadlock. Dette ble korrigert ved å implementere en vektor som til enhver tid indikerer om en kanal er aktiv eller ikke.

Et annet problem som ble oppdaget er at mikrokontrolleren bruker svært lang tid på å aktivere og deaktivere 8 kjerner i et DDS-design. Dette er et problem fordi det ofte er ønskelig at aktivering og deaktivering av kanalene foregår i sanntid. Løsningen ble å redusere tiden mikrokontrolleren bruker, ved å forhåndskalkulere data og adresser på vertsmaskinen før kjøringen starter. Dette eliminerte ikke problemet, men optimaliseringene reduserte overheaden for aktivering og deaktivering av hver kanal fra 200 ms til omtrent 20 ms.

5.2.4 Host-maskin

Koden for Host-maskinen ble testet gjennom breakpoints. Funksjonaliteten for Host-maskinen ble hovedsakelig verifisert under testing av de andre enhetene i systemet.

5.3 Genetisk algoritme

Den genetiske algoritmen utviklet og testet gjennom "Trial and error" metoden. Som indikert i forrige kapittel er det viktig at populasjonen konvergergerer i landskapet på en balansert måte for at evolusjon skal finne sted. Dersom utvalgstrykket blir for høyt vil algoritmen oppføre seg mer som en "hillclimbing"-algoritme. Dersom utvalgstrykket er for lavt vil algoritmen oppføre seg som et tilfeldig søk. Dette er også enkelt å observere gjennom forsøkene til Kauffman et al [26].

Testing av den genetiske algoritmen var derfor sentrert rundt de genetiske operatorene knyttet til utvalg og mutasjon. Metoden gikk ut på å gjøre en endring i en operator og deretter teste eksperimentet 2 ganger. Testene ble gjennomført med forskjellig signalform for forskjellige DDS-design for minst 20 generasjoner. Videre justeringer ble deretter gjort basert på fitness-verdiene og de fungerende enhetene som ble registrert under testene.

5.3.1 Observasjoner

Det ble oppdaget endringer i frekvensen på signalene hadde større påvirkning på fitnessverdiene sammenliknet med endringer i andre variabler. På grunn av dette ble det gjennomført flere tester for ulike (Hamming) mutasjons-distanser for den genetiske representasjonen av frekvensene. Testene avslørte at mutasjonene favoriserte sannsynlighetsfordelinger som over tid fører til en gjennomsnittlig økning av frekvensverdiene.

Et interessant funn som ble gjort under en av de første testene var at utvalgsfunksjonen for algoritmen ved et uhell var satt til å velge ut individer med dårligst fitness. På tross av dette klarte den genetiske algoritmen å finne 8 fungerende enheter i løpet av 100 generasjoner for sagtann og Design 1. Denne feilen ble ikke oppdaget før algoritmen ble testet for Design 2 der ingen enheter ble funnet.

5.3.2 Firkantbølger

Ettersom et av målene for denne oppgaven er å evaluere nye signalformer, er det i den forbindelse hensiktsmessig å foreta en sammenlikning med minst 1 av signalformene som vanligvis brukes. En løsning kunne vært å bruke resultatene fra

Lykkbø et al [20]. Ulempen med dette er at utvalgsfunksjonene er optimalisert for å oppnå en "uphill-walk" for fitnessen i forsøkene med de foreslåtte bølgeformene. Forsøket ble derfor gjennomført på nytt under de samme forholdene som med de foreslåtte bølgeformene.

Forskjellen var at konfigurasjonssignalene ble generert gjennom I/O pins i EM. Tilsammen 2 forsøk ble gjennomført, men resultatene indikerte at "utvalgstrykket" for algoritmen var for høyt. Under begge testene havnet algoritmen i et stabilt lokalt maksima etter svært få generasjoner. Algoritmen var fremdeles i stand til å finne 2 fungerende enheter. Ettersom algoritmen oppførte seg vesentlig forskjellig for firkantbølger er det derfor heller ikke hensiktsmessig med en sammenlikning etter denne metoden.

5.4 Forvrengning

Som nevnt i seksjon 4.1 skal det gjennomføres en analyse av 8 ulike frekvenser for hvert signal. Ettersom vi ønsker å måle støykomponenter på signalet, er det derfor ønskelig at den *dynamiske rekkevidden* er så høy som mulig. I de første testene ble derfor et blackman-vindu benyttet for alle målingene. Som forventet verdi for målingene ble signalets dynamiske rekkevidde brukt. Dette ble kombinert med visuell inspeksjon av PSD og signalenes tidsserier.

Vinduet fungerte for referansesignalene som ble generert gjennom Matlab, men analysene fra DDS-designene hadde forøvrig et betydelig avvik fra de forventede verdiene. En visuell inspeksjon av PSD indikerte at vinduet hadde for lav sensitivitet, noe som i en betydelig grad påvirket målingene for THD og SFDR. Et nytt forsøk med et Hann-vindu ga liknende resultat. En mulig måte å løse dette på kunne vært å øke tidslengden på samplene, men dette øker også betraktelig tiden det tar å gjennomføre simuleringene av signalene.

Et nytt forsøk ble derfor gjennomført med et Hamming-vindu, som bekreftet at de andre vinduene hadde for lav sensitivitet. En ting som bør nevnes i forbindelse med dette er at enkelte av estimatene som ble gjort av de høyeste frekvensene som ble målt, hadde avvik fra de forventede verdiene. Dette gjaldt hovedsakelig kun SNR. Årsaken er sannsynligvis manglende dynamisk rekkevidde. I slike tilfeller er det ikke uvanlig å se bort fra slike avvik, men blant annet fordi verdiene inngår i

et gjennomsnittskvadrat blir dette vanskelig. Løsningen ble istedet å bruke resultatene "slik de er". Av den grunn blir det lagt mer vekt på de andre estimatene i analysene.

Chapter 6

Resultat

6.1 Måling av forvrengning

I denne seksjonen presenteres resultatet fra signalanalysene. Resultatet presenteres i en normalisert og standardisert form. Som tidligere nevnt benytter THD, SNR og SFDR en fundamental frekvens av en ren sinusfunksjon. Dette resulterer i at funksjonene vil konvergere forskjellig for de ulike signalformene, noe som gjør det vanskelig å sammenlikne resultatet fra ulike funksjoner. Resultatene presenteres derfor som *standardisert* og *normalisert* form.

Den normaliserte formen er presentert i den hensikt å synliggjøre forskjeller med tanke på de ulike DDS-designene. Den standardiserte formen blir på den andre siden et objektivt mål på forvrengning, siden alle målingene benytter samme referansesignal. Dette vil synliggjøre forskjeller mellom de ulike signalene.

6.1.1 Standardisert form

Tabell 6.1,6.2 og 6.3 viser estimatene for total harmonisk forvrengning (THD), signal-til-støyforhold (SNR) og spurious-free dynamic range (SFDR) for hver signalform over de ulike designene. Hver av verdiene i tabellen er beregnet fra et kvadratisk gjennomsnitt av 8 målinger, samplet av èn bølgeform for hver av frekvensene presentert i seksjon 4.1. Kolonnen lengst til høyre indikerer konvergens for estimatene med hensyn på signalkvalitet.

THD (RMS,dB):	Design1:	Design2:	Design3:	Grenseverdi:
Sinus:	53.544	54.273	27.89	∞
Triangel:	23.439	20.329	22.749	18.526 (exTHD)
Sagtann:	5.0273	5.4534	9.1782	3.0654 (exTHD)

TABLE 6.1: Tabellen viser THD for de ulike bølgefunksjonene generert av de ulike DDS-designene. Alle verdiene i tabellen er representert som et kvadratisk gjennomsnitt av THD for 8 ulike frekvenser.

SNR (RMS,dB):	Design1:	Design2:	Design3:	Grenseverdi:
Sinus:	46.862	47.307	29.674	∞
Triangel:	24.114	27.418	19.129	33.593 (exSNR)
Sagtann:	6.7695	7.5723	7.5030	8.3551 (exSNR)

TABLE 6.2: Tabellen viser SNR for de ulike bølgefunksjonene generert av de ulike DDS-designene. Alle verdiene i tabellen er representert som et kvadratisk gjennomsnitt av SNR for 8 ulike frekvenser.

SFDR (RMS,dB):	Design1:	Design2:	Design3:	Grenseverdi:
Sinus:	45.926	47.317	30.162	∞
Triangel:	17.49	17.234	17.101	19.054 (exSFDR)
Sagtann:	6.7486	5.9670	5.1309	5.8859 (exSFDR)

TABLE 6.3: Tabellen viser SFDR for de ulike bølgefunksjonene generert av de ulike DDS-designene. Alle verdiene i tabellen er representert som et kvadratisk gjennomsnitt av SFDR for 8 ulike frekvenser.

6.1.2 Normalisert form

Tabell 6.4 viser normalisert estimat for total harmonisk forvrengning målt for hvert av DDS-designene. Som tabellen indikerer konvergerer alle estimatene mot ∞. Resultatet for hver signaltype i tabellen er beregnet av det kvadratiske gjennomsnittet av resultatet(THD) fra 8 ulike frekvenser. I den nederste raden vises et aritmetisk gjennomsnitt av målingene.

Tabell 6.5 og 6.6 viser signal-til-støyforholdet(SNR) og "spurious free dynamic range" (SFDR) målt av hvert design. Rutene i tabellen er beregnet fra et aritmetisk gjennomsnitt av 6 målinger. Som tabellen indikerer konvergerer SNR og SFDR forskjellig mellom signalformene. Ettersom styrkekonvergens er invers proporsjonalt beregnes gjennomsnittet i de nederste kolonnene etter formen $Gjennomsnitt_{SNR,SFDR} = (Sinus + (-Triangel) + (-Sagtann))dB$. Det må forøvrig understrekes at disse verdiene kun er en *relativ representasjon* av forvrengningen.

THD(RMS,dB):	Design1:	Design2:	Design3:
Sinus:	53.544	54.273	27.89
Triangel(norm.):	94.290	86.952	78.579
Sagtann(norm.):	103.27	118.69	107.10
Gjennomsnitt(aritm.):	83.701	86.638	71.19

TABLE 6.4: Tabellen viser THD for de ulike bølgefunksjonene generert av de ulike DDS-designene.For triangel og sagtann-funksjoner brukes en normalisert verdi. Verdiene som er vist på tabellen er et kvadratisk gjennomsnitt av alle frekvensene som ble testet for det respektive designet.

SNR(RMS,dB):	Design1:	Design2:	Design3:	Grenseverdi
Sinus:	46.862	47.307	29.674	∞
Triangel(norm.):	14.102	11.242	17.306	0 (ref)
Sagtann(norm.):	3.4607	2.3798	4.0189	0 (ref)
Gjennomsnitt(aritm.):	9.7664	11.228	2.7787	∞

TABLE 6.5: Tabellen viser SNR for de ulike bølgefunksjonene generert av de ulike DDS-designene.For triangel og sagtann-funksjoner brukes en normalisert verdi. Verdiene som er vist på tabellen er et kvadratisk gjennomsnitt av alle frekvensene som ble testet for det respektive designet.

SFDR(RMS,dB):	Design1:	Design2:	Design3:	Grenseverdi
Sinus:	45.926	47.317	30.162	∞
Triangel(norm.):	4.7326	3.0483	5.7968	0 (ref)
Sagtann(norm.):	2.0860	1.4489	1.7341	0 (ref)
Gjennomsnitt(aritm.):	13.036	14.273	7.5437	∞

TABLE 6.6: Tabellen viser SFDR for de ulike bølgefunksjonene generert av de ulike DDS-designene.For triangel og sagtann-funksjoner brukes en normalisert verdi. Verdiene som er vist på tabellen er et kvadratisk gjennomsnitt av alle frekvensene som ble testet for det respektive designet.

6.2 Genetisk algoritme

I denne seksjonen presenteres resultatet fra den genetiske algoritmen. Som tidligere nevnt er hensikten å indikere om sinus, sagtann eller triangel funksjoner er gode kandidater som konfigurasjonssignaler for nanokarbon-materialet. Av hensyn til oversiktlighet gjøres en kort visuell analyse av resultatene etterhvert som de presenteres.

Tabell 6.7 viser det aritmetiske gjennomsnittet av antall fungerende enheter som

Fungerende enheter(avg):	Design1:	Design2:	Design3:	Gjennomsnitt:
Sinus:	23	9	20	17
Triangel:	10	9	9	9
Sagtann:	31	25	5	20
Gjennomsnitt:	21	14	11	vises ikke

TABLE 6.7: Tabellen viser gjennomsnittlig antall fungerende enheter som ble oppdaget under 6 forsøk (for hver rute), kategorisert etter bølgeformen som ble brukt som konfigurasjonssignaler og DDS-design.

Fitness(avg):	Design1:	Design2:	Design3:	Gjennomsnitt:
Sinus:	18	15	19	17
Triangel:	15	40	-10	15
Sagtann:	21	21	22	21
Gjennomsnitt:	18	25	10	vises ikke

TABLE 6.8: Tabellen viser gjennomsnitts-fitness for hver fungerende enhet kategorisert etter DDS-design og hvilket konfigurasjonssignal som ble brukt. Det fjerde feltet i hver rad og kolonne viser gjennomsnittsverdien for hver kategori.

ble oppdaget i løpet av 6 identiske forsøk for hvert konfigurasjonssignal og DDS-design. Kolonnen lengst til høyre viser gjennomsnittet av antall oppdagede elementer for en gitt signaltype. Den nederste raden viser det aritmetiske gjennomsnittet av antall oppdagede enheter, kategorisert etter hvilket design som genererte konfigurasjonssignalene. Tabell 6.8 viser gjennomsnittet av fitness som var tildelt alle fungerende enheter, i løpet av 6 identiske forsøk for hvert konfigurasjonssignal og DDS-design.

