# Hoë Spoed Koppelvlak vir 'n Nuwe Generasie SUNSAT Mikrosatelliet.

H.L.Strydom

Desember 1999

Tesis ingelewer ter gedeeltelike voldoening aan die vereistes vir die graad van Magister in die Natuurwetenskappe in die Elektroniese Ingenieurswese aan die Universiteit van Stellenbosch.

Studieleier: Prof. P. J. Bakkes

# Verklaring

Ek, die ondergetekende, verklaar hiermee dat die werk in hierdie tesis vervat, my eie oorspronklike werk is, tensy anders genoem, en dat dit, tot my kennis, nog nie vantevore in die geheel of gedeeltelik by enige ander universiteit ter verkryging van 'n graad voorgelê is nie.

8/02/2000

Hendrik Strydom Datum

# Opsomming

'n Mikrosatelliet is 'n komplekse stelsel wat opgebou is uit 'n aantal substelsels. Kommunikasie koppelvlakke moet tussen hierdie substelsels bestaan vir die uitruil van data soos byvoorbeeld telemetrie en telebevel. Sulke koppelvlakke moet betroubaar en effektief wees, sodat data nie verlore gaan nie. Faling van 'n koppelvlak kan veroorsaak dat 'n korrek-funksionerende substelsel onbruikbaar word en sodoende die werking van die hele satelliet benadeel.

Die doel van hierdie werkstuk was die ondersoek na alternatiewe moontlikhede vir die implementering van 'n hoë spoed koppelvlak tussen die hoofkamera en die massageheue. Op die eerste generasie SUNSAT is die koppelvlak gerealiseer deur die verskillende kleure se beeldinligting parallel analoog tot by die massageheue oor te stuur. By die massageheue is dit versyfer en parallel ingeklok. Dit plaas beperkings op die stelsel en die nadele word in hierdie werkstuk bespreek. 'n Verskeidenheid parallelle en seriële tegnologieë is ondersoek en word bespreek.

LVDS is as die optimaalste tegnologie uitgewys en die koppelvlak is daar rondom ontwerp. Die tegnologie het die voordeel dat dit 'n seriële bus is wat gebruik maak van lae differensiële spannings, vir die transmissie van data. Sodoende word die kragverbruik en invloed van ruis geminimeer. Die hoë bandwydte van LVDS maak ook voorsiening vir 'n mate van foutprotokol.

As deel van die foutprotokol word 'n EDAC geïmplementeer om enkel bisfoute te korrigeer, maar het die nadeel dat 'n hoër bandwydte hiervoor benodig word. Vir alle ander moontlike foute word van 'n skuifregister met beperkte stoorarea gebruik gemaak om data te herversend. Dit is onprakties om 'n groter geheue aan die stuurkant te implementeer. Die foutprotokol is geïmplementeer en gesimuleer, en word in die werkstuk bespreek.

**Sleutel woorde:** mikrosatelliet, hoë spoed koppelvlak, hoofkamera, massageheue

# Synopsis

A micro-satellite is a complex system constructed of a number of subsystems. Communication interfaces must exist between these subsystems to exchange data, for example telemetric and telecommand. In order to receive all data, these interfaces must be efficient and reliable. Failure of an interface can result in an unuseable subsystem, which could affect the satellite.

The aim of this thesis was to investigate alternative possibilities for the implementation of a high-speed interface between the main camera and mass-memory. On the first generation SUNSAT the interface was realised by sending the analog image-information of the different colours, parallel to the mass-memory, where information is converted to digital data and latched into the main memory. This leads to limitations of the interface, and the disadvantages will be discussed. Various parallel and serial technologies are investigated and discussed.

LVDS was chosen as the most suitable technology and was used to design the interface. This technology has the advantage of implementing a serial bus, using low differential voltages for the transmission of data. This minimises the use of power and noise. The high bandwidth of LVDS also allows an errorprotocol to be implemented.

As part of the errorprotocol, an EDAC is implemented to correct single biterrors, but it needs a higher bandwidth, which is a disadvantage. A limited-queue can be used to resend data for any other errors, but it is impractical to implement a bigger memory on the sending side. The errorprotocol is implemented and simulated and is discussed in this thesis.

**Keywords**: micro-satellite, high-speed interface, main camera, mass-memory

# Bedankings

Ek wil graag die volgende bedank:

- Prof PJ Bakkes (studieleier) vir sy leiding.
- My ouers vir hul begrip en ondersteuning.
- Willemien vir haar liefde en motivering.
- My Hemelse Vader.

# Inhoudsopgawe

# Lys van Figure

# Lys van Tabelle

# Lys van Akronieme

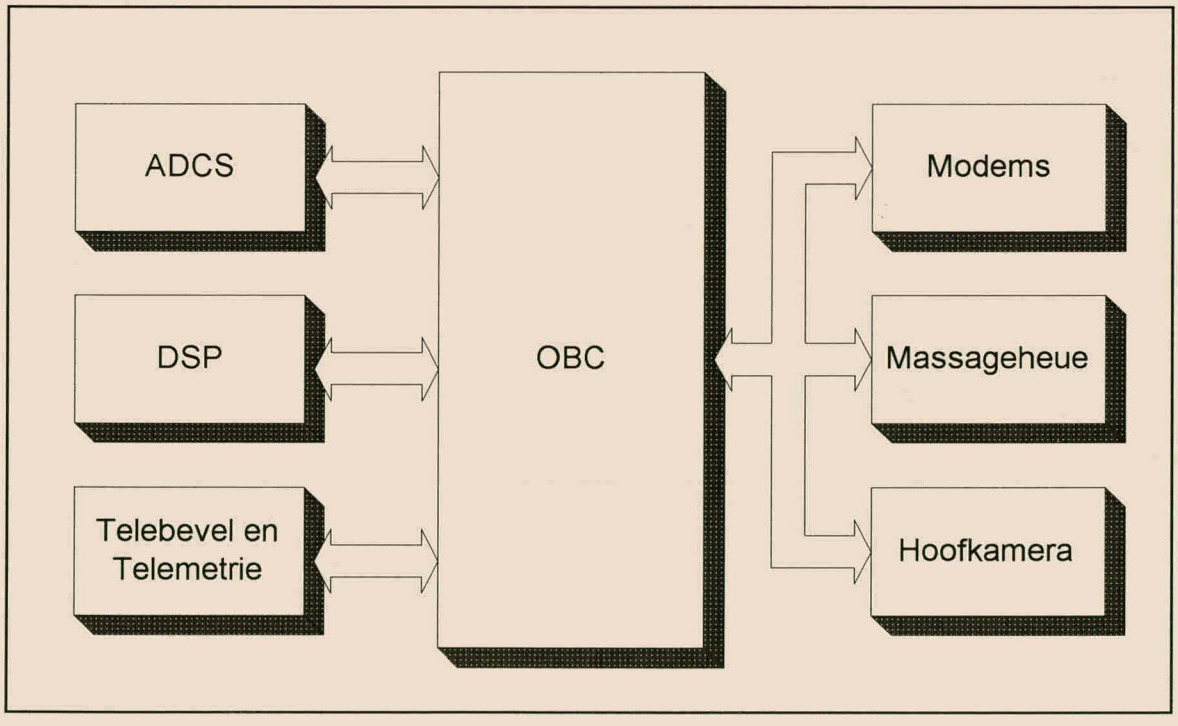
| | |
|---|---|
| A/D | Analoog na Digitaal |
| ADCS | "Attitude Determination and Control System" |
| AHDL | "Altera Hardware Description Language" |
| AMPP | "Altera Megafunction Partner Program" |
| ARGOS | "Advanced Research and Global Observation Satellite" |
| ASIC | "Application Specific Integrated Circuit" |
| bps | bisse per sekonde |
| CCD | "Charge Coupled Device" |
| CMOS | "Complementary Metal-Oxide Semiconductor" |
| CPLD | "Complex Programmable Logic Devices" |
| CRC | "Cyclic Redundancy Check" |
| DSP | Digitale Seinverwerking Prosessor |
| EAB | "Embedded Array Blocks" |
| EDAC | "Error Detection And Correction" |
| EDIF | "Electronic Design Interchange Format" |
| ESL | Elektroniese Stelsels Labarotorium |
| FIFO | "First-In-First-Out memory" |
| FLEX | "Flexible Logic MatriX" |
| FPDI | "Flat Panel Display Interface" |
| FPGA | "Field Programmable Gate Arrays" |
| GB | "GigaByte" |
| GPS | "Global Position System" |
| I/O | "Input-Output" |
| IC | "Integrated Circuit" |
| IOE | "I/O Element" |
| Kbps | Kilobisse per sekonde |
| LAB | "Logic Array blocks" |
| LE | "Logic Elements" |
| LPM | "Library of Parameterized modules" |
| LUT | "Look-Up Table" |
| LVDS | "Low-Voltage Differential Signalling" |

| | |
|---|---|
| MAX | "Multiple Array matriX" |
| MB/S | "Megabytes per second" |
| Mbps | Megabisse per sekonde |
| NiCd | Nikkel-Kadmium |
| OBC | "On-board Computer" |
| OBE | Ontvangbeheereenheid |
| PCI | "Peripheral Component Interconnect" |
| PECL | Pseudo-ECL |
| PLD | "Programmable Logic Devices" |
| PLL | "Phase Lock Loop" |
| PROM | "Programmable Read Only Memory" |
| RAM | "Random Access Memory" |
| SBE | Sendbeheereenheid |
| SCSI | "Small Computer System Interface" |
| SERDES | Serialiseerder-Deserialiseerder |
| SUNSAT | Stellenbosch Universiteit Satelliet |
| TTL | Transistor-Transistor Logika |
| UART | "Universal Asynchronous Receiver Transmitter" |
| USP | "Universal Serial Bus" |
| VHDC | "VHSIC Hardware Description Language" |

# 1. Inleiding

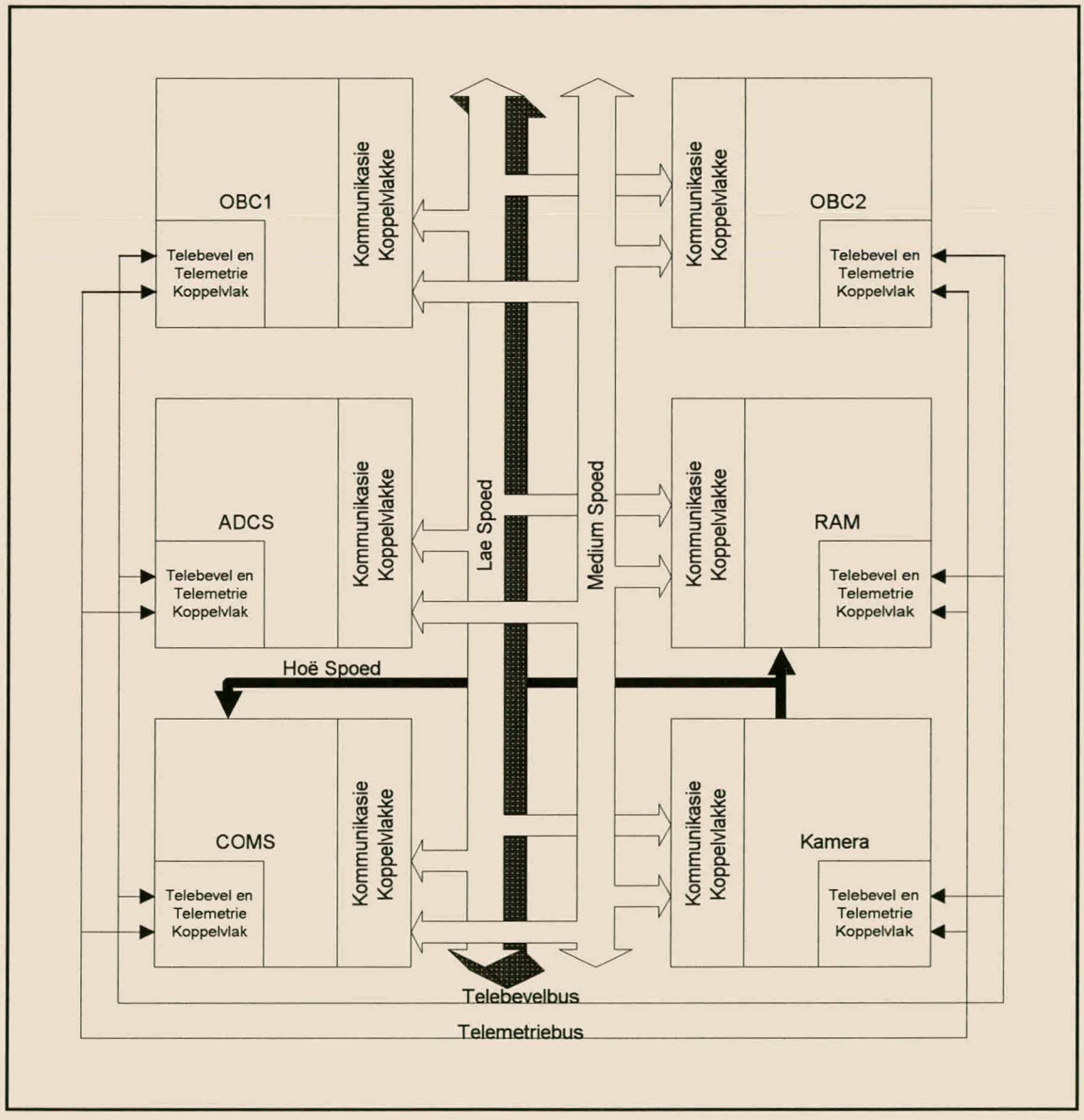
'n Mikrosatelliet is 'n uiter komplekse stelsel wat opgebou is uit 'n aantal substelsels. Die substelsels moet in staat wees om met mekaar te kommunikeer vir die uitruil van data. Hiervoor word 'n aantal betroubare en effektiewe kommunikasie-koppelvlakke benodig. Hierdie koppelvlakke is uniek, aangesien dit blootgestel is aan ekstreme kondisies in die ruimte, asook ander beperkende faktore wat deur die satellietplatvorm daarop geplaas word.

Op die eerste-generasie SUNSAT mikrosatelliet het die kommunikasie-koppelvlakke tussen substelsels onderling, en tussen substelsels en die aanboord rekenaars (OBC's), meestal bestaan uit 'n unieke parallelle verbinding wat ontwerp is rondom die spesifieke substelsels. Elke verbinding het dus sy eie unieke koppelvlak gehad en het sy eie drywer in die OBC benodig. Toevoeging van nuwe substelsels was moeilik en die komplekse kabelharnas wat hiermee gepaard gegaan het, was area-intensief. Figuur 1.1 toon 'n blokdiagram van die huidige kommunikasie-koppelvlakke tussen substelsels.




Figuur 1.1 : Blokdiagram van huidige kommunikasie-koppelvlakke tussen substelsels

Vir die volgende generasie SUNSAT word beplan om vir kommunikasie tussen substelsels, drie netwerkstelsels te implementeer, naamlik 'n lae spoed-, 'n medium spoed- en 'n hoë spoed netwerk. 'n Gestandaardiseerde koppelvlak na elke netwerk sal geïmplementeer word, met standaard drywers in die OBC's. Dit sal toevoeging van bykomende stelsels baie vereenvoudig. Figuur 1.2 toon 'n blokdiagram van die beplande netwerkstelsel van die volgende generasie SUNSAT.




Figuur 1.2 : Blokdiagram van volgende generasie SUNSAT se netwerkstelsel.

Die hoofvereistes van die lae spoed netwerk, is dat dit baie betroubaar moet wees. Die netwerk sal min data op 'n keer oordra, maar dit is van kardinale belang dat die data korrek oorgedra moet word. Om dit te verseker, is oortolligheid, 'n betroubare

verbinding en 'n effektiewe protokol nodig. Twee afsonderlike netwerke sal vir algemene kommunikasie en vir telemetrie- en telebevelinligting geïmplementeer word.

Die medium spoed netwerk sal groter volumes data hanteer teen 'n hoër tempo. Korrektheid van die data is net so belangrik. Hierdie netwerk implementeer nie dieselfde mate van oortolligheid as die lae spoed netwerk nie, maar 'n effektiewe protokol word gebruik om te verseker dat die data korrek is.

