NTNU
Det skapende universitet

Oppgradering av SiWa-Scan

Jim Kristian Malmhaug

Master i teknisk kybernetikk (2-årig)
Innlevert: januar 2014
Hovedveileder: Tor Engebret Onshus, ITK
Medveileder: Bendik Sægrov-Sorte, SINTEF Materialer og kjemi

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Oppgradering av SiWa-Scan

Jim-Kristian Malmhaug

12. januar 2014

Oppgave

SiWa-Scan er et instrument som finner dislokasjoner (krystallfeil) i prøver fra silisiumwafere. Å kunne vite hvor mye dislokasjoner det er i en silisiumwafer, kan være med på å bestemme kvaliteten på waferen. Instrumentet er en prototype som bygger på en idé av Gaute Stokkan ved SINTEF Materialer og kjemi.

Fra tidligere prosjekt er det avdekket at instrumentet inneholder uønskede refleksjoner fra kamera, og det er bestilt inn nye komponenter som skal testes ut for å fjerne disse.

Det skal også undersøkes om SINTEFs algoritme for gjenkjenning av tvillinger og korngrenser kan modifiseres for å få til en mer treffsikker algoritme, eller finne en bedre løsning for gjenkjenning.

Kameraet kan med full oppløsning ta 58 bilder i sekundet (FPS), men med nåværende program klarer man ikke mer enn 25 FPS. Dette er en flaskehals for systemet som fører til økt analysetid, og man ønsker å finne en løsning på dette. SINTEF har også fått tips fra National Instrument om å bruke parallellkjøring av operasjoner i SiWa-Scan-programmet for å redusere kjøretiden, og man ønsker å få testet ut dette.

Forord

Denne rapporten er resultat av min masteroppgave som ble avlagt ved Norges teknisk-naturvitenskapelige universitet, institutt for teknisk kybernetikk, høsten 2013. Masteroppgaven bygger videre på tidligere prosjekter som jeg har deltat i. Rapportene fra disse ligger som vedlegg 1 og 2.

Dette prosjektet har vært lærerikt og en meget god kilde til å skaffe seg mer kunnskap om programmering i LabVIEW og C, og lære seg å skrive i LaTeX.

Jeg vil takke SINTEF Materialer og kjemi for å ha fått lov til å ta masteroppgaven på deres prototype SiWa-Scan. Jeg vil samtidig takke veileder Bendik Sægrov-Sorte ved samme bedrift for mange gode tips og god hjelp i LabVIEW-programmeringen. Jeg vil også takke Tor Onshus ved institutt for teknisk kybernetikk for gode råd og veiledning av oppgaven. Til slutt vil jeg takke min samboer Line for god hjelp til korrekturlesing av rapporten.

Jim-Kristian Malmhaug

Trondheim, 12. januar 2014

Sammendrag og konklusjon

Kapittel 1 gir en kort introduksjon til fenomenene dislokajoner, tvillinger og korngrenser. Kapitlet gir også en kort beskrivelse av tidligere prosjekter på SiWa-Scan. I kapittel 2 gjennomgås jobben med å oppgradere optikken på instrumentet, og målinger gjort med referansesensoren. Det tar også for seg støy som oppstår i målingene og hvordan man teller dislokasjoner i SiWa-Scan. Resultatet av dette var at man fikk til en bedre måling fra referansesensoren, og man kom frem til en ny korreksjonfaktor for telling av dislokasjoner. I kapittel 3 gjennomgås teori for digitalkamera og kameraets parametre. Ved uttesting og endring av lukketid kom man frem til at kameraet kunne økes fra 25 FPS til 40 FPS. Dette førte til en kortere analysetid for SiWa-Scan. Fjerde kapittel viser hvordan brukergrensesnittet er laget i SiWa-Scan. Brukergrensesnitt-versjonen som tilhører denne rapporten er en enklere og mer strukturert versjon enn den tidligere. Kapittel 5 går gjennom oppbyggingen av nytt LabVIEW-program for SiWa-Scan, og innføringen av FIFO-kø. Ved hjelp av FIFO-køen gjorde man analysen av bilder uavhengig fra bildetakingen. Kapittel 6 viser hvordan bilder av dislokajoner og tvillinger blir i SiWa-Scan, og hva man må tenke på når man skal lage en kode for strekgjenkjenning i dette instrumentet. Det sier også litt om en påbegynt C-kode, og den resulterende LabVIEW-koden for gjenkjenning av tvillinger. Kapittel 7 inneholder fire forskjellige tester gjort på SiWa-Scan. Kapittel 8 og 9 diskuterer og konkluderer oppgaven.

Abstract

Chapter one gives an short introduction to dislocations, twins and grain boundaries. It also gives an overview of earlier projects on SiWa-Scan. Chapter two goes through the optical upgrade, and measurements done with the referrence sensor. The chapter also shows the noise in SiWa-Scan, and how dislocations are counted. The result from this job is that the author managed to get an better measuring from the sensor than earlier. In the next chapter, the theory for camera is presented. It also goes through the camera parameters. The camera's FPS was set to 40 by adjusting the time for the shutter. This lowered the analyse time on SiWa-Scan. Chapter four shows the user interface. This interface is more simple and structured than earlier versions. Chapter five goes through the new SiWa-Scan program, and the implementation of a FIFO queue. This queue made the picture analyse independent from the camera acquisition. Chapter six shows how pictures of dislocations and twins are seen in SiWa-Scan, and what to have in mind before designing a recognition algorithm. A C code, which not is finished, is also presented here. In the end of the chapter, the final LabVIEW code for recognition is presented. Chapter seven shows four tests done on SiWa-Scan, and chapter eight and nine discuss and concludes the thesis.

Innhold

1 Innledning **1**
- 1.1 Dislokasjoner, tvillinger og korngrenser . . . 1
- 1.2 Tidligere arbeid . . . 2
  - 1.2.1 Bacheloroppgave, våren 2011 . . . 2
  - 1.2.2 Sommerjobb 2011, og SINTEF-oppgraderinger . . . 5
  - 1.2.3 Prosjektoppgave TTK4551, våren 2013 . . . 5

2 Oppgradering av optikk **6**
- 2.1 Nye optiske komponenter . . . 6
- 2.2 Bedre måling fra referansesensor . . . 10
- 2.3 Støy i bildet . . . 13
- 2.4 Telling av dislokasjoner . . . 16

3 Kamera **19**
- 3.1 Teori . . . 19
- 3.2 Kameraparametre . . . 21
  - 3.2.1 Innsamlingsegenskaper . . . 22
  - 3.2.2 Kontroller . . . 22
  - 3.2.3 Kameraegenskaper . . . 24
  - 3.2.4 Status- og kamerainformasjon . . . 25
- 3.3 Endring av lukketid . . . 25

4 Brukergrensesnitt **27**
- 4.1 Waferinspeksjon . . . 27
- 4.2 Innstillinger . . . 28
- 4.3 Lasersensor . . . 28
- 4.4 Bildekalibrering . . . 29

5 Parallelle operasjoner og LabVIEW-programmets virkemåte **30**
- 5.1 Parallelle sløyfer . . . 30
  - 5.1.1 Ikke-kritiske oppgaver (Sløyfe 1) . . . 33

5.1.2 XY-bord (Sløyfe 2) 33
5.1.3 Innsamlingsløkke (Sløyfe 3) 33
5.1.4 Analyseløkke (Sløyfe 4) 35
5.1.5 Lasersensor (Sløyfe 5) 35

6 Gjenkjenning av tvillinger og korngrenser 36
6.1 Bildemønster i SiWa-Scan 36
6.2 C-kode 41
6.3 LabVIEW-kode 41

7 Tester 44
7.1 Programsammenligning 46
7.2 Dislokajonstelling med og uten støykorreksjon 49
7.3 Kjøretid, kamerasløyfe og analysesløyfe 51
7.4 Oppløsning 53

8 Diskusjon 56

9 Konklusjon 59

10 Videre arbeid 61

Bibliografi 63

Figurliste 65

A Vedleggsoversikt 66

Kapittel 1

Innledning

1.1 Dislokasjoner, tvillinger og korngrenser

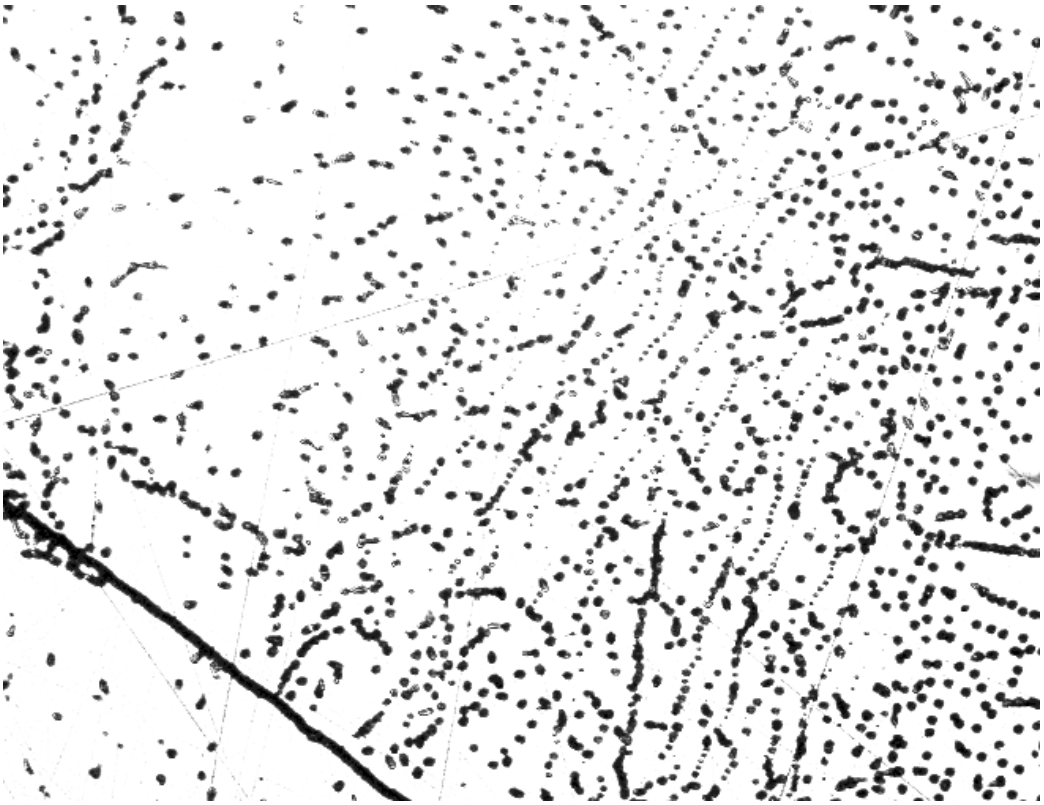
I solcelleindustrien blir halvmetallet silisium brukt mye som en komponent i solceller. Grunnen er at silisium er lett å fremskaffe via god prosessteknologi og at det finnes rike forekomster av metallet. Ofte produseres silisiumet i enheter kalt ingotter. Disse blir som oftest kuttet opp i wafere, som er tynne runde skiver av materialet. Waferene kan også bli videreforedlet og brukt i for eksempel integrerte kretser.[3]

Under prosesseringen av silisiumet blir materialet utsatt for en høy temperatur som gjør stoffet flytende, før det så trekkes ut av en silisiumovn med en stav og kjølt ned til de faste ingottene. Denne prosessen må overvåkes nøye ettersom den har mye å si for hvordan kvaliteten på ingottene blir. Hvis stukturen til silisiumingottene blir utsatt av for mye stress under prosessen, kan det dannes krystalldefekter som dislokasjoner. Det finnes to typer dislokasjoner, alt etter hvordan materialet har vridd seg. Hovedtypene er kantdislokasjoner og skrudislokasjoner. Dislokasjonene kan sees som «hull» i materialet.[1, side. 5-11]

Tvillinggrenser er en struktur som oppstår i materialet der krystaller av samme type gror inn i hverandre, og gir «misorientation» i materialet.[4] Korngrenser er defekter som fører til svekket elektrisk egenskap, og til at styrken i materialet blir dårligere. Det vil si at materialet kan bli mer porøst. Dislokasjoner, tvillinger og korngrenser vil være «bi-effekter» i materialet som dannes under støypingen av silisiumingottene.

SINTEF Materialer og kjemi tilbyr kompetanse på å karakterisere wafere.[5]

For å finne ut noe om hvor god kvalitet en wafer eller ingott har, vil det være interessant for SINTEF å se på hvor mye dislokasjoner som befinner seg på en waferprøve. Dislokasjoner fører til at materialet får en svekket elektrisk egenskap, og derfor er det ønskelig å ha wafere med så lav dislokasjonstetthet som mulig. En wafer med lav dislokasjonstetthet er som oftest en wafer med god kvalitet.



Figur 1.1: Dislokasjoner oppstår som prikker i en silisiumwafer. Sett ifra mikroskop.

1.2 Tidligere arbeid

1.2.1 Bacheloroppgave, våren 2011

Våren 2011 ble det gitt en bacheloroppgave til undertegnede og fire andre HiST-studenter om å utvikle et instrumet for SINTEF og NTNU. Instrumentet skulle kunne brukes til å lete opp og telle krystallfeil i silisium av

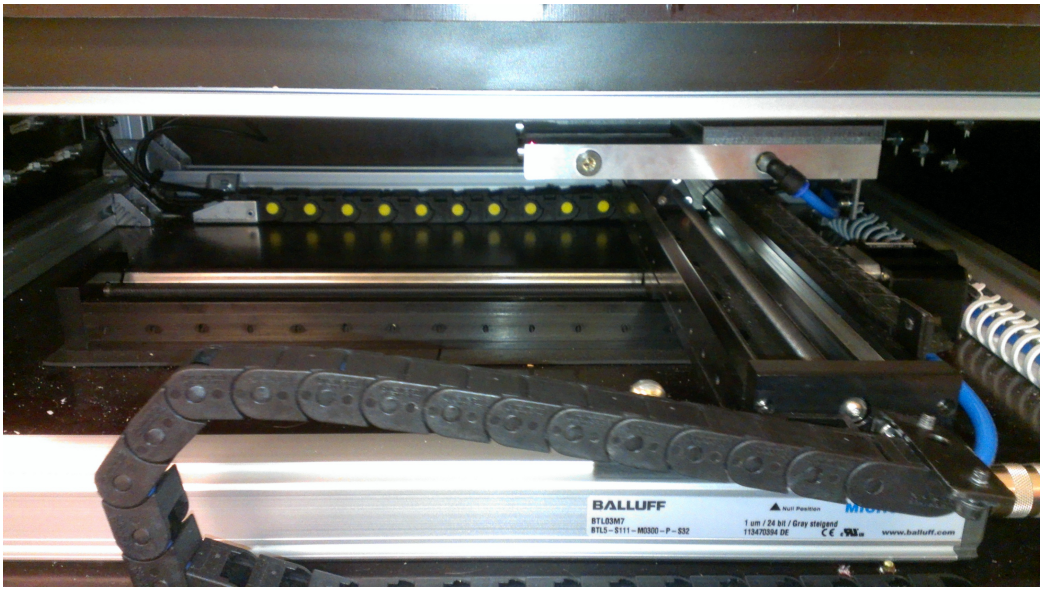
typen dislokasjoner. Denne oppgaven gikk under et større prosjekt ved navn *Defect Engineering for Crystalline Silicon Solar Cells*. Det nye instrumentet ble kalt SiWa-Scan og besto av en kasse som i hovedsak inneholdt optiske komponenter, et kamera, en laser og et XY-bord med fire frihetsgrader. Disse komponentene ble satt samme slik at det kunne lyse opp en waferprøve med laseren, og fange opp refleksjonen via de optiske komponentene og føre det til kameraet. Deretter ble lyssignalet gjort om til elektriske signaler og sendt videre via FireWire for analyse i et brukergrensesnitt. Brukergrensesnittet ble satt opp i LabVIEW. SINTEFs ønske var å få utviklet et slikt instrument for å kunne karakterisere silisiumwafere bedre. Før SiWa-Scan ble utviklet brukte de et annet instrument for gjenkjenning og telling av dislokasjoner. Dette instrumentet heter PVScan.

PVScan analyserte i hovedsak wafere via bruk av laserlys, en integrerende kule og lasersensorer. PVScan fanget opp spredningen av laserlyset og kunne via dette telle hvor mange dislokasjoner som befant seg i et bestemt område. En ulempe med dette instrumentet var at den fanget opp krystallformasjonene tvillinger og korngrenser, og førte til en feiltelling av hvor mange dislokasjoner som befant seg i en prøve.

Med det nye instrumentet ønsket man å kunne utelukke tvillinger og korngrenser fra tellingen av dislokasjoner. Instrumentet ble utviklet etter de krav og spesifikasjoner gitt av SINTEF og NTNU. Kostnadsrammen for utviklingen lå på 200 000 NOK, og ble ikke overskredet. Det nye instrumentet ble ferdigstilt 25.5.2011, og kunne telle dislokasjoner på ca lik linje med PVScan.

Under bacheloroppgaven viste det seg å være utfordrene å få til en algoritme som utelukket tvillinger og korngrenser fra tellingen av dislokjoner. Denne oppgaven ble ikke løst, og det ble anbefallt å sette av mere tid til dette i neste fase av SiWa-Scan-prosjektet.[6, side. 90]

Figur 1.2: SiWa-Scan med tilhørende PC og brukergrensesnitt.



Figur 1.3: XY-bordet som brukes til å flytte waferprøven rundt i SiWa-Scan.

1.2.2 Sommerjobb 2011, og SINTEF-oppgraderinger

Undertegnede hadde i forbindelse med en sommerjobb for SINTEF sommeren 2011 i oppgave å forbedre SiWa-Scan. Under denne jobben ble en uønsket refleksjon fjernet, og det ble installert filtre for å redusere intensiteten som kom fra waferoverflaten. Refleksjonen kom fra noen komponenter kalt «slits», og disse ble fjernet. Det ble også avklart at posisjonsmålingen som kan hentes ut fra XY-bordet var for treg til å kunne brukes i en forrigling mellom posisjon og bildetakingen til kamereat.

SINTEF har i ettertid av sommerjobben installert en ekstern posisjonssensor og lagt inn en algoritme for strekgjenkjenning. Et annet mål som SINTEF ønsket å bruke possisjonsensoren til, var å fjerne et sikk-sakk mønster som dannet seg i dislokasjonskartet under skanning av wafer. Oppgaven ble ikke ferdigstilt på daværende tidspunkt. (Sikk-sakk mønsteret ble fjernet i dette prosjektet, høsten 2013.)

1.2.3 Prosjektoppgave TTK4551, våren 2013

Våren 2013 fikk undertegnede i oppgave å vidreføre en ny fase i utviklingen av SiWa-Scan gjennom faget TTK4551, gitt av NTNU og SINTEF Materialer og kjemi.

Under dette prosjektet kom det frem at man ikke kunne bruke intensitet som eneste kriterie for å skille tvillinger og korngrenser fra tellingen av dislokasjoner. Årsaken var at intensiteten i bilder med dislokasjoner var sammenfallende med intensiteten i bilder av tvillinger/korngrenser.

Det ble også avdekket at instrumentet inneholdt en uønsket refleksjon fra kamereaet som forstyrret referansesensoren. Dette førte til at man ikke kunne bruke refereansesensoren til å korrigere for svingninger i laserintensiteten. Thorlabs kom med forslag til å prøve ut nye optiske komponenter for å se om man ble kvitt disse refleksjonene.

I dette prosjektet ble det ikke utviklet en ny algoritme som var bedre en den SINTEF hadde laget. Dermed ble denne oppgaven stående åpen.

Det ble gjort målinger av hastigheten til kameraets FPS, og sporet opp at hastigheten ikke skyldtes noe annet enn kameraet eller kameraets drivere.