Tabellene indikerer at Design 1 presterer best med hensyn på fungerende enheter. Av signaltypene er det tydelig at Sagtann generelt gir høyest gjennomsnittsfitness og høyest antall fungerende enheter. Som vist på figuren er det generelt en høy spredning mellom antall fungerende enheter som ble funnet mellom forsøkene. For å finne et svar på dette vil en grundigere gjennomgang av forsøksdataene finne sted i resten av denne seksjonen.

6.2.1 Resultat for Design 1

Figur 6.1 viser det totale antall fungerende enheter som ble oppdaget på Design 1. Hvert av punktene på figuren angir hvor mange fungerende enheter som ble funnet i en bestemt generasjon, i løpet av 6 forsøk for hver bølgeform. Fra figuren ser det

ut til at algoritmen fant flest fungerende enheter for sinus mellom generasjon 1 og 30 generasjon. Etter generasjon 40 er det en tydelig overvekt av antall enheter som ble oppdaget i forsøkene der sagtann ble benyttet som konfigurasjonsignal, og det forekommer en gradvis reduksjon av antall oppdagede enheter oppdages i de øvrige forsøkene. Etter generasjon 60 oppdages et større antall fungerende enheter i forsøkene for sinus, omtrent samtidig som antall enheter for sagtann brått reduseres.

Fra figuren ser det ut til at det oppstår en betydelig endring i fenotypene blant de dominante individene i populasjonene i regelmessige intervaller på 20-40 generasjoner. Det ser ut til at denne *kaotiske oppførselen* i genotypene holder populasjonene i bevegelse og hindrer at de konvergerer fullstendig.

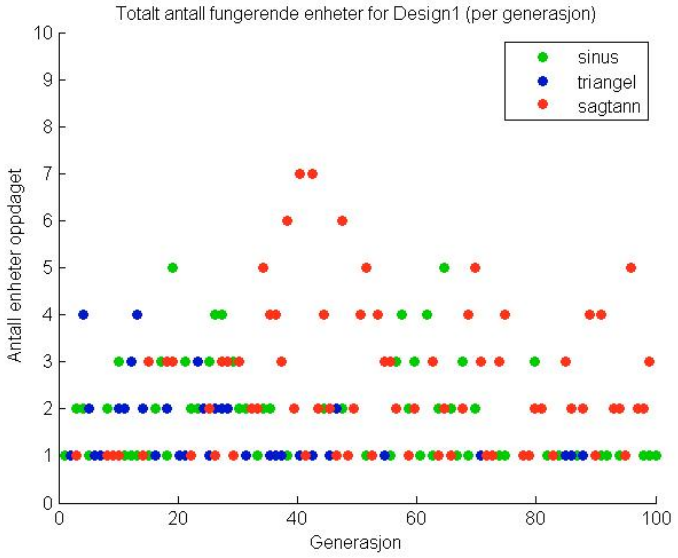

FIGURE 6.1: Grafen viser totalt antall enheter som ble oppdaget på Design 1, sammenlagt fra 6 forsøk med hver bølgeform.

Figur 6.2 viser også de samme tendensene til konvergent oppførsel. Figuren viser det aritmetiske gjennomsnittet av fitnessen for alle de 18 forsøkene der konfigurasjonssignalene ble generert av Design 1. Det kvadratiske gjennomsnittet(RMS) av fitnessverdiene fra kurven på figuren beregnet til $RMS = 660.29$, og fitnessverdiene ser ut til å stabilisere seg etter 40 generasjoner. Det ser også ut til at det oppstår endringer i variansen for fitnessverdiene ved generasjon 60 og 80.

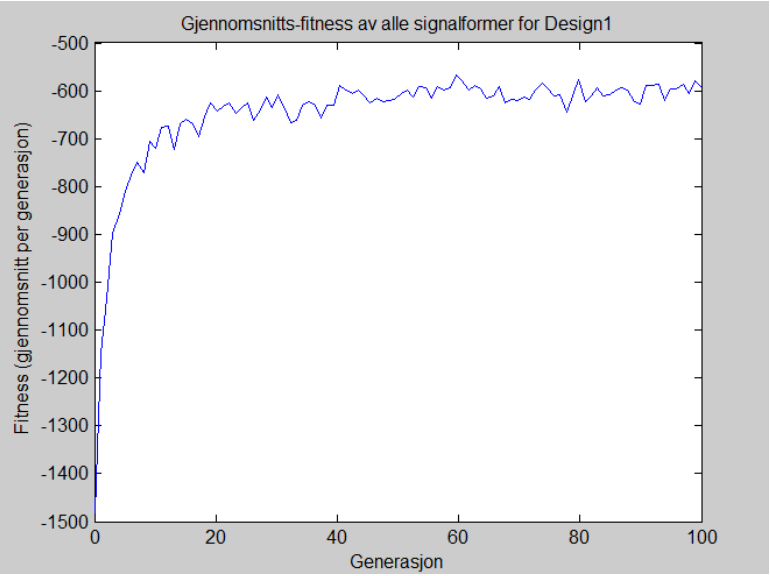

FIGURE 6.2: Grafen på figuren viser gjennomsnitt-fitness for alle individene i hver generasjon. Fitnessen er et aritmetisk gjennomsnitt av forsøk fra alle bølgetypene i Design 1.

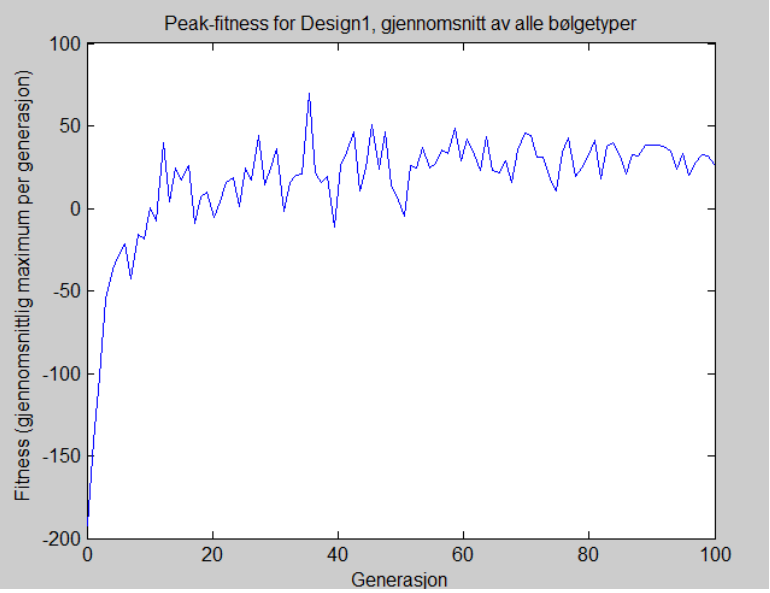

FIGURE 6.3: Grafen på figuren viser gjennomsnittet for maksimal fitness langs hver generasjon for alle bølgetypene i Design 1. Gjennomsnittet er beregnet fra 6 forsøk for hver bølgeform.

Figur 6.3 viser gjennomsnittet for maksimal fitness langs hver generasjon for alle forsøkene som ble utført med Design 1. Fra figuren kan vi se at denne fitness-verdien på grafen øker fra -200 til 0 mellom generasjon 1 og 20. Etter generasjon 50 reduseres variansen i fitness gradevis. Det kan også argumenteres for at fitness-verdiene på grafen har en viss *ringe-effekt*.

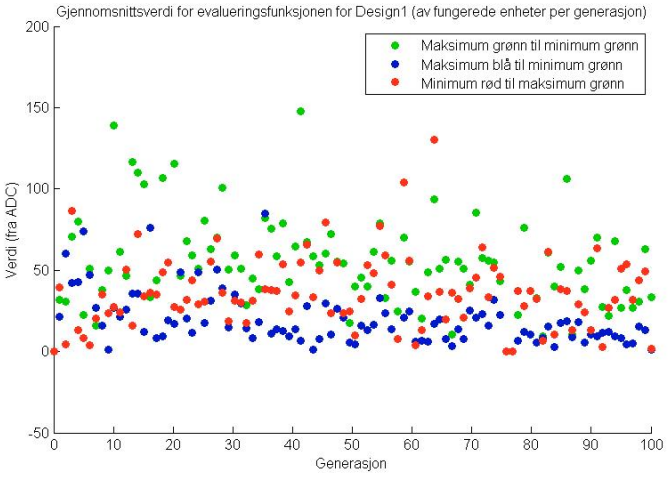

FIGURE 6.4: Vannfallgrafen på figuren viser gjennomsnittet av verdifordelingen i fitnessfunksjonen for alle fungerende enheter funnet for Design 1.

Figur 6.4 viser en vannfallgraf som representerer gjennomsnittet av verdifordelingen for fitnessfunksjonen i alle forsøkene der konfigurasjonssignalene ble generert av Design 1. I grafen er viser den genetiske algoritmen en tendens til å optimalisere fitness-funksjonen for størst mulig spenningsforskjell mellom rød og grønn node, samt minst mulig spenningsforskjell mellom de minste og største målte spenningene for grønn node. Spenningsforskjellen mellom grønn og blå node optimaliseres i en vesentlig mindre grad.

6.2.1.1 Forsøk med sinus

Figur 6.5 viser gjennomsnittet av fitness-verdien for populasjonene i alle forsøkene for Design 1 der sinus ble brukt som konfigurasjonssignal. Et spesielt trekk ved denne grafen er at variansen mellom fitnessverdiene er lav sammenliknet med de andre bølgeformene. Samtidig er differansen mellom gjennomsnittsfitness ved generasjon 0 og generasjon 100 lik 1000. Dette er en indikasjon på at det er enkelt for den genetiske algoritmen å optimalisere fitness for denne signaltypen.

6.2.1.2 Forsøk med Sagtann

Figur 6.6 viser gjennomsnittet av fitness-verdien for de 6 forsøkene der sagtann ble brukt som konfigurasjonssignal. Grafen indikerer at det er formmessige likheter mellom fitnesslandskapene i forsøkene for sinus og sagtann. Blant annet ser vi at

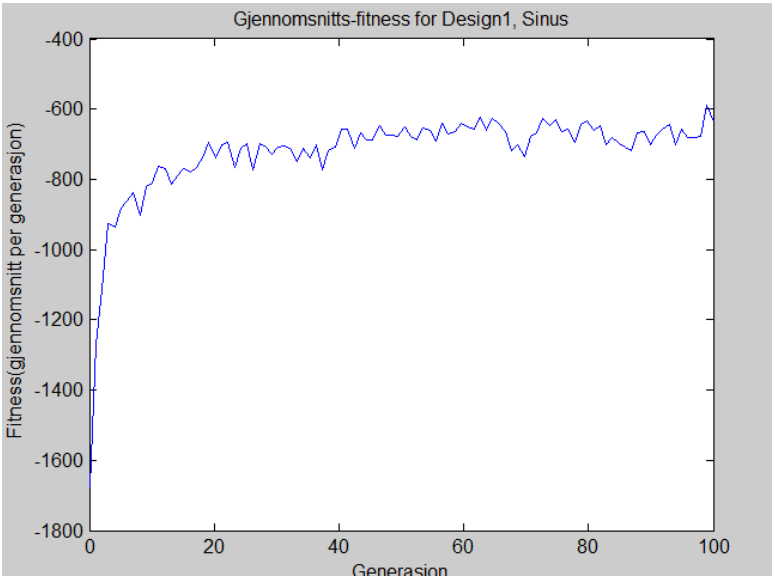

FIGURE 6.5: Figuren viser gjennomsnittlig fitness per generasjon der sinus er brukt som konfigurasjonsignal. Merk at oppløsningen på aksene av figuren er forskjellig fra de øvrige figurene.

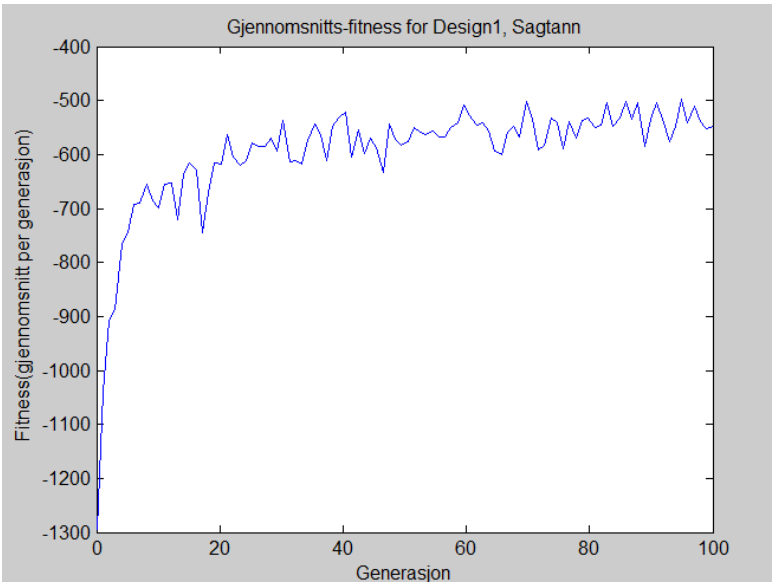

FIGURE 6.6: Figuren viser gjennomsnittet av fitness over populasjonene per generasjon i forsøk der sagtann er brukt som konfigurasjonsignal.

fitnessfunksjonen for begge grafene stabiliserer seg ved generasjon 40, og at landskapet hovedsakelig er svært stabilt mellom generasjon 40 og generasjon 100. En viktig forskjell er fitnessfunksjonen for sagtann stabiliserer seg for en større fitnessverdi. Dette gir en indikasjon på hvordan fitnesslandskapenes høyde og gjennomsnittsverdi påvirker antall fungerende enheter, dersom vi anser disse forsøkene som estimater av fitnesslandskapet.