Die hoë spoed netwerk word benodig waar groot volumes data teen 'n baie hoë tempo oorgedra moet word. Daar bestaan op hierdie stadium egter nie baie sulke verbindings nie. Tipiese data wat met so 'n netwerk oorgedra sal word, is beelddata vanaf die hoofkamera na die massageheue, en vanaf die massageheue na die hoë spoed modems.

Die doel met hierdie tesis is die ondersoek na moontlike tegnologieë wat vir die hoë spoed netwerk gebruik kan word, en die ontwerp en implementering daarvan. Die ondersoek en ontwerp word op spesifiek die verbinding tussen die hoofkamera en die massageheue gebaseer.

## 1.1 Beskrywing van hoofstukke wat volg

**Hoofstuk 2**

In hoofstuk 2 word die agtergrond van die SUNSAT projek bespreek. 'n Beskrywing van die hoofkamera word gegee. Laastens word die beplande eienskappe van die volgende generasie kamera genoem.

**Hoofstuk 3**

In hoofstuk 3 word die beperkings en eienskappe van die hoë spoed netwerk bespreek. Die transmissiemediums, tipe verbindings, en netwerk topologieë word ondersoek. Daarna word verskillende moontlike tegnologieë evalueer volgens die bespreekte beperkings en eienskappe. Die mees geskikte tegnologie word uitgewys en in meer diepte bespreek.

**Hoofstuk 4**

Hoofstuk 4 handel oor die stelselontwerp van die hoë spoed koppelvlak. Die ontwerp word op die verbinding tussen die hoofkamera en die massageheue gebaseer en word rondom die verkose tegnologie ontwikkel. Aannames ten opsigte van die koppelvlak is gemaak en daar rondom is ontwerp. 'n Aantal moontlike opstellings word gegee en die voordele en nadele van elk word bespreek.

**Hoofstuk 5**

Hoofstuk 5 bespreek die apparatuurontwerp van die stelsel wat in hoofstuk 4 voorgestel is. Aanpassings is gemaak sodat die stelsel deeglik getoets en geëvalueer kan word. Die komponente wat gebruik word, en die keuse daarvan, word bespreek.

**Hoofstuk 6**

In hoofstuk 6 word die VHDL-ontwerp van die stelsel bespreek. Opsies wat bestaan met die gebruik van voorafontwikkelde VHDL-komponente word bespreek en die voordele, nadele en werkverrigting van elk word uitgelig.

**Hoofstuk 7**

Die gevolgtrekkings wat aan die einde van die werkstuk gemaak is, word in hierdie hoofstuk bespreek. Die hoofstuk sluit af met sekere aanbevelings wat ten opsigte van die ontwikkelde koppelvlak, asook volgende koppelvlakke, gemaak kan word.

# 2. Agtergrond oor die SUNSAT projek

Die SUNSAT (Stellenbosch UNiversiteit SATelliet) projek is in 1992 begin deur die departement Elektriese en Elektroniese Ingenieurswese, met die doel om nagraadse studente praktiese ondervinding te bied in die elektronika. Die ESL (Elektroniese Stelsels Laboratorium) is gestig waar 15 studente met die projek begin het, en tot op datum, het meer as 50 studente al daaraan gewerk.

SUNSAT is op 23 Februarie 1999 om 10:29 UTC op 'n Boeing-Delta II vuurpyl vanaf Vandenberg-lugmagbasis aan die weskus van Kalifornië, VSA, gelanseer en is Suid-Afrika (en Afrika) se eerste mikrosatelliet. Die lansering is gedeel met ARGOS (Advanced Research and Global Observation Satellite) van die Amerikaanse lugmag en Ørsted van Denemarke. ARGOS het die hoofloonvrag gevorm en weeg ongeveer 2,7 ton.

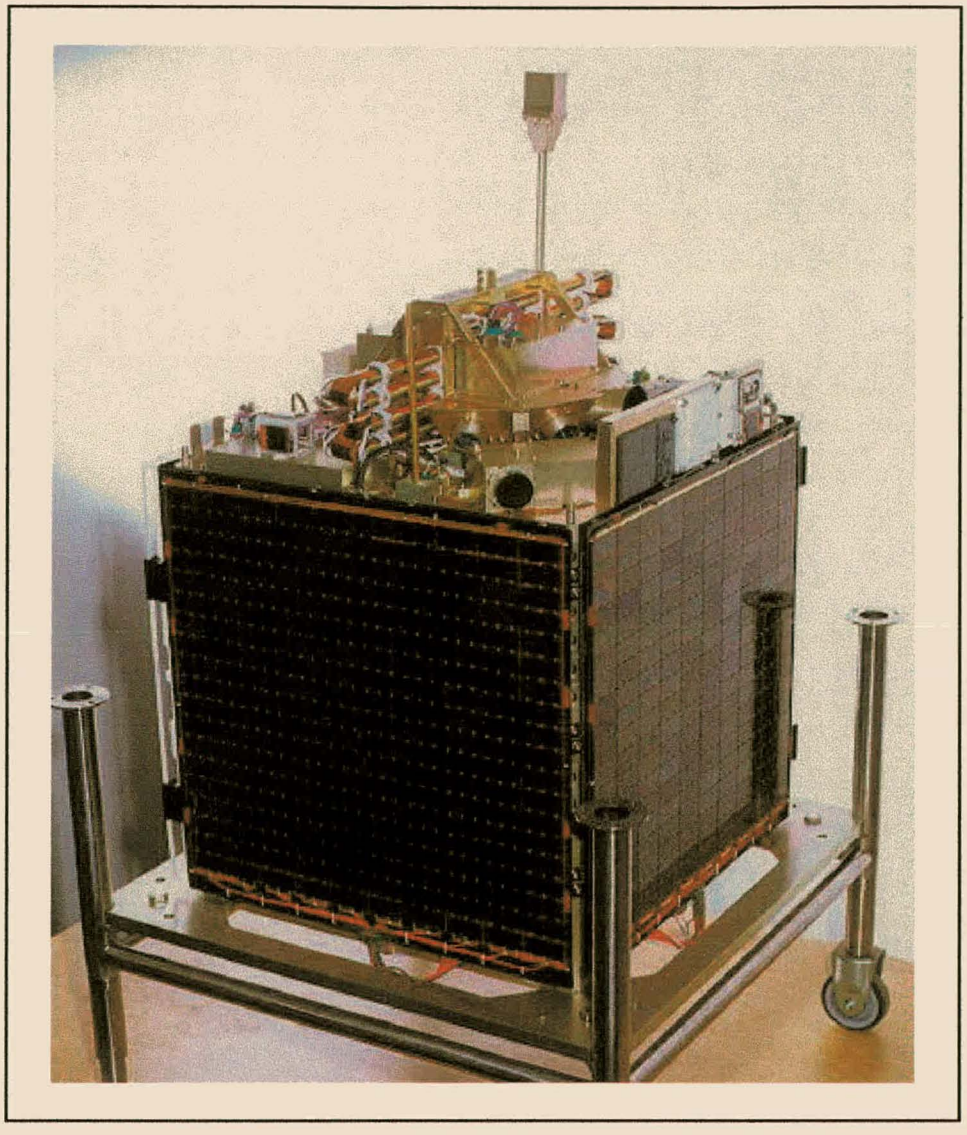
## 2.1 Wentelbaan en Struktuur

### Wentelbaan

SUNSAT is in 'n effens ellipties polêre lae-aarde wentelbaan, op 'n hoogte wat wissel tussen 520 km en 850 km en beweeg teen 'n snelheid van 7,224 km/s. Die wentelbaan is nie sonsinkroon nie en dryf ongeveer 0,8° per dag ten opsigte van die ewenaar-assestelsel. As gevolg van die rotasie van die aarde word die hele oppervlak gedek. Die periode van een omwenteling is naastenby 100 minute en dus is SUNSAT vir slegs 10 minute per keer sigbaar vanaf die grondstasie.

### Struktuur

SUNSAT is 'n kubusvormige mikrosatelliet met dimensies 45 × 45 × 60 cm en weeg 63,4 kg. Dit het 'n modulêre struktuur wat bestaan uit 11 aluminium-laaie wat aan mekaar vasgebout is. Sonpanele is aan die vier sykante geheg. Op die topplaat is verskeie oriëntasie-sensors geheg, asook 'n ontplooibare gravitasie-arm met tipmassa. Die vashegtingsring van die lanseerder is aan die onderkant vasgeheg. 'n Foto van SUNSAT word in figuur 2.1 getoon.



Figuur 2.1 : Vlugmodel van SUNSAT

## 2.2 Hoofloonvragte

### Hoë resolusie kamera

Die hoofkamera is in staat om stereoskopiese beelde van die aarde te neem met resolusie van ongeveer 15m vanaf 'n hoogte van 800 km. Dit beteken dat voorwerpe groter as 15m as 'n spikkel ("pixel") op die foto verteenwoordig word. Die kamera kan drie spektrale bande onderskei, naamlik rooi, naby-infrarooi en groen. Daar word veral belang gestel in beelde van landbou- en bosbougebiede. Die spektrale reflektiwiteit van plante is veral hoog by dié drie kleure, en daarom is hierdie kleure gekies.

**Kommunikasie**

Die kommunikasiestelsel bestaan uit verskeie radiosenders en -ontvangers wat soos volg daar uitsien:

- VHF (145 MHz) : Twee ontvangers en twee senders wat teen 1 W of 5 W kan uitsaai met 'n maksimum oordragtempo van 1200 bps
- UHV (436 MHz) : Twee ontvangers en twee senders wat teen 2 W of 10 W kan uitsaai met 'n maksimum oordragtempo van 9600 bps
- L-band (1,265 GHz) : Ontvanger met oordragtempo van 2 Mbps
- S-band : Sender wat teen 5 W gedryf kan word en met oordragtempo van 40 Mbps.

## 2.3 Ondersteuningstelsels

**Algemene beheer**

Twee aanboord rekenaars is verantwoordelik vir die algemene beheer van die satelliet. Die primêre rekenaar (OBC1) is 'n 80188EC verwerker en verrig die belangrikste verwerkingsfunksies, prosesskedulering, en is ook in staat om telebevellyne te beheer. Die sekondêre rekenaar (OBC2) is 'n 80386EX verwerker en dien as ondersteuning vir OBC1.

**Massageheue**

Statiese geheue met 64 MB kapasiteit dien as massastoor-area vir beelddata afkomstig vanaf die hoofkamera, en telemetriedata afkomstig van sensore op verskeie dele van die satelliet. Die data word hierop gestoor en kan later weer na die grondstasie afgelaai word.

**Telemetrie en Telebevel**

Met die telemetriestelsel word die toestand van die satelliet gerapporteer. Satellietdata word versamel en versyfer, en kan of in die massageheue gestoor word, of deur die radios na die grondstasie uitgesaai word. Die telebevelstelsel word gebruik om bevele vanaf die grondstasie uit te voer. Verskillende skakelaars, verwerkers, radios en sensore kan met hierdie stelsel aan- of afgeskakel word.

**Oriëntasie-beheer**

Die ADCS ("Attitude Determination and Control") stelsel is verantwoordelik vir oriëntasie afskatting en beheer van die satelliet. Dit bestaan uit vyf tipes sensore en twee tipes aktueerders. Passiewe stabilisasie word deur die 2,2 m gravitasie-arm verskaf. Die vyf tipes sensore is as volg:

- 'n Drie-as magnetometer bepaal die oriëntasie vanaf die aarde se magneetveld.
- 'n Sonsensor bepaal die posisie van die son.
- Twee horisonsensors bepaal die posisie van die aarde se horison.
- Twee sterkameras neem beelde van sterkonstellasies om oriëntasie daarvolgens te bereken.
- Ses sonselle is gemonteer op die satelliet en dien as rowwe sonsensore.

Ses magneetspoele word gebruik om die satelliet met behulp van die aarde se magneetveld te oriënteer. Vir fyner oriëntasie word van vier reaksiewiele gebruik gemaak.

**Kragstelsel**

Kragvoorsiening van die satelliet is afkomstig van twee batterypakke. Elke pak bestaan uit vyf 1,2 V NiCd selle en beskik oor 'n kapasiteit van 6 Ah. Die vier sonpanele is verantwoordelik vir die herlaai van die batterypakke. Elke paneel is in staat om 'n maksimum van 30 W drywing te lewer en die opgewekte stroom word direk terug gevoer na die batterypakke.

**Eksperimente**

In ruil vir die lansering word twee eksperimente van NASA saamgevlieg. Die eerste is 'n baie akkurate GPS-ontvanger wat ook deur die satelliet vir posisiebepaling gebruik word. Die tweede is 'n stel laser retro-reflektors geheg rondom die tipmassa en dié word gebruik vir posisionele afskatting vanaf die aarde. Drie skooleksperimente word ook op die satelliet gevind.

**PAL televisiekamera**

'n PAL televisiekamera is aan die onderste rak gemonteer. Dit neem intyds beelde van die aarde en stuur dit oor die S-band na die grondstasie. Hierdie inligting kan egter nie gestoor word nie, dus kan slegs beelde van Suid-Afrika gesien word. [1][2][3]

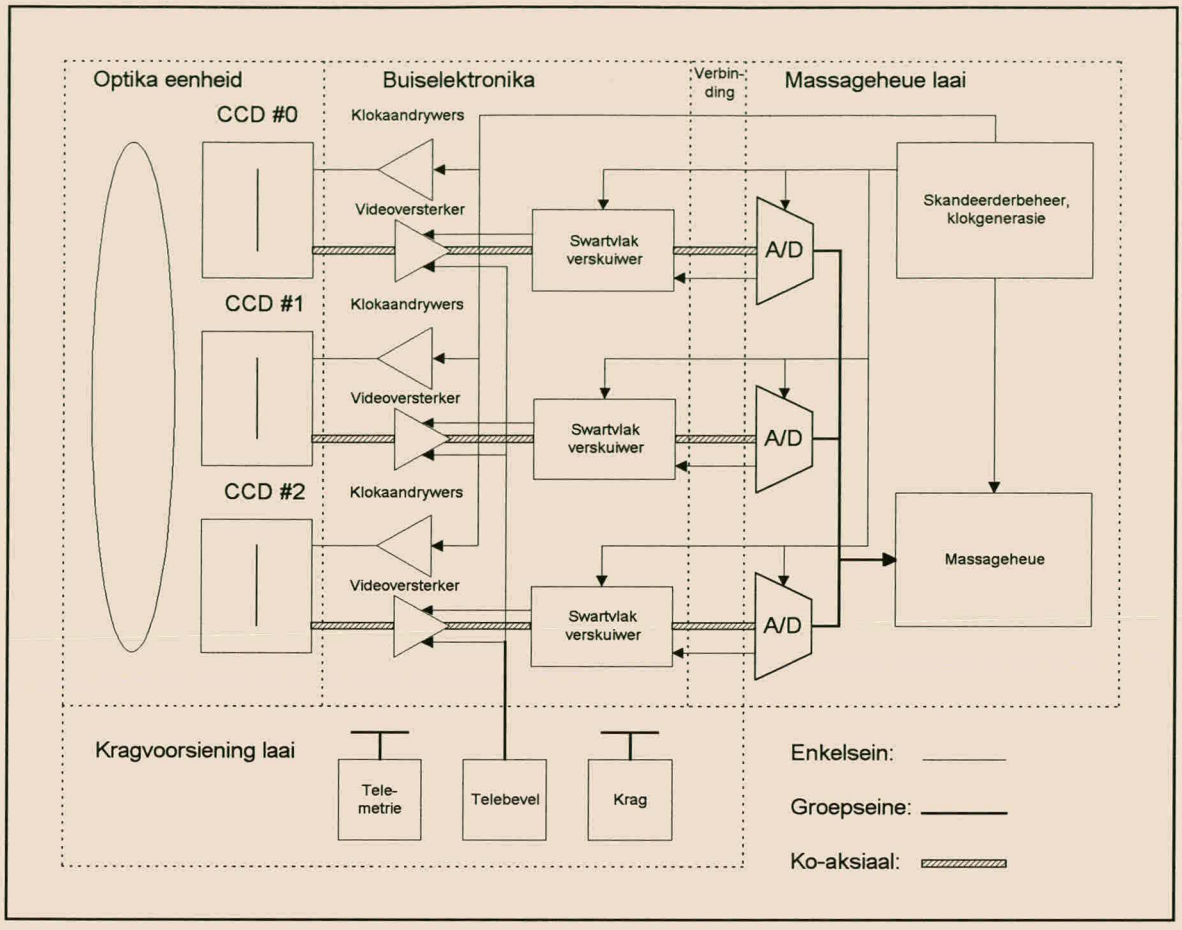
## 2.4 Die Hoofkamera

Die hoofkamera is geleë in die onderste rak van die satelliet. Dit bestaan uit elektronika en optika wat binne 'n buis gehuisves is. Die buis is diagonaal in die rak gemonteer, met 'n 45° spieël by die een end, en die optika en elektronika by die ander end daarvan. Die spieël projekteer weerkaatste lig vanaf die aarde deur 'n opening in die bodemplaat op die optika. Die buis kan deur stappermotors geroteer word, wat die kamera in staat stel om stereoskopiese beelde van die aarde te neem.