Til slutt ble det besluttet at noe måtte gjøres med datastrukturen som samlet inn og analyserte bilder i LabVIEW-programmet.

Kapittel 2

Oppgradering av optikk

Dette kapitlet tar for seg fremgangsmåten for å komme frem til en lineær modell for målestøy. Det tar også for seg hvordan man fant fram til en korreksjonsfaktor som kan brukes for å telle opp antall dislokasjoner i bildet.

2.1 Nye optiske komponenter

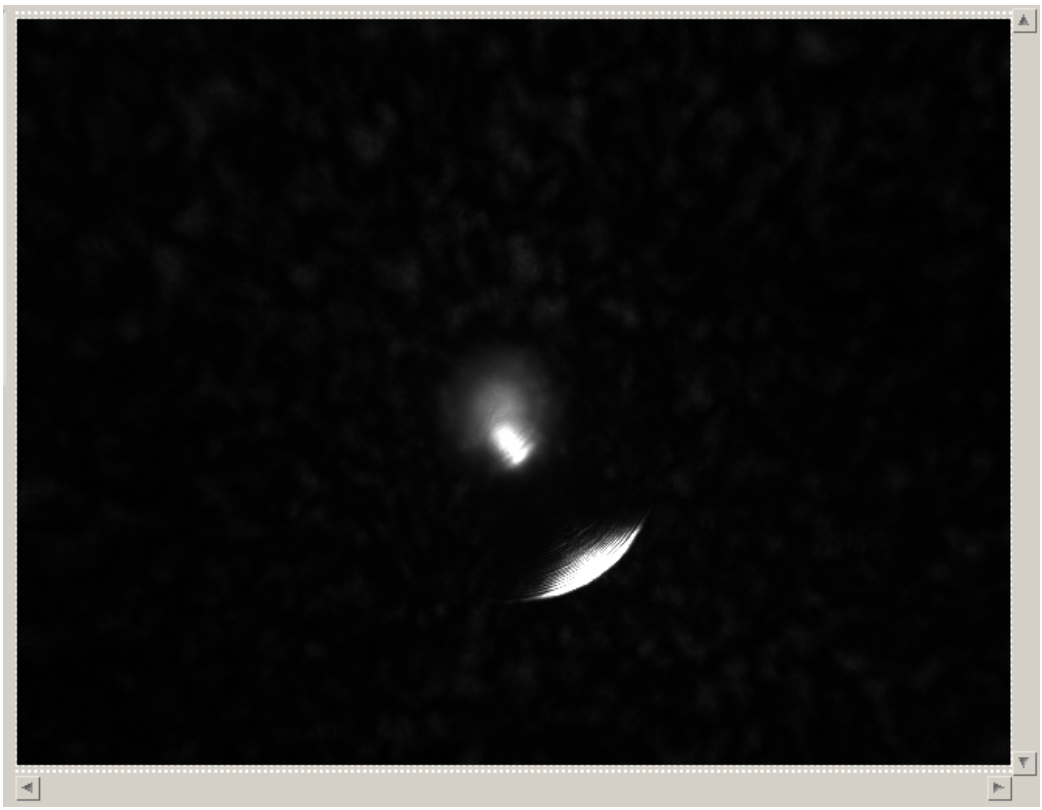
Fra prosjektoppgaven våren 2013 ble det oppdaget at instrumentet inneholdt uønskede refleksjoner som forstyrret signalet til referansesensoren i SiWa-Scan.[1, side. 38-39] Det førte til at man ikke kunne bruke sensoren til å kompensere for svingninger i laserintensiteten. Av denne grunnen ble det kjøpt inn en non-polarized beamsplitter, best-form linse (fokallengde = 40 mm) og en komponent som skråstiller kameraet.

Disse komponentene ble montert og stilt inn slik at det viste skarpest mulig bilde av tvillinger og korngrenser. En svakhet som instrumentet har er at man må demontere alt av optiske komponenter for å kunne montere beamsplitteren. Dette gjør det tungvint å teste ut ulike posisjoner av beamsplitteren.

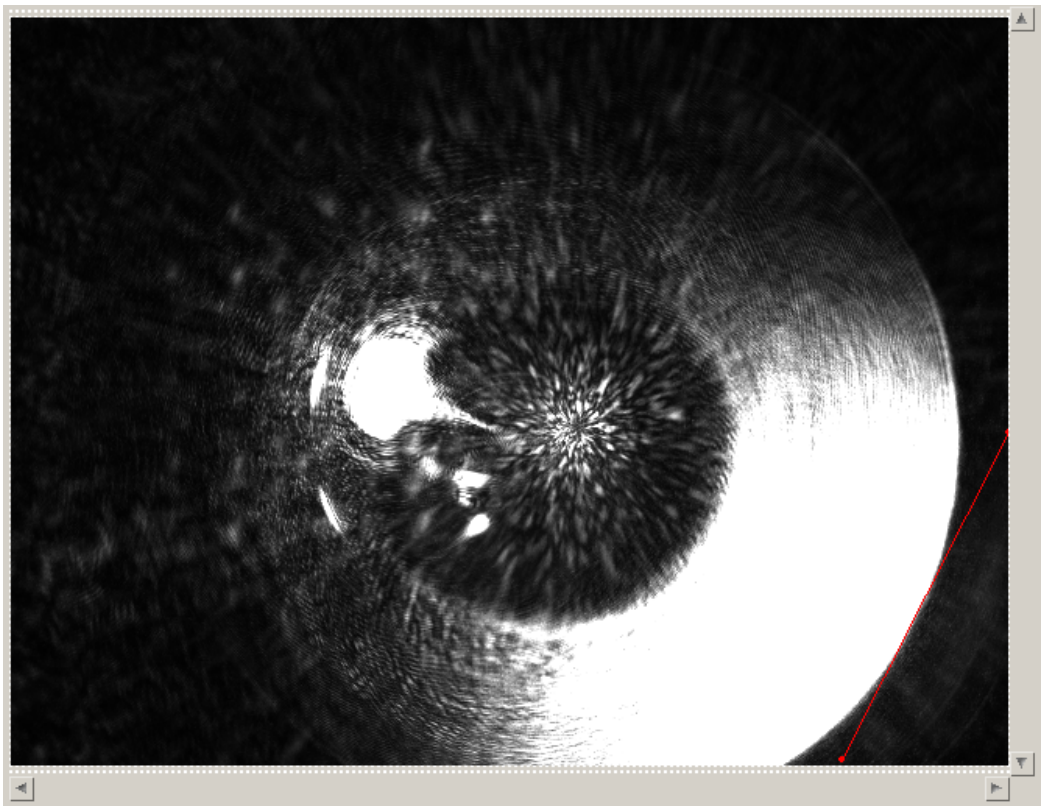
Fra før av har instrumentet et «blindpunkt» som skal sperre for en intens lysstråle som går i midten av linsene. Blindpunktet er montert i midten av en glassplate som er montert over beamsplitteren, og er laget av hvit modelleire. Ettersom denne leiren var hvit og hadde liten størrelse, trodde man at denne kunne gi uønskede refleksjoner både på grunn av størrelse og farge. Derfor ble den byttet ut med en sortfarget leire som hadde større form. Denne ble formet som en kule.

Resultatet ble at bildet ble forverret med den nye linsen som ble kjøpt inn. Figur 2.1 viser hvordan kamerabildet så ut med linsen som sto i fra før, mens figur 2.2 viser kamerabildet etter den nye linsen ble satt i.

Fra figurene ser man at bildene med den nye linsen blir mer opplyst av en «lysring». En slik ring fører til at et større del av bildet blir ubrukelig. Det ble derfor valgt å la den gamle linsen stå i. Man har under testingen lært at ringen kommer fra den nederste linsen som er nærmest wafer-prøven, noe man tidligere ikke viste.

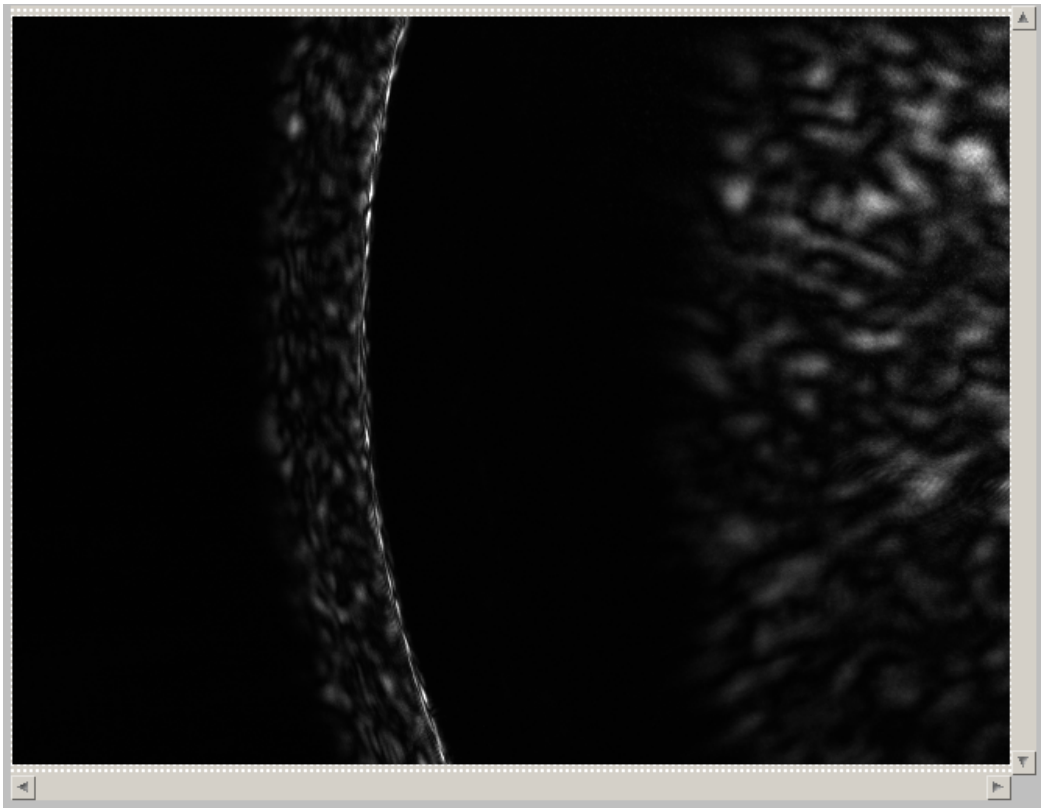


Figur 2.1: Kamerabilde av mørk bakgrunn med gammel linse.



Figur 2.2: Kamerabilde av mørk bakgrunn med ny linse.

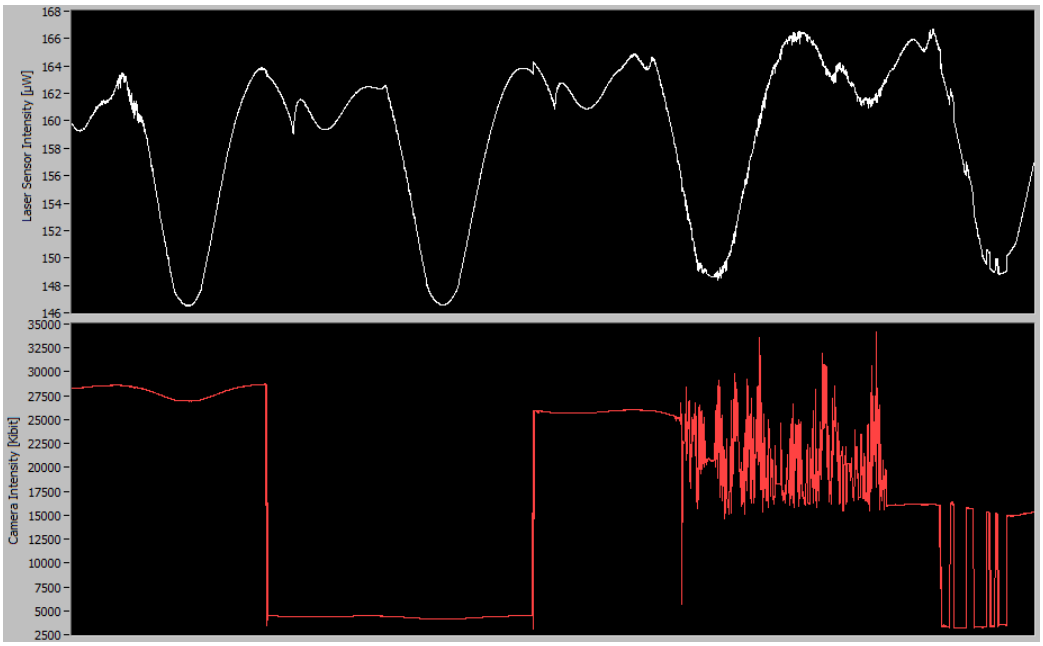
Et annet forslag som ble anbefalt av SINTEF å prøve etter prosjektoppgaven i vår 2013[1, side. 38], var å skråstille kameraet, slik at de reflekterende strålene fra kameraet ikke ble sendt ned tilbake til referansesensoren, men rett i røret som holder på plass linsene. Etter å ha montert komponenten som skråstiller kameraet 10 grader, så man at kameraet bare så kanten av bildet, slik figur 2.3 viser. Siden dette fører til tap av infomasjon, ble det derfor valgte å ikke gå videre med denne løsningen.



Figur 2.3: Kameraet ser bare kanten av bildet når det er skråstilt 10 grader.

2.2 Bedre måling fra referansesensor

Etter å ha valgt å gå videre med den nye beamsplitteren og nytt «blindpunkt» i den optiske banen, ble referansesensoren testet opp mot kameraet for å se om de uønskede refleksjonene ennå var tilstede. Til det ble det kjørt en sprangtest på systemet, der man målte signalet som kom fra et område med tilfeldig antall dislokasjoner, og med mørk bakgrunn (sort papir). Figur 2.4 viser at de uønskede refleskjonene ikke gir store utslag og forstyrrer referansensoren som før. Dette er en klar forbedring fra hva man har målt tidligere. [1, side. 26]

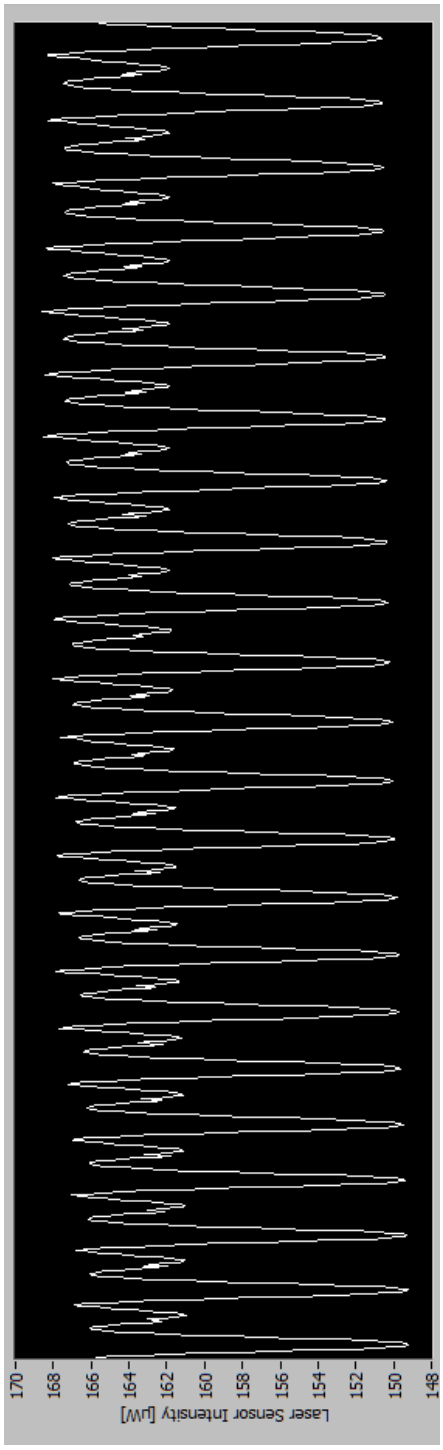


Figur 2.4: Bedre måling fra referansesensor under forstyrrelse.

Den nederste grafen i figuren viser signalet som kameraet fanger opp (sum av intensitet), og den øverste viser referansesensoren. Man ser også at laserintensiteten svinger i en syklus med tre topper og en bunn. Den verste forsyrrelsen sees i den nederste grafen der det er mye «spiker» og «firkantpulser», i siste halvdel av grafen. Alle disse hurtige pulsene er forårsaket ved at man drar et sort papir frem og tilbake i det området som man henter inn refleksjonene fra på SiWa-Scan. Det vil si før lysstrålene samles opp i den nederste linsen.

Etter at man fikk fjernet mestparten av de uønskede refleksjonene, så man at referansesensoren målte mye mindre svingning i lasersens intensitet. Figur 2.5 viser måling av laserintensiteten like etter at laser er slått på. Den er i dette tidspunktet mest ustabil, og intensiteten svinger mest.

Fra prosjektet i vår 2013 ble det målt at laseren svinger mellom 100 mikrowatt og 700 mikrowatt[1, side. 19], mens den nå ligger mellom ca. 149 mikrowatt og 169 mikrowatt.



Figur 2.5: Måling av ustabil laser etter at de nye optiske komponentene er satt inn.

2.3 Støy i bildet

Med stabil måling av laserintensitet kan man bruke verdien som referansesensoren måler for å kompensere waferanalysen for ustabilt lasersignal. Etter råd fra Tor Onshus må man gjøre to kompenseringer. Den første går ut på å kompensere for den bakgrunnsstøyen som er konstant i bildet, det vil si «lysringen» som nevnt tidligere. Den andre er å kompensere for svingningen i lysintensiteten fra prøver av silisiumwafere. Deretter trekker man fra forstyrrelsen fra totalbildet. For å kunne gjøre det må man finne frem til en kameraverdi som funksjon av referansesensoren.

$$c(r(i)), \quad i = 0...n \tag{2.1}$$

I dette tilfellet er verdien i nummeret på iterasjonen den gjennomgår. Siden systemets signaler ikke er konstant, vil kamera og referansesensor ha ulike verdier av i. Iterasjonene gjøres i while-løkkene i dataprogrammet til SiWaScan.

Det resulterende bildet etter støykorreksjon vil være rådata fra kameraet minus likning (2.1).

$$c_{res}(i) = c_{raw}(i) - c(r(i)), \quad i = 0...n \tag{2.2}$$

For å finne en funksjon for bakgrunnsstøyen ble det brukt et sort papir som testobjekt for å hindre at refleksjoner ble sendt opp gjennom den nederste linsen. For best mulig resultat anbefales det å kjøre XY-bordet helt i venstre ende, og deretter skråstille det sorte papiret i skinnene. Legges det for nært den nederste linsene kan man oppleve å få refleksjoner, selv om papiret er sort og matt. Deretter tok man ut 20 forskjellige målinger av verdier fra referansesensoren og kameraet. Verdiene vises i tabell 2.1.

For å finne kameraverdien som funksjon av referansesensoren, er det brukt regresjonsanalyse. Denne analysen er gjort via programmet STAT på en CASIO CFX-9850GB PLUS kalkulator. Analysen viser at kameraverdien stiger proporsjonalt med verdien til referansesensoren, og beste funksjon for dette ser ut til å være den lineære funksjonen:

$$c_1(r(i)) = ar(i) + b, \quad i = 0...n \tag{2.3}$$

a er stigningstall og b estimert nedre grense for kameraverdien når referansesensoren er 0 µW.

Referansesensor[µW]	Kameraverdi[Kbit]
175,87	6423,59
158,37	6318,59
163,31	6349,65
171,08	6388,36
171,47	6393,54
156,60	6310,30
160,05	6332,23
169,51	6382,95
161,12	6335,86
168,29	6379,92
163,93	6355,47
158,68	6327,38
155,21	6308,15
158,80	6327,18
154,52	6307,73
170,91	6402,91
167,70	6385,40
162,93	6353,96
170,43	6405,96
157,54	6327,73

Tabell 2.1: Måleverdier fra referansesensor og kamera ved måling av støy fra linse.