6.2.1.3 Forsøk med Triangel

Figur 6.7 viser gjennomsnittet av fitness-verdien for populasjonene der triangel ble brukt som konfigurasjonssignal. Figuren viser at fitness-verdien for populasjonene når et topp-punkt ved generasjon 20, og stabiliserer seg ytterligere ved generasjon 40. Denne oppførselen avviker fra grafene for sinus og sagtann, som konvergerer mellom generasjon 0 og 40. Det ser også ut til at det er mer stokastisk støy med tanke på hvordan fitness evalueres. Dette være forårsaket av høy varians fra målingene i materialet og dermed et mer "stokastisk" fitnesslandskap som er mindre optimaliserbart.

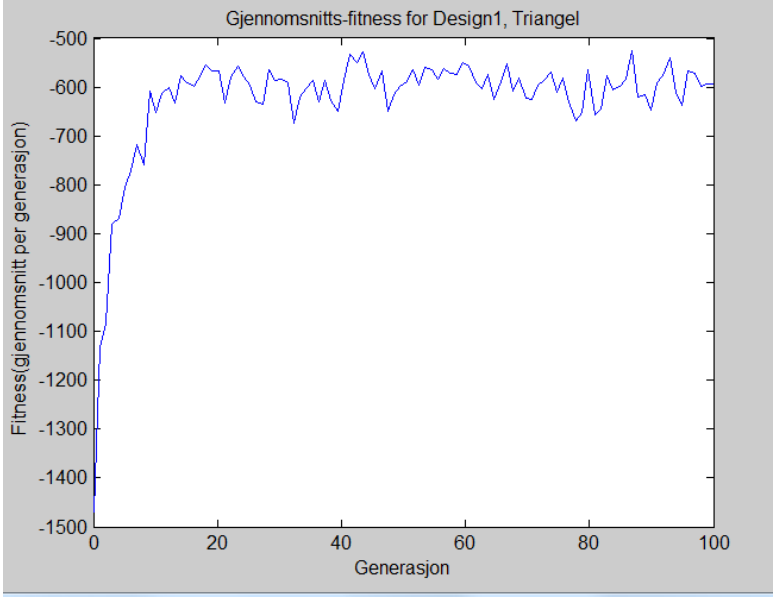

FIGURE 6.7

6.2.2 Resultat for Design 2

Fra tabell 6.7 er det tydelig at det i gjennomsnitt ble oppdaget vesentlig færre enheter i forsøkene der konfigurasjonssignalene ble generert av Design 2. For hvert forsøk ble det i gjennomsnitt oppdaget 14 fungerende enheter over 100 generasjoner. Figur 6.8 viser totalt antall oppdagede enheter for hver generasjon. Resultatet for hver fargekode er sammenlagt fra 6 forsøk der konfigurasjonssignalene for karbon-materialet ble representert gjennom de ulike bølgeformene. figuren viser en høy forekomst av fungerende enheter for sagtann mellom generasjon 10 og generasjon 40. Etter generasjon 50 reduseres antall oppdagede enheter i blant fenotypene i populasjonen for sagtann, mens det samtidig oppdages flere enheter for triangel og sinus. Figuren viser de samme tendensene for periodisk endring eller*kaotisk oppførsel* i fenotypene for de dominante, men i en vesentlig mindre grad.

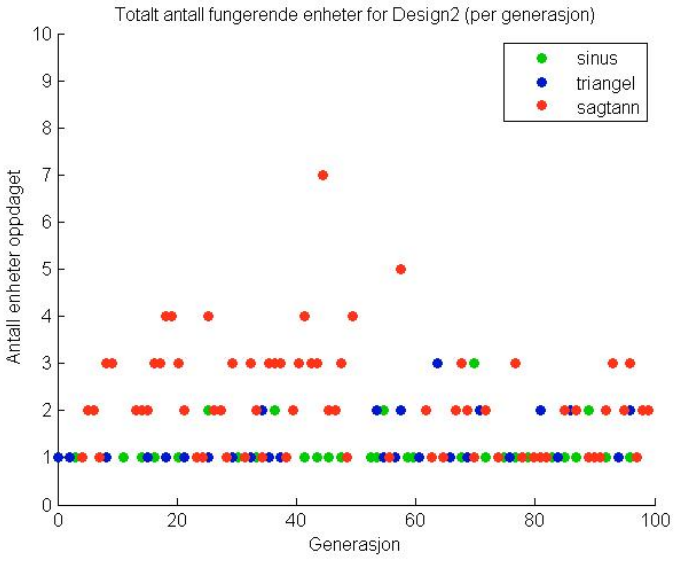

FIGURE 6.8: Grafen viser totalt antall enheter som ble oppdaget på Design 2, sammenlagt fra 6 forsøk med hver bølgeform.

Figur 6.11 viser gjennomsnittet av maksimal fitness for alle forsøkene som ble gjort for design 2. Figuren viser at fitness-landskapet konvergerer fram til generasjon 50-60. Det oppstår også tilsynelatende en brå endring i landskapet ved generasjon 45, noe som sannsynligvis har sammenheng med endringene i antall oppdagede enheter ved generasjon 50. En vesentlig forskjell fra Design 1 er ringe-effekten som ble observert for det tilsvarende resultatet for Design 1. Istedet konvergerer

funksjonen over tid fram til generasjon 60. En "bølge-effekt" oppstår mellom generasjon 60 og 100.

Dersom vi ser på fitness-landskapet for hele den genetiske populasjonen i figur 6.9, er det også tydelig at det oppstår endringer i landskapets kontinuitet ved generasjon 10, generasjon 20 og generasjon 40. En vesentlig forskjell fra Design 1 er at varians i gjennomsnittsfitness er betydelig større. Dette kan gi uttrykk for at det er mer støy i fitnesslandskapet når konfigurasjonssignalene genereres av Design 2. Hvis dette kommer av mindre forutsigbarhet i målingene fra materialet, vil forskjellene i antall enheter som ble oppdaget kunne forklares ved at populasjonens fenotypiske motstandsdyktighet mot endringer i genotypen, er lavere sammenliknet med Design 1. Det kan også nevnes at det kvadratiske gjennomsnittet av fitnessverdiene på figuren er beregnet til 734.16, noe som er lavere sammenliknet med Design 1.

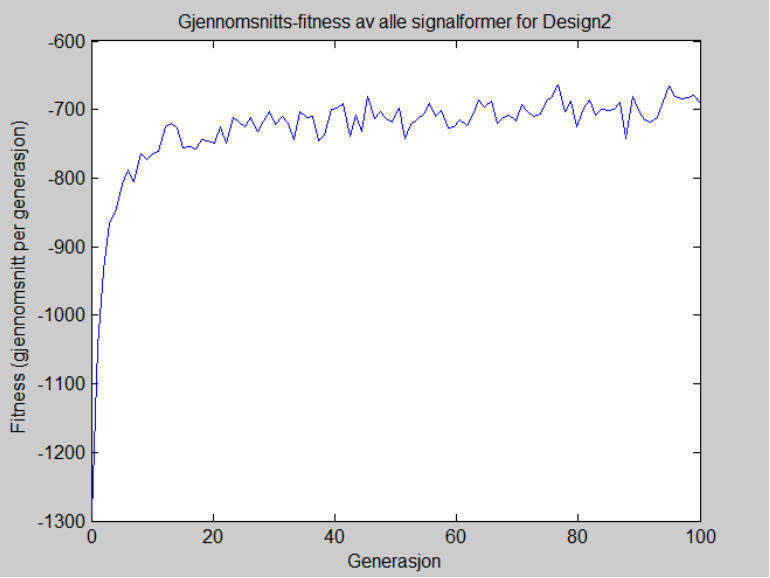

FIGURE 6.9: Grafen på figuren viser gjennomsnitt-fitness for alle individene i hver generasjon for alle bølgetypene i Design 2.

Figur 6.10 viser hvordan verdien av de 3 variablene som utgjør fitnessfunksjonen er fordelt per generasjon for alle fungerende enheter som ble oppdaget i forsøkene med Design 2. Sammenliknet med figur 6.4 er det ingen vesentlige forskjeller. Blant annet optimaliserer algoritmen fitness for de samme variablene, noe som tyder på at fitnesslandskapene i forsøkene for Design 1 og Design 2 har visse likheter (se [27]). Det er allikevel noen mindre forskjeller mellom figurene. En av dem er at

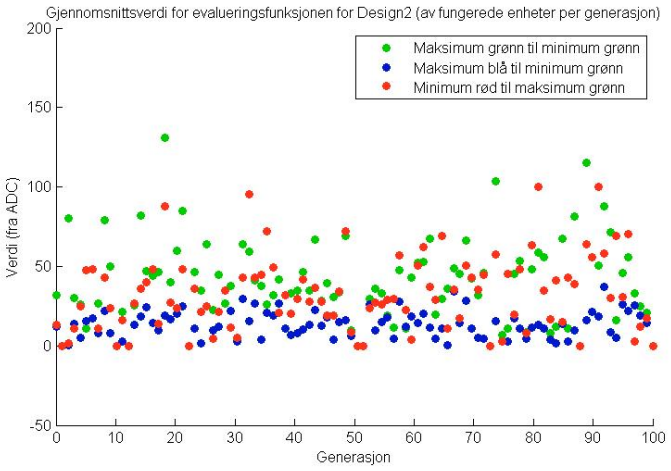

FIGURE 6.10: Vannfallgrafen på figuren viser gjennomsnittet av fitnessfunksjonen for alle fungerende enheter funnet for Design 2, dekomponert i form av 3 ortogonale vektorer per generasjon.

det er mindre spredning mellom punktene i figur 6.10. Dette gjelder spesielt de første 40 generasjonene. Det er også synlig diskontiunitet ved generasjon 50.

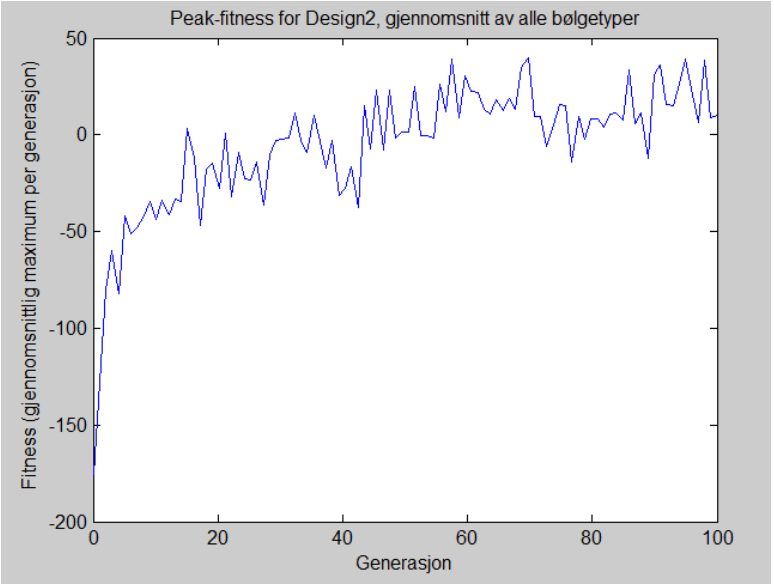

FIGURE 6.11: Grafen på figuren viser gjennomsnittet for maksimal fitness langs hver generasjon for alle bølgetypene i Design 2. Gjennomsnittet er beregnet fra 6 forsøk for hver bølgeform.

De øvrige resultatene som er assosiert med Design 2 viser hovedsakelig de samme tendensene som som påpekt tidligere, og blir derfor ikke gjennomgått eksplisitt. De øvrige grafene som er generert i forbindelse med forsøkene for Design 2 er vedlagt i Appendix C.1. Det samme er gjeldende for Design 3.

6.2.3 Resultat for Design 3

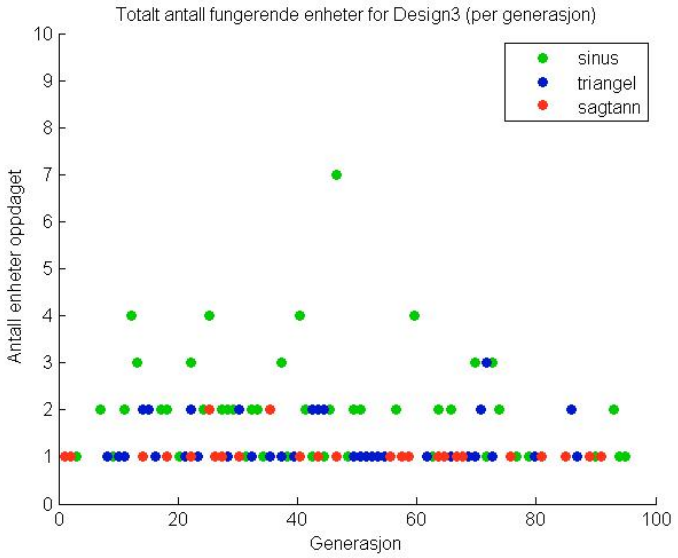

FIGURE 6.12: Grafen viser totalt antall fungerende enheter som ble oppdaget i forsøk der konfigurasjonssignaler generert gjennom Design 3. Hvert av punktene representerer summen av antall enheter sammenlagt fra 6 forsøk med en spesifikk bølgeform.