Die optika bestaan uit 'n penta-prisma met hoofsaaklik dichroïse filters, wat die inkomende lig projekteer op drie lyn CCD's. Elke CCD neem 'n ander spektrale band waar. Die drie spektrale bande wat waargeneem kan word, is rooi, naby-infrarooi en groen. 'n Beeld word nie as geheel geneem nie, maar slegs een lyn op 'n keer. Die kamera werk dus baie soos 'n faksmasjien, deurdat 'n beeld saamgestel word deur die gebied lyn vir lyn te skandeer.

Die CCD's wat gebruik is, is die TC104 lineêre CCD vervaardig deur Texas Instruments. Elke CCD besit 3456 aktiewe spikkels en funksioneer by golflengtes tussen 400 nm en 1000 nm. Die CCD benodig vier klokseine naamlik "Reset clock" (RCK), "Transport clock" (TCK), "Transfer clock" (XCK) en "White Reference clock" (WRCK). Analoogpulse word teen die frekwensie van RCK by die uittree uitgeklok en stel die ligintensiteit van elke spikkel in die lyn voor.

Figuur 2.2 toon 'n blokdiagram van die kamera-elektronika en die ondersteunende stelsels daarvan. Die datapaaie van die drie kleure is apart geïmplementeer om betroubaarheid te verhoog. Sodoende sal die kamera nie onbruikbaar wees as een van die datapaaie faal nie.




Figuur 2.2 : Blokdiagram van kamera- en ondersteunende elektronika [3]

As gevolg van beperkte ruimte binne die kamerabuis, moes die meeste van die elektronika geskei word van die CCD's, wat aan die optika vasgeheg is. Die enigste elektronika binne die buis was die CCD's, videosein uittree-versterkers met swartvlak verskuiwers, en temperatuursensors vir elke CCD. Die lang verbindings het die ontwerp van die klokdrywers bemoeilik as gevolg van die induktansie by die hoë klokfrekwensies.

**Kamerabeheerder**

Die kamerabeheerder genereer die nodige klokseine na die CCD's en die ondersteunende elektronika. Om vierkantige spikkels te behou vanaf verskillende hoogtes, moet die frekwensie van die klokseine verstelbaar wees. 'n Spikkelfrekwensie kan wissel vanaf 1,39 MHz by hoogte van 850 km tot 2,94 MHz by hoogte van 450 km. Tensy anders gemeld, word 'n spikkelfrekwensie van 2 MHz aanvaar. Die beheerder is ook in staat om intydse telemetriedata soos posisie, oriëntasie en tyd, in die data van 'n lyn in te voeg.

Die data-oordragtempo word soos volg bereken:

$$\begin{aligned}\text{Data oordragtempo} &= \text{Aantal kleure} \times \text{Bisse per kleur} \times \text{Spikkelfrekwensie} \\ &= 3 \times 8 \times 2\,\text{MHz} \\ &= 48\,\text{Mbps}\end{aligned}$$

**Analoë videoseinverwerker**

Die analoë uittree van elke CCD word deur 'n analoë videoversterker versterk. Die aanwins van die versterker is vanaf telebevel beheerbaar. Die spanningswaai word ook daardeur beperk vir die beskerming van die analoog-na-syfer-omsetter. 'n Swartvlak verskuiwer met die analoë videosein, die swart spanningsvlak van die analoog-na-syfer-omsetter, en 'n digitale kloksein vanaf die kamerabeheerder, as intree, lewer as uittree 'n verwysing vir die videoversterker.

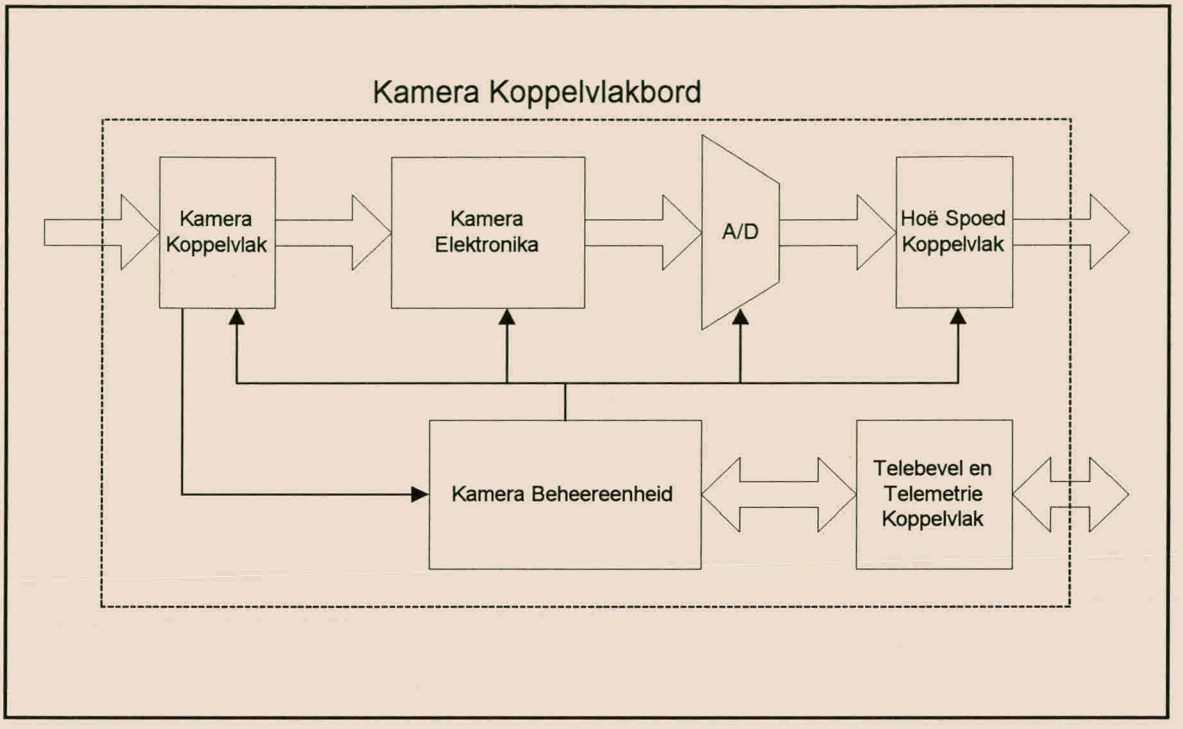
**Analoog-na-syfer-omsetter**

'n Analoog-na-syfer-omsetter word gebruik om die analoë videosein na digitale data om te skakel wat in die massageheue gestoor kan word. Plessey Semiconductors se SP973T8 A/D omsetter is hiervoor gebruik. Dit is 'n 5 V "Flash" 8-bis omsetter wat van bipolêre transistors gebruik maak. Die A/D is op die massageheuelaai geleë en die analoë videosein word deur ko-aksiale verbindings daaraan voorsien. [3]

## 2.5 Die Volgende Generasie Kamera

Die grootste verandering vir die volgende generasie kamera is dat dit as 'n funksionele eenheid ontwikkel sal word. Dit sal toetsing en integrasie daarvan baie vereenvoudig. 'n Kamera-koppelvlakbord met 'n hoë spoed digitale dataskakel sal deel vorm van die funksionele eenheid, en dit sal slegs van krag en telebeveldata voorsien moet word.

Al die nodige klokseine sal deur die eenheid self gegenereer word en saam met die beelddata uitgestuur word. Die eenheid sal ook verantwoordelik wees vir al die analoog-na-syfer-omskakelings, asook die byvoeging van ekstra data, soos telemetriedata, by die beelddata. 'n Digitale datastroom word dus as uittree verkry wat die beelddata en bygevoegde data bevat. 'n Blokdiagram van so 'n koppelvlakbord word in figuur 2.3 getoon.




Figuur 2.3 : Blokdiagram van beplande kamera-koppelvlakbord

Veranderinge as gevolg van beter tegnologie beskikbaar sal ook gemaak word. So byvoorbeeld sal beter CCD's gebruik word met tot 7000 spikkels in 'n lyn. Groter beelde met hoër resolusie kan dus verkry word. Verbetering in die optika sal ook tot gevolg hê dat meer spektrale bande waargeneem kan word. Waarneming van tot ag spektrale bande word vir die toekoms beplan.

Die belangrikste gevolg van bogenoemde veranderings is dat die datavolume aansienlik toeneem. Nie net word groter stoorarea benodig nie, maar ook hoër data-oordragtempo's. Vir die volgende generasie massageheues bestaan 1 GB 3,3 V "Flash" geheue-ontwerpe reeds. Die doel van hierdie tesis is dan die ondersoek na moontlike tegnologieë en die implementering daarvan, om as hoë spoed verbinding te gebruik. [2][4]

# 3. Tegnologie ondersoek

Soos reeds in hoofstuk 1 genoem, bestaan die hoë spoed verbindings op die SUNSAT I uit direk gekoppelde analoë of digitale parallelle verbindings. Dit is nie 'n ideale opstelling nie, as gevolg van die area wat dit in beslag neem en die kompleksiteit van 'n unieke koppelvlak vir elke eenheid. 'n Baie beter en vereenvoudigde oplossing sal wees om 'n hoë spoed netwerk daar te stel met 'n goed gedefinieerde koppelvlak en protokol wat maklik uitgebrei kan word.

In hierdie hoofstuk gaan na verskillende tegnologieë gekyk word wat as verbinding vir so 'n netwerk kan dien. Die hoofstuk begin met 'n bespreking van die doel van die tesis. Daarna gaan gekyk word na die beperkings en eienskappe van 'n hoë spoed netwerk. Verskillende tipe verbindings, transmissiemediums, en netwerk topologieë word ondersoek. Die tegnologieë word evalueer volgens die eienskappe waaraan die netwerk moet voldoen en die hoofstuk sluit af met 'n detail-ondersoek van die verkose tegnologie.

## 3.1 Doel

Die doel van hierdie tesis is die ontwerp van 'n hoë spoed netwerk, met 'n gewaarborgde data-oordragtempo vir gebruik op die volgende generasie SUNSAT. Daar word spesifiek gekyk na wat die mees optimale tegnologie en opstelling sal wees vir die verbinding tussen die hoofkamera en die massageheue. Faktore wat bydra tot die groot volume data wat deur die kamera gegenereer word, is die aantal spikkels per lyn, die monsterfrekwensie van 'n lyn en die aantal kleure. 'n Hoë bandwydte verbinding vir die oordrag van die data na die massageheue word benodig, aangesien die kamera oor geen stoorarea beskik nie.

Die medium spoed netwerk is geskik vir die meeste substelsels wat groter volumes data wil versend. Daar is slegs enkele substelsels wat baie groot volumes data teen baie hoë oordragtempo's moet uitruil, en almal is punt-tot-punt verbindings tussen twee spesifieke substelsels. 'n Algemene hoë spoed netwerk word dus nie benodig nie. Die moontlikheid van 'n algemene hoë spoed netwerk word egter in gedagte gehou met die ondersoek na die mees geskikte tegnologie.

## 3.2 Beperkings

'n Hele aantal beperkende faktore moet met die ontwerp van 'n stelsel op 'n mikrosatelliet ingedagte gehou word. Beide die omgewing waarbinne dit funksioneer en die vereistes ten opsigte van die satellietplatvorm, speel 'n belangrike rol in die keuse van tegnologieë en implementasie.

Die belangrikste faktore wat deur die werksomgewing op die ontwerp geplaas word, is die volgende:

- Vakuum
- Gewigloosheid
- Temperatuur
- Bestraling
- Meteoriete en partikels
- Elektromagnetiese versoenbaarheid
- Lansering.

Die volgende is die hoofvereistes wat deur die satellietplatvorm op die ontwerp geplaas word:

- Min area in beslag neem
- Nie baie bydra tot die totale massa nie
- Lae kragverbruik handhaaf
- Groot temperatuurwisseling kan handhaaf
- Koste moet laag wees
- Komponente moet beskikbaar wees.

## 3.3 Eienskappe van die hoë spoed netwerk

Op die huidige en die volgende generasie SUNSAT is daar nog nie die behoefte aan 'n algemene hoë spoed netwerk nie. Vir die oordrag van kritieke data, kan die lae spoed netwerk gebruik word en vir groter volumes data, die medium spoed netwerk. Die hoë spoed netwerk sal dus slegs vir enkele verbindings benodig word waar baie groot volumes data teen baie hoë tempo oorgedra moet word.

Tipiese verbindings wat van so 'n hoë spoed netwerk gebruik sal maak, is die oordrag van beelddata vanaf die hoofkamera na die massageheue, en vanaf die massageheue na die hoë spoed modems. Dit is ook belangrik om daarop te let dat dit altyd net punt-tot-punt verbindings is en dat dit 'n hoë oordragtempo gewoonlik ook net in een rigting benodig.

Die netwerk moet nie net in die ekstreme omgewing van die ruimte korrek funksioneer nie, maar moet ook voldoen aan die algemene vereistes wat deur die satellietplatvorm daarop geplaas word. Die netwerk is ook onderhewig aan vereistes ten opsigte van spesifikasies waaraan dit moet voldoen, soos byvoorbeeld die oordragtempo daarvan.

Die kenmerke van die hoë spoed netwerk is soos volg geïdentifiseer:

**1. Punt-tot-punt verbinding**

Op die huidige SUNSAT, en op die volgende generasie SUNSAT, is daar nie baie substelsels wat teen 'n hoë tempo met mekaar moet kommunikeer nie. Sulke hoë spoed verbindings is gewoonlik baie spesifiek en slegs tussen spesifieke substelsels. Daar bestaan dus geen nut vir 'n algemene hoë spoed netwerk, met komplekse adressering en fouthanteringsprotokolle nie. 'n Beter opsie sal wees om 'n hoë spoed punt-tot-punt verbinding met 'n mate van fouthanteringsprotokol te ontwikkel, waar die grootste deel van die bandwydte vir die oordrag van data gebruik kan word.

**2. Hoë spoed slegs in een rigting**

Soos reeds genoem, word hoë oordragtempo's meestal benodig tussen twee spesifieke eenhede en vir die oordrag van spesifieke data. In al die gevalle is dit slegs nodig om 'n hoë oordragtempo in een rigting te implementeer. Enkele beheerlyne word egter terugwaarts benodig om die sender van moontlike foute in kennis te stel.

**3. Eenvoudige opstelling**

'n Eenvoudige opstelling word benodig wat die huidige parallelle opstelling kan vervang. Vir die substelsels moet dit lyk asof hulle steeds direk aan mekaar geskakel is deur middel van die parallelle koppelvlak. Geen ekstra las soos fouthantering moet op die substelsels geplaas word nie.

**4. Min plek in beslag neem**

Alhoewel baie hoër oordragtempo's verkry kan word deur van 'n parallelle opstelling gebruik te maak, het dit die nadeel dat dit meer plek in beslag neem. Met 'n seriële verbinding is dit moeiliker om sulke hoë bistempo's te bereik, maar dit minimeer die area wat in beslag geneem word, en is meer bestand teen omgewingsfaktore.