For å kunne fjerne støy som oppstår fra testwafere, ble det brukt en lignende framgangsmåte som ovenfor. Først sammenlignet man en piksel med referansesensoren for å se om man kunne se noen sammenheng når laserintensiteten svingte. Dette pikselet befant seg i venstresiden av bildet, og var ikke berørt av støyet som oppstår i optikken. Det vil si ikke en piksel som lysringen dekker. Siden det var små variasjoner i det ene pikselet, ble denne testen kjørt rett etter oppstart med mest mulig svingning i laserintensiteten. Testen ble gjort med mørk bakgrunn, og med en wafer for å skaffe refleksjoner til kameraet.

Resultatet ble ikke som forventet. Når referansesensoren målte stigning i laserintensitet kunne pikselet falle i verdi, og omvendt. Dermed kunne man ikke lage en lineær modell basert på denne informasjonen.

I neste omgang ble et større område med piksler på venstresiden i bildet

tatt ut. Dette området var heller ikke berørt av støyen i optikken. Deretter sammenlignet man referansesensoren med gjennomsnittet per piksel i det utvalgte området. Testen ble også her kjørt med mest mulig ustabil laser for å få mest mulig svingninger i målingene, og testresultatet ble mer som forventet enn med metoden beskrevet ovenfor. Når referansesensoren målte stigning i laserintensiteten målte man også økning i gjennomsnittet, og omvendt.

Med denne informasjonen kunne man finne frem til en lineær funksjon for støyen:

$$c_2(r(i)) = dr(i) + e, \quad i = 0...n, e = 0 \tag{2.4}$$

e er satt til 0 ettersom man allerede har funnet frem til verdien b. En ny konstant e vil ikke gi mening, og man er her kun interessert i å vite hvordan svingningene i laserintensitet fra waferoverflate opptrer. Videre antar man at den gjennomsnitlige pikselverdiens svingninger fra waferoverflaten er gjeldende for alle kameraets 640 x 480 piksler.

Konstantene man kom frem til var:

- a = 5,65551697 [Kbits/µW]
- b = 5429,18937 [Kbits]
- d = 17,55959194 [Kbits/µW]

Den lineære modellen for støyen fra linse og waferoverflate vil da bli:

$$c_{res}(i) = c_{raw}(i) - c(r(i)) = c_{raw}(i) - c_1(r(i)) - c_2(r(i)), \quad i = 0...n \tag{2.5}$$

$$c_{res}(i) = c_{raw}(i) - (a + d)r(i) - b, \quad i = 0...n \tag{2.6}$$

2.4 Telling av dislokasjoner

For telling av dislokasjoner, etter at bakgrunnsstøyen er trukket fra, brukes formelen:

$$n_{dislocations}(i) = \begin{cases} \lfloor \frac{c_{res}(i)}{k_d} \rfloor & \text{if } \lfloor \frac{c_{res}(i)}{k_d} \rfloor \geq 0 \\ 0 & \text{if } \lfloor \frac{c_{res}(i)}{k_d} \rfloor \leq 0 \end{cases} \tag{2.7}$$

Med likning (2.7) får man en nedre grense på 0 dislokasjoner i bildet. Minusverdier av dislokasjoner vil ikke gi noen mening for den som skal analysere waferprøven. Man får også ut antall dislokasjoner i heltall.

Ettersom det ble installert nye optiske komponenter og gjort justeringer, måtte man finne frem til en ny korreksjonsfaktor for å telle dislokajoner basert på lysintensiteten i bildet. Kalibreringen ble gjort ut i fra to områder på en testwafer, hvorav det er stor forskjell på antall dislokasjoner i disse områdene. I figur 2.6 og 2.8 kan man se hvordan disse områdene så ut i mikroskop. Figur 2.7 og 2.9 viser hvordan områdene så ut i SiWa-Scan.

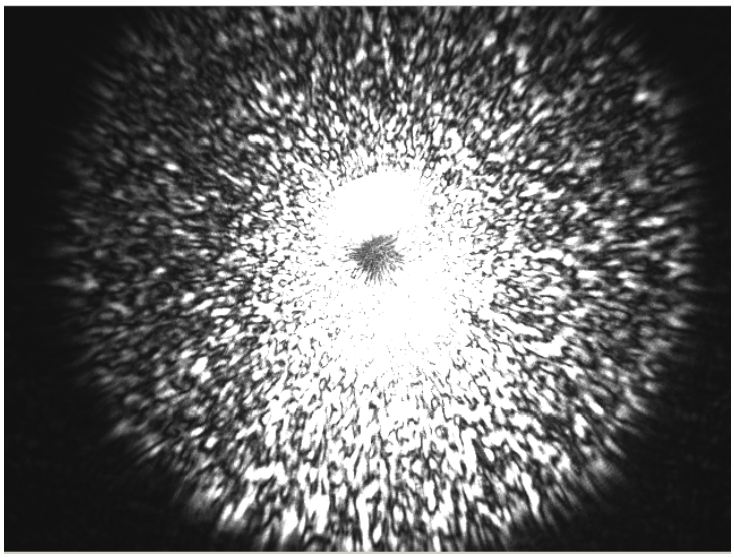
Område 1 og 2 ble estimert ved manuell telling av sorte prikker i bildet. Man antar at laserens nedslagspunkt er sirkulær og at denne vil ha en diameter på 570 µW (radius = 285 µW). For område 1 målte SiWa-Scan en intensitet minus støy på 21422,169[Kbit], og 1407 prikker i mikroskop. For område 2 telte man 45 prikker i mikroskop.

Formelen for å regne ut ny korreksjonsfaktor basert på målinger i område 1 vil være:

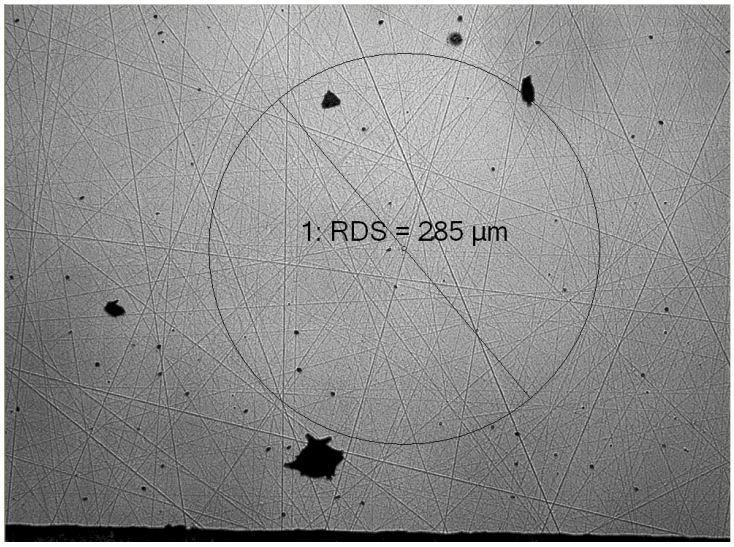
$$k_d = \frac{c_{res}}{n_{mcd}} = \frac{21422,169[Kbit]}{1407[dislocations]} = 15,225[Kbit/dislocations] \tag{2.8}$$

Denne verdien ble så prøvd ut mot område 2 i SiWa-Scan og man fant ut at det ble telt for mange dislokasjoner i området. Dermed prøvde man å korrigere korreksjonsfaktoren til 28,225, og testet det så ut mot område 2. Denne gangen kom man ut med antall dislokasjoner på rundt 80 i området. Etter diskusjon med veileder fra SINTEF kom man frem til at man skulle holde korreksjonsfaktoren på sistnevnte verdi.

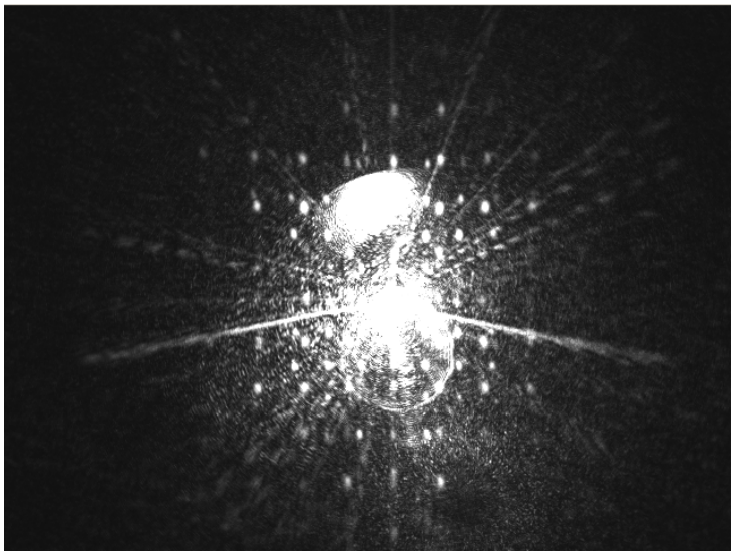
Figur 2.6: Område 1 med mye dislokasjoner sett fra mikroskop.



Figur 2.7: Område 1 med mye dislokasjoner sett fra SiWa-Scan.



Figur 2.8: Område 2 med lite dislokasjoner sett fra mikroskop.



Figur 2.9: Område 2 med lite dislokasjoner sett fra SiWa-Scan.

Kapittel 3

Kamera

Dette kapitlet tar først for seg teorien bak digitalkameraet. Deretter gjennomgås SiWa-Scan-kameraets parametre, og hvordan man ved hjelp av å endre lukketiden på kameraet klarte å øke kameraets hastighet.

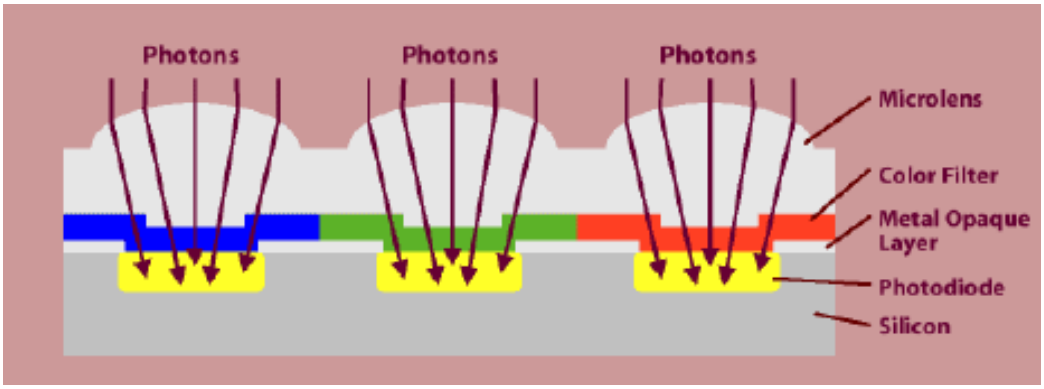
3.1 Teori

Kameraet som brukes i SiWa-Scan er av type GUPPY F-033B med en Sony ICX424-sensor, og er CCD Progressive. Kameraet bruker en Bayer-filter-teknikk for å skape digitale bilder.

Når et digitalkamera tar bilde, sendes det inn lys-fotoner til kamerabrikken. Når fotonene treffer brikken blir de målt ut i fra tre forskjellige farger. Disse er rødt, grønt og blått.

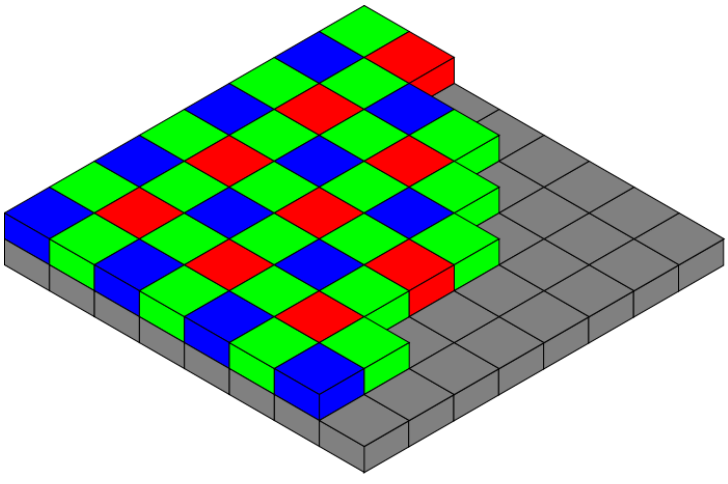
Figur 3.1 viser hvordan filteret kan være bygd opp. Her ser man at fotonene først treffer en mikrolinse som avbøyer og innsnevrer banen til disse, og på denne måten klarer å fange inn lysspektrumet. Deretter går de gjennom fargefilteret. Hvis for eksempel et foton av fargen grønn treffer det blå filteret, vil det bli hindret av filteret i å gå videre. Videre vil de fotonene som slipper gjennom filteret treffe en fotodiode som måler hvor mye fotoner som kommer gjennom. Filteret inneholder i dette tilfellet like mye grønt som rødt og blått tilsammen. Dette fordi det må brukes mer grønt for å danne en farge som øyet oppfatter bedre (true color)[7][9].

Bayer-filteret kan for eksempel være bygd opp slik figur 3.2 viser. Hver rute i denne figuren representerer en «sensel», også kalt pikselsensor. Når mengden



Figur 3.1: Fargefilter, filtrering av fotoner(siliconimaging.com/RGB Bayer.htm).

med lys er målt av hver pikselsensor, kjøres det en «demosaicing»-algortime som finner frem til de riktige fargene i bildets piksel, via interpolering og hvitbalansering. For eksempel bruker de fire sensorene, sett øverst til venstre i figur 3.2, informasjon om mengde lys fra de fire nærliggende pikslene, inkludert seg selv, til å gjennskape den riktige fargen i bildets piksel. Fra RGB-systemet vet vi at man kan gjennskape alle mulige farger bare ved hjelp av fargene rødt, grønt og blått. Hvilke farger man ønsker, kommer ann på mengden av de tre ulike fargene man blander sammen. Andre filtre som også brukes for å finne frem til fargen i et piksel er CYYM, CYGM, RGBW-Bayer og RGBW (#1, #2, #3)[8].



Figur 3.2: Bayerfilter(en.wikipedia.org/wiki/Bayer_filter).

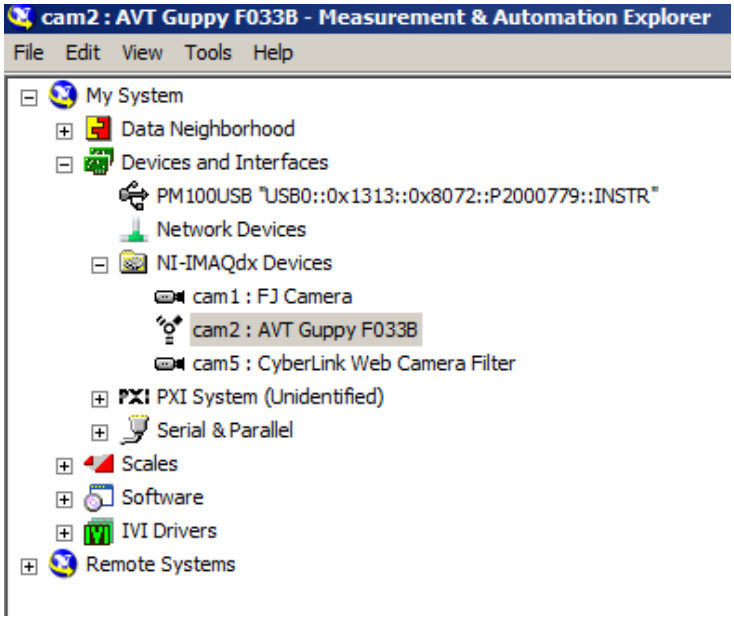
Hver piksel måler fotonene etter mengden 0- til 255-bit, hvor 0 er 0 % og

255 er 100 %. Hvor mange forskjellige farger man kan gjennskape kommer ann på bit-dybden til kameraet. Hvis bit-dybden på kameraet er 8-bit, vil det binære tallet 11111111 (2^8) gi 256 mulige intensiteter av en farge. Bruker man dette videre for å blande 3 forskjellige farger, kan man oppnå $256^3 = 16777216$ farger. Jo høyere bit-dybde man har, jo flere antall farger kan man oppnå[10].

CCD (Charge Coupled Device) er en teknologi som konverterer lys til elektroner[11]. Denne teknologien leser av fargen fra sensorer i en piksel og sender den videre til en ADC (Analog-to-Digital Converter) som gjør om hver piksel til en digital verdi. Med CCD kan man lage bilder med høy kvalitet og lite støy. En annen teknologi er CMOS, men denne teknologien er mer følsom for støy. CCD bruker ca. 100 ganger mer strøm enn CMOS for å lage bilder.

3.2 Kameraparametre

Innstillinger av Guppy-kameraets parametre kan gjøres via National Instruments program «Measuremt and Automation». I dette prosjektet er det versjon 4.7.7f0 som er brukt. For å komme inn til kamera og parametre kan man følge figur 3.3.



Figur 3.3: Veien til parametrene.

Når man åpner mappen «Devices and Interfaces» kan det ta litt tid før pcen finner mappen «NI-IMAQdx Devices», men når den er funnet, velger man kameraet AVT Guppy F033B som er SiWa-Scan sitt kamera. Når kameraet er valgt får man opp et vindu på høyresiden med oversikt over alle parametre.

3.2.1 Innsamlingsegenskaper

Parameteren i denne kategorien tar for seg innsamlingsegenskaper til kameraet, som i hovedsak endrer innstillingen på Bayer-filteret til kameraet.

Parametrene «Gain B», «Gain G» og «Gain R», som kan velges under «Bayer», brukes til å forsterke kameraets fargekomponenter i forhold til hverandre. Alle disse er satt til 1.0 i forsterkning.

Parametret «Pattern» brukes til å velge hvordan mønsteret på Bayer-filteret skal være. Slik figur 3.2 viser, starter denne med blå og grønn som repeterende mønster. Parameteret er i dette tilfelle valgt til å være BG, men man kan også velge om man vil stille inn mønsteret til å vise et annet repeterende mønster. Parametret er valgt til «Use hardware value», og las stå slik.

Det neste parametret er «Bits Per Pixel» og lar deg stille inn bit-dypden til pikslene. Denne kan kun settes til 8-bit for dette kameraet.

3.2.2 Kontroller

Ved hjelp av kontroller-kategorien kan man styre og manipulere dataen som sendes ut fra kameraet.

«Desired Stream Channel» og «Stream Cahnnel Mode» er to parametere som er valgt til «0» og «Automatic». Denne har med dataoverføringen av FireWire å gjøre, og las stå slik. Hvis man for eksempel velger en verdi over 15 kan man oppleve å få error-meldinger.

Neste parameter i parameterlisten er en read-only parameter kalt «Frame Intervall» og er konstant satt til 16. Dette er en parameter som skal være til hjelp for å hente ut tiden i millisekund mellom suksesfulle bilder. Dette parametret er utgrået i listen kan ikke endres direkte.

«Height»-parametret setter høyde på bildet og er valgt til 480. Den kan velges mellom 1 til 480, men las stå på sistnevnte ettersom man er interessert i å hente ut så mye av bildet som mulig.

«Ignore First Frame» lar deg ignorere første bildet som tas dersom man av en eller annen grunn skulle trenge det. Den ignorer kun første bildet som taes etter at kameraet har startet å ta bilder, og er valgt til å ikke ignorere første bildet.

Parametrene «Offset x» og «Offset y» lar deg bestemme hvilke piksler som skal være startpunkt. Begge disse las stå på 0.

Parametret «Overwrite Mode» brukes til å bestemme hva som skal gjøres når det interne minnet i kameraet er fullt. Dette parametret kan settes til «Get oldest», «Fail» eller «Get newest». Det er innstilt på sistnevnte.