Antall fungerende enheter som ble oppdaget i forsøkene der Design 3 ble brukt er vist på figur 6.12. Figuren viser at det ble funnet et høyt antall fungerende enheter for sinus mellom generasjon 1 og 80. For triangel og sagtann er antall fungerende enheter lavt og tilsynelatende uniformt distribuert langs generasjonene. Forsøkene for sinus gav derimot bedre resultat. Figuren indikerer forøvrig at det er relativt stor variasjon mellom antall enheter som blir funnet for nærliggende generasjoner. Dette kan være en indikasjon på at den fenotypen for populasjonen er veldig følsom for genetiske mutasjoner.

Figur 6.13 viser at den gjennomsnittlige fitnessen i populasjonene konvergerer mot en grenseverdi i likhet med forsøkene gjort for de andre DDS-designene. En vesentlig forskjell er at konvergensen er mye svakere og det tar omtrent 40 generasjoner før veksten avtar. Det kvadratiske gjennomsnittet av fitnessverdiene på figuren er beregnet til 660.29, som er den laveste verdien blant de 3 DDS-designene. En annen ting er at den lokale variansen ser ut til å være noe høyere sammenliknet med de andre Designene. I tråd med observasjonene som er gjort for de tidligere resultatene kan dette forklare hvorfor algoritmen ga et lavt antall fungerende enheter for forsøkene der signalene ble generert av dette DDS-designet.

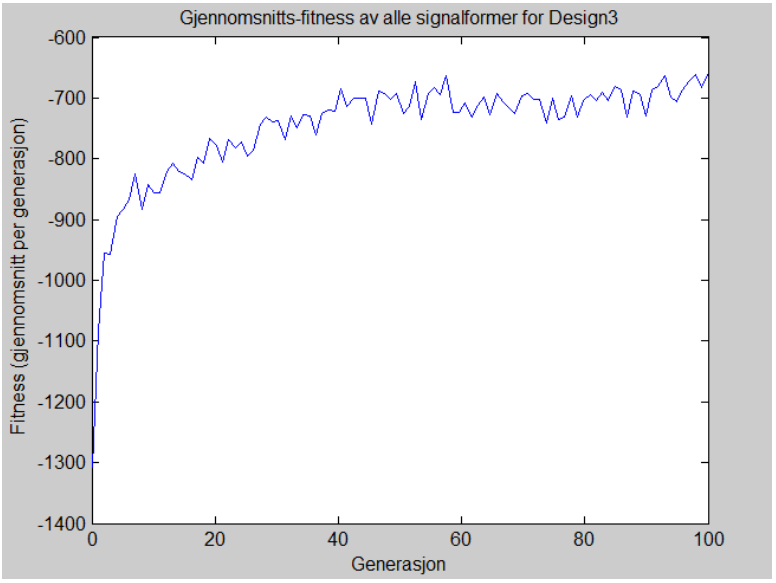

FIGURE 6.13: Grafen på figuren viser gjennomsnitt-fitness for alle individene i hver generasjon for alle bølgetypene i Design 3.

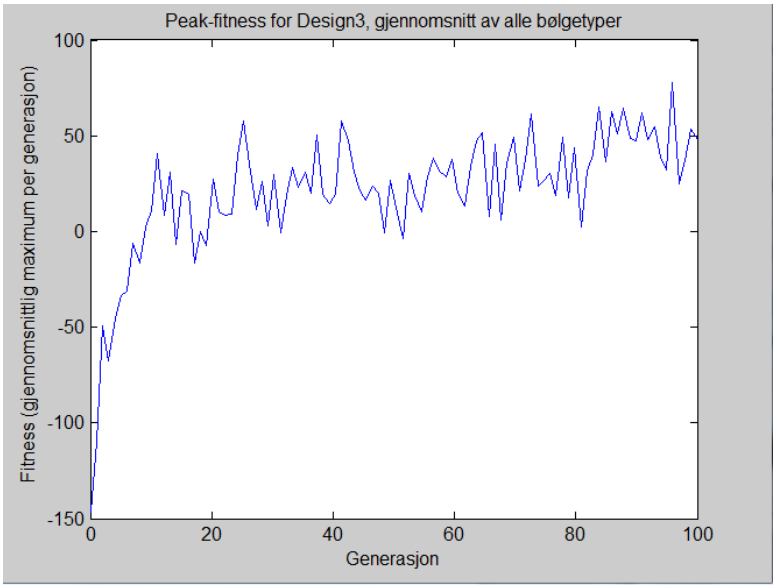

FIGURE 6.14: Grafen på figuren viser gjennomsnittet for maksimal fitness langs hver generasjon for alle bølgetypene i Design 3. Gjennomsnittet er beregnet fra 6 forsøk for hver bølgeform.

Gjennomsnittet av maksimal fitness for populasjonene er vist på figur 6.14. Grafen viser at maksimalfitness konvergerer mellom generasjon 1 og generasjon 10. En svak konvergens kan observeres mellom mellom generasjon 10 og 100. Dersom grafen sammenliknes med de tilsvarende grafene for forsøkene utført med de øvrige

designene, er det tydelig at variasjon i fitnessverdier oppstår tilfeldig. Til sammenlikning var det i figur 6.3 (Design 1) mulig å observere en "ringe-effekt" som gradvis avtok, mens i figur 6.11 (Design 2) kan det observeres periodiske tendenser i måten fitness-verdiene konvergerer. Selv om dette i mange tilfeller kan betraktes som støy, er den ikke stokastisk men *kaotisk*. Ren stokastisk "Støy" eller "bråk" i grafene ser derfor generelt ut til å ha en negativ påvirkning på algoritmens evne til å optimalisere fitness.

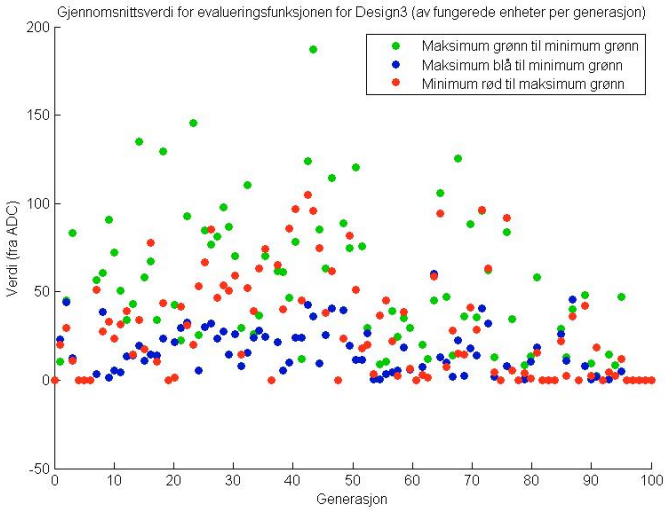

FIGURE 6.15: Vannfallgrafen på figuren viser gjennomsnittet av verdifordelingen i fitnessfunksjonen for alle fungerende enheter funnet for Design 3.

Sammenliknet med de andre DDS-designene, viser vannfallgrafen på figur 6.15 at den genetiske algoritmen hadde en større tendens til å optimalisere fitness for de røde og grønne nodene. Bidragene til fitnessverdier fra blå og rød node synker etter generasjon 50, mens den øker noe for grønn node (lav verdi for grønn node gir høy fitness). Dette kan tyde på at algoritmens evne til å finne fungerende enheter under forsøkene er i større grad er avhengig av start-tilstandene sammenliknet med de andre DDS-designene.

Chapter 7

Diskusjon

Dette kapittlet konkluderer arbeidet som er gjort i forbindelse med denne oppgaven. De viktigste momentene fra kapittel 6 oppsummeres, analyseres og konkluderes i seksjon 7.1. Seksjon 7.2 inneholder en kort oppsummering av de tidligere kapittlene samt en konklusjon av arbeidet.

7.1 Oppsummering og analyse av resultatene

Resultatene fra seksjon 6.2 tyder på at det er en sterk korrelasjon mellom fitness og antall fungerende enheter i forsøkene som ble gjennomført. I den forbindelse var den kvadratiske gjennomsnittsverdien av populasjonens fitness er vesentlig høyere i forsøkene med Design 1 sammenliknet med de andre DDS-designene. Fitnessgrafene for gjennomsnittspopulasjonene var også "glattere" for Design 1. Det ble i gjennomsnitt funnet minst 33 % flere fungerende enheter i forsøkene der Design 1 ble brukt, sammenliknet med de andre DDS-designene. Gjennomsnittet av maksimal-fitness for populasjonene konvergerte raskest og var også generelt høyest for Design 1.

Seksjon 6.1.2 viser forvrengning av de ulike signalene definert etter hvilken grad signalene som genereres avviker fra et referansesignal for sin respektive bølgeform. Estimatene for THD, SNR og SFDR indikerer at signalene som genereres av Design 2 har lavest forvrengning. Dette skyldes sannsynligvis dither-generatoren ved utgangen fra faseakkumulatoren, som kun ble implementert for dette designet. Sammenliknet med Design 2 viste gjennomsnittet av estimatene for Design 1 mer

periodisk og ikke-periodisk og støy på signalet. Estimatene indikerer også at det er mest forvrengning og støy på signalene som genereres av Design 3.

7.1.1 Er Sinus, sagtann og triangel signaler egnet som konfigurasjonssignaler?

Resultatene fra eksperimentene med den genetiske algoritmen indikerer at sagtann er det beste valget av konfigurasjonssignal, ettersom algoritmen fant flest fungerende enheter for disse forsøkene da signalene ble generert gjennom Design 1 og Design 2. En analyse av fitnessverdiene for populasjonene viste også i gjennomsnitt høyere gjennomsnittsverdier og maksverdier for populasjonene i forsøkene.

Et betydelig antall enheter ble også funnet i forsøkene der andre signalformer ble generert gjennom de ulike DDS-designene. Det kan det derfor konkluderes med at alle de valgte bølgeformene er egnet som konfigurasjonssignaler for det valgte materialet og den genetiske algoritmen. Som tidligere nevnt bør det være tilstrekkelig å vise at algoritmen er i stand til å oppdage minst 1 fungerende enhet for hver bølgeform for at denne formuleringen skal være sann.

Et forbehold er at det sannsynligvis forekommer statistisk støy på landskapet [27]. Det vil si at resultatene ikke nødvendigvis er reproduserbare, selv om populasjonen har en identisk bane i faserommet. I tillegg er det ikke nødvendigvis slik dette resultatet er portabelt mellom forskjellige materialer og fitnessfunksjoner. Det vil derfor naturligvis være en ukjent risiko for at hypotesen aksepteres på feil grunnlag. De 3 første arbeidshypotesene som ble presentert i innledningen aksepteres derfor for et ukjent konfidensintervall.

7.1.2 Kan forvrengning på konfigurasjonssignalene utnyttes i en genetisk algoritme?

Fra resultatene i forrige seksjon kan det konkluderes at forsøkene med Design 3 indikerer at clipping for sagtann og triangel gir et dårligere resultat med hensyn på både fitnessverdiene som ble målt, og antall enheter som ble oppdaget. Det ble også observert vesentlig flere fungerende enheter sammenliknet med Design 2 i forsøkene der sinus ble brukt som konfigurasjonssignaler.

Forvrengning som oppstår i signalene som genereres av Design 1 kan ses på som en form for "dynamisk forvrengning". Med "dynamisk forvrengning" menes det at Design 1 varierer oppløsningen på signalet som genereres som en funksjon av frekvensen. Dette resulterer i at svært nærliggende frekvenser kan ha store variasjoner i oppløsningen på signalet. I denne sammenheng kan forvrengning som oppstår i Design 3 defineres som "statisk" ettersom det ikke oppstår slike variasjoner som følger av "clipping".

Dersom forvrengning settes i kontekst med resultatene fra den genetiske algoritmen impliserer dette at den genetiske algoritmen utnytter den dynamiske oppløsningen. Dette kommer av at Design 1 i gjennomsnitt ga 33 % flere fungerende enheter sammenliknet med Design 2. Denne observasjonen kan forklares gjennom NK-modellen presentert i seksjon 2.1.2.1. Som tidligere nevnt presiserer modellen at "ulendtheten" i fitness-landskapet kan beskrives gjennom N og K. Verdien K kontrollerer epistasen i modellen, eller hvor mange K andre nærliggende loki som påvirker en gitt lokus. For den genetiske algoritmen representerer N lengden på den genetiske representasjonen av hvert individ, og en høy verdi av K gir generelt lavere "glatthet".