**5. Hoë data-oordragtempo**

Die netwerk moet voorsiening maak vir 'n minimum oordragtempo van 80 Mbps. Dit sal genoeg bandwydte laat om vyf spektrale kleure, gemonster teen 2 MHz, oor te dra. Indien hoër bandwydtes beskikbaar is, kan protokoldata ook bygevoeg word.

**6. Hoë akkuraatheid**

Dit is nie moontlik om die data weer vanaf die sender aan te vra nie, omdat dit nie aan die stuurkant gestoor word nie. Die netwerk moet dus voorsiening maak dat data nie verlore gaan indien 'n fout voorkom nie. Alhoewel enkele foute in die data nie kritiek is nie, verswak dit die kwaliteit van die beeld.

Die doel van die projek is dus om 'n betroubare eenrigting hoë spoed punt-tot-punt verbinding te ontwikkel, wat data teen 80 Mbps kan oordra. Die verbinding moet 'n eenvoudige koppelvlak na die substelsels implementeer en so min as moontlik plek in beslag neem. Die verbinding is verantwoordelik om te verseker dat die data korrek oorgedra word.

## 3.4 Tipe verbindings

Verbindings kan in twee hoofklasse ingedeel word naamlik parallelle en seriële verbindings. Faktore van belang in die keuse van watter tipe verbinding gebruik moet word, is die volgende:

- Oordragtempo
- Afstand
- Area wat in beslag geneem word
- Kompleksiteit van koppelvlak
- Koste daaraan verbonde.

**Seriële verbindings**

Met die tipe verbinding word die databisse serieel agter mekaar oor die verbinding gestuur. Min area word in beslag geneem, omdat gewoonlik net twee geleiers benodig word. Dit veroorsaak ook dat die fisiese koppelvlak nie baie kompleks is nie en dat die koste daarvan laag is.

**Parallelle verbindings**

Met 'n parallelle verbinding word die databisse parallel met mekaar gestuur. Veel meer area word in beslag geneem omdat vir elke bis ten minste een geleier benodig word. Om die rede is die fisiese koppelvlak en koste ook hoër. Die voordeel van parallelle verbindings is dat baie hoër oordragtempo's behaal kan word. 'n Belangrike nadeel vanuit 'n satelliet ontwerpsoogpunt, is die groter moontlikheid van foutiewe data oordrag as gevolg van omgewingsfaktore, omdat meer area blootgestel word.

## 3.5 Transmissiemedium

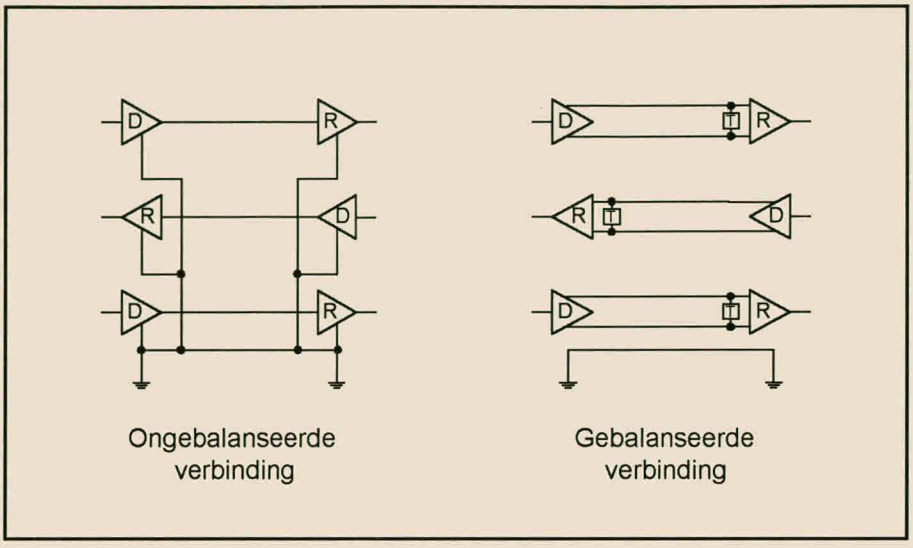
Die medium wat gebruik word vir die fisiese verbinding speel 'n uiters belangrike rol in die werkverrigting van die verbinding. Belangrike faktore wat in gedagte gehou moet word met die keuse van 'n transmissiemedium is die volgende:

- Seinverswakking deur die medium
- Beperkte bandwydte van die medium
- Vertragingsdistorsie van die medium
- Ruisverwerping van die medium.

Die verskillende transmissiemediums gaan ten opsigte van die volgende vyf faktore evalueer word:

- Maksimum data oordragtempo
- Maksimum kabellengte
- Invloed van ruis
- Elektromagnetiese uitstraling
- Koste.

Twee begrippe naamlik ongebalanseerde en gebalanseerde verbindings moet eers gedefinieer word. Figuur 3.1 toon die verskil tussen die twee tipes verbindings.




Figuur 3.1 : verskil tussen Ongebalanseerde en Gebalanseerde verbindings [5]

**Ongebalanseerde (enkelpunt) verbinding** : Die sein word in een lyn oorgestuur met 'n spanningsverskil ten opsigte van die grondlyn. Slegs dié een lyn skakel tussen data. Dit het die voordeel dat parallelle verbindings dieselfde grondlyn kan gebruik, en dus word minder lyne benodig. Die verbinding vaar egter swak in 'n ruisige omgewing as gevolg van kruiskoppelling tussen die seine en grond potensiaalverskille.

**Gebalanseerde (differensiële) verbinding** : Twee lyne vir elke sein word gebruik en beide skakel tussen data. Die potensiaalverskil tussen die twee lyne dui die toestand van die data aan. Die verbinding doen goed in 'n ruisige omgewing, omdat geen kruiskoppelling en grond potensiaalverskille voorkom nie. Differensiële verbindings word gebruik in toepassings waar hoë oordragtempo's benodig word. [5]

Die fisiese verbinding kan in twee hoofklasse verdeel word, naamlik gerigte en ongerigte verbindings. Die gerigte verbindings maak van 'n gerigte medium gebruik vir die transmissie van die sein. Voorbeelde hiervan is metaalgeleiers en optiese vesel. Die ongerigte verbinding saai die sein in 'n ongerigte medium uit. Voorbeelde hiervan is radiogolwe, mikrogolwe en infrarooi kommunikasie. [6]

Vir gebruik op 'n mikrosatelliet word slegs van gerigte mediums gebruik gemaak. Die verskillende mediums word hieronder bespreek.

**1. Dubbel-geleier-oop-lyn**

Dit word gebruik vir ongebalanseerde verbindings. Die sein word voorgestel met 'n spanning of stroom in die een geleier ten opsigte van grondpotensiaal in die ander geleier. 'n Maksimum oordragtempo van 19,2 kbps en 'n afstand van 50 m is haalbaar, waarna seinverswakking te hoog is. Die verbinding vaar swak in die teenwoordigheid van ruis en het 'n hoë uitstraling van elektromagnetiese energie.

**2. Gedraaide-paar-lyn ("Twisted pair lines")**

Dit word gebruik vir gebalanseerde verbindings. Die sein word voorgestel deur die spanningsverskil in die twee geleiers. 'n Oordragtempo van etlike honderd Mbps is haalbaar, afhangende van die seinvoorstelling. Die afstand bereik is minder as 100 m en is afhanklik van die oordragtempo. Die voordeel van die transmissiemedium is die goeie ruisverwerping en lae elektromagnetiese uitstraling.

**3. Ko-aksiale kabel**

Dit word meestal gebruik vir die oordrag van analoë data as gevolg van die goeie isolasie wat dit bied. Die nadeel van gedraaide-paar-kabel is die rand effek ("skin effect"), en word deur ko-aksiale kabel geëlimineer. Hoër bandwydtes kan oor langer afstande behaal word met ko-aksiale kabel.

**4. Optiese vesel**

Optiese vesel word meestal gebruik vir die oordrag van digitale data. Met die verbinding word die sein deur lig voorgestel en transmissie vind binne 'n glas vesel plaas. Oordragtempo's van Gbps is moontlik oor honderde kilometers. Dit is bestand teen elektromagnetiese ruis en straal geen elektromagnetiese energie uit nie. Die nadeel is die kragverbruik en koste verbonde aan die verbinding. [7]

Tabel 3.1 toon 'n vergelyking tussen die verskillende transmissiemediums. Van die tabel kan gesien word dat optiese vesel die mees geskikte medium is om vir hoë spoed toepassings te gebruik, maar die koste verbonde daaraan is baie hoog. As alternatief

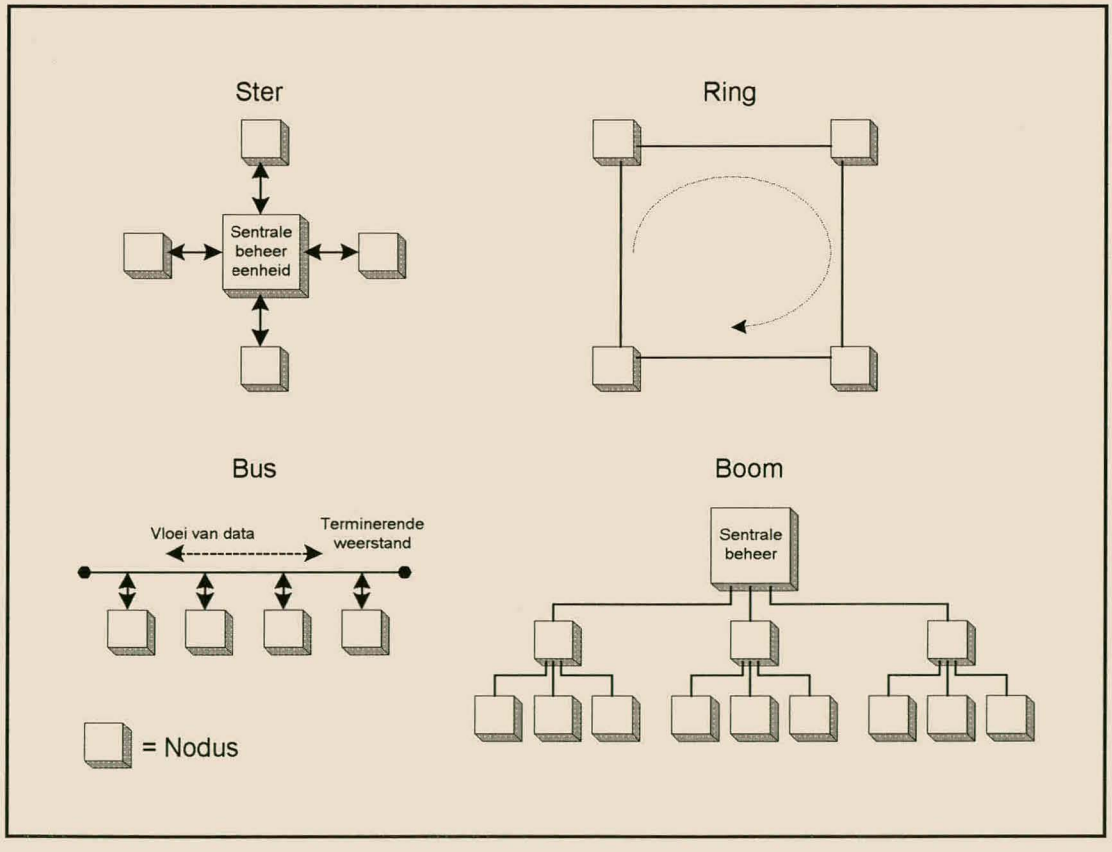
kan gedraaide-paar-lyn of ko-aksiale kabel gebruik word, maar ten koste van die bereikbare afstand.

Tabel 3.1 : Vergelyking tussen die verskillende transmissiemediums

| Transmissiemedium | Data-oordragtempo | Bereikbare afstand | Invloed van ruis | Elektromagnetiese uitstraling | Koste |
|---|---|---|---|---|---|
| Dubbel geleier oop lyn | Laag | Laag | Baie hoog | Hoog | Baie laag |
| Gedraaide-paar-lyn | Hoog | Medium | Laag | Laag | Laag |
| Ko-aksiale kabel | Hoog | Hoog | Laag | Laag | Medium |
| Optiese vesel | Baie hoog | Baie hoog | Geen | Geen | Hoog |

## 3.6 Netwerk topologieë

Daar bestaan verskeie netwerk topologieë. Die geskiktheid daarvan word bepaal deur faktore soos die kompleksiteit van die protokol, die aantal nodusse in die netwerk en die toepassing waarvoor dit gebruik moet word. Figuur 3.2 toon die verskillende netwerk topologieë.




Figuur 3.2 : Netwerk topologieë [6]

**Ster**

Vir hierdie opstelling is al die nodusse verbind aan 'n sentrale eenheid deur 'n stuur- en 'n ontvang punt-tot-punt verbinding. Beheer oor die netwerk word vanaf die sentrale eenheid gedoen. Die topologie is nie geskik vir gebruik op 'n satelliet nie, omdat die hele netwerk faal indien die sentrale eenheid faal.

**Ring**

Die netwerk verbind een eenheid aan die volgende totdat al die eenhede in die vorm van 'n lus of ring verbind is. Dit maak gebruik van 'n direkte eenrigting punt-tot-punt verbinding tussen aangrensende eenhede. Toegangsbeheer tot die ring word benodig om regverdige toegang aan elke eenheid te gee. Die topologie is ook nie geskik vir gebruik op 'n satelliet nie, aangesien die hele netwerk onbruikbaar word indien een van die skakels of een van die eenhede faal.

**Bus**

Die verbinding maak gebruik van 'n multitap medium wat op elke punt getermineer is. Die eenhede verbind direk aan die medium met 'n kort vol-dupleks verbinding. Toegangsbeheer word ook benodig, aangesien die bus deur al die eenhede gedeel word. Dit is die beste opstelling vir gebruik as algemene netwerk op 'n satelliet, aangesien die netwerk nie faal as 'n eenheid onklaar raak nie. Indien 'n onderbreking in die bus sou voorkom, sal eenhede weerskante van die onderbreking nie meer met mekaar kan kommunikeer nie, maar met die toevoeging van 'n tweede bus word die probleem uitgeskakel.

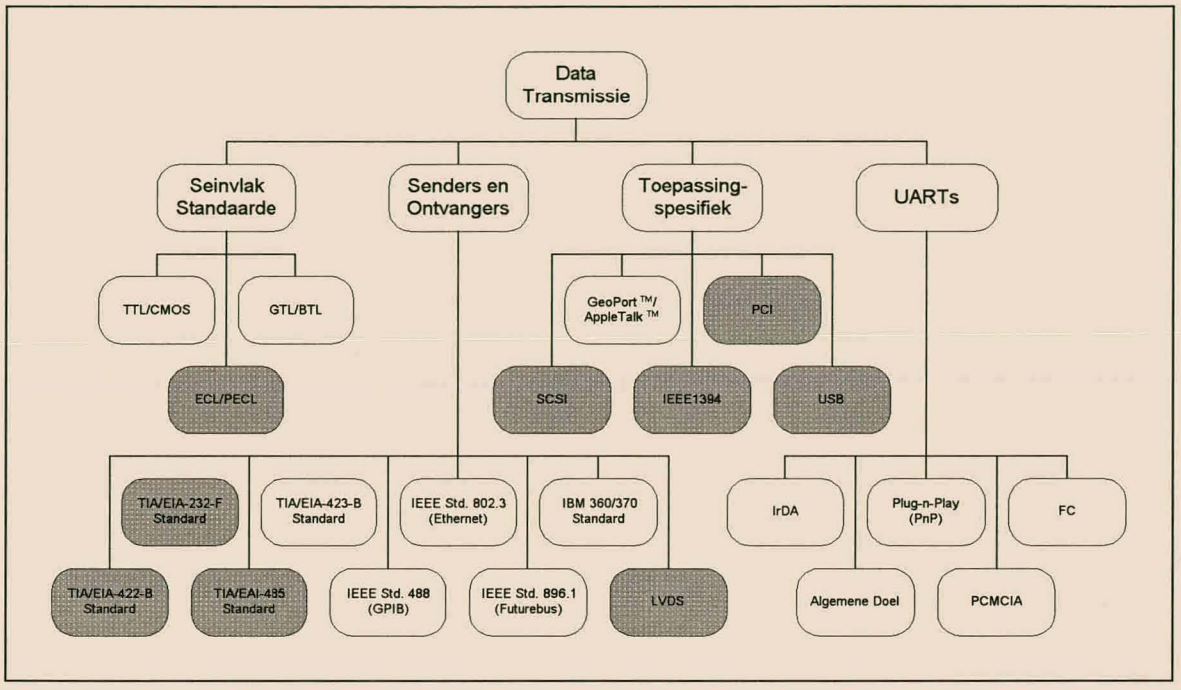
**Boom**

Die topologie word opgebou deur eenhede in 'n boomstruktuur te skakel. Elke eenheid word aan 'n eenheid bo dit verbind en bied weer verbindingsplek vir ander eenhede. Die topologie is ook nie geskik vir gebruik op 'n satelliet nie, omdat die tak onder 'n eenheid onbruikbaar word as die eenheid, of die skakel na die eenheid, faal. [6][7]

## 3.7 Tegnologieë

'n Verskeidenheid van tegnologieë bestaan vir die transmissie van data tussen twee eenhede. Die tegnologieë kan in die volgende vier hoofkatagorieë ingedeel word

naamlik: seinvlak standaarde, algemene senders en ontvangers, toepassing-spesifieke netwerke, en algemene UARTs. Figuur 3.3 toon 'n uiteensetting van die indeling van die tegnologieë in die vier katagorieë. 'n Gedetaileerde bespreking van die gearseerde tegnologieë volg.