«Packet Size» er en parameter som ikke kan stilles direkte. Den endrer seg selv etter hvor mye data som skal sendes ut fra kameraet, og det bestemmes igjen av andre parametre, som for eksempel av «Offset x», «Offset y» og «Height». Denne står nå på 2560.

«Payload Size» bestemmes etter rammen på bildet, og er nå konstant innstilt på $640 * 480 = 307200$. Denne endrer seg hvis man endrer på antall piksler som analyseres.

Neste parameter er «Pixel Format». Denne bestemmer hvilket format på bildet som kameraet sender ut skal ha. Siden kameraet er et mono 8-bit kamera er denne satt til «Mono 8». Ingen andre valg kan gjøres i dette parametret.

«Pixel Signedness» kan settes til «signed» eller «unsigned». For et 8-bit word kan en unsigned variabel av typen int ha en verdi fra 0 til $2^8 - 1$, men for en signed vil samme type ha en verdi fra -2^7 til $+2^7 - 1$. Dette parametret er satt til signed.

«Receive Time stamp Mode» er et parameter som gjør det mulig tidsfeste bilder. Denne kan settes til «None», «Tick Count» eller «System Time». Den er satt til førstnevnte.

Parametret «Reserve Dual Pack» brukes for store FireWire pakker. Denne kan velges til av eller på, hvorav den er valgt til førstnevnte.

«Shift Pixel Bits»-parametret setter oppstillingen til 16-bits kamera. Hvis man mottar mest signifikante bit fra oppstilt data, bør man «downshift» piksel-bitene. Denne kan velges av eller på, hvorav den er valgt til på.

I neste parameter «speed» velger man hastigheten i [Mbps] for dataen som skal sendes over FireWire. Denne er satt til maks 400 [Mbps], men kan også velges til de lavere hastighetene 100 og 200.

Neste parameter «Swap Pixel Bytes» brukes hvis kameraet returnerer «little endian»-data. Hvis man ser på det hexadesimale tallet 12AB er dette skrevet som «big endian» fordi AB som er det største tallet er skrevet på slutten. «Little endian» får vi når tallet er skrevet som AB12. Dette parametret er valgt til å være av.

I «Timeout» velger man tiden (millisekund) man ønsker det skal ta før kameraet returnerer feilmelding. Denne er satt til å ta 5000 millisekund.

For å velge videomodus bruker man parametret «Video Mode». Her kan man for «640 x 480 Mono 8»-bilder velge 3.75, 7.5, 15, 30 og 60 FPS. Man kan også velge «Format 7, mode 0». Denne vil ha en bildestørrelse på 656 x 494 piksler og åpner for endringer av utgrået parametre i parameterlisten. Å velge denne fører også til at «Packet Size» blir større. Dette parametret er satt til «640 x 480 Mono 8 60.00 FPS».

For å velge bredden på bildet bruker man parameter «Width». Denne er satt til å være 640.

3.2.3 Kameraegenskaper

Under «Brightness» kan man velge verdier for mode og value. For den førstnevnte kan man velge mellom «realative» og «ignored». I den andre kan man velge en verdi mellom 0 til 255. Disse to er satt til å være relative og verdi lik 16.

For «Gain» kan man velge mellom «relative», «auto» og «ignored mode», og en verdi mellom 0-680 for «value». Disse er satt til relative og 0.

«Gamma» kan velges til en verdi mellom 0 og 1. Mens dens «mode» kan velges mellom «relative», «off» eller «ignored». Den er satt til off.

Neste parameter «Shutter» gir mulighet for å stille lukketiden til kameraet. Her kan «mode» velges til «auto», «relative» eller «ignored». «Value» kan settes til en verdi mellom 1 og 4095. Den har frem til nå vært stilt inn på relative og value lik 2000, men fordi denne har innvirkning på hvor mange bilder kameraet kan ta i sekundet, har value blitt endret til 800. Se kapittel 3.3 for gjennomgang av denne endringen.

«Trigger» bestemmer når kameraet skal aktiveres og man kan sette «trigger activation» til «level high» eller «level low». Den er satt til level high. «Trigger mode» er satt til «off», men kan også settes til «mode 0», «mode 1» og

«mode 15». Settes trigger mode til andre verdier enn off vil man få feilmelding. Trigger parameter kan settes til en verdi mellom 0 og 4095, hvorav den er satt til 0.

«Mode» i «Trigger Delay» er satt til «off», men kan også settes til «absolute», «relative» eller «ignored». «Value» for dette parametret kan også velges til en verdi mellom 0 og 4095.

3.2.4 Status- og kamerainformasjon

I parameterlisten kan man finne informasjon om hvilken «base adress» kameraet har. Man kan lese av hvilken bustype som brukes, og man kan hente ut levrandørinformasjon, serienummer og modellnavn.

Listen gir også oversikt over hvor mange bufre som er brukt, og gir beskjed om noen av bufrene er gått tapt. Man kan også se om kameraet er aktivert. Bufrene lagres i en FIFO-kø (First In First Out) som sender ut bildene fra kameraet så snart det er ledig på bussen.

3.3 Endring av lukketid

Lukketiden bestemmer hvor lenge kameraets sensorer skal være åpne for mottak av lys. Etter undersøkelse av forskjellige verdier for lukketid ser man at kameraets totale intensitet stiger tilnærmet lineært med økende lukketid. Under denne undersøkelsen er det funnet frem til den beste verdien som gjør at kameraet kan oppnå høyest mulig FPS, samtidig som at det mottar mest mulig lys. Tabell 3.1 viser at den mest optimale lukketiden for dette kameraet vil være når shutter value er innstilt på 800. En lavere verdi enn dette vil ikke føre til noen forbedring i hastighet av bilder i sekundet. En lavere verdi vil kun føre til at mindre lys slipper gjennom. Ved økende verdi av shutter value vil hastigheten på kameraet gå ned. Hastigheten vil på dette kameraet, bestemt ut i fra lukkertiden, ligge mellom 12 og 58 FPS. Ved å senke shutter value til 800 la man også merke til at bildene ble mye skarpere.

S.V.	Kjøretid while-løkke [ms]	Bilder i sekunder [FPS]	Intensitet [bit]
400	16,5	58,5	21'017'315
800	16,5	58,5	23'357'875
1200	24,5	41,5	28'796'500
1600	32,5	30,5	33'242'833
2000	40,5	24,5	36'707'600
2500	50,5	20	40'778'225
3000	60,5	16,5	44'119'975

Tabell 3.1: Kjøretidsmåling av kamera ved endring av Shutter Value(S.V.).

Kapittel 4

Brukergrensesnitt

I dette kapitlet gjennomgås de forskjellige funksjonene i brukergrensesnittet. Brukergrensesnittet som blir brukt i den nyeste versjonen av programmet til SiWa-Scan, ligner på det gamle. Undertegnede har gått gjennom det gamle og gitt det en ryddigere og enklere struktur. All tekst er gjort om til engelsk slik at ikke-norskspråkelige personer kan bruke det lettere. Det er også fjernet funksjoner som ikke ansees for å være vesentlige for personer som skal bruke instrumentet. Dette var funksjoner som bruktes til debugging.

4.1 Waferinspeksjon

Vedlegg 3 og 4 viser oppstartsbilde når man starter opp NewSiWa-OS v2.9 (vedlegg 8). I dette bildet kan man legge inn data for hvor stor wafer størrelse man ønsker å skanne (Wafer Size). Disse verdiene er med på å sette instillingene på hvor langt bordet skal gå.

For å legge inn en ny wafer må man trykke «Change wafer». Da kjøres bordet ut mot operatøren, og man kan legge inn en waferbit. Når dette er gjort kan man trykke «Load wafer», og bordet kjøres inn til startpossisjon for skanning.

Når alt er klart for skanning trykker man på «Start counting» og bordet begynner å telle opp dislokajoner og tvillinger (streker) i bildet. Under hele skanningen kan man følge med på hvor mye av waferen som er skannet (Progress). Nedenfor denne baren vises også hvor lang tid som har gått siden skanningen startet.

Man kan i dette vinduet også bruke «Step Mode» for å forflytte seg med

småe steg i x/y-retning. Ved startposisjon bør man ikke flytte bordet mot høyre i x-retning eller oppover i y-retning (bordets fysiske bevegelse). Dette kan føre til feilmelding og programkrasj.

Det er to fliker som kan velges av brukeren for å se hva som skjer i SiWa-scan under skanning. Dette er «Camera View» og «Dislocation Map». Den førstnevnte viser refleksjonsbildet som kameraet fanger opp under skanning. Den andre viser et kart over dislokasjoner på waferen. I «Camera View» kan man lese av hvor mange tvillinger og dislokasjoner som er funnet. Man kan også velge å nullstille tellerne, og man kan ta bilde av refleksjonen fra et bestemt område ved å trykke på «Capture picture». Ved å trykke på denne knappen vil det lagres en bmp-fil som viser bildet, og en xls-fil som viser pikselverdiene til kameraet i ett 640 x 480 array. Dette lagres på d-disken på pcen som tilhører SiWa-Scan. I «Dislocation Map» har man en funksjon som lar brukeren kunne lagre array-verdiene i kartet til en xls-fil. Det er også en bar ved siden av dislokasjonskartet som viser hvor mange dislokasjoner som befinner seg i de spesifikke områdene på kartet.

4.2 Innstillinger

I vedlegg 5 vises instillinger for å justere noen parametre i SiWa-Scan. Her kan man blant annet justere hvor stor oppløsning man vil ha på bildene ved å endre på «Resolution». Default resolution er satt til 300 µm. Man kan også endre hastigheten på bildetakingen til kameraet ved å endre på instillingen for «FPS». Default FPS er satt til 40 bilder i sekundet. Ved å endre på disse to instillingene vil while-sløyfen som styrer XY-bordet regne ut ny konstant hastighet som bordet skal ha under skanningen av en wafer.

I dette skjermbildet er det også mulig å endre på instillingene til XY-bordet, men dette anbefales ikke. Feile instillinger for XY-bordet kan føre til at bordet endrer oppførsel. Dersom man skulle ha behov for å endre instillingene, kan man se brukermanualen for XY-bordet (vedlegg 9).

4.3 Lasersensor

Skjermbildet som vises i vedlegg 6 viser målinger som er gjort av lasersensoren. Boks som vises til venstre, med muligheter for å endre «sensor flags» og «read out config», er opprinnelig fra leverandøren av lasersensoren. Det

eneste som er forandret her er «avraging rate» som er satt til 10 millisekund. Dette er gjort for å få en raskere oppdatering av sensorens verdi. Grafen til høyre viser laserintensiteten i [µW] ved hver iterasjon som while-løkken til lasersensoren går gjennom.

4.4 Bildekalibrering

Vedlegg 7 viser et vindu som lar brukeren kunne endre parametre som styrer hvor mye som skal trekkes av fra målingene for å få bort intensiteten til støyen. Default-verdiene til disse parametrene er de som man har kommet frem til i kapittel 2.

Når man endrer på parametrene kan man samtidig se på hvor mange dislokasjoner som telles i bildet. Ved for eksempel målinger rundt et ønsket nullpunkt kan man se om antall dislokasjoner ligger på/eller rundt null. Samtidig som man gjør dette kan man sjekke hvilket kamerabilde man har.

Kapittel 5

Parallelle operasjoner og LabVIEW-programmets virkemåte

Hensikten med dette kapitlet er å gi leseren et overblikk over hvordan datastrukturen til SiWa-Scan er bygd. Her gjennomgås kun de viktigste funksjonene som brukes i de ulike sløyfene i LabVIEW-programmet. Prosessoren som brukes i SiWa-Scans PC er en Intel (R) CORE™ i7-2620M CPU med 2 kjerner og fire tråder hver. I teorien skal alle sløyfer i LabVIEW-programmet kunne gå parallelt, men de har også fått ulike prioriteringer for å sikre at de viktigste oppgavene kjøres først.

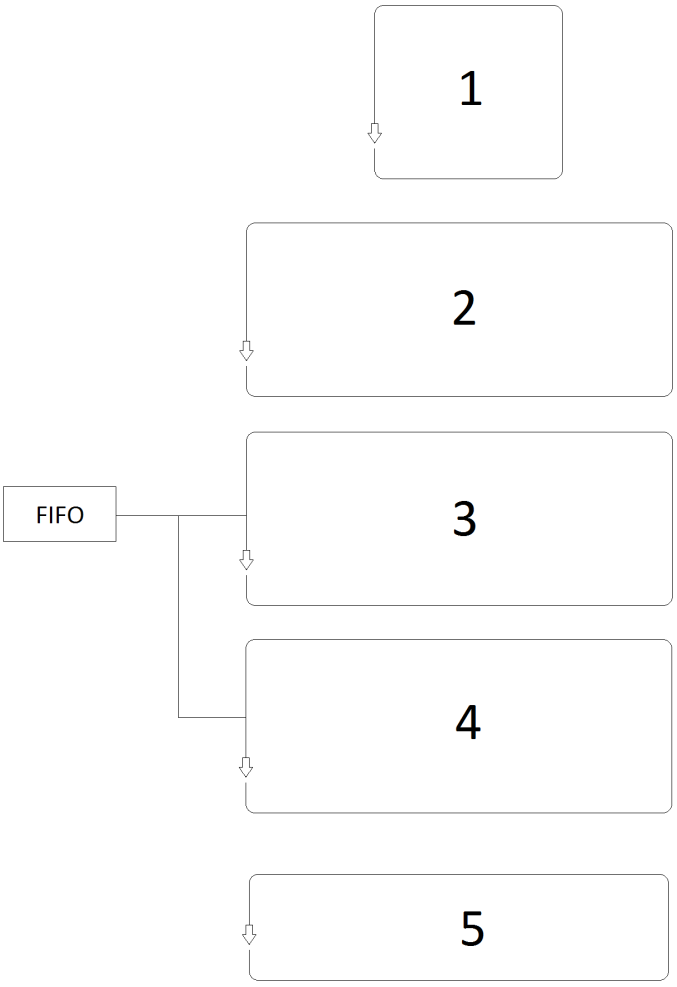
5.1 Parallelle sløyfer

Fra tidligere har SINTEF fått tips fra National Instrument om å endre på datastrukturen som samler inn og analyserer bilder i SiWa-Scan. Samtidig ble det i prosjektoppgaven våren 2013 oppdaget at samme while-løkke hadde ustabil kjøretid[1, side. 37]. På bakgrunn av dette er det i dette prosjektet opprettet en ny fil i LabVIEW for å lage et nytt program for SiWa-Scan, og siste versjon som er gjeldende for denne rapporten er NewSIWA-OS 2.9 (vedlegg 8).

Figur 5.1 viser en enkel skisse av hvordan programmet er bygd opp i LabVIEW. Sløyfene har en prioritering slik at kritiske oppgaver kjøres før mindre kritiske oppgaver. Jo høyere tall man har på prioritering av sløyfen, jo vikti-

gere er sløyfen[12]. Default-prioritering er ved opprettelse av sløyfer en verdi på 100.

Sløyfe 1 inneholder ikke-kritiske oppgaver, og har en prioritering på 98. Sløyfe 2 inneholder kjøringen av XY-bordet til SiWa-Scan og har en prioritering på 100. Sløyfe nummer 3 styrer bildeinnhentingen fra kameraet og har fått den høysete prioriteringen 101. Oppgaven med å hente inn bilder ansees for å være den viktigste oppgaven. Sløyfe 4 inneholder jobben med å analysere bildene som kommer inn fra kameraet. Prioritering på denne er satt til 99. Siste sløyfe, nummer 5, jobber med å hente inn verdier fra referansesensoren, og har en prioritering på 100. Det vil si at sløyfe 5 og 2 ansees for å være like viktig.



Figur 5.1: Skisse av datastruktur til SiWa-Scan. Datastrukturen er i hovedsak bygd opp med fem «timed while loops», hvor sløyfe 3 og 4 bruker en FIFO-kø (First-in-first-out).

5.1.1 Ikke-kritiske oppgaver (Sløyfe 1)

Som nevnt ovenfor, fikk sløyfe 1 lavestet prioritering på sine oppgaver. Sløyfens oppgaver er å la brukeren lagre en fil av dislokasjonskartet som SiWa-Scan lager etter skanning av en wafer. En annen oppgave som er lagt til, er å skifte skjermbilder. Det kan gjøres ved å trykke på en av de knappene som er til venstre for bildet i brukergrensesnittet (Se vedlegg 3-7). En tredje oppgave er å nullstille tellere for dislokasjoner og tvillinger, og dislokasjonskartet. Etter ett scan må man nullstille disse før man kan skanne en wafer på nytt. Den fjerde oppgaven som er lagt til er å vise hvor lang tid det har tatt siden man startet skanning.

5.1.2 XY-bord (Sløyfe 2)

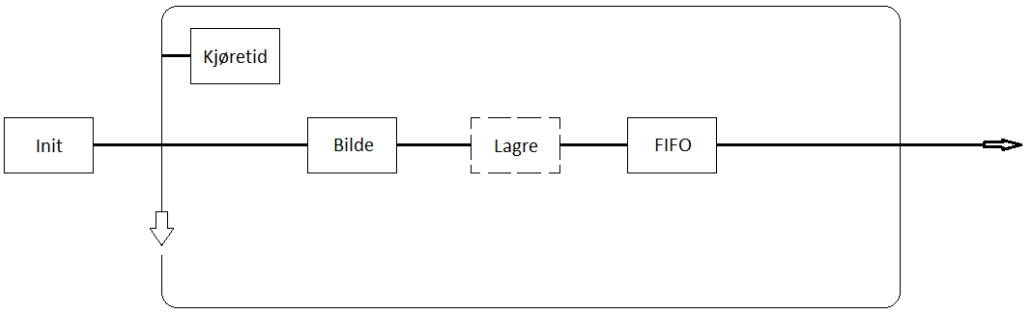
XY-bordet, som styres i sløyfe 2, er fra før av bygd opp av to Zaber T-LSQ300B og gir bordet fire frihetsgrader. Sløyfe 2 er ikke bygd opp på nytt, men er den samme sløyfen som i det gamle SiWa-programmet med noen modifikasjoner. Undertegnede har fjernet noen funksjoner som ikke er viktige for det nye programmet, mens SINTEFs veileder for prosjektet har lagt til noen modifikasjoner(forklart i neste avsnitt).

Fra tidligere var man plaget med at dislokasjonskartet viste et sikk-sakk mønster ved skanning av waferprøve. Dette er også et kjent problem fra PV-Scan ifølge veileder fra SINTEF. I PVScan løste de problemet med å la bordet skanne og lagre data til dislokasjonskartet i en retning, og da fikk de bort sikk-sakk-mønsteret som dannet seg i kartet. Veileder fra SINTEF har fulgt samme oppskrift og lagt inn denne funksjonen i styringen til XY-bordet. Det viste seg å fungere bra, og ingen sikk-sakk mønster danner seg i dislokasjonskartet. Når bordet beveger seg i motsatt retning enn skanneretning, så vil det gå med full hastighet.

5.1.3 Innsamlingsløkke (Sløyfe 3)

Innsamlingsløkken, sløyfe 3, er den sløyfen som har fått høyest prioritet i programmet. Det fordi man ønsker at kameraet ikke skal hoppe over områder som skal være med i skanningen. Dette kan i teorien skje dersom sløyfen får en lavere eller lik prioritet som andre sløyfer i programmet.

I figur 5.2 ser man at startbetingelser for parametre som tilhører løkken settes først (Init). Under kjøring av løkken kan man måle kjøretiden på løkken.



Figur 5.2: Skisse av sløyfe 3 som henter inn bilder fra kameraet.

Dette kan brukes av person som er kjent med LabVIEW-programmering og skal drive med debugging, og er ikke en del av brukergrensesnittet for operatøren. Videre hentes det inn bilder fra kameraet. Disse kan lagres på datamaskinen i en bmp-fil og xls-fil (Se kapittel 4.1). Til slutt vil bildene bli sendt videre i en «cluster» til en FIFO-kø som analyseløkken kan hente ut bilder fra.