Figur 7.1 representerer en visualisering av fitness-landskapet og er basert på resultatene i forrige seksjon samt en tolkning av NK-modellen, for ukjente verdier av N og K. "Generasjonsvektorene" på figuren representerer frekvensen i hver de digitale genotypene. En mulig forklaring på resultatene er at i Design 1 vil dette frekvens-genet generelt inneholde mer informasjon sammenliknet med de andre designene. Dette er forklart utifra euklidiansk distansenorm på figuren, som resulterer i at frekvensgenet er større i Design 1 sammenliknet med Design 2 og Design 3. Alternativt kan dette også beskrives direkte gjennom NK-modellen dersom vi legger til innholdet i formel 7.1. Tensoren S_i i formelen beskriver en mapping av hvert digitale lokus C til et digitalt gen, og tensoren S i den orginale modellen beskriver hvordan hvert digitale gen realiseres og gir en fitness i det fysiske domenet. Det bør også nevnes at $C_{Design1|res}$ kan oversettes som C_{res} gitt Design1. Leddet $(C_{frek} + (Design1|C_{res}))^i$ i formelen tilsier at dette lokuset er betydelig større for Design 1 sammenliknet med de andre designene. Dersom vi antar at dette er ensbetydene med at genet har større påvirkning på fitness målt av det fysiske domenet, kan det enkelt forklares gjennom *Shannon-entropi*[] at variablen K har en lavere verdi som følger av at de andre genene er*recessive*. Denne tankegangen kan også enkelt formaliseres ved å utvide NK modellen til et

NKp-landskap, hvor p er sannsyneligheten for at en gitt allele ikke påvirker fitnessverdien [27]. At K er lavere i fitnesslandskapene for Design 1 støttes også av observasjonene om at gjennomsnittspopulasjonene konvergerer noe raskere og har lavere varianse sammenliknet med de andre DDS-designene. Dette er egenskaper som gir et "glatt" fitnesslandskap i følge Kauffmans modell.

$$S_i = g(S_i, (C_{frek} + C_{Design1|res})^i, C^i_{pin}, ...C^i_{ampl}) \tag{7.1}$$

Hvordan et glattere landskap kan oppnå høyere fitness er forøvrig vanskelig å si. På grunn av at det er en direkte relasjon mellom fitness og spenningen som måles, er én mulighet for at det er forårsaket av mindre variasjon i spenningen målt fra materialet for identiske eller svært like "dominante" gener. Lavere varians gjør at fitnessverdiene som angies blir mindre usikre og at fenotypene derfor generelt er mer stabile. På den måten kan også høyere stabilitet gi høyere fitness uten at det øker sannsynligheten for at algoritmen setter seg fast i lokale maksima. Dette prinsippet er også dokumentert i studiene Levinthal gjennomførte i forbindelse med "Ulendte fitnesslandskap"[24].I den forbindelse kan det nevnes at testene med firkantbølger i gjennomsnitt ga høyere fitnessverdier sammenliknet med de andre periodiske funksjonene (se seksjon 5.3.2). På tross av dette ble det oppdaget 2 fungerende enheter med lave fitnessverdier sammenliknet med forsøkene gjort for de andre designene. Ettersom det var mulig å observere høy konvergens i grafene for gjennomsnittsfitness og maksimal fitness var det tydelig at populasjonen gikk inn i et lokalt maksimum. Dette kan tyde på at fitnesslandskapet for firkantbølger er glattere. Et ubesvart spørsmål i forbindelse med dette er hvilken grad og rolle skaleringsvarians av landskapet påvirker resultatene.

Mindre variasjon i spenningen målt fra materialet i forsøkene der Design 1 ble brukt er forøvrig vanskelig å forklare. Det er heller ikke usannsynelig at det kan ha oppstått som en emergent egenskap som forekommer regelmessig under disse forholdene. I den forbindelse påpeker Levinthal: "The ability of established organizations to respond to changing environments is importantly conditioned by the extent to which elements of organizational form interact in their effect on organizational fitness. Tightly coupled organizations are subject to high rates of failure in changing environments [...] but not for more loosely coupled organizations that are able to engage in effective local adaptation"[24]. I denne sammenheng kan

det bety at signalene påvirker materialet slik at den molekylære strukturen organiseres på mer effektiv måte, med hensyn på hele systemets potensial til å finne fungerende enheter. Analysen av fitness-landskapet er forøvrig kun en hypotese, men kan forklare hvorfor det ble funnet flest fungerende enheter i Design 1 for alle bølgeformene.

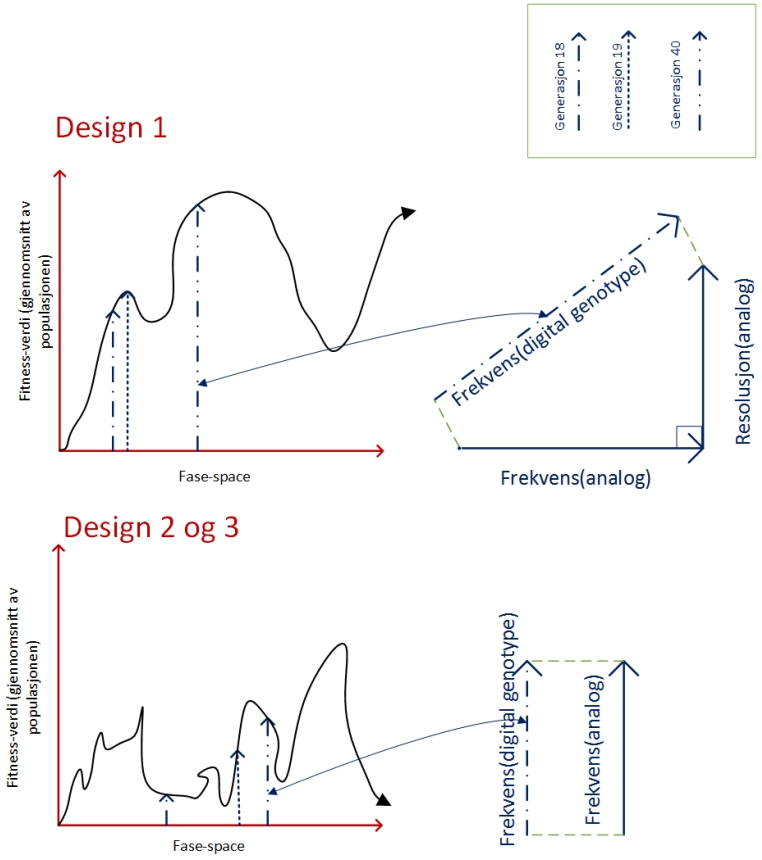

FIGURE 7.1: Figuren viser en grafisk forenkling av hvordan relasjonen mellom genotype og fitness i faserommet kan se ut mellom de forskjellige designene. Den røde horisontale aksen på grafene representerer en hypotetisk forutbestemt retning i faserommet. Det bør presiseres at faserommet er visualisert som et "pre-Hilbertrom".

Fra denne analysen kan det konkluderes med at resultatene viser gode indikasjoner med hensyn på arbeidshypotesen om at en genetisk algoritme er i stand til å utnytte periodiske artefakter på signalene. I realiteten tyder resultatene på at den genetiske algoritmen er i stand til å optimalisere periodiske artefakter for signalformene. I den forbindelse bør det nevnes at indikasjonene på at forvrengning kan gi en lavere K verdi kan være viktig. Dette er sett i sammenheng med teoriene for beregninger "ved kanten av kaos" (se kapittel 2). Ettersom K er proporsjonal med variasjon i fitnesslandskapet betyr dette at variabel resolusjon på signalet skaper

en høyere grad av orden på fitnesslandskapet. Dette er derfor en positiv indikasjon for denne teorien med hensyn på signalvalg, ettersom det er enkelt å vise gjennom en entropi-modell eller tidsserieanalyse, at forvrengning på signalet påfører mer "kaotisk" informasjon og dermed *kaotisk oppførsel*. Dersom vi ser dette i kontekst med Nichele et als arbeid hvor det ble påvist kaotisk oppførsel i materialet for ulike signaler[6], kan dette bety at det er mulig å **indusere den kaotiske oppførselen i systemet som er nødvendig for å utføre beregninger ved å påføre periodiske artefakter på konfigurasjonssignalene, med forbehold om at disse kan varieres av den genetiske algoritmen**.

Selv om disse forsøkene har gitt et godt grunnlag for å anta at hypotesen om at periodiske artefakter i konfigurasjonssignalene kan utnyttes, bør det forøvrig konkluderes at arbeidshypotesen ikke bør aksepteres basert på resultatene som ble funnet. Ettersom den statistiske fordelingen, forventning og standardavvik for datasettene i resultatet er ukjent, er det nødvendig at en standard T-test utføres for en frihetsgrad av 29 eller høyere[36]. Det er derfor nødvendig å utføre flere forsøk før hypotesen eventuelt kan aksepteres på et rent kvantitativt grunnlag.

7.2 Konklusjon

For denne oppgaven var målet å verifisere hypotesen om at triangel, sinus og sagtannfunksjoner er egnet som konfigurasjonssignaler mot et nanokarbonmateriale for en genetisk algoritme. I tilegg var målet å undersøke hypotesen om at periodiske artefakter på signalet med fordel kan utnyttes i en genetisk algoritme. 3 varianter av direkte digitale syntesere ble implementert på Mecobo for å generere de foreslåtte periodiske funksjonene. Gjennom digitale simuleringer ble forvrengning estimert for hvert DDS-design og hver periodiske funksjon. Hver av de nye signalene ble evaluert gjennom en genetisk algoritme. Forsøkene ble gjennomført med antall fungerende enheter som algoritmen oppdaget for et gitt signal som evalueringskriterium. Resultatene ble deretter analysert ved å studere sammenhengen mellom konfigurasjonssignalene som ble brukt i forsøkene og alfa/dominante individer i populasjonene med utgangspunkt i NK modellen.

Fra analysene av resultatene gjøres følgende oppsummering av de viktigste momentene:

- Det ble observert at høyest fitness og flest fungerende enheter for forsøkene der sagtannbølger ble generert gjennom Design 1.
- Blant de nye signalformene som ble utprøvd ble det funnet flest fungerende enheter i forsøkene når konfigurasjonssignalene ble generert gjennom Design 1.
- Den er ingen synlig forskjell i hvordan den genetiske algoritmen optimaliserer fenotypen for populasjonene av dominante individer mellom forsøkene gjort for Design 1 og Design 2.
- De gjennomsnittlige fitness-verdiene for populasjonene var høyest for Design 1.
- Det ser ut til å være en positiv korrelasjon mellom "glatthet" på fitness-grafene, variabel resolusjon på konfigurasjonssignalene, gjennomsnittlig fitnessverdi og gjennomsnittlig maksimumsverdi for fitness.
- Den estimerte gjennomsnittlige "høyden" på fitnesslandskapet er størst for Design 1.
- Det ser ut til at det er en sammenheng mellom lokal varians eller "glatthet" på fitness-grafene og algoritmens generelle prestasjon.
- Basert på en kvalitativ gjennomgang av øvrige observasjonene ble det estimert at fitnesslandskapet for forsøkene der Design 1 ble brukt var "glattere" sammenliknet med fitnesslandskapene for de øvrige forsøkene.

Fra denne listen kan det utifra resultatene konkluderes at dersom periodiske artefakter på signalet kan kontrolleres av den genetiske algoritmen, kan dette tilføre kaotisk oppførsel i SWNCT-materialet som med fordel kan brukes til å utføre beregninger. Faktisk ser det ut til å redusere stokastisk støy på fitnessgrafene fra forsøkene. Dersom fitnessgrafene er forventningsrette vil arbeidshypotesen kunne aksepteres. På den andre siden må vi gå utifra at det er en ukjent grad av usikkerhet grunnet statistisk støy. Det er også en mulighet for at resultatene og observasjonene som ble gjort ikke er portable for andre fitnessfunksjoner og materialer. På grunn av dette kunne forøvrig ikke denne arbeidshypotesen entydig aksepteres. Av samme grunn er det heller ikke beregnet statistisk signifikans for observasjonene i listen over. Det ble derfor konkludert med at eksperimentene ikke kunne falsifisere arbeidshypotesene.

De samme problemene gjelder også for de øvrige arbeidshypotesene om at "de nye signalene er egnede basiser for konfigurasjonssignaler", men ettersom disse ble utformet med hensyn på å indikere om videre arbeid er hensiktsmessig, ble hypotesene akseptert for et ukjent konfidensintervall. På bakgrunn av dette anbefales det derfor å gjennomføre flere forsøk og analyser for sagtann og eventuelt sinus-funskjoner. Flere forsøk i forbindelse med periodiske forstyrrelser på signaler bør også gjennomføres med hensyn på statistisk signifikans.

Resten av denne seksjonen er dedikert til forslag til videre arbeid for å forbedre implementasjonene og metodene brukt under denne avhandlingen.

7.2.1 Videre arbeid

For å akseptere de framsatte hypotesene for et konfidensnivå på et generelt kvantitativt grunnlag, bør det foretas en hypotesetest. For å utføre en standard T-test for begge hypotesene er det forøvrig nødvendig å utføre minst 30 forsøk for hver bølgeform og hver DDS-design[36].

Det kunne også vært interessant å se hvordan resultatet påvirkes dersom man varierer mellom de ulike signaltypene i løpet av samme forsøk. I den forbindelse kan det også være interessant å se hva som skjer når man varierer antall konfigurasjonssignaler over forskjellige individer i samme populasjon. Dette kan for eksempel implementeres som en variabel som den genetiske algoritmen kan optimalisere.