Figuur 3.3 : Indeling van datatransmissie-tegnologieë [29]

### 3.7.1 ECL/PECL (pseudo-ECL)

Soos TTL en CMOS definieer ECL en PECL seinvlakke. Die sein word differensieel voorgestel deur die spanningsverskil tussen twee lyne waar beide lyne skakel tussen verskillende data. Komplekse terminasie word benodig by die ontvangkant, en bestaan gewoonlik uit 'n 100 Ω weerstand tussen die twee lyne en 220 Ω aftrekweerstande vanaf elke lyn na grond. ECL en PECL voldoen aan hoë oordragtempo's van tot 400 Mbps, maar ten koste van hoë kragverbruik. Dit is ook besonder afhanklik van die kragtoevoer en bemoeilik omskakeling na laer toevoerspannings.

'n Verskeidenheid van serialiseerders-deserialiseerders (SERDES), wat parallelle TTL-seine omskakel na seriële ECL of PECL seine, en weer terug na parallelle TTL-seine, word deur verskillende maatskappye vervaardig. 'n Voorbeeld van so 'n komponent is SN75FC1000 vervaardig deur Texas Instruments. Hierdie komponent implementeer 'n ultra hoë spoed PECL punt-tot-punt verbinding in beide rigtings. Die hoofdoel van die

komponent is egter om as SERDES vir die fisiese laag van 'n optiese kanaal te dien. Oordragtempo's van 1,0625 Gbps is haalbaar. [8][9]

Tabel 3.2 : Voordele en nadele van ECL/PECL

| Voordele | Nadele |
|---|---|
| • Hoë oordragtempo<br>• Lae invloed van ruis | • Baie hoë kragverbruik<br>• Komplekse terminasie benodig<br>• Meestal afhanklik van 5 V kragtoevoer |

### 3.7.2 TIA/EIA-232-F

TIA/EIA-232 is van die oudste gedefinieerde seriële koppelvlakke en word as vertrekpunt van die kategorie, senders en ontvangers, gebruik. Dit is 'n ongebalanseerde, eenrigting, punt-tot-punt koppelvlak. Die seine word simmetries ten opsigte van die grondlyn voorgestel met die intreedrempels van die ontvanger gelyk aan +3 V vir binêr 0 en –3 V vir binêr 1, en 'n maksimum intreespanning van ±25 V. 'n Maksimum oordragtempo van 20 kbps en 'n afstand van 15 m is haalbaar met TIA/EIA-232. TIA/EIA-694 is soortgelyk aan TIA/EIA-232 met 'n maksimum oordragtempo van 512 kbps.

Tabel 3.3 : Voordele en nadele van TIA/EIA-232-F

| Voordele | Nadele |
|---|---|
| • Goed gedefinieer<br>• Maklik implementeerbaar<br>• Goedkoop | • Ongebalanseerd en dus sensitief vir ruis<br>• Hoë elektromagnetiese uitstraling<br>• Lae oordragtempo<br>• Kort afstande<br>• Hoë kragverbruik |

### 3.7.3 TIA/EIA-422-B en TIA/EIA-485

TIA/EIA-422 is ook 'n seriële koppelvlak. Dit maak gebruik van 'n eenrigting, getermineerde, gebalanseerde koppelvlak. Terminasie vind plaas aan die endpunte van die kabel. TIA/EIA-422 maak voorsiening vir 'n enkelsender met veelvuldige ontvangers. 'n Maksimum oordragtempo van 10 Mbps (teen 10 m) en maksimum afstand van 1000 m (teen 100 kbps) is haalbaar.

TIA/EIA-485 is identies aan TIA/EIA-422, maar maak ook voorsiening vir veelvuldige drywers. 'n Bidireksionele, half-dupleks, veelvuldige puntkoppelvlak is dus moontlik. Dieselfde maksimum oordragtempo en afstand as TIA/EIA-422 word behaal.

Tabel 3.4 : Voordele en nadele van TIA/EIA-422-B en TIA/EIA-485

| Voordele | Nadele |
|---|---|
| • Gebalanseerde (differensiële) verbinding<br>• Goeie ruisverwerping<br>• Lae elektromagnetiese uitstraling<br>• Veelvuldige ontvangers<br>• Goedkoop<br>• Veelvuldige drywers (slegs TIA/EIA-485) | • Lae oordragtempo |

### 3.7.4 TIA/EIA-612

Die standaard spesifiseer 'n eenrigting, getermineerde, punt-tot-punt koppelvlak met 'n gebalanseerde drywer en ontvanger. Die terminasieweerstand is geleë by die ontvanger. Dit maak gebruik van ECL tegnologie en kan data-oordragtempo's van 52 Mbps handhaaf. Die standaard word as verwysing vir TIA/EIA-613 gebruik en implementeer saam HSSI ("High Speed Serial Interface").

Tabel 3.5 : Voordele en nadele van TIA/EIA-612

| Voordele | Nadele |
|---|---|
| • Gebalanseerde (differensiële) verbinding<br>• Goeie ruisverwerping<br>• Lae elektromagnetiese uitstraling | • Lae data-oordragtempo<br>• Hoë kragverbruik |

### 3.7.5 TIA/EIA-644 (LVDS)

LVDS is slegs 'n elektriese standaard. Dit maak gebruik van gebalanseerde drywers en ontvangers. Dit is 'n eenrigting, punt-tot-punt koppelvlak en veelvuldige senders is onder sekere toestande moontlik. Spanningsvlakke is so gekies dat dit geskik is vir hoë spoed, lae kragverbruiktoepassings, en onafhanklik is van die kragtoevoerspanning. Dit kan gebruik word by +5 V, +3,3 V en selfs +2,5 V toevoerspannings. Die lae spanningswaai (330 mV) minimeer ook elektromagnetiese uitstraling. Terminasie word by die ontvangerintree, of eindpunt van die lyn, benodig. 'n Maksimum oordragtempo van 655 Mbps word gespesifiseer. [5][8]

Tabel 3.6 : Voordele en nadele van TIA/EIA-622 (LVDS)

| Voordele | Nadele |
|---|---|
| • Gebalanseerde verbinding<br>• Hoë oordragtempo<br>• Lae kragverbruik<br>• Ruishantering | • Kort afstande |

### 3.7.6 SCSI

SCSI ("Small Computer System Interface") is 'n toepassing-spesifieke busnetwerk en maak van 'n parallelle ongebalanseerde of gebalanseerde verbinding gebruik. Vir ongebalanseerde verbindings word TTL-vlakke gebruik en 'n maksimum afstand van 6 m en 'n oordragtempo van 10 Mxfers/s is haalbaar. Die differensiële verbinding maak gebruik van die RS-485 standaard en 'n maksimum afstand van 25 m en 'n oordragtempo van 20 Mxfers/s is haalbaar. 'n Sentrale beheereenheid word benodig om die toegang na die bus te reguleer. [10]

Tabel 3.7 : Voordele en nadele van SCSI

| Voordele | Nadele |
|---|---|
| • Baie hoë oordragtempo | • Parallelle verbinding<br>• Toepassing-spesifiek<br>• Hoë kragverbruik<br>• Hoë koste<br>• Kompleks |

### 3.7.7 PCI

PCI ("Peripheral Component Interconnect") is ook 'n toepassing-spesifieke parallelle busnetwerk met ongebalanseerde verbindings. Die netwerk word egter so ontwerp dat dit van seinweerkaatsing by die endpunte van die bus gebruik maak om die nodige seinsterkte te verkry. 'n Sein van halfsterkte word op die bus geskryf. Wanneer die weerkaatste sein dan superponeer op die oorspronklike sein, word die drempelwaarde oorskrei, en die sein waargeneem. Hierdie eienskap van PCI bemoeilik die gebruik daarvan, omdat dit baie afhanklik is van die uitleg van die bus.

Baie hoë oordragtempo's kan egter bereik word. Die standaardbus is 32 bisse wyd en skakel teen 33 MHz wat dus 'n oordragtempo van 132 MB/s (1056 Mbps) lewer. 'n Wyer bus met 64 bisse, wat teen 66 MHz skakel en dus 'n oordragtempo van 528 MB/s (4224 Mbps) lewer, bestaan ook. Die kragverbruik is egter baie hoog aangesien teen sulke hoë tempo's gewerk word. PCI benodig ook 'n sentrale beheereenheid wat verantwoordelik is vir die toegang tot die bus. [11]

Tabel 3.8 : Voordele en nadele van PCI

| Voordele | Nadele |
|---|---|
| • Baie hoë oordragtempo | • Parallelle verbinding<br>• Toepassing-spesifiek<br>• Hoë kragverbruik<br>• Baie hoë koste<br>• Beheereenheid benodig<br>• Komplekse uitleg |

### 3.7.8 USB

USB ("Universal Serial Bus") is 'n algemene hoër spoed seriële bus vir gebruik in 'n persoonlike rekenaar (toepassing-spesifiek), en bied verbinding vir tot 127 komponente. Dit is 'n lae koste implementasie met 'n oordragtempo van 12 Mbps. Dit bied beide isochroon en asinchrone oordrag.

USB implementeer 'n protokol wat vloeibeheer en databuffering toepas. Datapakkies van verskillende groottes en verskillende oordragtempo's word ondersteun. Fouthantering word ook deur die protokol behartig. USB implementeer 'n sternetwerk en benodig 'n netwerk beheereenheid. [30]

Tabel 3.9 : Voordele en nadele van USB

| Voordele | Nadele |
|---|---|
| • Seriële gebalanseerde verbinding<br>• Reeds geïmplementeerde protokol<br>• Lae koste<br>• Eenvoudige uitbreibaarheid | • Gewaarborgde bandwydte te laag<br>• Sternetwerk implementasie<br>• Benodig netwerk beheereenheid |

### 3.7.9 IEEE 1394

IEEE 1394 of Firewire is ook 'n toepassing-spesifieke netwerk. Dit is 'n persoonlike rekenaar stelsel I/O standaard en implementeer 'n seriële busnetwerk wat beide isochroon en asinchrone oordrag ondersteun. Drie verskillende oordragtempo's is moontlik naamlik 100, 200 of 400 Mbps. Die standaard definieer die transmissie metode, medium en protokol. Adressering is 64 bisse wyd en kan dus 1023 netwerke met 63 nodusse elk met 281 TB geheue, adresseer.

Die drywers en ontvangers word in die fisiese laag van die netwerk geïmplementeer en die protokol in die verbindingslaag. Om die netwerk meer veeldoelig te maak, is die twee lae gewoonlik nie in dieselfde komponent geïmplementeer nie, maar dit verhoog die kompleksiteit en koste van die netwerk.

Tabel 3.10 : Voordele en nadele van IEEE 1394

| Voordele | Nadele |
|---|---|
| • Hoë oordragtempo's | • Koste is hoog |
| • Gebalanseerde seriële verbinding | • Implementeer boomnetwerk |
| • Reeds geïmplementeerde protokol | • Benodig fisiese- en verbindingslaag eenhede |
| • Eenvoudige uitbreibaarheid | |

### 3.7.10 Optiese veselkanaal

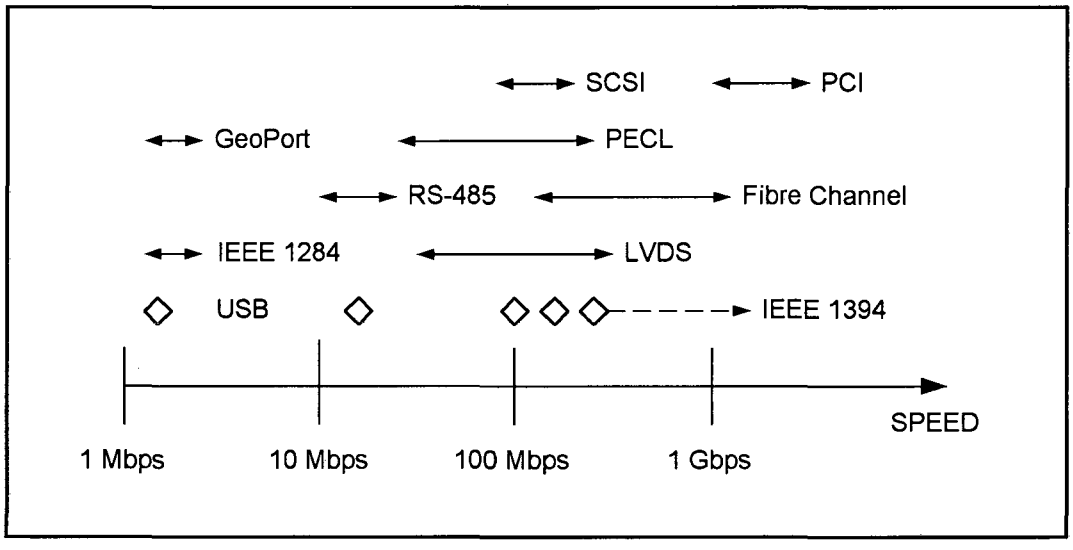
ANSI X3T11 definieer die optiese kanaal ("Fibre Channel") standaard. Optiese vesel word as verbindingsmedium gebruik. Die data intrees na die optiese drywers, en die data uittrees vanaf die optiese ontvangers maak egter gebruik van ECL/PECL vlakke. Dit beteken dat ECL/PECL drywers en ontvangers ook gebruik moet word indien data oor optiese vesel gestuur moet word. [10]

Tabel 3.11 : Voordele en nadele van Optiese veselkanaal

| Voordele | Nadele |
|---|---|
| • Baie hoë oordragtempo | • Baie hoë kragverbruik |
| • Baie lang afstand | • Baie hoë koste |
| • Geen invloed deur ruis | • Benodig ECL/PECL drywers en ontvangers |
| • Geen elektromagnetiese uitstraling | |

## 3.8 Vergelyking tussen tegnologieë

Die belangrikste eienskap van die netwerk is die data-oordragtempo daarvan. Figuur 3.4 toon 'n vergelyking tussen die tegnologieë ten opsigte van data-oordragtempo.