Kameraets bildetaking er ikke koblet opp og bestemt etter hvor langt XY-bordet har gått. Det fordi det er vanskelig å få til en god lengdemåling av bordet under høy hastighet. I stede lar man bordet gå med konstant hastighet bestemt av hvor mye FPS kameraet er stilt inn på. I teorien vil bordet flytte seg langt nok før neste bilde av waferen tas, og dermed klarer man å dekke hele waferen.

I det gamle SiWa-Scan-programmet lå bildeinnhentingen og analyse av bilder i samme sløyfe, og analysen kunne bruke ulik tid på å finne ut om det var streker i bildet. Brukte analysen for lang tid var det fare for at bordet hadde gått for langt når neste bilde ble tatt, og man måtte holde en lav FPS for å forhindre at dette skjedde.

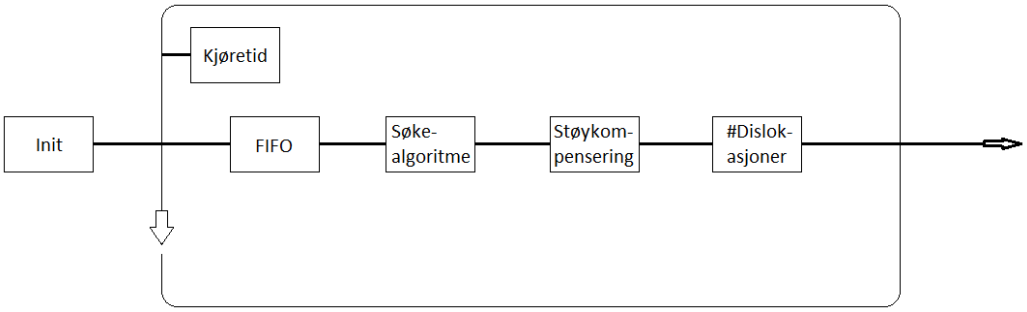
Innsamlingsløkken og analyseløkken(kapittel 5.1.4) er i dette prosjektet bygd opp etter inspirasjon fra et «produsent-konsument-mønster»[13]. FIFO-køen er også satt til å kunne lagre uendelig (teoretisk) mange bilder. Ved å bruke dette mønsteret kan man la analyseløkken bruke så mye tid den trenger til å analysere bilder, uten at det går utover innhentingen av bilder fra kameraet.

Før man kom frem til denne løsningen, forsøkte man å bruke en funksjon som lagret bildene til en xls-fil til hardisken. Tanken var å lage en slags FIFO-kø mellom sløyfe 3 og 4, men dette fungerte ikke som forventet. For å lagre en slik fil måtte man bruke en driver i LabVIEW-bibiloteket som går under navnet «ArrayToSpreadSheetString». Denne driveren klarer maksimalt å lagre fem

filer til hardisken i sekundet. Noe som igjen tilsvarer at man ikke klarer å hente inn mer enn fem bilder i sekunder(5 FPS). Altså en mye dårligere løsning enn det som finnes i det gamle SiWa-Scan-programmet.

5.1.4 Analyseløkke (Sløyfe 4)

Sløyfe 4 sin hovedoppgave er å finne ut hvor mange dislokasjoner som befinner seg i et bilde. Den skal også finne og trekke fra eventuelle tegn på tvillinger i bildet.



Figur 5.3: Skisse av analyseløkken, sløyfe 4.

Figur 5.3 viser en skisse av denne løkken. Her kan man se at startbetingelser for parametre som tilhører løkken blir satt utfor løkken. Man kan i likhet med forrige løkke måle kjøretiden på hver iterasjon. Inni løkken vil den først starte med å hente ut bilder fra FIFO-køen. Hvert bilde som blir hentet ut vil også bli slettet i køen, slik at man får frigjort plassen som det tar opp. Bildet sendes så videre til en søkealgoritme som leter opp eventuelle streker i bildet, og fjerner disse fra bildet. Neste steg er å kompensere for støy fra linse og waferoverflate. Til slutt vil den regne ut hvor mye dislokasjoner som befinner seg i bildet.

5.1.5 Lasersensor (Sløyfe 5)

Sløyfe nummer fem har som oppgave å måle laserintensiteten slik at dette kan brukes videre for kompensering for ustabil laser. Denne sløyfen er lik den som det gamle SiWa-Scan programmet bruker.

Kapittel 6

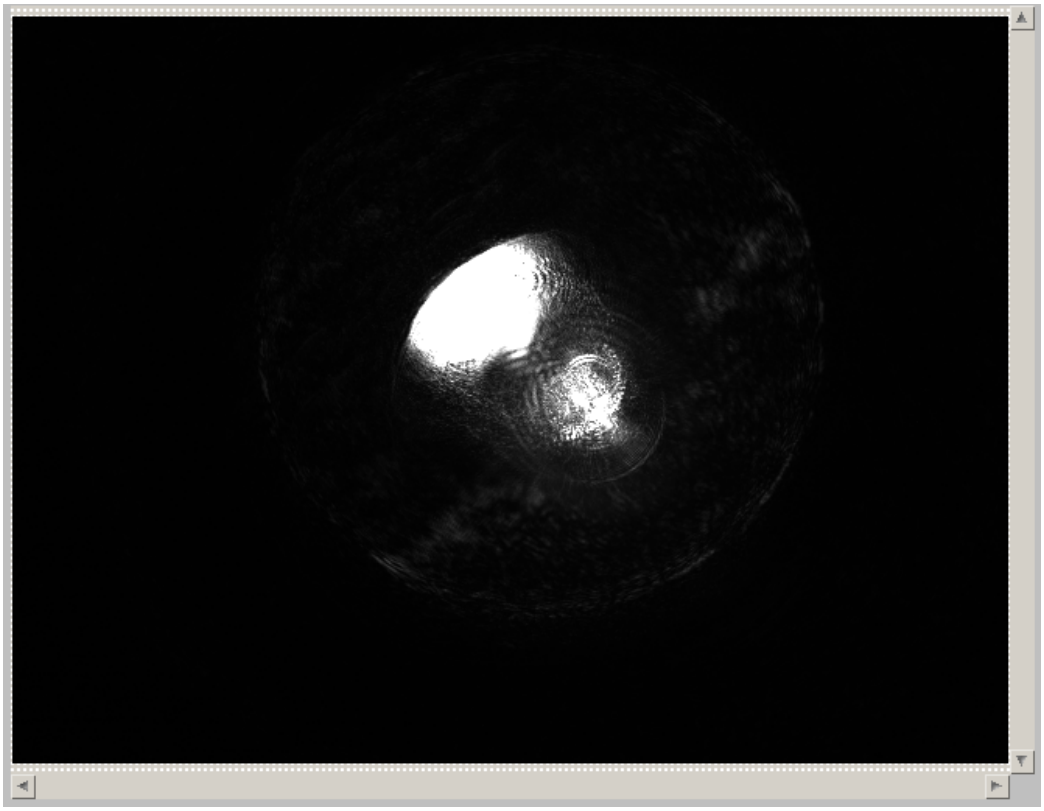
Gjenkjenning av tvillinger og korngrenser

Dette kapitlet går gjennom og beskriver hvordan bildemønstre kan oppstå i SiWa-scan. Det tar også for seg en påbegynt C-kode, og den resulterende LabVIEW-koden som ble den endelig løsning på SiWa-Scans gjennkjennings-algoritme.

6.1 Bildemønster i SiWa-Scan

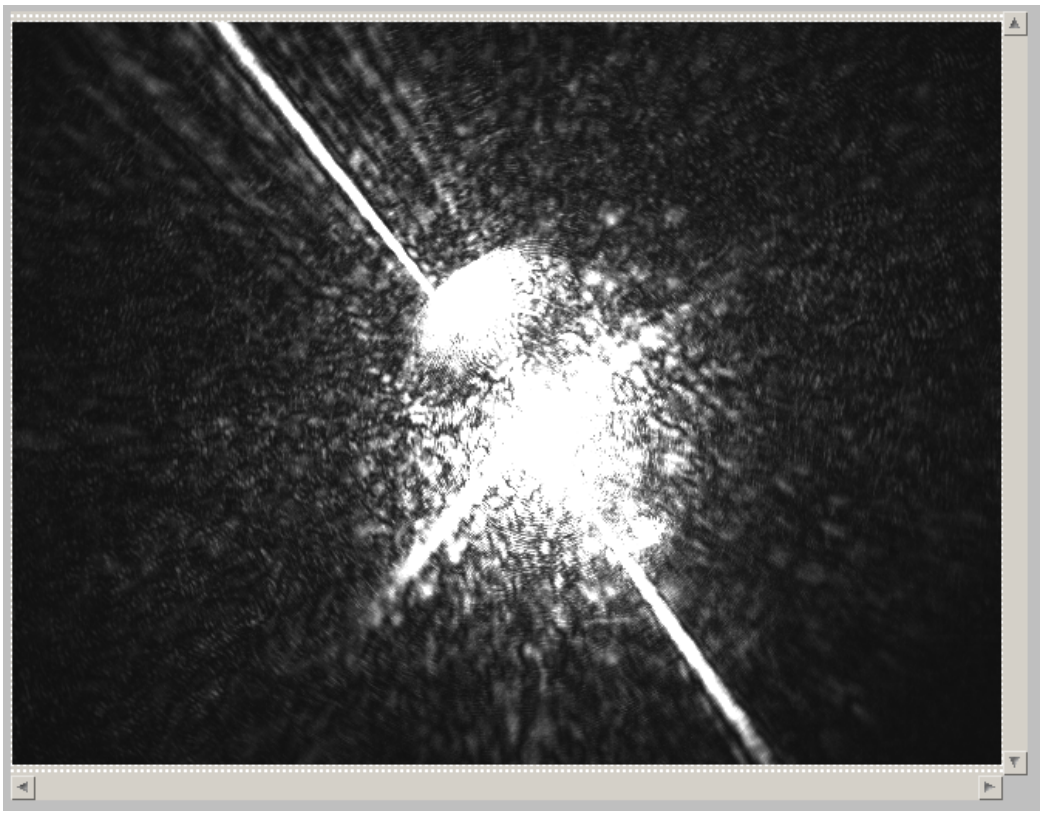
For å kunne lage en tilpasset kode til SiWa-Scan som fjerner uønskede objekter fra bildene, må man vite hvordan objekter viser seg i bildet. Bildene vil også være unike i forhold til hverandre, og kan bli svært komplekse. Figur 6.1 - 6.4 viser fire ulike tilfeller for hvordan objekter kan vise seg i SiWa-Scan.

Figur 6.1 viser hvordan støy fra linser opptrer i et bilde. Det er tidligere gjort forsøk på å fjerne dette, men man har foreløpig ikke klart det ettersom det er vanskelig å finne en løsning. Thorlabs, leverandøren av optikken, har også sett på dette. De foreslo at man kunne vri litt på beamsplitteren for å se om det hjalp. Dette har blitt prøvd, men ga ingen god løsning. Bildet kom da ut av senter og man tapte data. Det har også vært prøvd å bruke linser som er «refleksjonsfri», men dette førte til forverring av bildet(kapittel 1). Den beste løsningen for å utelukke denne refleksjonen er å utelukke det i software. Det kan man enten gjøre ved å utelukke ett rektangulært/kvadratisk område i bildet som støyet befinner seg i, eller å finne frem til og utelukke hvert enkelt piksel til støyet.



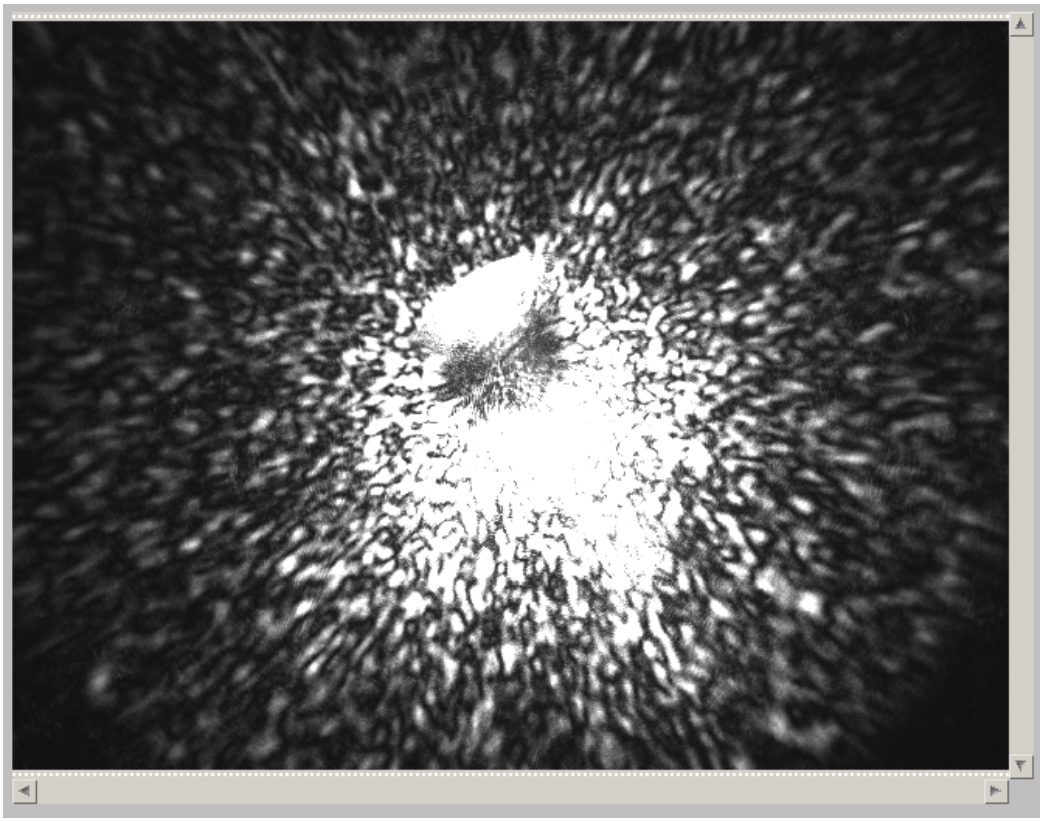
Figur 6.1: Viser hvordan støy fra linsen opptrer i bildet (Ingen refleksjon fra wafer).

Figur 6.2 viser hvordan en tvilling eller korngrense kan opptre i et bilde. Her ser man at tvillingen går skrått fra topp til bunn. Man ser også at intensiten fra tvillingen lyser godt opp, og den er sammenhengende. Tvillingen i figuren vil i teorien være mindre kompleks å skille ut enn tvillingen i figur 6.4. Det antas at prikker som er i bildet er dislokasjoner som skal være med i tellingen.



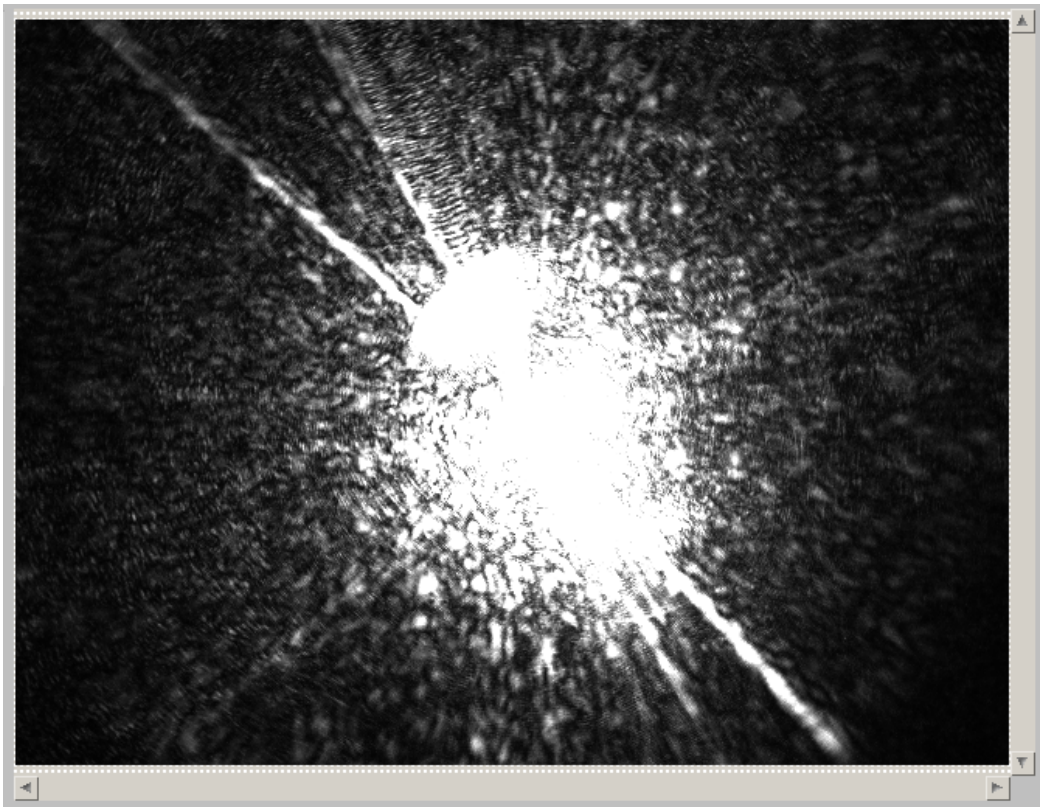
Figur 6.2: Tvilling vises som en lys strek i bildet.

Figur 6.3 viser et bilde av et område på waferen med høyt antall dislokasjoner, men uten tvilling i bildet.



Figur 6.3: Bilde av dislokasjoner.

Figur 6.4 viser et mer komplekst bilde som kan oppstå i SiWa-Scan. Her ser man flere tvillinger i bildet. Man ser også antydning til små subtile streker med lavere intensitet enn de tydligste strekene. Streken som er mest intensiv ser også ut til å miste sin intensitet mot kanten av bildet, og kan bli vanskelig å oppdage for en gjenkjenningsalgoritme. Av erfaring viser det seg at slike bilder ofte oppstår i SiWa-Scan.



Figur 6.4: Komplekst bilde.

6.2 C-kode

Det ble gjort et forsøk på å lage en kode i C som er tilpasset SiWa-Scans bilder. Programmet som ble brukt til å skrive koden er NetBeans IDE versjon 7.4, og koden ble kompilert i Cygwin. Før programmeringen satte man NetBeans opp mot Cygwin.[14]

Det ble brukt et enkelt 10 x 10 array istedet for 640 x 480 array som pikslene til bildene fra SiWa-Scan er lagret i. Dette gjør det enklere å jobbe med kodingen, og man kan senere utvide til 640 x 480 array for å teste på SiWa-Scan-bilder.

Koden inneholder en funksjon som gjør det mulig å lese data fra en csv.fil (comma seperated values) fra harddisken. Denne filen kan man hente ut fra SiWa-Scans funksjon for lagring av bildedata. I denne funksjonen blir det da lagret som en xls-fil, men ved å åpne denne filen i excell og lagre den som csv-fil, så har man riktig fil-type for C-koden.