Det kan nevnes at det bør gjøres en tidsserie-analyse av signalet ved å sample signalet direkte fra DACen. Grunnen til at dette bør gjøres er at Design 1 har variabel resolusjon over frekvensdomenet. Resolusjonen har ingen praktisk betydning under fourier transformasjoner i det digitale domenet, men vil påvirke rekonstruksjon av signalet gjennom DACen. Av hensyn til dette bør det gjennomføres en analyse av den analoge spenningen som generes gjennom DACen. I tillegg til å gi en bedre kvantitativ beskrivelse av designene, kan metoden som ble brukt av Nichele et al gi mer informasjon om faserommet[6]. For raskere gjennomføring av eksperimenter bør det vurderes å redesigne mikrokontrolleren på EM-plattformen for å redusere overhead for håndtering av sekvenser. Det kan nevnes at det tok mikrokontrolleren gjennomsnittlig 300 ms for å starte og deaktivere 16 sekvenser, som er 3 ganger så lang tid som den reelle tidsvarigheten for sekvensene. Hovedårsakene til dette er mest antakelig minneaksesser. En effektive måte å redusere "overheaden" er

gjennom parallellisering av oppgavene. En mulighet er å inkludere ekstra prosessorer med parallellt minne slik at flere sekvenser kan aktiveres og deaktiveres samtidig. Dette kan også gjennomføres med minimal overhead ettersom det er svært få strukturelle avhengigheter mellom dataene for de forskjellige sekvensene.

Appendix A

Skjemaer

I dette vedlegget ligger støtteillustrasjoner og skjemaer.

A.1 Arkitektur

A.1.1 Generatormodul for sinus

A.1.2 Generatormodul for triangel

A.2 FSM

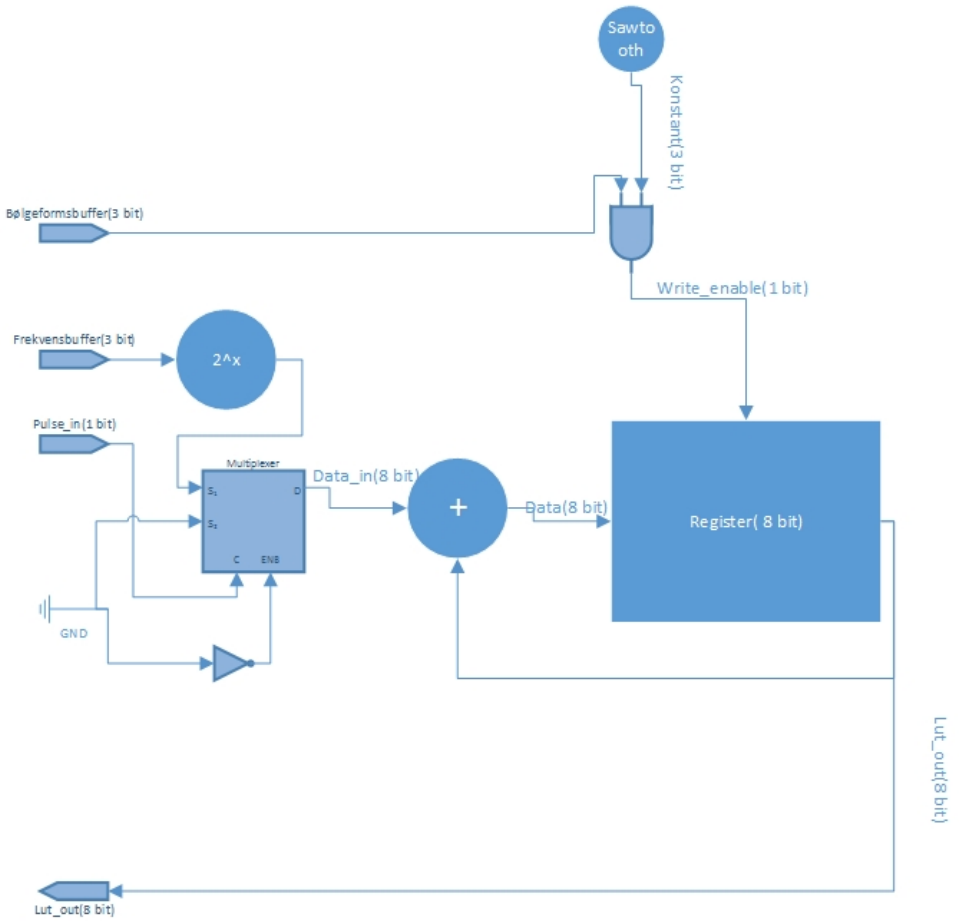

FIGURE A.1: Figuren viser oversikt over utgangsmodul for sagtann for design 1.

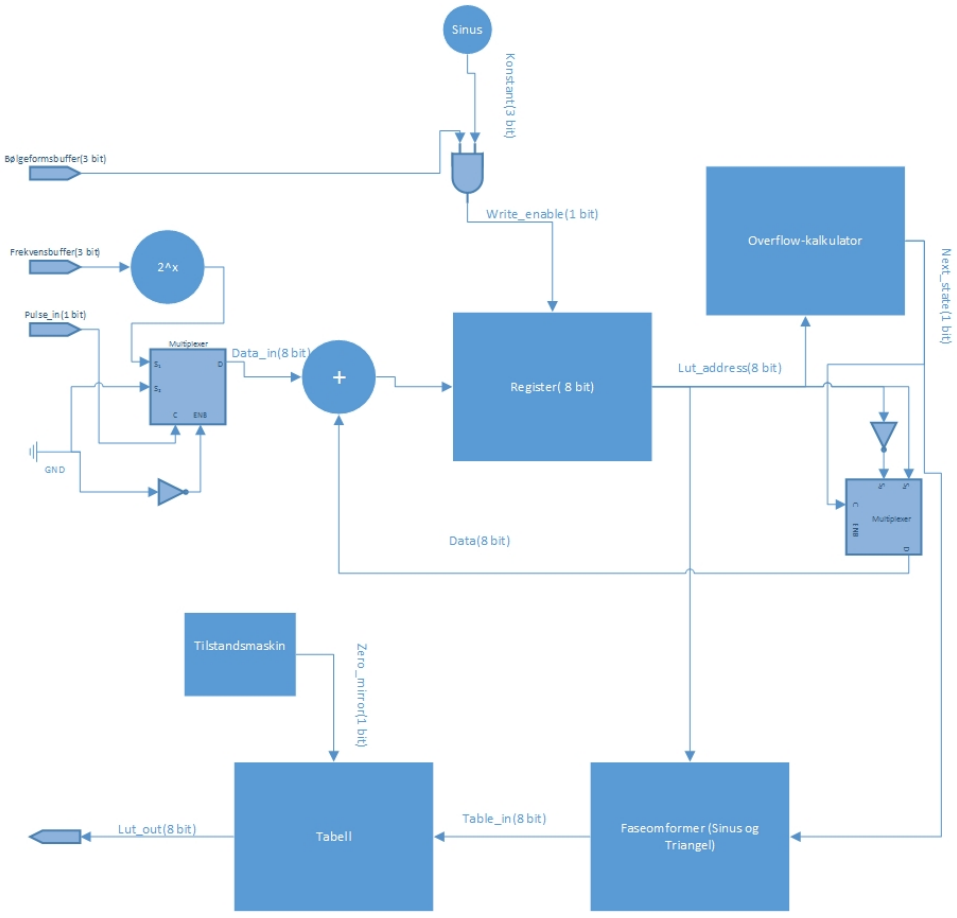

FIGURE A.2: Figuren viser en oversikt av generatormodul for sinus for design 1.

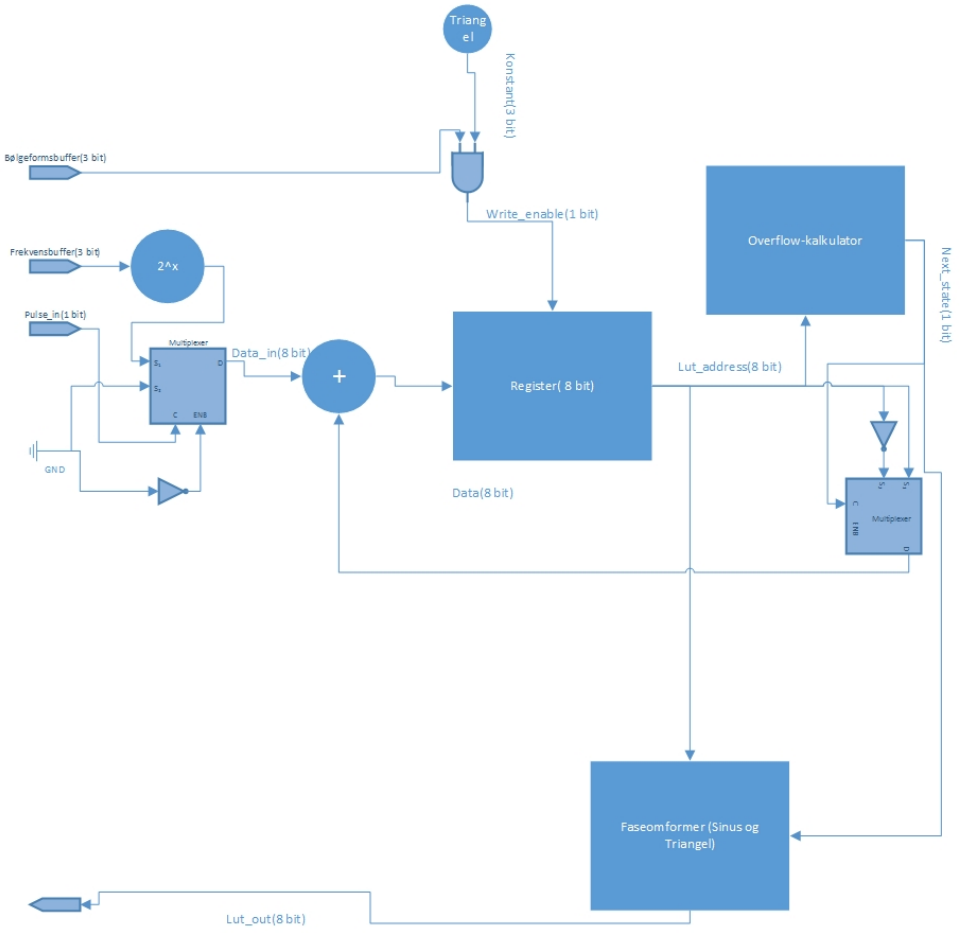

FIGURE A.3: Figuren viser en oversikt av generatormodul for triangel for design 1.

Appendix B

Brukerveiledning

B.1 Brukermanual

Dette vedlegget er ment å gi en brukerveiledning for oppstart av eksperiment. Oppsett av utviklermiljø og drivere er forklart i [32]. Det bør opplyses at brukerveiledningen og rammeverket er skrevet for Windows 7. Det bør nevnes at filbanene i enkelte av prosjektfilene muligens må forandres for å kompilere koden på nytt. De kjørebare filene skal i hovedsak fungere på alle windows 7 systemer etter oppsett av miljø og drivere.

B.1.1 Nødvendig utstyr

- Vertsmaskin med oppsatt miljø.
- EFM32GG STK3700
- Mecobo hovedkort
- Mecobo Datterbrett
- SWD kabel
- Mini-usb kabel
- USB kabel

B.1.2 Framgangsmåte

1. Installer drivere for EM-brettet og koble en USB-kabel mellom EM-brettet (se [32]).

1.1 Koble opp STK3700 debuggeren.

2. Åpne Simplicity Studio og overfør filen "mecobo_design[X].hex til EM-brettet.

3. Åpne ledetekst og gå til mappen kalt "Host".

3.1. Skriv "winhost1 1 -b "[...]FPGA/mecobo_design1.bit for å bruke Design 1.

3.2. Skriv "winhost1 0 -b "[...]FPGA/mecobo_design[X].bit for å bruke Design 2 eller Design 3.

3.3. Påse at designvalget har vært konsistent for alle stegene hittil. Rammeverket vil fremdeles starte opp og kjøre på tross av inkonsistens i designvalg, men vil ikke fungere slik det er ment.

4. Dobbelklikk på kjøringsfilen for den genetiske algoritmen. Kjøringsfilen kalles EMClient og plassert i mappen "EMServer/bin" Pass på at filen ligger i den samme mappen som den opprinnelig kom i.

5. Data fra eksperimentene vil dukke opp i undermappene "sinus", "triangel" og "sagtann".

B.2 Digitalt vedlegg

Vedlagt i det digitale vedlegget er kildekode for eksperimentene, datasett for resultatene, og de oppdaterte versjonene av Mecobo (DDS-designene). Med unntak av kildekoden for den genetiske algoritmen er alle ressursene fordelt mellom mappene "Design1", "Design2" og "Design3". Disse er organisert i en hierarkisk struktur hvor de fleste av dem inkluderer en "README"-fil som forklarer kort innholdet på engelsk. Disse er alltid skrevet i ".txt"-format.

Det anbefales å benytte det digitale vedlegget for følgende tilfeller:

- Dersom noe av innholdet i denne avhandlingen er uklar.

- Ved påbygning/videreføring av arbeidet.
- For å gjenta eksperiment og metodene som er presentert i denne avhandlingen.

Det bør nevnes at Design1 er basert på forprosjekt høsten 2014 (videreutvikling av eget arbeid). Av den grunn vil noen av filene som er assosiert direkte med DDS-designet være datert til den perioden.

Appendix C

Data og målinger

Dette vedlegget inneholder data fra eksperimentene som er gjennomført. Rådata fra målinger og estimater som er gjort i forbindelse med forvrengning er vedlagt i det digitale vedlegget.

C.1 Fitnessgrafer for GA

Denne seksjonen innholder fitnessgrafer for den genetiske algoritmen som ikke ble omtalt direkte i teksten. Som tidligere nevnt er grafene og resultatene presentert i denne avhandlingen generert fra større datasett. Disse er lagret som "rådata" i det digitale vedlegget.