Figuur 3.4 : Vergelyking tussen tegnologieë ten opsigte van data-oordragtempo [10]

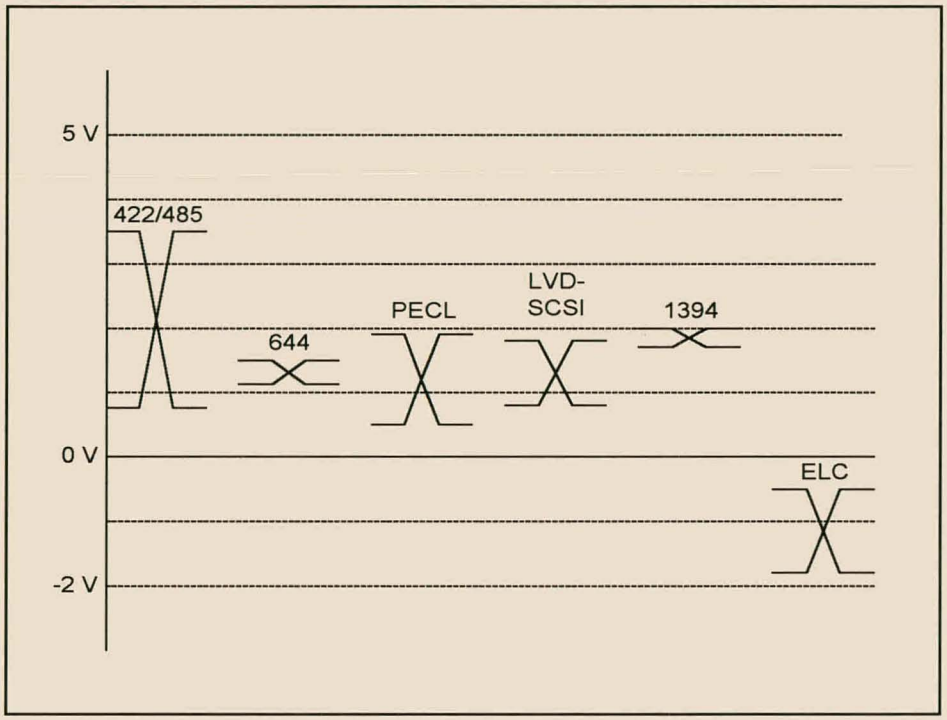
### Vergelyking tussen LVDS en ander tegnologieë

Tabel 3.12 toon 'n vergelyking tussen LVDS, ander seinvlak standaarde, en algemene senders en ontvangers. Van die tabel kan gesien word dat slegs LVDS voldoen aan die vereistes van hoë spoed, lae kragverbruik en lae koste. PECL kan data teen hoë spoed oordra, maar gebruik te veel krag en die koste daarvan is te hoog. TTL is weer baie goedkoper, maar is baie sensitief vir ruis en die datatempo is te laag. Optiese vesel het die nadeel van hoë koste en kompleksiteit.

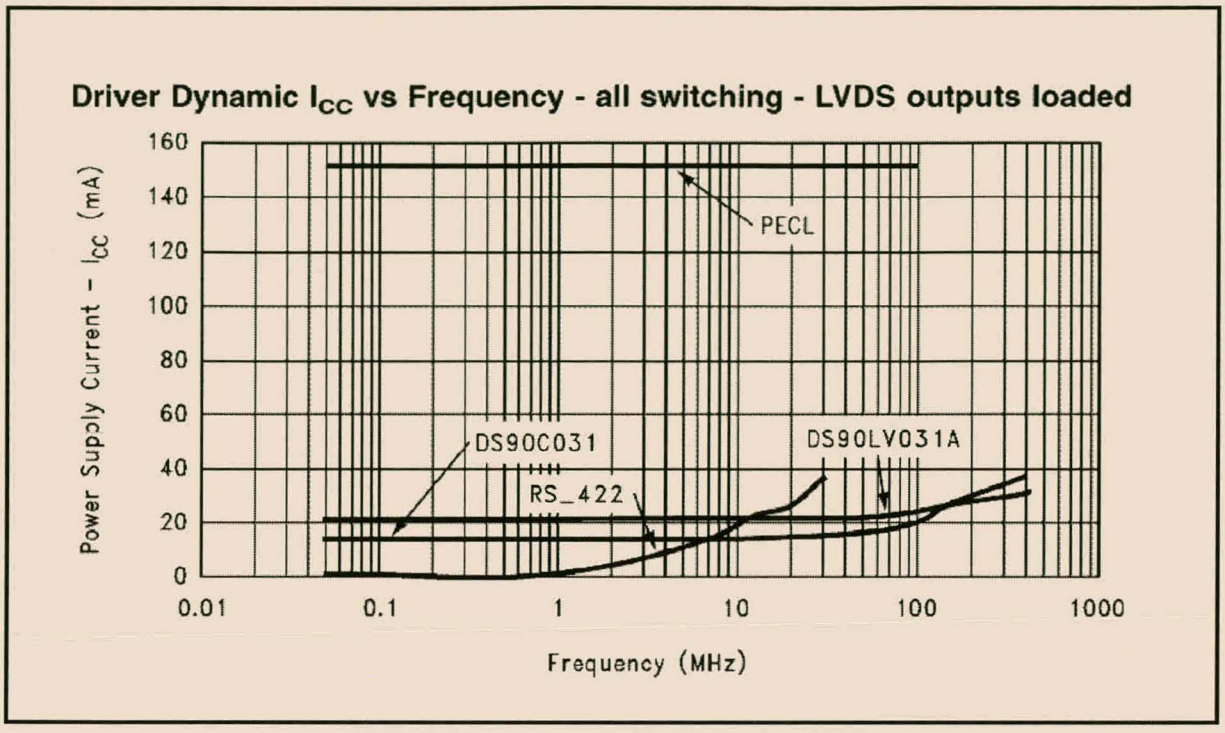
Tabel 3.12 : Vergelyking tussen LVDS en ander standaarde

| Parameters | LVDS | PECL | Opties | RS-422 | GTL | TTL |
|---|---|---|---|---|---|---|
| Differensiële uittreespanning | ±350 mV | ±800 mV | Nvt | ±2-5 V | 1,2 V | 2,4-5 V |
| Ontvanger intreedrempel | ±100 mV | ±200 mV | Nvt | ±200 mV | 100 mV | 1,2 V |
| Oordragtempo (Mbps) | >400 | >400 | 1000 | <30 | <200 | <50 |
| Dinamiese kragverbruik | Laag | Hoog | Hoog | Laag | Hoog | Hoog |
| Ruis | Laag | Laag | Geen | Laag | Medium | Hoog |
| Koste | Laag | Hoog | Hoog | Laag | Laag | Laag |

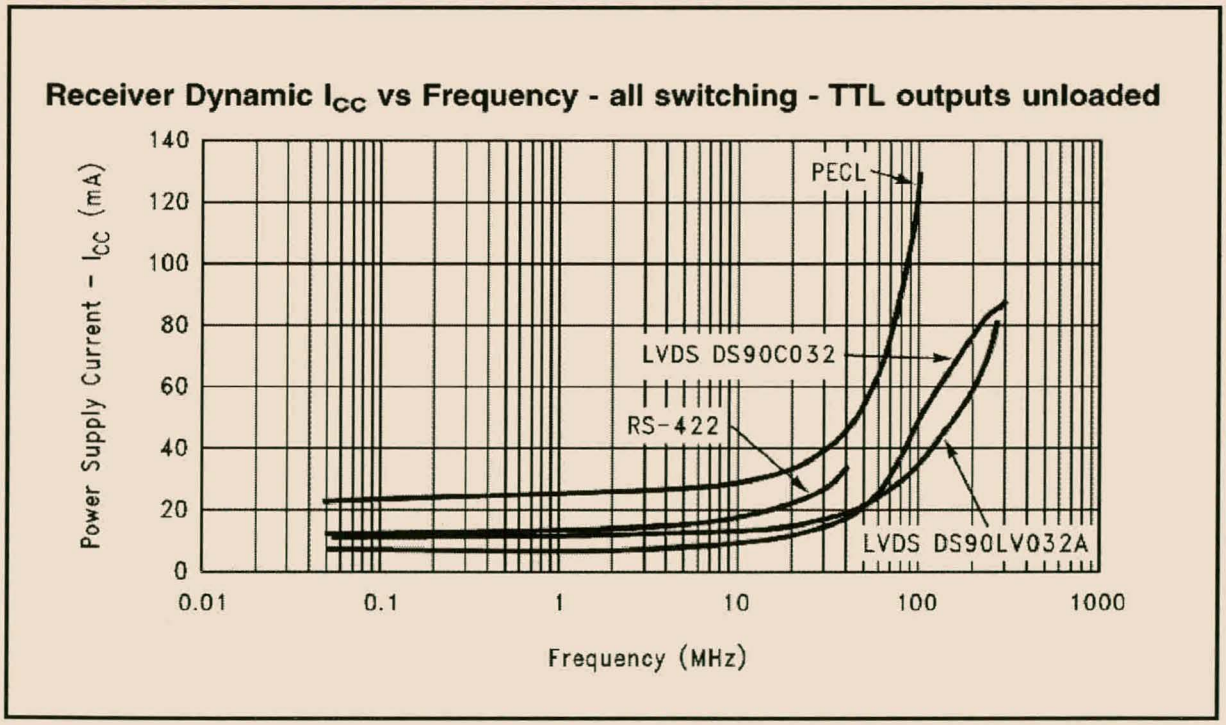
Figuur 3.5 toon die seinvlakke van die verskillende tegnologieë. Soos op die figuur getoon, is die differensiële spanningsverskil van LVDS die kleinste. Figuur 3.6 en figuur 3.7 toon grafieke van die dinamiese stroom ($I_{CC}$) verbruik teenoor frekwensie van LVDS, PECL, en RS-422 drywers en ontvangers. Die grafieke is gebaseer op National Semiconductor se 5 V DS90C031 en 3,3 V DS90LV031A drywers, en 5 V DS90C032 en 3,3 V DS90LV032A senders.




Figuur 3.5 : Seinvlakke van die verskillende tegnologieë




Figuur 3.6 : Dinamiese stroom ($I_{CC}$) verbruik teenoor frekwensie van LVDS, PECL, en RS-422 drywers [9]




Figuur 3.7 : Dinamiese stroom ($I_{CC}$) verbruik teenoor frekwensie van LVDS, PECL, en RS-422 ontvangers [9]

**Vergelyking tussen toepassing-spesifieke netwerke**

Tabel 3.13 toon 'n vergelyking tussen die verskillende toepassing-spesifieke netwerke en Fibre Channel. Soos vanaf die tabel gesien kan word, is USB se bandwydte te laag om as hoë spoed verbinding te dien. SCSI en PCI is parallelle verbindings, en die kompleksiteit en koste daarvan maak dit ook onprakties. Optiese veselkanaal is bedoel om as hoë spoed verbinding oor lang afstande te dien. Die kragverbruik en kompleksiteit daarvan, skakel dit ook uit. IEEE 1394 is die mees geskikte van die toepassings-spesifieke stelsels.

Tabel 3.13 : Vergelyking tussen toepassing-spesifieke netwerke en Fibre Channel

| Kenmerk | USB | IEEE 1394 | SCSI | PCI | Opties veselkanaal |
|---|---|---|---|---|---|
| Oordragtempo | 12 Mbps | 400 Mbps | 20 Mxfers/s | 132 MB/s | 1,0625 Gbps |
| Struktuur | Serieel | Serieel | Parallel | Parallel | Serieel / Opties |
| Maksimum lynlengte | 5 m | 4,5 m | 6 m Ongebalanseer 25 m Differensieel | Opstelling spesifiek | 10 km |
| Aantal konneksies | 127 | 1023 busse, 63 nodusse | Gewoonlik 9 | Opstelling spesifiek | 1 |
| Argitektuur | Sternetwerk | Boomnetwerk | Busnetwerk | Busnetwerk | Punt-tot-punt |
| Stelselkoste | Laag | Medium | Hoog | Hoog | Medium |
| Kragverbruik | Laag | Medium | Hoog | Hoog | Hoog |

## 3.9 TIA/EIA-644 (LVDS) Standaard

Die LVDS ("Low Voltage Differential Signalling") standaard spesifiseer slegs die elektriese eienskappe van die koppelvlak en is ontwikkel om as verwysing vir volledige koppelvlakke te dien. LVDS word gewoonlik gebruik as 'n punt-tot-punt verbinding vir die oordrag van binêre seine tussen stelsels, maar veelvuldige ontvangers en senders word onder sekere ontwerpstoestande ondersteun. Die koppelvlak bestaan uit 'n sender wat deur 'n gebalanseerde verbindingsmedia aan 'n ontvanger en terminasieweerstand verbind is.

LVDS is bedoel om gebruik te word, en funksioneer goed, waar enige van die volgende geld:

- Die data-oordragtempo te hoog is vir ongebalanseerde verbindings.
- Die benodigde oordragtempo te hoog is vir gebalanseerde (differensiële) standaarde soos TIA/EIA-422-B, EIA-485 en TIA/EIA-612.
- Die koppelvlak blootgestel is aan hoë ruisvlakke.
- Waar dit nodig is om elektromagnetiese uitstraling te minimeer.

**Toepaslikheid**

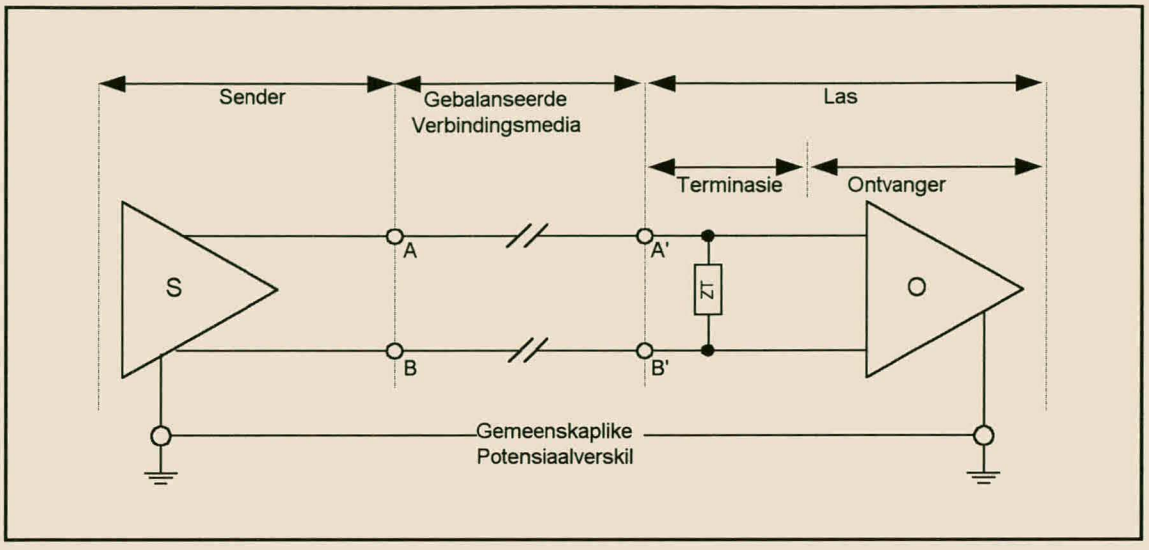
Die maksimum oordragtempo word bepaal deur die sender se transmissietyd eienskap, die verbindingsmedia eienskap, en die afstand tussen die sender en die ontvanger. 'n Maksimum van 655 Mbps word deur die spesifikasie aanbeveel, maar die teoretiese maksimum oor 'n verlieslose medium word bereken as 1,923Gbps. Die maksimum lengte van 'n verbinding word nie in die standaard gespesifiseer nie.

Om LVDS so veelsydig as moontlik te maak, is die standaard onafhanklik van tegnologie, transmissiemedium, en kragtoevoer gedefinieer. LVDS kan geïmplementeer word in BiCMOS, CMOS, of GaAs. Dit kan gebruik word by +5 V, +3,3 V en selfs +2,5 V toevoerspannings en kan data oor PCB of enige tipe kabel stuur.

LVDS-standaard is toegerus met 'n foutbeskermings eienskap om te verseker dat die ontvangeruittrees 'n bekende toestand (HOOG) aanneem indien die intrees gekortsluit, oopsluit, of getermineerd is. Sulke kondisies kan ontstaan indien die ontvanger aktief is en die sender se kragtoevoer verwyder word, die sender onaktief is, of uit die stelsel verwyder word.