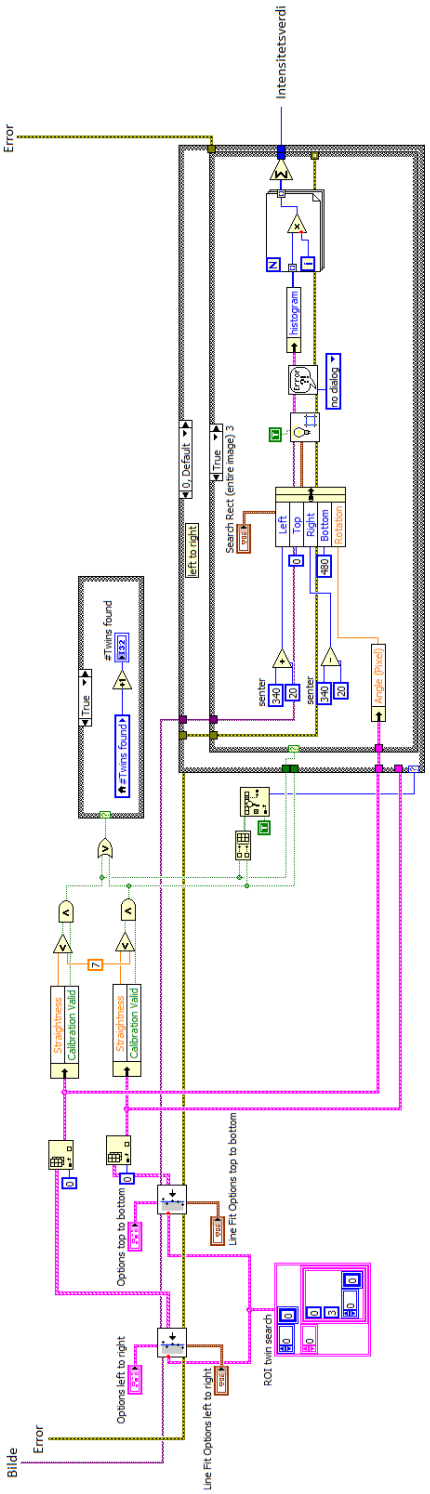
Programmet inneholder også en funksjon som gjør det mulig å hente ut et array fra en fil som beskriver hvilke piksler som er «døde». Døde piksler er piksler som er berørt av forstyrrelsen fra linsene i SiWa-Scan. Etter at de døde pikslene er fjernet, lages det et nytt 2D-array uten disse pikslene. Det er også lagt inn en funksjon for å fjerne et rektangulært/kvadratisk område i arrayet. Parametrene for den velges i main()-filen.

Når arrayet har gjennomgått fjerning av døde elementer og utelukking av område, blir resterende array sendt til en funksjon som lagrer de interessante pikslene i en liste. De interessante pikslene vil være piksler som er bestemt etter en maksverdi. Dette ble gjort for å videre kunne brukes til å finne ut hvilke piksler som var naboer, for så bruke det videre i arbeidet med å finne streker i bildet. Men siden oppgaven med å utvikle denne koden ble mer komplekst og tidkrevende etter dette punktet, ble det istedet valgt å se på om man kunne gjøre noe med den eksisterende LabVIEW-koden for gjenkjenning. Prosjektmappen for C-koden ligger som vedlegg 10.

6.3 LabVIEW-kode

Figur 6.5 viser hvordan algoritmen for gjenkjenning av tvillinger/korngrenser er satt opp i LabVIEW. Denne koden er laget av veileder fra SINTEF, og verdiene i de ulike parametrene er funnet via uttesting på forskjellige tvillinger.

Denne algoritmen ligger i sløyfe 4 og henter ut bilder fra FIFO-køen. Det første den gjør er å skanne to ganger etter kanter i bildet. Den søker fra venstre til høyre, og fra topp til bunn. Dersom den finner en strek, regner den ut en «straightness» for den. Ligger denne verdien under syv, anses den som en gyldig tvilling/korngrense. I neste steg finner den frem til en vinkel for denne streken, og bruker denne informasjonen til å legge en boks rundt streken. Vinkelen regnes ut fra et fast sentrum i bildet. Intensiteten som er i denne boksen blir summert og sendt videre i programmet, hvor verdien blir trukket fra før man fjerner støydata og regner ut antall dislokasjoner.



Figur 6.5: Strekgjenkjenningsalgoritme laget i LabVIEW.

Kapittel 7

Tester

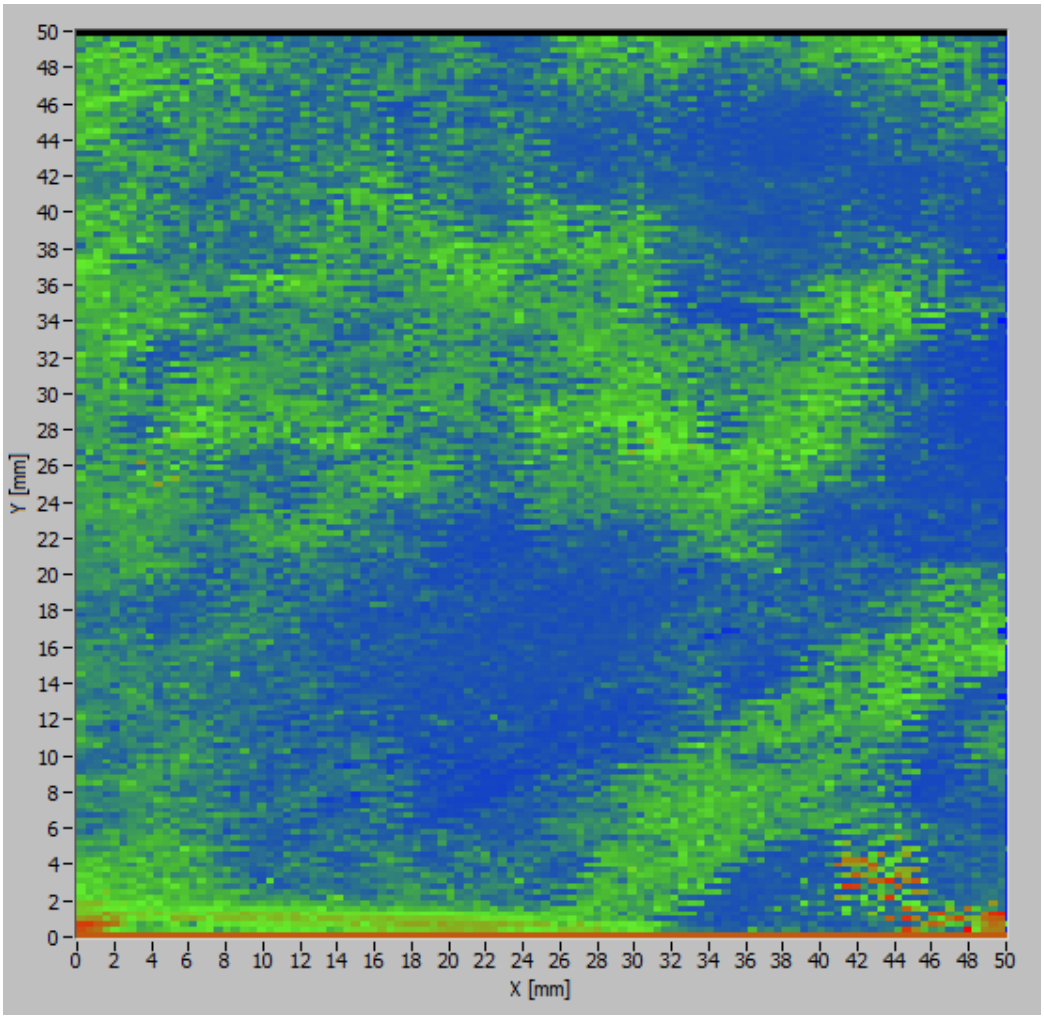
I dette kapitlet presenteres fire ulike tester som er gjort på SiWa-Scan. Alle testene er gjort på waferen vist i figur 7.1. Denne waferens totale størrelse er 98 x 100 mm. I testene ble kun et område på 20 x 20 mm, og 50 x 50 mm brukt.

Figur 7.1: Bilde av overflaten på wafer «KMB8 185 etset».

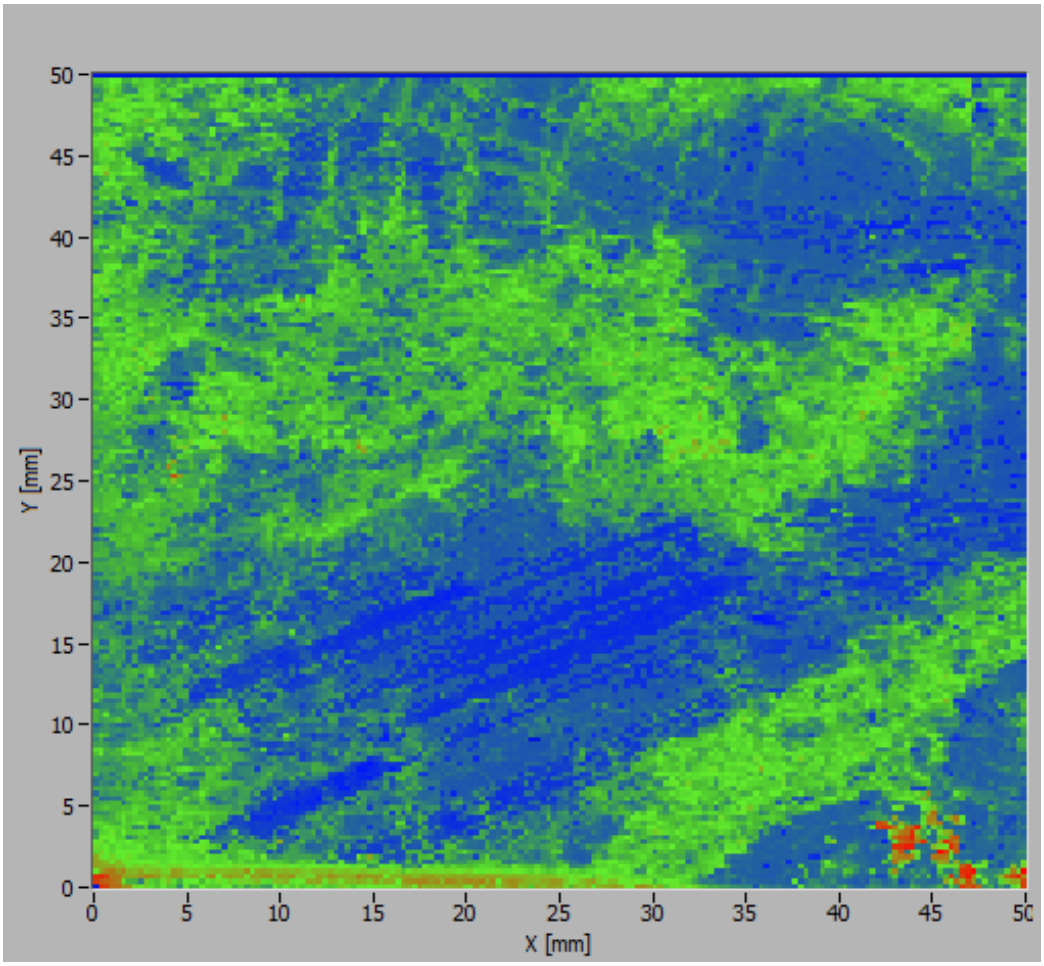
7.1 Programsammenligning

I denne testen ble alle skann gjort med en hastighet som bestemt etter 40 FPS, og oppløsning på 300 µm. Den første testen er gjort med programversjonen som var ferdig etter Bacheloroppgaven våren 2011, figur 7.2. Den siste er fra dette prosjektet, og vises i figur 7.3. Området på waferen som ble skannet var 50 x 50 mm av nedre, venstre hjørne.

Ved å sammenligne disse to bildene, ser man at sistnevnte figur viser et betydlig klarere skann. Man ser at sikk-sakk-mønsteret ikke er tilstede i dette bildet, og tvillingene trekkes fra. Tvillingene er de blå strekene som strekker seg fra midten og ned til venstre hjørne.



Figur 7.2: Skann av wafer med programvare fra bacheloroppgaven.



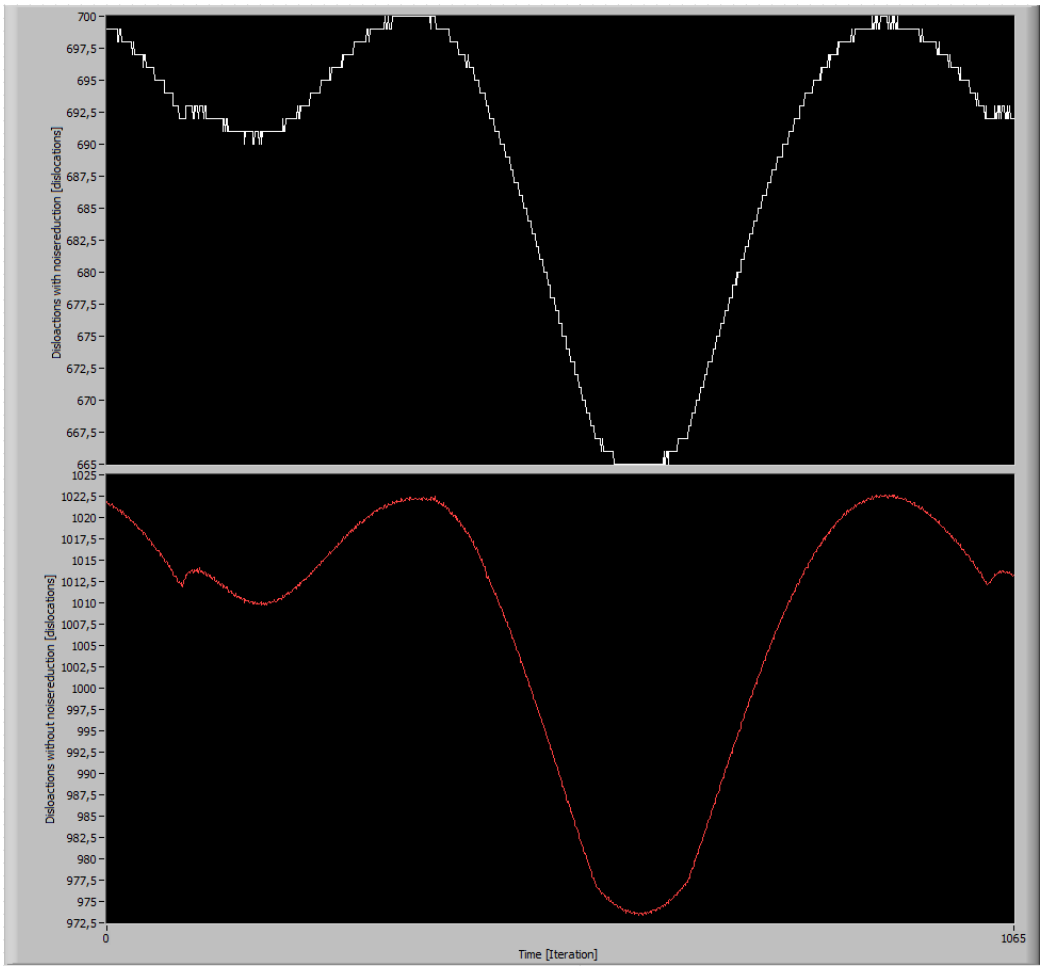
Figur 7.3: Skann av wafer med programvare laget i dette prosjektet (NewSiWa-OS v2.9).

7.2 Dislokajonstelling med og uten støykorreksjon

I figur 7.4 ser man hvordan SiWa-Scan teller dislokasjoner i et fast punkt over 1065 iterasjoner av løkke 4. I nederste graf ser man hvordan SiWa-Scan ville telt ved å bruke kun rådata, mens øverste viser hvordan det telles med støykorreksjon.

I øverste graf ser man at tellingen svinger mellom 665 og 700 dislokasjoner. Antar man at riktig antall dislokasjoner er 682,5, vil målingen avvike 2,56 % fra dette punktet.

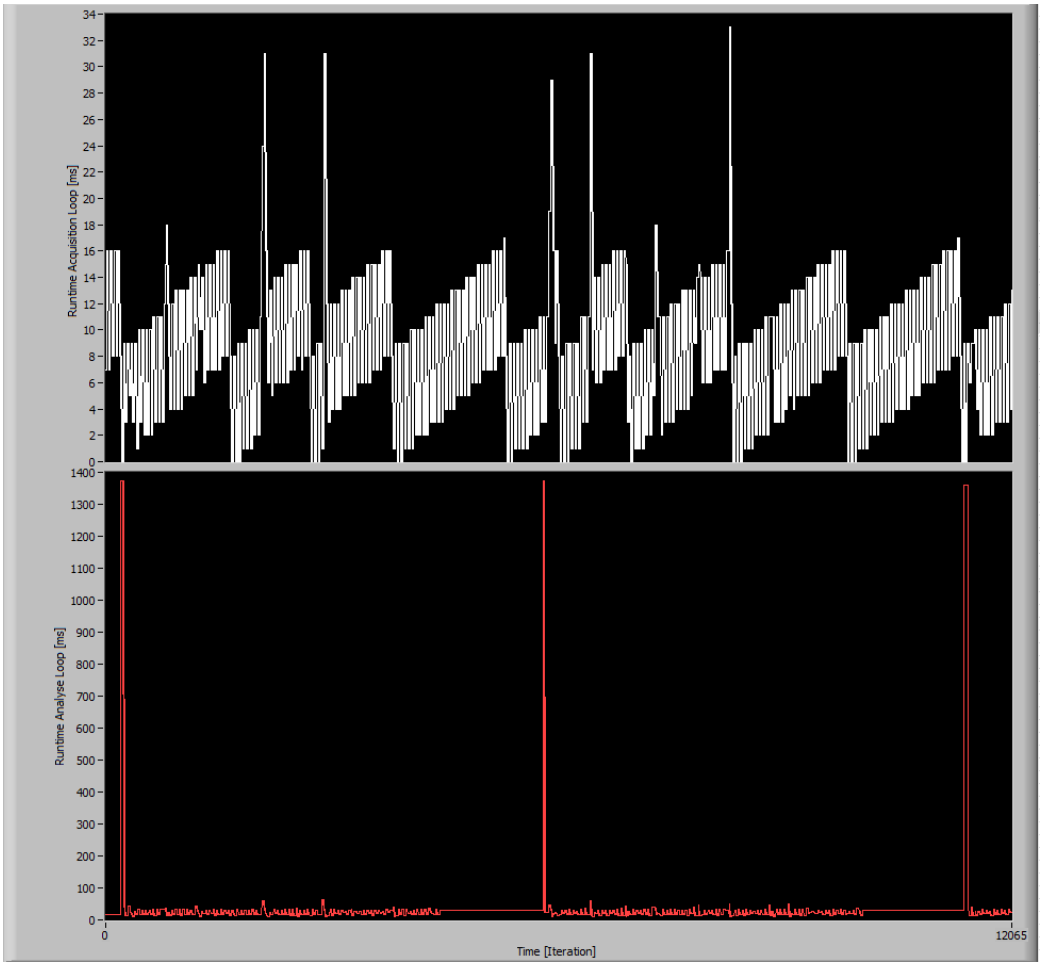
I nederste graf ser man at tellingen svinger rundt et mye høyere midtpunkt enn i den øverste. Tellingen svinger her mellom 973 og 1023 dislokasjoner. Middelverdien er 998 dislokasjoner, og målingen avviker 2,51 % fra denne verdien.