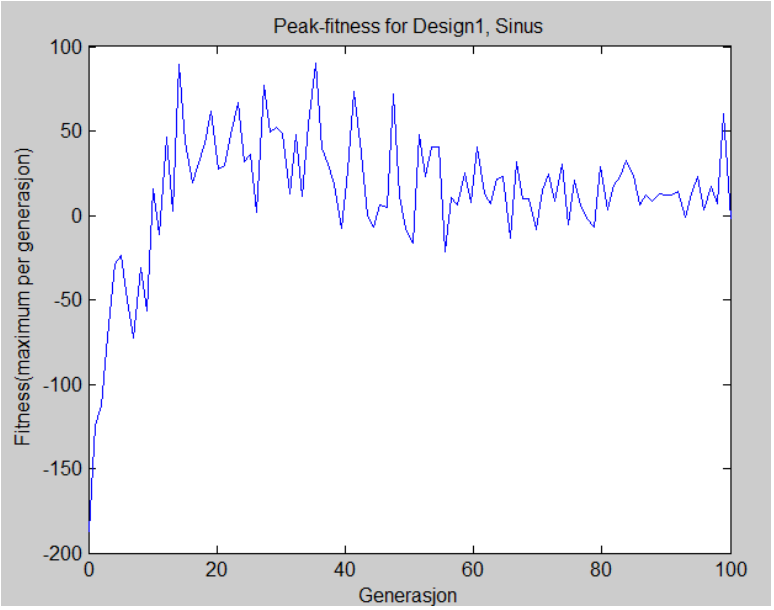

FIGURE C.1: Figuren viser maksimal fitness per generasjon der sinus er brukt som konfigurasjonsignal.

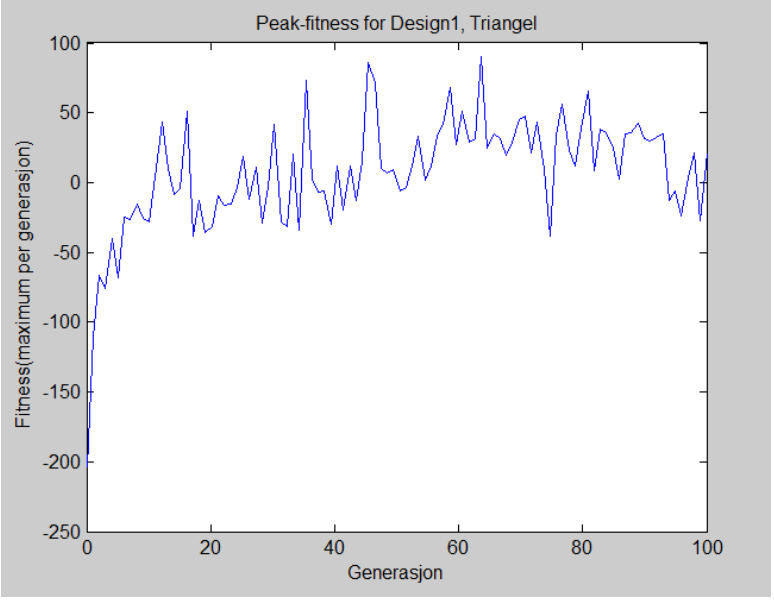

FIGURE C.2

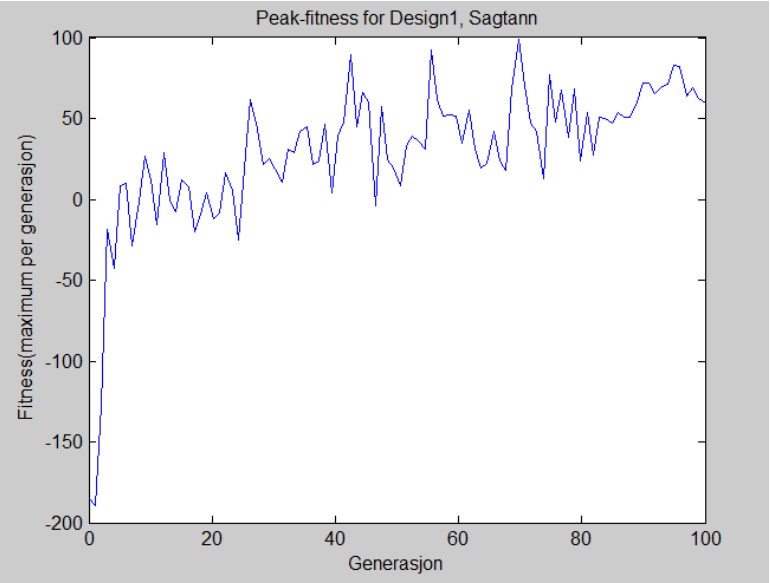
Peak-fitness for Design1, Sagtann
Fitness(maximum per generasjon)
Generasjon
100
50
0
-50
-100
-150
-200
0
20
40
60
80
100