**Elektriese eienskap**

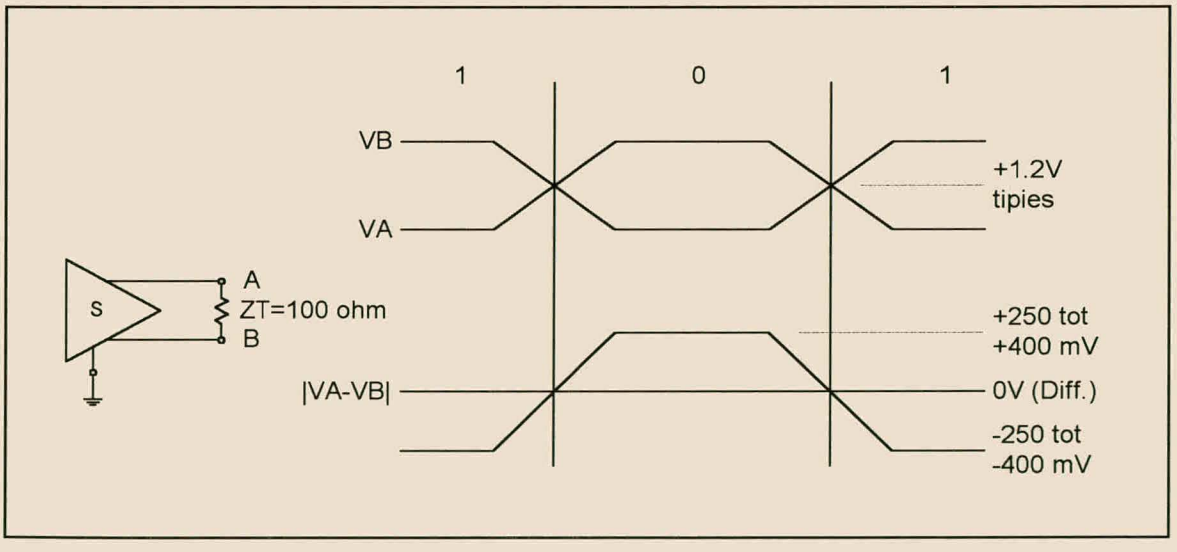
Figuur 3.8 toon die LVDS-koppelvlakbaan. Die baan bestaan uit drie dele naamlik die sender (S), die gebalanseerde verbindings media, en die las. Die las bestaan uit 'n terminasie-impedansie en 'n ontvanger (O). Tabel 3.14 aan die einde van die hoofstuk toon die elektriese eienskap van die LVDS-standaard.




Figuur 3.8 : LVDS-koppelvlakbaan [12]

Sender:

Volgens spesifikasie is die sender 'n gebalanseerde stroombron wat 'n differensiële spanning van 250 mV tot 450 mV oor 'n terminasie-las van 100 Ω lewer. Figuur 3.9 toon die spanningsvlakke by die senderuittree met 'n binêre 1, voorgestel deur 'n negatiewe spanning by punt A met verwysing tot B.




Figuur 3.9 : Senderuittree spanningsvlakke [12]

Verbindingsmedia:

Die verbindingsmedia kan bestaan uit enige paar metaalgeleiers wat gebalanseerde transmissie sal behou soos byvoorbeeld gedraaide-paar-kabel, ko-aksiale kabel, "ribbon

cable", of PCB lyne. Die maksimum DC lusweerstand word as 50 Ω gespesifiseer en die karakteristieke impedansie as 110 Ω ±20% vanaf 10 MHz tot die toepassing se boonste frekwensielimiet.

Las:

Die las word gedefinieer as die impedansie tussen A' en B' op figuur 3.8 en bestaan uit 'n terminasie-impedansie en die ontvanger. Dit is 'n gebalanseerde ontvanger met 'n hoë intree-impedansie en 'n lae intree drempelwaarde van ±100 mV. Die terminasie-impedansie (ZT) word gespesifiseer as tussen 90 Ω en 132 Ω, maar moet gelyk aan die verbindingsmedia-impedansie, by die toepassingsfrekwensie, gekies word. Die terminasie moet so na as moontlik aan die ontvanger geplaas word, en oppervlak gemonteerde komponente word aanbeveel. [12]

Tabel 3.14 : Elektriese eienskap van die TIA/EIA-644 (LVDS) Standaard [8]

| Parameter | Beskrywing | Min | Maks | Eenheid |
|---|---|---|---|---|
| $V_t$ | Differensiële Uittree Spanning | 247 | 454 | mV |
| $V_{OS}$ | Afset Spanning | 1,125 | 1,375 | V |
| $\Delta V_t$ | \|Verandering in $V_t$\| | | 50 | \|mV\| |
| $\Delta V_{OS}$ | \|Verandering in $V_{OS}$\| | | 50 | \|mV\| |
| $I_{SC}$ | \|Kortsluit stroom\| | | 24 | \|µA \| |
| $t_r$ / $t_f$ | Uittree Styg / Val Tyd (<200 Mbps) | | 30% van $t_{ui}$† | |
| | Uittree Styg / Val Tyd (≥200 Mbps) | 0,26 | 1,5 | ns |
| $I_{IN}$ | \|Intree Stroom\| | | 20 | \|µA\| |
| $V_{TH}$ | \|Drempel spanning\| | | 100 | \|mV\| |
| $V_{ID}$ | Differensiële Intree Spanningsbereik | 100 | 600 | mV |
| $V_{IN}$ | Intree Spanningsbereik | 0 | 2,4 | V |

† $t_{ui}$ is eenheid interval (biswydte)

# 4. Stelselontwerp

Soos reeds genoem is die doel van hierdie ondersoek om 'n hoë bandwydte koppelvlak te ontwikkel. Die verbinding tussen die hoofkamera en die massageheue word as uitgangspunt gebruik waarom die stelsel ontwerp word. Die verbinding tussen die twee substelsels op die huidige SUNSAT, is reeds in hoofstuk 2 bespreek en die nadele daarvan is uitgewys. In die vorige hoofstuk is verskillende hoë spoed verbindings-tegnologieë bespreek en LVDS is as die geskikste vir die doel uitgewys.

LVDS het die voordele dat dit 'n baie hoë spoed seriële verbinding is, met 'n hoë ruis immuniteit en lae kragverbruik. As gevolg van die hoë bandwydte beskikbaar met LVDS, is dit moontlik om 'n mate van protokol ook te implementeer. Dit is egter moeilik om komplekse protokolle teen sulke hoë datatempo's te implementeer.

Sekere aannames word gemaak ten opsigte van die stuureenheid (hoofkamera) en die ontvangeenheid (massageheue) . Ideaal moet dit vir die stuur en ontvangeenhede lyk asof hulle direk aan mekaar gekoppel is. 'n Geskikte koppelvlak tussen die twee eenhede en die LVDS-verbinding word in hierdie hoofstuk bespreek.

## 4.1 Aannames

Die volgende aannames word ten opsigte van die stuureenheid (hoofkamera) en die ontvangeenheid (massageheue) gemaak. 'n Beskrywing van die volgende generasie kamera is reeds in hoofstuk 2 gegee en die aannames is daarop gebaseer.

Aannames ten opsigte van volgende generasie kamera:

- Die hoofkamera vorm 'n funksionerende eenheid.
- Die neem van 'n beeld geskied onafhanklik van die hoë spoed koppelvlak.
- 'n Betroubare lae spoed bevelbus word gebruik om die kamera te aktiveer en beheer.
- Alle klokseine benodig, word deur die kamera-eenheid self opgewek.
- Alle ekstra inligting is reeds by die beelddata gevoeg wanneer dit uitgeklok word.
- Geen data kan deur die kamera gestoor word nie.
- Korrupte data kan nie weer vanaf die kamera aangevra word nie.

Die aannames ten opsigte van die massageheue is die volgende:

- Die massageheue beskik oor 'n eenheid wat die adressering en toegang beheer.
- Die data kan parallel, sinchroon met 'n beheersein, aan die eenheid gegee word.
- Die geheue-eenheid is verantwoordelik daarvoor dat data nie verlore gaan indien die massageheue nie gereed is om dit te ontvang nie.

Ander algemene aannames:

- Die tydvertraging geassosieer met die oordrag van die data is nie kritiek nie.
- Ernstige foute word aan die sentrale beheereenheid gerapporteer en dié sal verantwoordelik wees vir die toepaslike reaksie. Dit het egter tot gevolg dat die beeld onvolledig sal wees.

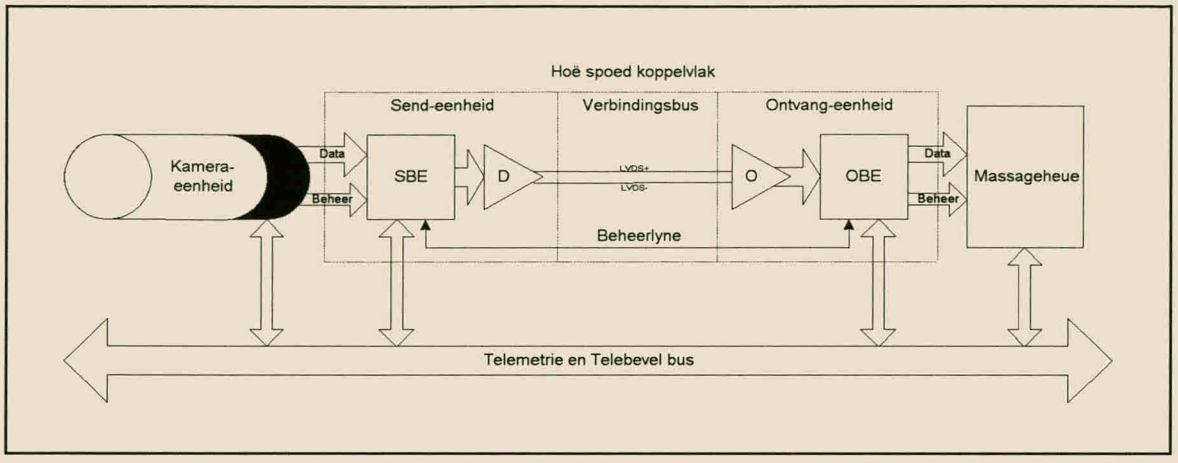
Om dus op te som, sal die kamera die beelddata konstant, sinchroon met beheerseine, uitklok tydens die neem van 'n foto. Die massageheue sal vooraf opgestel wees om die data te ontvang, en dit sal parallel, sinchroon met beheerseine, daarin ingeklok word. 'n Koppelvlak word dus benodig wat vir die kamera en die geheue lyk asof dit direk parallel aan mekaar gekoppel is.

## 4.2 Oorhoofse ontwerp

Figuur 4.1 toon 'n blokdiagram van die beplande stelsel wat ontwerp word. Die stelsel bestaan uit drie eenhede naamlik die sender, die verbindingsbus, en die ontvanger. Dit sal 'n hoë spoed punt-tot-punt koppelvlak vanaf die sender- na die ontvangereenheid implementeer deur van LVDS gebruik te maak. Enkele beheerlyne vanaf die ontvanger terug na die sender vorm ook deel van die verbindingsbus.

Die sendeenheid bestaan uit 'n beheereenheid (SBE) en die LVDS-drywer. Die beheereenheid is verantwoordelik vir die ontvang van die data vanaf die kamera, enige manipulasie of tydelike storing van die data, die beheer van die LVDS-drywer en die oorgee van data aan die drywer.

Die ontvangeenheid bestaan uit 'n beheereenheid (OBE) en die LVDS-ontvanger. Die ontvanger beheereenheid beheer die LVDS-ontvanger, ontvang die data, doen die nodige verwerking, en gee dit oor aan die massageheue.




Figuur 4.1 : Blokdiagram van die beplande hoë bandwydte koppelvlak

Daar word aanvaar dat die bandwydte van die LVDS-verbinding hoog genoeg is om die beelddata, asook protokoldata te akkommodeer. Aangesien 'n punt-tot-punt verbinding geïmplementeer word, is 'n komplekse adresseringsprotokol nie nodig nie. Die hoofdoel van die protokol is om te verseker dat die data korrek vanaf die kamera tot in die massageheue oorgedra word.

Die twee belangrikste aspekte wat in die protokol vervat moet word is die volgende:

- **Eenheidkoppeling** : Wanneer 'n foto geneem word, genereer die kamera-eenheid data teen 'n konstante tempo. Eerstens kan die proses nie tydelik gestop word nie, aangesien data verlore sal gaan. Die koppeling tussen die kamera-eenheid en die hoë spoed verbinding is dus uiters belangrik. Die protokol moet voorsiening maak dat data teen 'n konstante tempo vanaf die kamera-eenheid ontvang kan word.

- **Data betroubaarheid** : Tweedens kan die data nie weer vanaf die kamera-eenheid aangevra word nie, omdat dit nie oor enige stoorarea beskik nie. Die protokol moet dus ook sorg dat die data betroubaar na die ontvangeenheid oorgedra word.

Vir die eerste vereiste word stoorarea benodig. Wanneer 'n foto geneem word, word beelddata, teen 'n konstante tempo, in die geheue ingeskryf. Sodra die beheereenheid waarneem dat data in die geheue teenwoordig is, kan dit teen 'n hoër tempo uitgelees en

gestuur word. Data word dus tydelik in die geheue gestoor indien dit nie gestuur kan word nie, en die kamera word dus nie onderbreek nie.

Vir die tweede vereiste word foutdeteksie en herstelling van foutiewe data benodig. Die ontvangeenheid moet in staat wees om die korrektheid van die ontvangde data te toets. Indien 'n fout voorgekom het, moet 'n metode bestaan om dit te herstel.

## 4.3 Eenheidkoppeling

Komplekse stelsels is gewoonlik opgebou uit 'n aantal eenhede wat elk verantwoordelik is vir 'n spesifieke funksie. Vir die korrekte funksionering van die stelsel, moet data tussen hierdie eenhede oorgedra kan word. Die eenhede implementeer gewoonlik elk 'n unieke koppelvlak vir die oordrag van data. Dus hang die koppeling tussen twee eenhede af van fisiese faktore soos die beheerlyne van die koppelvlak, asook van dinamiese faktore soos die oordragtempo. Twee algemene metodes bestaan om die datapaaie van twee eenhede te koppel naamlik dubbelpoort geheue ("dual port RAM") en eerste-in-eerste-uit geheue (FIFO RAM).

**Dubbelpoort geheue**

Dubbelpoort geheue is geheue met twee onafhanklike toegangspoorte. Elke poort beskik oor sy eie datalyne, adreslyne en beheerlyne. Data kan dus op enige plek vanaf die een poort in die geheue geskryf word, terwyl data op enige ander plek vanaf die tweede poort gelees word.

So 'n geheue word as 'n buffer gebruik om 'n blok data tydelik te stoor. Die geheue word tussen die twee eenhede geïmplementeer met die stuureenheid verbind aan die skryfpoort en die ontvangeenheid aan die leespoort. Wanneer die stuureenheid beheer van die geheue het, skryf dit 'n blok data daarin. Daarna word die beheer oorgegee aan die ontvangeenheid wat weer die blok data lees. Semafoor logika word gebruik om die beheer toe te ken en dit verhoed dat die stuureenheid data probeer skryf wanneer die ontvangeenheid besig is om die data te lees.

Omdat slegs een buffer geïmplementeer word, het hierdie opstelling die nadeel dat die sender moet wag totdat die ontvanger die hele blok data gelees het, voordat dit weer

data daarheen kan skryf. Deur die geheue in twee te deel en van 'n tweede stel semafoor logika gebruik te maak, kan die oordrag van data meer effektief plaasvind. Die metode om van dubbelpoort geheue gebruik te maak, word veral by stelsels, wat reeds beskik oor geheue-adresseringseenhede, toegepas.

**Eerste-in-eerste-uit geheue FIFO**

FIFO buffers word meestal gebruik om datastrome tussen twee eenhede te koppel, waarvan die oordrageienskappe, soos die oordragtempo, verskil. Die stuureenheid kan dus data teen sy konstante tempo daarheen skryf. Wanneer data beskikbaar is, kan die ontvangeenheid teen sy datatempo die data begin lees.

Daar word van 'n eerste-in-eerste-uit rystelsel gebruik gemaak vir die stoor van die data. Data kan dus slegs gelees word indien daar reeds data geskryf is. Soos dubbelpoort geheue, het FIFO's ook twee toegangspoorte. Die skryfpoort word gebruik om data agter in die ry te plaas en die leespoort word gebruik om data voor uit die ry te haal. Elke poort beskik oor sy eie datalyne en beheerlyne. Die beheerlyne word gebruik om te versoek dat data in die FIFO geskryf kan word, of daaruit gelees kan word. Die adressering van waar die data in die geheue gelees of geskryf word, word intern tot die FIFO gedoen. Geen eksterne adreslyne is dus beskikbaar nie.