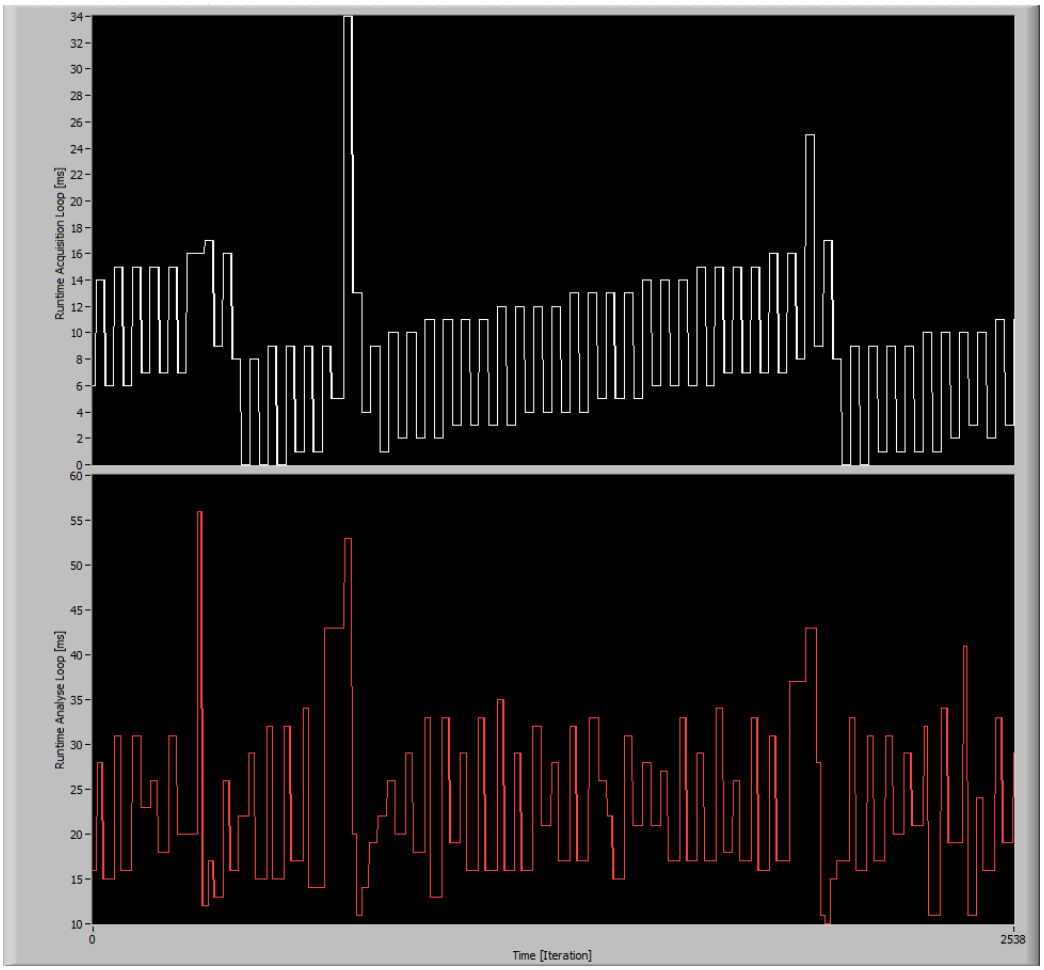
Figur 7.4: Nederste figur viser telling av dislokasjoner uten korreksjon for støy, mens øverste viser med korreksjon for støy.

7.3 Kjøretid, kamerasløyfe og analysesløyfe

I figur 7.5 ser man kjøretiden målt for sløyfe 3 og 4. Det vil si sløyfen for innhentingen av bilder, og sløyfen for analyse av bilder. Øverste graf viser at kjøretiden for innhenting av bilder stort sett ligger under 18 millisekund, men at den kan hoppe helt opp til 34 millisekund(figur 7.6). Nederste graf i figur 7.5 viser at kjøretiden for analyse av bilder kan hoppe opp mot ca. 1375 millisekund. Dette skjer hver gang før analysesløyfen starter med å hente ut bilder fra FIFO-køen.



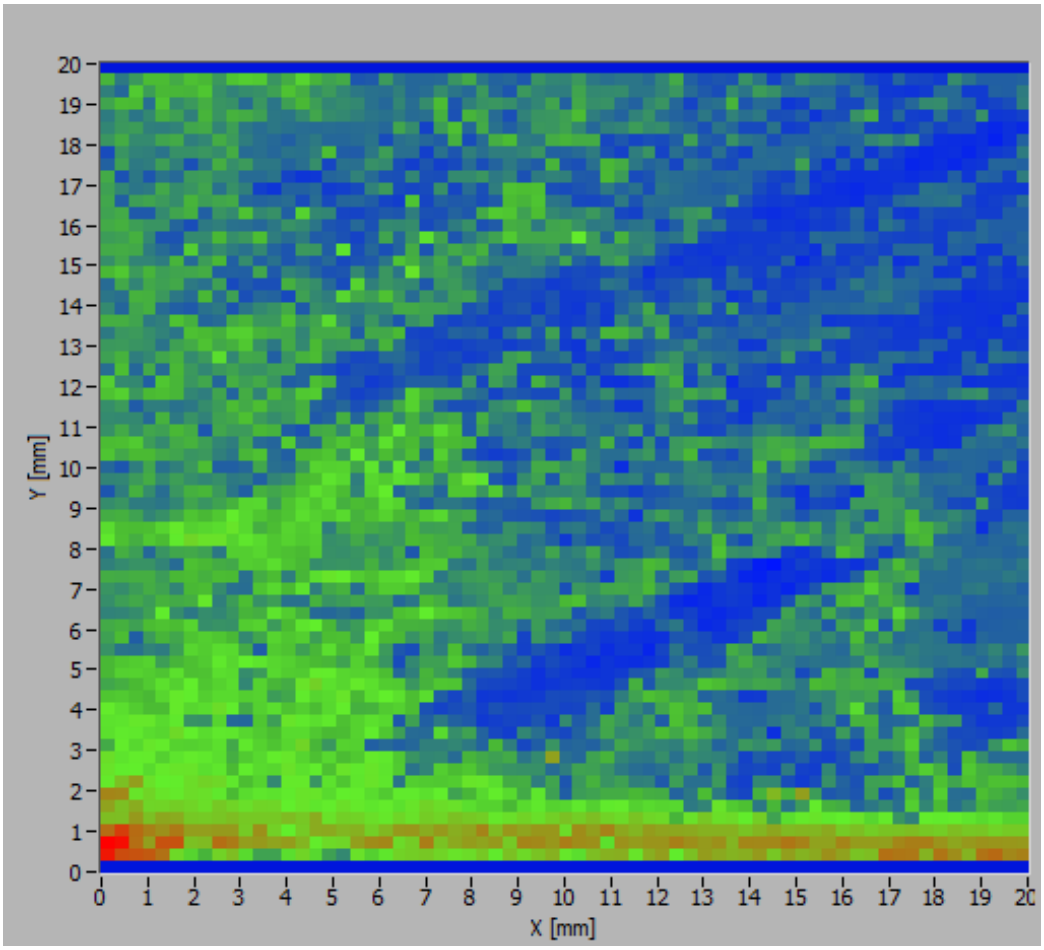
Figur 7.5: Kjøretid målt fra sløyfe 3 og 4 under tre skann i x-retning.



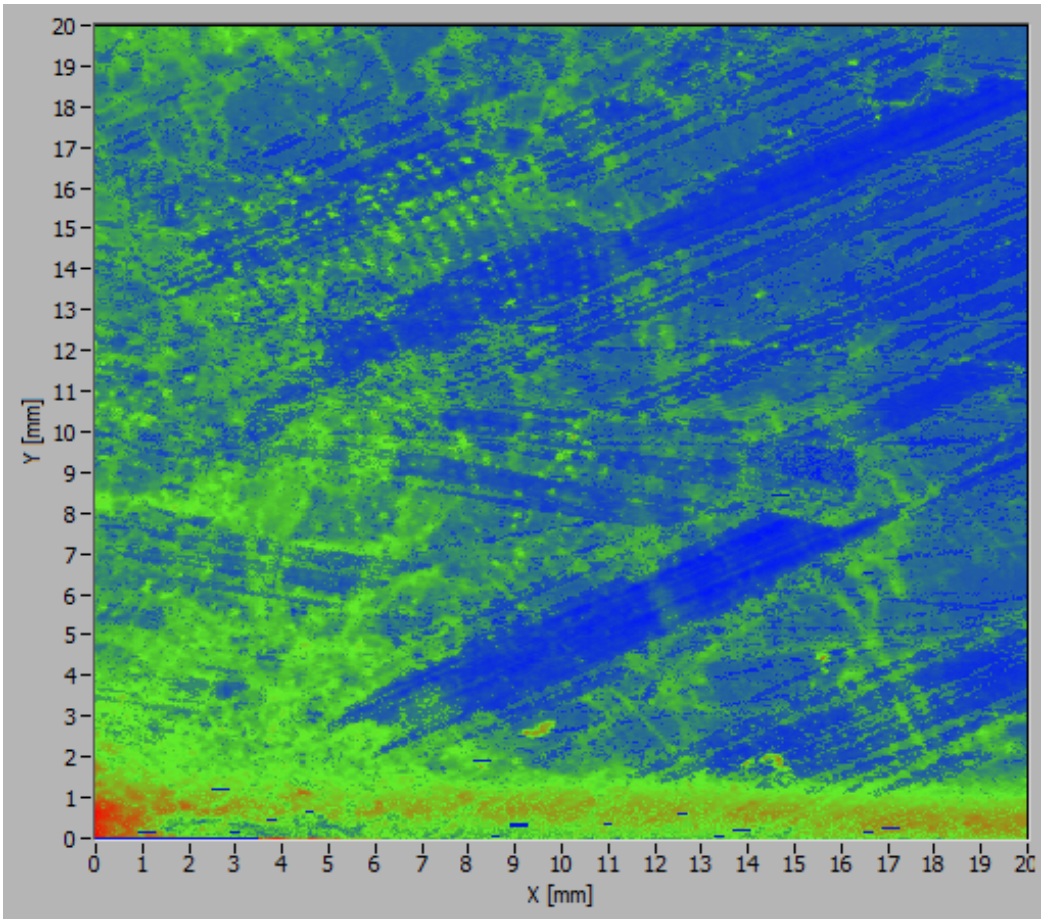
Figur 7.6: Kjøretid målt fra sløyfe 3 og 4 under ett skann i x-retning.

7.4 Oppløsning

I figur 7.7 ser man et skann gjort med oppløsning stilt inn på 300 µm og i figur 7.8 med 50 µm. Med en oppløsning på 300 tar det 2 min og 41 sekunder å skanne et område på 20 x 20 mm. For samme område med oppløsning innstilt på 50 vil det ta 72 minutter og 45 sekunder. For begge skann er kameraet satt til 40 FPS. I figurene ser man at det er tap av data ved første og siste skanning. I figur 7.8 ser man noen blå rektangler nede i bildet, noe som mulignes kan tyde på tap av data i disse områdene.



Figur 7.7: Skann gjort med oppløsning på 300 µm.



Figur 7.8: Skann gjort med oppløsning på 50 µm.

Kapittel 8

Diskusjon

Av erfaring har man kommet frem til at beste hastighet på bordet er når kameraet er innstilt på 40 FPS. En høyere hastighet enn dette viser tegn på at kameraet ikke klarer å henge med og man taper data. For å se om hastigheten på PC'en er årsaken til dette, kunne det vært interessant å teste ut programmet på en PC med raskere prosessor. Det ble ikke tid til å gjennomføre en slik test i dette prosjektet.

Man kom i kapittel 2.1 frem til at den nye linsen økte lysringe i kamerabildet. Lysringen stammer fra den nederste linsen. Hvis man bare ved å endre linse på optikken også øker lysringen, kan det tenkes at det er mulig å gjøre det motsatte. Det kunne vært interessant å se om man ved å sette inn linser med andre fokallengder oppnåde reduksjon i størrelsen på lysringen.

Fra kapittel 2.2 har man kommet frem til at referansesensoren måler laserintensiteten bedre enn før. Målingene viser noe påvirkning fra underlaget man analyserte, men sett i forhold til før oppgradering av optikken, kan man se bort fra denne forstyrrelsen. Man har også kommet frem til at referansesensoren svinger mellom 150 µW - 168 µW, og dette er mindre enn før oppgradering. I følge spec-sheet fra Thorlabs (vedlegg 11) skal laseren ligge mellom 0,5 mW og 1 mW. Laseren går gjennom beamsplitteren som sender 50 % til referansesensoren og 50 % til waferoverflaten. Det vil i teorien si at referansesensoren skal måle mellom 0,25 mW og 0,5 mW. Årsaken kan muligens være at beamsplitteren ikke sender lik mengde lys til sensor og waferoverflate. En annen årsak til dette kan være at laseren treffer kanten på referansesensoren, slik at ikke hele strålen treffer fotodioden. Laseren har vært inne til reperasjon flere ganger før dette prosjektet, og av den grunn kan det være mulig at den ikke stemmer med spec-sheet.

I kapittel 2.3 er en lineær modell for støyen regnet ut for hånd. Dette er en metode som tar tid, og det burde sees på om man kan gjøre dette automatisk i LabVIEW. SiWa-scan kan være følsom for eksternt lys, og parametrene for støy kan være påvirket av dette. Ved å legge til en autofunksjon som regner ut nye konstanter raskt og enkelt for støyparametrene, kan man kjøre denne autokorrigeringen før skanning, og øke treffsikkeheten på tellingen.

I det nye SiWa-scan-programmet er det brukt en modifisert versjon av løkken til XY-bordet. Denne løkken inneholder en dokumentasjon som er på norsk. Det burde skrives om til engelsk slik at ikke-norskspråklige personer forstår hvordan programmet fungerer.

Kapittel 2.4 gir en beskrivelse av hvordan man har gått frem for å finne en korreksjonsfaktor for tellingen av dislokasjoner. Punktene som brukes på testwaferen er vanskelig å treffe eksakt slik som mikroskopbildene viser. Dette fordi det er vanskelig å se hvilke områder man treffer med laseren i SiWa-Scan, og fordi det gjøres for hånd på områder i µm-nivå. Tellingen gjort i SiWa-Scan kan derfor avvike noe fra tellingen i mikroskop. Dersom man i fremtiden skal finne frem til en ny korreksjonsfaktor, vil det være gunstig å bruke flere områder enn to. Det kan være med på bedre bestemmelsen av en ny korreksjonsfaktor.

SiWa-Scan-programmet er bygd opp på nytt slik som beskrevet i kapittel 5. Ved å gjøre det får man med seg hele tankegangen på hvordan SiWa-Scan må fungere, og man slipper å bruke mye tid på finne ut hvordan ting er gjort fra før.

Ideeinnehaver av Siwa-Scan har nevnt at streker muligens kan oppstå i kamerabildet hvis laseren treffer på et område som har dislokasjoner i en array- eller flaskehalslignende formasjon. På grunn av mangel på tid, ble det ikke undersøkt om dette var tilfelle.

Den nye koden for gjenkjenning av tvillinger bruker et fast punkt (senter) i bildet. Dette punktet kan endre seg hvis optikken i SiWa-Scan blir utsatt for bevegelse, som for eksempel kan skje under flytting av SiWa-Scan. Det burde derfor sjekkes med jevne mellomrom om punktet er på riktig plass. Under uttesting av denne algoritmen, så det ut for at algoritmen fant streker oftest i bildet når den skannet fra topp til bunn.

Under testingen av støykorreksjonen i SiWa-Scan i kapittel 7.2 kom man frem til at det er vanskelig å få kompensert for svingninger i laserintensiteten. Det er mulig at modellen for støyen må være ulineær.

Det ble ikke tid til å sjekke hvordan tellingen av dislokasjoner blir når man

endrer på oppløsningen av bildet. Det er mulig at når man øker oppløsningen (300 til 50 µm) så vil intensiteten fra flere bilder summeres opp, og man får et høyere totaltall for dislokasjoner. En mulig løsning på dette er å øke/senke korreksjonsfaktoren, eller innskrenke bildet i software, etter hva oppløsningen er satt til. Det er ikke anbefalt å bruke høy oppløsning hvis man må gjøre raske skann, siden bedre oppløsning krever mer skanntid.

Under kjøring av instrumentet flere ganger på rad, viste det seg at instrumentet kunne stoppe opp midt i et skann. For å forhindre at noe slikt skjer må man slå av og på programmet før et nytt skann kjøres. Det burde også fjernes at man må trykke «load wafer» når XY-bordet allerede står i startpossisjon.

Kapittel 9

Konklusjon

Undertegnede av rapporten har:

- fjernet de uønskede refleksjonene ved å bytte ut blindpunkt og beamsplitter. Dette har ført til at man kan måle laserintensiteten uten påvirkning av forstyrrelse fra underlaget som skannes. Den nye linsen som Thorlabs anbefalte forverret bildet, og man beholdt da den gamle i stedet.

- laget en lineær funksjon for støy fra linse og waferoverflate. Denne funksjonen klarer å fjerne noe støy, men klarer ikke å kompensere for å svingninger i laserintensiteten.

- kommet frem til en ny korreksjonsfaktor for telling av dislokasjoner.

- satt seg inn i teori om digitalkamera, endret lukketiden på kameraet og økt hastighet på FPS. Dette har igjen senket tiden som SiWa-Scan trenger for å analysere en wafer.

- forbedret brukergrensesnittet ved å gjøre det enklere og mer strukturert.

- bygget opp LabVIEW-programmet på nytt, og innført FIFO-kø i systemet.

- påbegynt en C-kode for å finne streker i bilder.

Veileder fra SINTEF har:

- fjernet sikk-sakk mønsteret.

- lagt til dislokasjonskartet.

- programmert koden for gjenkjenning av tvillinger/korngrenser.

Fredag 20.12.2013 viste undertegnede og veileder fra SINTEF en demonstrasjon av SiWa-Scan for ideinnehaveren Gaute Stokkan. Han mente at framgangsmåten som ble gjort for å finne frem til en korreksjonsfaktor for telling av dislokasjoner var riktig. En lignende måte gjøres også i PVScan. Han ville også ha inn en funksjon som gjorde det mulig å flytte bordet til en bestemt posisjon, samt en indikator på dislokasjonstetthet. Etter demonstrasjonen meddelte han at SINTEF nå skal ta i bruk SiWa-Scan.

Kapittel 10

Videre arbeid

Nedenfor vises forslag på oppgaver man kan jobbe videre med:

- Legge inn en funksjon som hindrer bruker i å kjøre bordet fysisk mot venstre og/eller opp når det står i startpossisjon og step mode.

- Endre på funksjonen som lagrer bilder fra kamera i en bmp- og xls-fil til å la bruker selv velge navn, plass og fil-type.

- Lage en funksjon som viser bordets posisjon i [µm].

- Legge til en funksjon som lar brukeren gå til ønsket posisjon på en wafer.

- Endre dokumentasjonen for XY-bordet i SiWa-OS v2.9 til engelsk tekst.

- Hindre at data går tapt i første og siste skann av en wafer.

- Legge inn en indikator for dislokasjonstetthet.

- Finne ut av og ordne opp i feilen som gjør at SiWa-Scan stopper midt i skanningen når det kjøres flere analyser på rad.

Bibliografi

[1] Malmhaug, Jim-Kristian, *Videreutvikling av SiWa-Scan*. Trondheim, Norges teknisk-naturvitenskapelige universitet, 2013.

[2] SiliconImaging.com, *RGB Bayer"Color and MicroLenses*. Lastet ned fra http://www.siliconimaging.com/RGB 20Bayer.htm [21.10.2013].

[3] Wikipedia.org, *Wafer (electronics)*. Lastet ned fra http://en.wikipedia.org/wiki/Wafer_(electronics) [26.12.2013].

[4] Wikipedia.org, *Crystal twinning*. Lastet ned fra http://en.wikipedia.org/wiki/Crystal_twinning [5.1.2014].

[5] SINTEF Materialer og kjemi, *Silisium*. Lastet ned fra https://www.sintef.no/SINTEF-Materialer-og-kjemi/Tema/Materialer-og-nanotek/Silisium/ [5.1.2014].

[6] Hansen, Alexander Engum m.fl., *Utvikling av nytt instrument for karakterisering av silisiumwafere*. Trondheim, Høgskolen i Sør-Trøndelag, 2011.

[7] Karim Nice, Tracy V. Wilson og Gerald Gurevich, *How Digital Cameras Work*. howstuffworks.com, Lastet ned fra http://electronics.howstuffworks.com/cameras-photography/digital/digital-camera.htm [7.1.2014].

[8] Wikipedia.org, *Color filter array*. Lastet ned fra http://en.wikipedia.org/wiki/Color_filter_array [7.1.2014].