FIGURE C.3

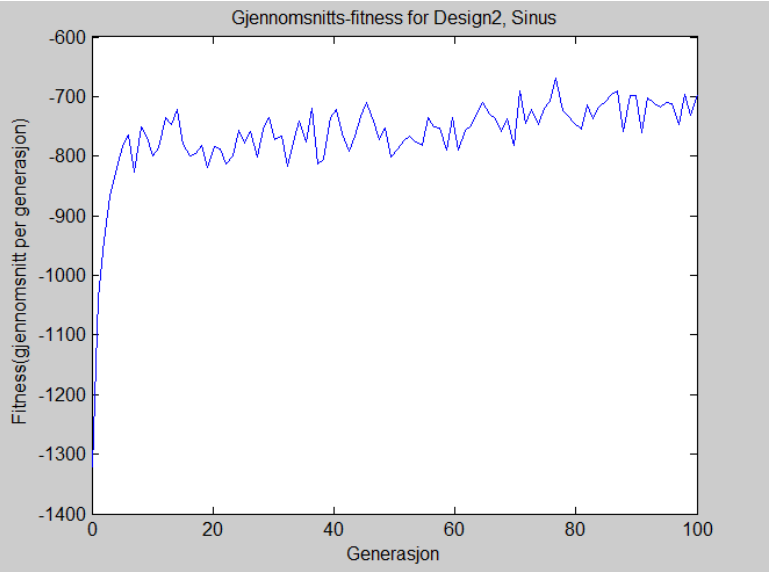

FIGURE C.4

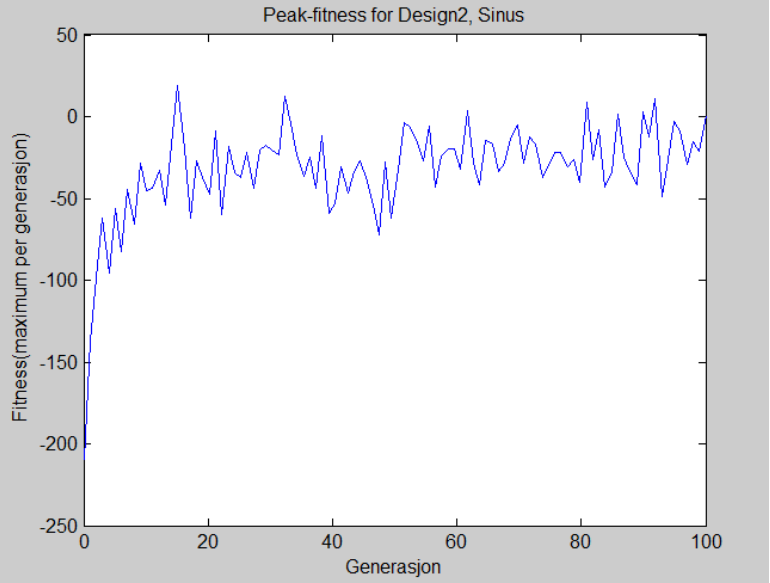

FIGURE C.5

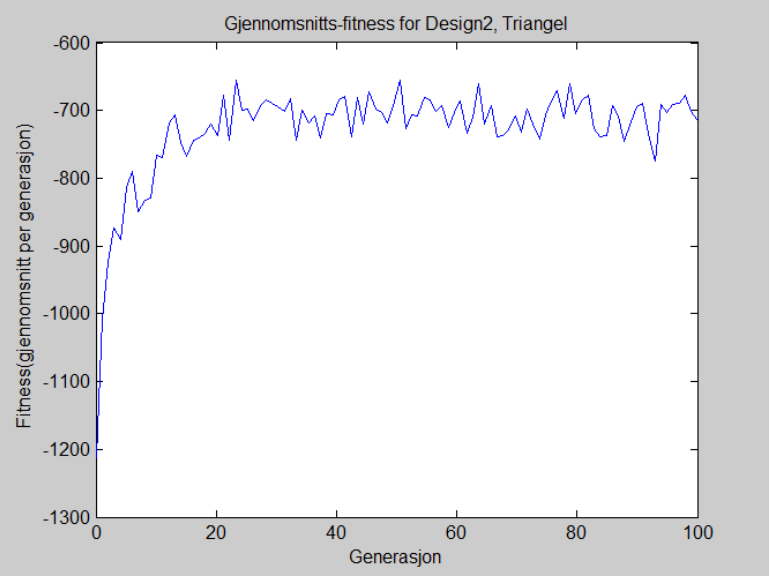

FIGURE C.6

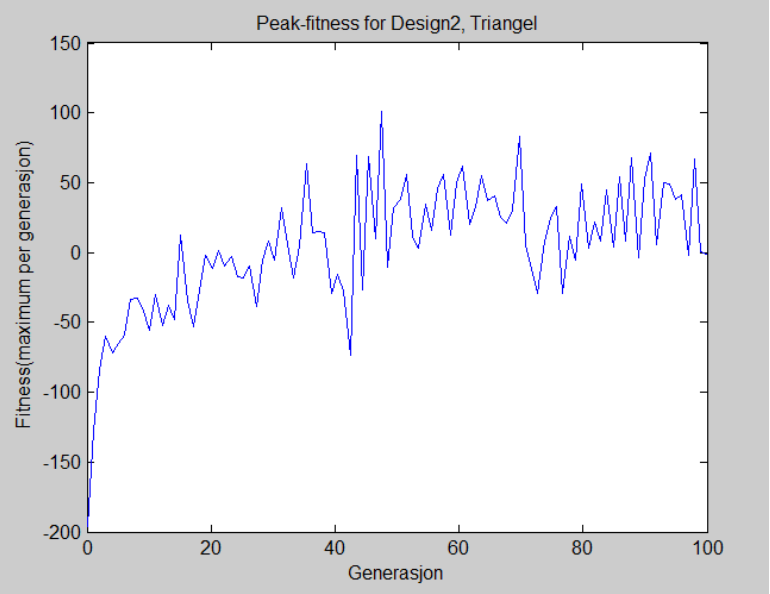

FIGURE C.7

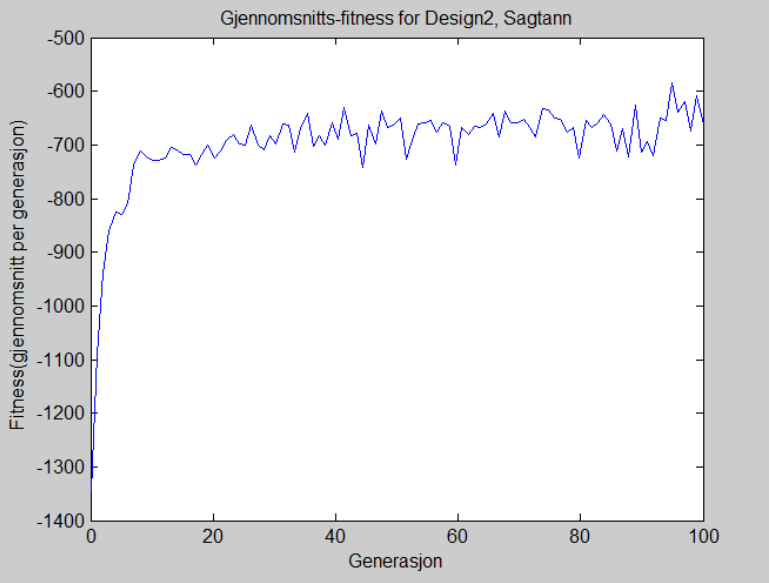

FIGURE C.8

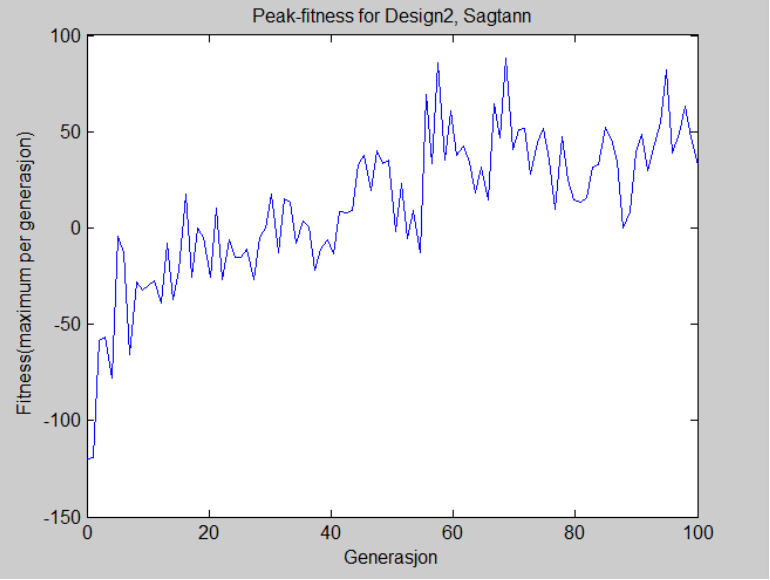

FIGURE C.9

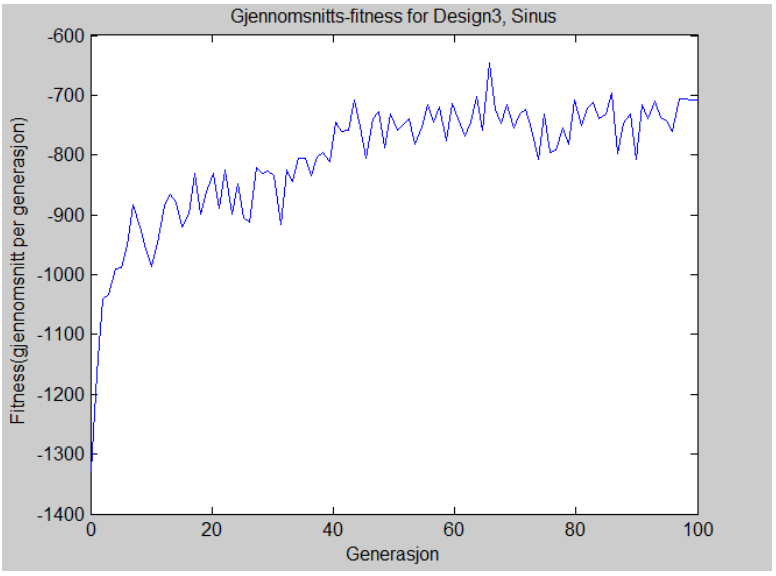

FIGURE C.10

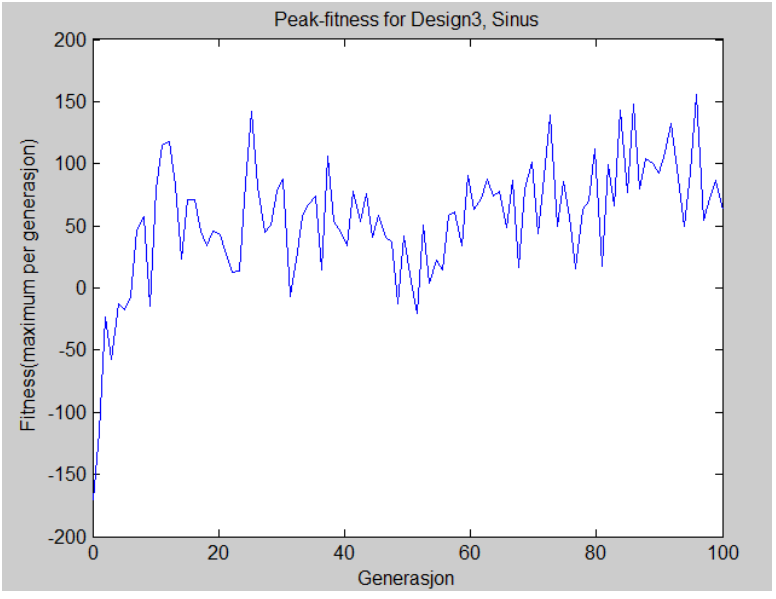

FIGURE C.11

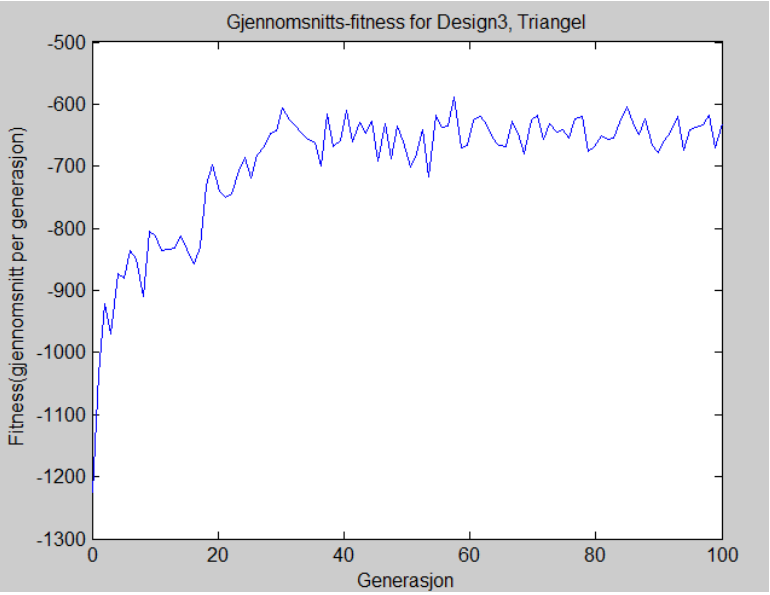

FIGURE C.12

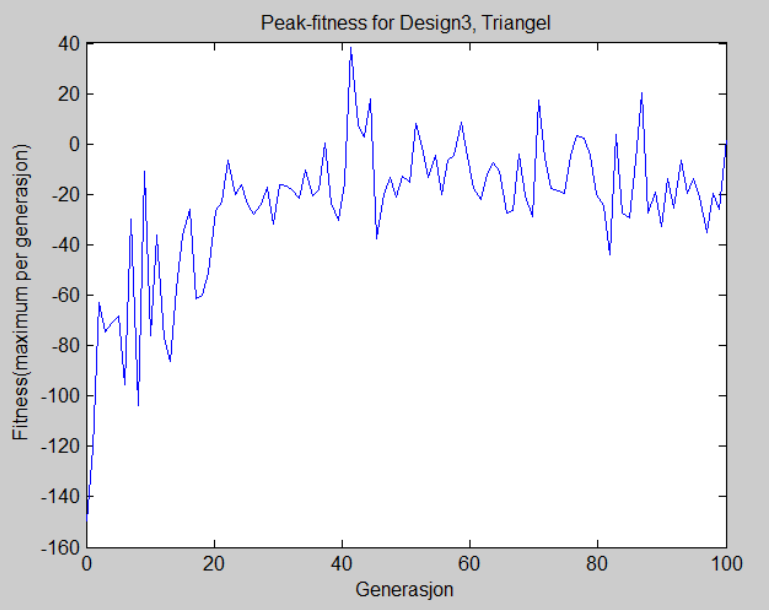

FIGURE C.13

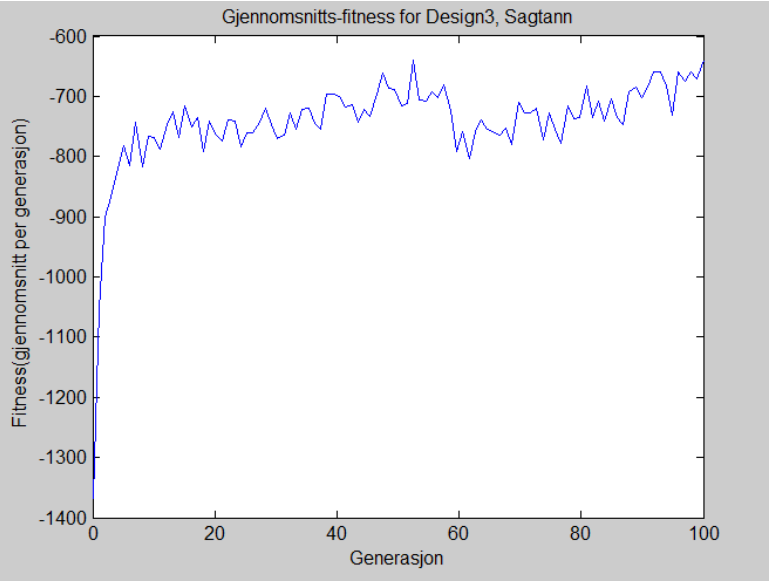

FIGURE C.14

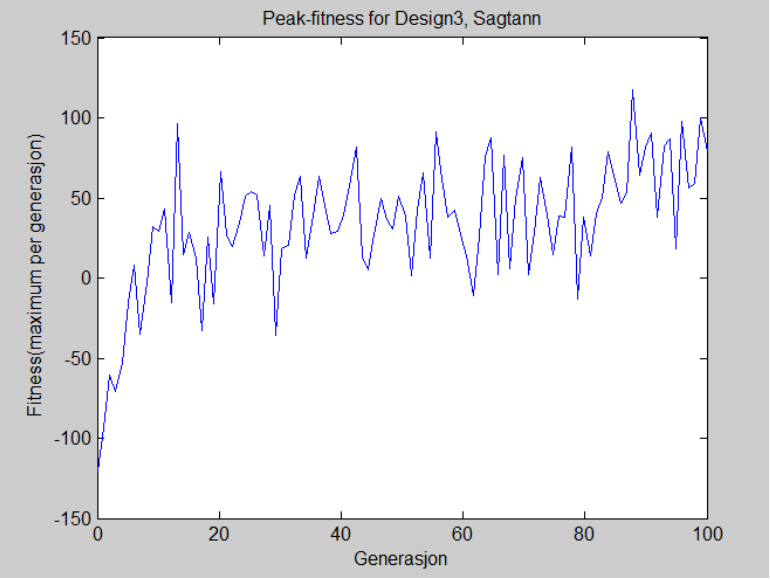

FIGURE C.15

Bibliography

[1] Nascence prosjektside. http://www.nascence.eu/.

[2] Sevall Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. 1932.

[3] Gordon Pask. Physical analogues to the growth of a concept. 1958.

[4] Adrian Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. 1996.

[5] Paul Layzell. A new research tool for intrinsic hardware evolution. 1998.

[6] Stefano Nichele. Dragana Laketić. Odd Rune Lykkebø. Gunnar Tufte. Is there chaos in blobs of carbon nanotubes used to perform computation? *Future Computing*, 2015.

[7] Illustrasjon, vindusfunksjon. https://en.wikipedia.org/wiki/Window_function#/media/File:Spectral_leakage_from_a_sinusoid_and_rectangular_window.png.

[8] Blackman window, illustrasjon. https://upload.wikimedia.org/wikipedia/commons/3/38/Window_function_and_frequency_response_-_Blackman.svg.

[9] Hamming window, illustrasjon. https://upload.wikimedia.org/wikipedia/commons/7/76/Window_function_and_frequency_response_-_Hamming_%28alpha_%3D_0.53836%29.svg.

[10] Alan Turing. On computatable numbers, with an application to the entscheidungsproblem. 1936.

[11] Boaz Barak Sanjeev Arora. *Computational Complexity: A Modern Approach*. 2009.

[12] John Von Neumann. The first draft report on edvac. 1945.

[13] Hadi Esmaeilzadeh. Emily Blem. Renée Amant. Karthikeyan Sankaralingam. Doug Burger. Dark silicon and the end of multicore scaling.

[14] Shekhar Borkar. Andrew A. Chien. The future of microprocessors.

[15] Cyril Crassin. Fabrice Neyret. Miguel Sainz. Simon Green. Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. 2011.

[16] Francis Heylighen. The science of selforganization and adaptivity.

[17] Julian F. Miller. Simon L. Harding. Gunnar Tufte. Evolution-in-materio: evolving computation in materials. 2014.

[18] Francesco Petruccione Heinz-Peter Breuer. *The Theory of Open Quantum Systems*. 2002.

[19] What is the genetic algorithm? http://se.mathworks.com/help/gads/what-is-the-genetic-algorithm.html.

[20] Odd Rune Lykkebø. Gunnar Tufte. Comparison and evaluation of signal representations for a carbon nanotube computational device. 2014.

[21] Michael Conrad. The price of programmability. 1988.

[22] Warren Weaver. Science and complexity. 1948.

[23] Arneud Le Ny. Introduction to (generalized) gibbs measures.

[24] Daniel Levinthal. Adaptation on rugged landscapes. 1997.

[25] Simon Levin Stuart Kauffman. Towards a general theory of adaptive walks on rugged landscapes. 1987.

[26] Edvard Weinberger Stuart Kauffman. The nk model of rugged fitness landscapes and its application to maturation of the immune response. 1989.

[27] Lionel Barnett. Tangled webs, evolutionary dynamics on fitness landscapes with neutrality. 1997.

[28] Chris G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. 1990.

[29] Tom Holme Larry Brown. *Chemistry for Engineering Students, 2nd International Edition*. 2011.

[30] John Hennesey. David Patterson. *Computer Architecture, A Quantitative Approach ed. 5*.

[31] Odd Rune Lykkebø. Design and implementation of a prototype platform for evolution in materio. 2010.

[32] Github repository for mecobo. https://github.com/NASCENCE/mecobo/tree/v4.0.

[33] Jouko Vankka. *Digital Synthesizers and Transmitters for Software Radio*. 2005.

[34] Jerry Gibson. Toby Berger. Tom Lookabaugh. Dave Lindbergh. Richard Baker. *Digital compression for multimedia, principles and standards*. 1998.

[35] V.F. Koupa. *Direct Digital Frequency Synthesizers*. 1999.

[36] Gunnar Løvås. Statistikk for universiteter og høyskoler utg 2. 2004.

[37] Apache thrift. https://thrift.apache.org/.

[38] D. G. Karczub M. P. Norton. *Fundamentals of Noise and Vibration Analysis for Engineers*.

[39] G. Heinzel. A. Rudiger and R. Schilling. Spectrum and spectral density estimation by the discrete fourier transform including a comprehensive list of window functions and some new at-top windows. 2002.

[40] Arthur Lohwater. *Introduction to inequalities*. 1982.