[9] Wikipedia.org, *Bayer filter*. Lastet ned fra http://en.wikipedia.org/wiki/Bayer_filter [7.1.2014].

[10] Cambridgeincolour.com, *Bit Depth Tutorial*. Lastet ned fra http://www.cambridgeincolour.com/tutorials/bit-depth.htm [7.1.2014].

[11] Wikipedia.org, *Charge-coupled device*. Lastet ned fra http://en.wikipedia.org/wiki/Charge-coupled_device [7.1.2014].

[12] National Instruments, *Configure Timed Loop Dialog Box*. Lastet ned fra
http://zone.ni.com/reference/en-XX/help/371361J-01/lvdialog/loop_configuration_db/ [8.1.2014].

[13] National Instruments, *Application Design Patterns: Producer/ Consumer*. Lastet ned fra
http://www.ni.com/white-paper/3023/en/ [8.1.2014]

[14] NetBeans.org, *Configuring NetBeans IDE 7.4 for C/C++/Fortran*. Lastet ned fra
https://netbeans.org/community/releases/74/cpp-setup-instructions.html# verifying [9.1.2014]

Figurer

1.1 Dislokasjoner oppstår som prikker i en silisiumwafer. Sett ifra mikroskop. 2
1.2 SiWa-Scan med tilhørende PC og brukergrensesnitt. 4
1.3 XY-bordet som brukes til å flytte waferprøven rundt i SiWa-Scan. 4

2.1 Kamerabilde av mørk bakgrunn med gammel linse. 7
2.2 Kamerabilde av mørk bakgrunn med ny linse. 8
2.3 Kameraet ser bare kanten av bildet når det er skråstilt 10 grader. 9
2.4 Bedre måling fra referansesensor under forstyrrelse. 10
2.5 Måling av ustabil laser etter at de nye optiske komponentene er satt inn. 12
2.6 Område 1 med mye dislokasjoner sett fra mikroskop. 17
2.7 Område 1 med mye dislokasjoner sett fra SiWa-Scan. 17
2.8 Område 2 med lite dislokasjoner sett fra mikroskop. 18
2.9 Område 2 med lite dislokasjoner sett fra SiWa-Scan. 18

3.1 Fargefilter, filtrering av fotoner(siliconimaging.com/RGB Bayer.htm). 20
3.2 Bayerfilter(en.wikipedia.org/wiki/Bayer_filter). 20
3.3 Veien til parametrene. 21

5.1 Skisse av datastruktur til SiWa-Scan. Datastrukturen er i hovedsak bygd opp med fem «timed while loops», hvor sløyfe 3 og 4 bruker en FIFO-kø (First-in-first-out). 32
5.2 Skisse av sløyfe 3 som henter inn bilder fra kameraet. 34
5.3 Skisse av analyseløkken, sløyfe 4. 35

6.1 Viser hvordan støy fra linsen opptrer i bildet (Ingen refleksjon fra wafer). 37
6.2 Tvilling vises som en lys strek i bildet. 38
6.3 Bilde av dislokasjoner. 39

6.4 Komplekst bilde. 40
6.5 Strekgjenkjenningsalgoritme laget i LabVIEW. 43

7.1 Bilde av overflaten på wafer «KMB8 185 etset». 45
7.2 Skann av wafer med programvare fra bacheloroppgaven. 47
7.3 Skann av wafer med programvare laget i dette prosjektet (NewSiWa-OS v2.9). 48
7.4 Nederste figur viser telling av dislokasjoner uten korreksjon for støy, mens øverste viser med korreksjon for støy. 50
7.5 Kjøretid målt fra sløyfe 3 og 4 under tre skann i x-retning. . . 51
7.6 Kjøretid målt fra sløyfe 3 og 4 under ett skann i x-retning. . . 52
7.7 Skann gjort med oppløsning på 300 µm. 54
7.8 Skann gjort med oppløsning på 50 µm. 55

Tillegg A

Vedleggsoversikt

Vedlegg 1 - Bacheloroppgaven vår 2011 (CD-ROM)

Vedlegg 2 - Prosjektrapport vår 2013 (CD-ROM)

Vedlegg 3 - Brukergrensesnitt bilde 1

Vedlegg 4 - Brukergrensesnitt bilde 2

Vedlegg 5 - Brukergrensesnitt bilde 3

Vedlegg 6 - Brukergrensesnitt bilde 4

Vedlegg 7 - Brukergrensesnitt bilde 5

Vedlegg 8 - NewSiWa-OS v2.9 (CD-ROM)

Vedlegg 9 - Brukermanual XY-bord (CD-ROM)

Vedlegg 10 - C-kode (CD-ROM)

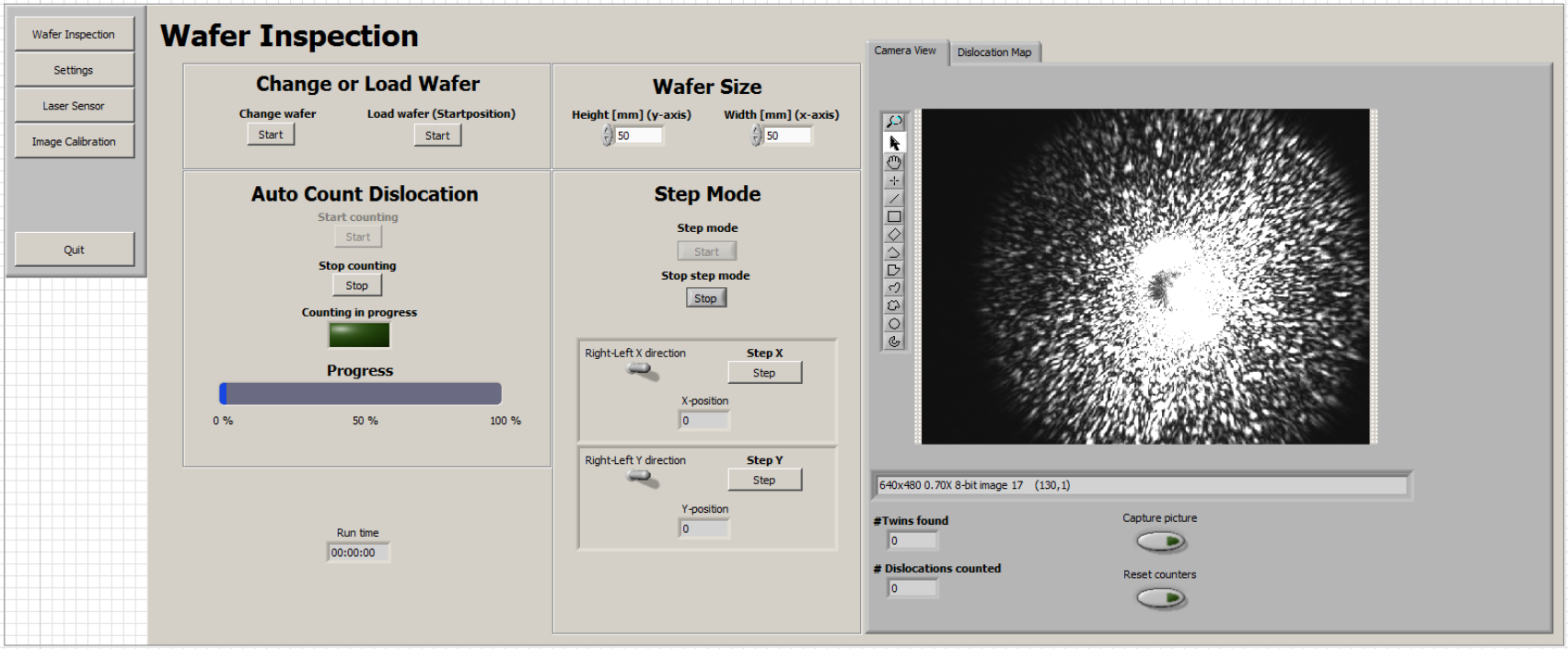
Vedlegg 11 - Laser Spec-sheet (CD-ROM)

Vedlegg 12 - LaTeX-mappe (CD-ROM)

Vedlegg 13 - Rapport fra masteroppgave (CD-ROM)

SINTEF

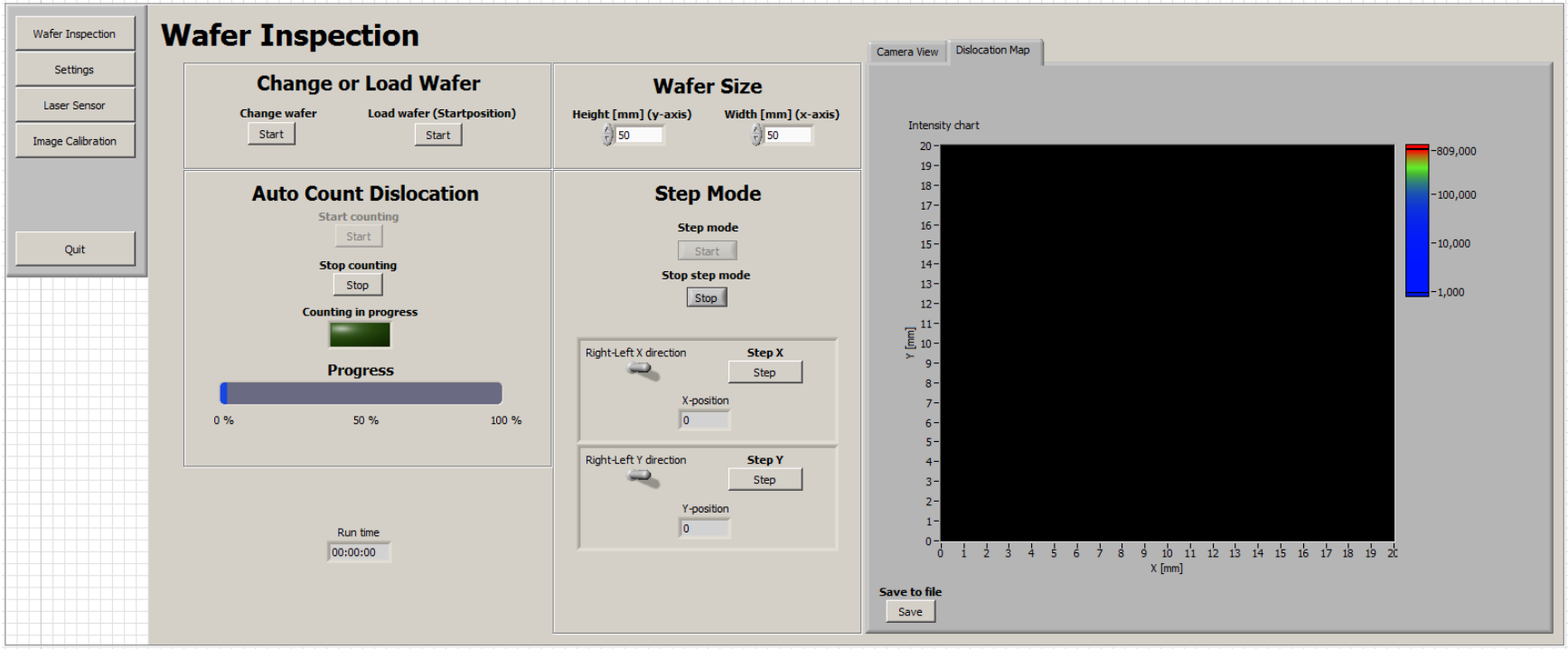
SiWa-scan



Vedlegg 3

SINTEF

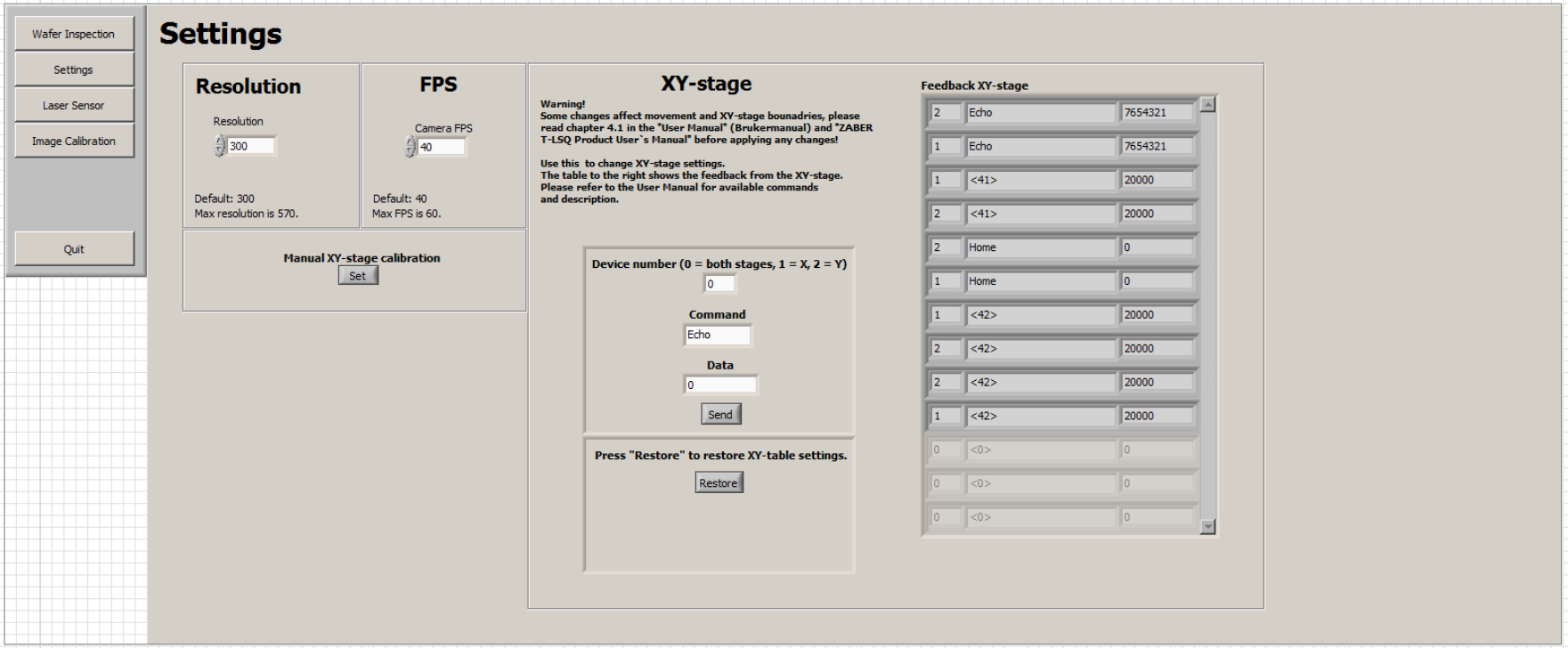
SiWa-scan



Vedlegg 4

SINTEF

SiWa-scan



Vedlegg 5

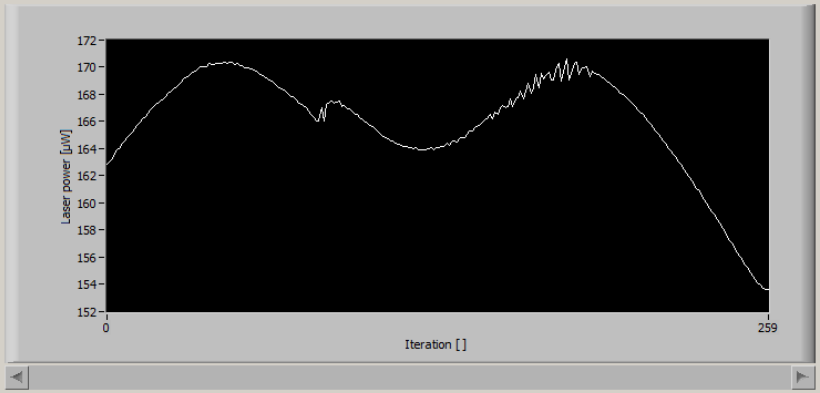
SINTEF

SiWa-scan

Wafer Inspection
Settings
Laser Sensor
Image Calibration
Quit

Laser Sensor

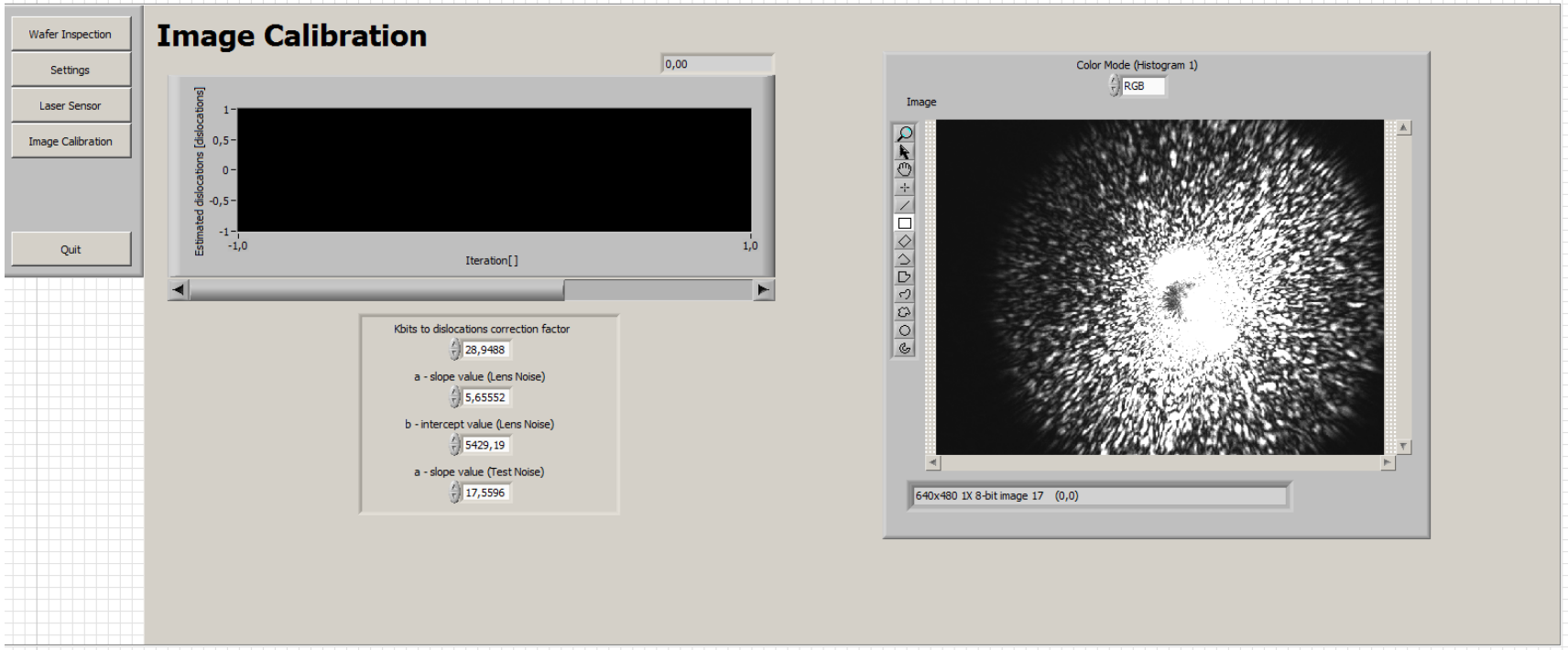
IDN String: Thorlabs,PM100USB,P2000779,
Sensor SN: 14-Mrz-2011
Sensor Name: S120C
Sensor Power [µW]: 153,603556E+0
Sensor Flags:
1 Power Sensor
2 Energy Sensor
4
8
16 Response settable
32 Wavelength settable
64 Tau (time constant) settable
128
256 With temperature sensor
Error log
Readout string: 153.6uW
Readout value: 153,603556E-6
New
Readout Config: POW
Averaging Rate: 10
Timeout [ms]: 1000
STOP



Vedlegg 6

SINTEF

SiWa-scan



Vedlegg 7