'n Stel vlaggies bestaan om die toestand van die geheue aan te toon. 'n Vol vlaggie en leë vlaggie toon aan as al die geheue gebruik is, of indien niks van die geheue gebruik is nie. Data kan nie na die geheue geskryf word indien dit vol is nie en data kan ook nie daarvan gelees word indien dit leeg is nie. Die amper vol vlaggie en amper leeg vlaggie toon aan as die geheue 'n spesifieke hoeveelheid plek oor het, of gebruik het, en funksioneer as waarskuwing. Die hoeveelheid plek wat oor moet wees, of in beslag geneem moet wees, vir die aktivering van dié vlaggies, is of staties vas, of kan dinamies verander word.

**Gevolgtrekking**

Vir die implementering van die koppelvlak tussen die kamera en die hoë spoed stelsel, is die FIFO die mees geskikte geheue om van gebruik te maak. Eerstens kan beelddata teen 'n konstante tempo in die FIFO ingeskryf word, al is die sendeenheid nie gereed

om die data onmiddellik te stuur nie. Wanneer die eenheid data kan stuur, en data is in die FIFO teenwoordig, kan dit teen 'n hoë tempo daaruit gelees en gestuur word.

Tweedens is geen geheuebeheer, behalwe vir die lees- en skryfaksie, nodig nie. Al die adressering word deur die FIFO self gedoen. Daar word slegs van die lees- en skryflyne gebruik gemaak vir die oordrag van die data, en van vlaggielyne om die toestand van die FIFO aan te dui.

## 4.4 Data betroubaarheid

Die belangrikste eienskap van 'n datanetwerk en protokol is die betroubaarheid daarvan. Om foute in die data te kan herken en sinvol daarop te reageer, word al hoe belangriker soos datavolume en spoed toeneem. Twee algemene metodes van foutbeheer is: Fout-Deteksie-en-Herversending, en Fout-Deteksie-en-Korreksie. Die eerste metode word algemeen gebruik in datakommunikasie en maak gebruik van 'n sikliese oortolligheidstoets ("Cyclic Redundancy Check" - CRC) tegniek. Die tweede metode word algemeen gebruik wanneer data tydelik in geheue gestoor moet word.

**Sikliese oortolligheidstoets**

Hierdie tegniek maak gebruik van 'n wiskundige algoritme op die data wat gestuur moet word om oortollige inligting te genereer. Dit word agter die data gelas en saam oorgestuur en verander dus nie aan die oorspronklike data nie. Aan die ontvangkant word dieselfde algoritme op die ontvangde data gedoen om te bepaal of 'n fout voorgekom het. Indien 'n fout voorgekom het, is die tegniek nie in staat om die fout te korrigeer nie. 'n Versoek moet aan die sender gestuur word om die blok data oor te stuur. Dit het die nadeel dat die data weer gestuur moet word en dat dit aan die stuurkant gestoor moet word totdat seker is dat die data korrek ontvang is.

CRC hanteer die data wat gestuur moet word as 'n enkele groot getal. Die wiskundige algoritme doen 'n vermenigvuldiging met $2^N$ en deling met 'n spesiale getal (die polinoom), wat dieselfde in grootte is as die vermenigvuldiger (N bisse), en lewer as res 'n getal wat, as dit by die data getal getel word, gelyk deelbaar is deur die polinoom.

Hieronder volg 'n desimale voorbeeld wat die proses illustreer. Die data is 725913 en word vermenigvuldig met $10^2$, en die polinoom is gelyk aan 83.

Voorbeeld om werking van CRC te verduidelik

| | |
|---|---|
| Data om te stuur: | 725913 |
| Data vermenigvuldig met 100/83: | 874593 (res 81) |
| Toets syfers = 83 – res: | 83-81=2 |
| Data met toets syfers: | 72591302 |
| Korrekte data ontvang × 100/83: | 7259130200/83=87459400 (res 0) |
| Verkeerde data ontvang × 100/83: | 7254130200/83=87399159 (res **3**) |
| Verkeerde data ontvang × 100/83: | 72**1359**0200/83=86910725 (res **25**) |

CRC wys nie net enkel bisfoute uit soos "9" na "4" nie, maar ook volgordefoute soos "5913" na "1359", wat nie deur gewone toets-som-metodes geïdentifiseer word nie. Die effektiwiteit van CRC hang af van die grootte en keuse van die polinoom. Die data kommunikasie en netwerk industrie het op 'n standaard polinoom besluit wat bekend staan as CRC-32 en bestaan uit 32 bisse. Dit word gebruik in Ethernet, FDDI, optiese verbindings, en ander protokolle. [28]

Die polinoom lyk soos volg:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

**Fout-deteksie-en-korreksie**

In die 1950s het Hamming 'n wiskundige tegniek ontwikkel vir die opstel van kodes met die eienskap waar enkelfoute herstel kan word en dubbelfoute geïdentifiseer kan word. 'n Soortgelyke metode word alledaags gebruik waar data in geheue gestoor moet word. Die metode staan bekend as "Single Error Correcting, Double Error Detecting (SECDED) Modified Hamming Error Correction Code (ECC)" en maak gebruik van eksklusiewe-of sommasies op die data om die oortollige inligting (toetsbisse) te bereken.

'n Aantal oortollige bisse word bereken, elk op verskillende kombinasies van ongeveer helfte van die databisse. Die kombinasie van databisse wat gebruik word om elke toetsbis te genereer word so gekies dat 'n enkelfout geïdentifiseer kan word deur te kyk watter toetsbisse verkeerd is. 'n Addisionele toetsbis word gebruik om die

moontlikheid waar 'n dubbelfout as 'n enkelfout herken word, uit te skakel. 'n Enkel- of dubbelfout in die toetsbisse kan met hierdie metode net so effektief herken word. Die minimum toetsbisse benodig vir verskillende datawydtes word in tabel 4.1 getoon.

Tabel 4.1 : Minimum aantal toetsbisse benodig

| Datawydte | Toetsbisse |
|---|---|
| 3 – 4 | 4 |
| 5 – 11 | 5 |
| 12 – 26 | 6 |
| 27 – 57 | 7 |
| 58 – 120 | 8 |
| 121 – 247 | 9 |
| 248 – 502 | 10 |

Die werking word in die volgende voorbeeld geïllustreer. Drie databisse (101) word beskerm deur vier toetsbisse.

Voorbeeld om werking van EDAC te verduidelik

**Enkodering**

Toetsbis 0 : T0 = D0 XOF D1
Toetsbis 1 : T1 = D0 XOF D2
Toetsbis 2 : T2 = D1 XOF D2
Toetsbis 3 : T3 = D0 XOF D1 XOF D2
Na enkodering : 101.0101

**Dekodering**

Sindroombis 0 : S0 = D0 XOF D1 XOF C0
Sindroombis 1 : S1 = D0 XOF D2 XOF C1
Sindroombis 2 : S2 = D1 XOF D2 XOF C2
Sindroombis 3 : S3 = D0 XOF D1 XOF D2 XOF C3
Dekodering van 101.0101 eindig met sindroom van 0000 dus geen foute
Dekodering van 111.0101 eindig met sindroom van 1101 wat fout in D1 identifiseer

## Gevolgtrekking

Soos reeds genoem, kan data nie weer van die kamera aangevra word nie. Om dus 'n CRC foutdeteksieskema te gebruik, moet die data by die sendeenheid gestoor word, wat

ekstra geheue hiervoor benodig. Alhoewel die effektiewe bandwydte van 'n CRC skema hoër is as die van 'n EDAC skema, word die verrigting van die netwerk baie verswak indien herversending van die data moet plaasvind.

'n Beter opstelling is om van 'n EDAC gebruik te maak om te verseker dat alle enkelbisfoute herstel word. Voorsiening moet gemaak word vir die herversending van klein hoeveelhede data indien die EDAC 'n dubbel bisfout waarneem, of 'n fout met die LVDS-verbinding ontstaan. Hiervoor word van 'n data skuifregister gebruik gemaak. Die skuifregister kan net 'n sekere hoeveelheid data stoor. Soos data by die een punt ingeskryf word, word ou data by die ander punt verwyder.

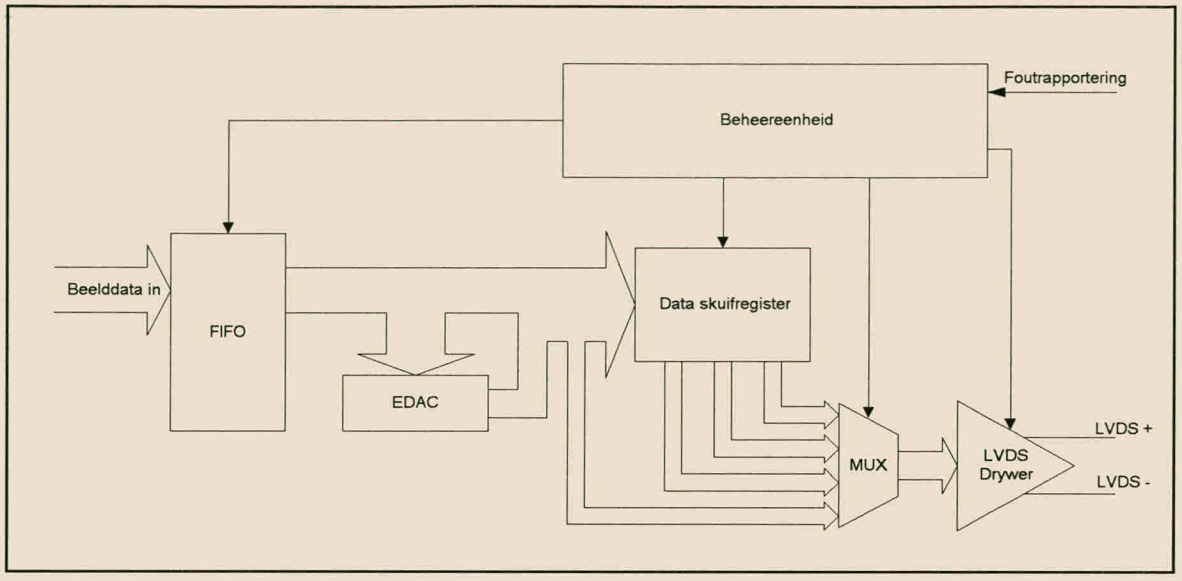
Indien data herversoek word, word dit uit die pyp gelees vanaf die oudste tot die nuutste, totdat al die data weer gestuur is. Daarna gaan die proses van data lees uit die FIFO en die stuur daarvan normaal voort.

## 4.5 Sendeenheid-ontwerp

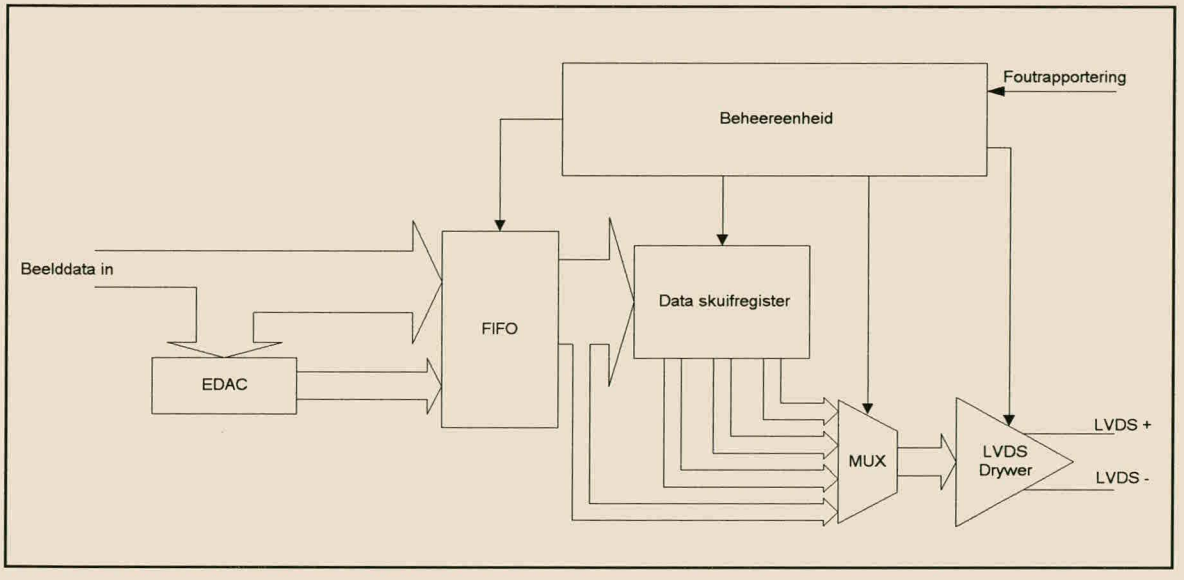
Soos hierbo genoem, bestaan die sendeenheid uit 'n send-beheereenheid en die LVDS-drywer. Die send-beheereenheid kan weer op sy beurt in die volgende vyf eenhede opgedeel word:

- FIFO
- EDAC
- Data skuifregister
- Multiplekser
- Toestandsmasjien-beheereenheid

Twee moontlike opstellings bestaan vir die skakeling van die eenhede naamlik die FIFO-EDAC-opstelling en die EDAC-FIFO-opstelling. Figuur 4.2 toon die FIFO-EDAC-opstelling en figuur 4.3 die EDAC-FIFO-opstelling.




Figuur 4.2 : Blokdiagram van FIFO-EDAC-opstelling




Figuur 4.3 : Blokdiagram van EDAC-FIFO -opstelling

**FIFO-EDAC-opstelling**

Met hierdie opstelling word die kamera-koppelvlak eers aan die FIFO verbind. Die FIFO word dan aan die EDAC verbind. Die uittree van die EDAC is verbind aan die multiplekser, asook aan die data skuifregister waar die data tydelik gestoor word. Afhangende of die stelsel in normale modus is, of in foutherstellings-modus, selekteer die toestandsmasjien-beheereenheid een van die multiplekser intrees en gee dit aan die LVDS-drywer.

Die ontvangde data word teen 'n konstante tempo van die kamera in die FIFO geskryf. Wanneer die data gestuur moet word, word die data uit die FIFO gelees en die EDAC funksie daarop toegepas. Beide die beelddata en die oortollige data word aan die LVDS-drywer gegee sodat dit gestuur kan word.

Die voordeel van hierdie opstelling is dat die FIFO slegs nodig het om die beelddata te stoor en dus meer effektief benut word. Die nadeel is egter dat slegs enkel bisfoute wat oor die verbinding ontstaan, geïdentifiseer en korrigeer kan word. Indien 'n fout in die FIFO sou ontstaan as gevolg van ruimtelike omstandighede, sal die ontvangeenheid nie in staat wees om die fout te herken of korrigeer nie.

**EDAC-FIFO-opstelling**

Hierdie opstelling is dieselfde as die FIFO-EDAC-opstelling, maar die EDAC-funksie word voor die FIFO geïmplementeer. Die beelddata, asook die oortollige data, word tydelik in die FIFO gestoor, totdat dit gestuur kan word. Beide die beeld, sowel as die oortollige data word dan uit die FIFO gelees en aan die multiplekser en die data skuifregister gegee. Die toestandsmasjien-beheereenheid selekteer een van die multiplekser intrees en gee dit aan die LVDS-drywer sodat dit gestuur kan word.

Die data wat vanaf die kamera ontvang word, kan steeds teen 'n konstante tempo ingelees en na die FIFO geskryf word. Geen tydvertraging is nodig vir die berekening van die oortollige data nie, aangesien die EDAC dit asinchroon bereken. Sodra die beelddata verskyn, word die oortollige data bereken en saam met die beelddata in die FIFO ingeskryf.

Die voordeel van hierdie opstelling is dat alle enkel bisfoute wat mag ontstaan tussen die kamera en die massageheue, identifiseer en herstel kan word. Die nadeel is egter dat 'n groter FIFO benodig word om beide die beelddata en die oortollige data te stoor.

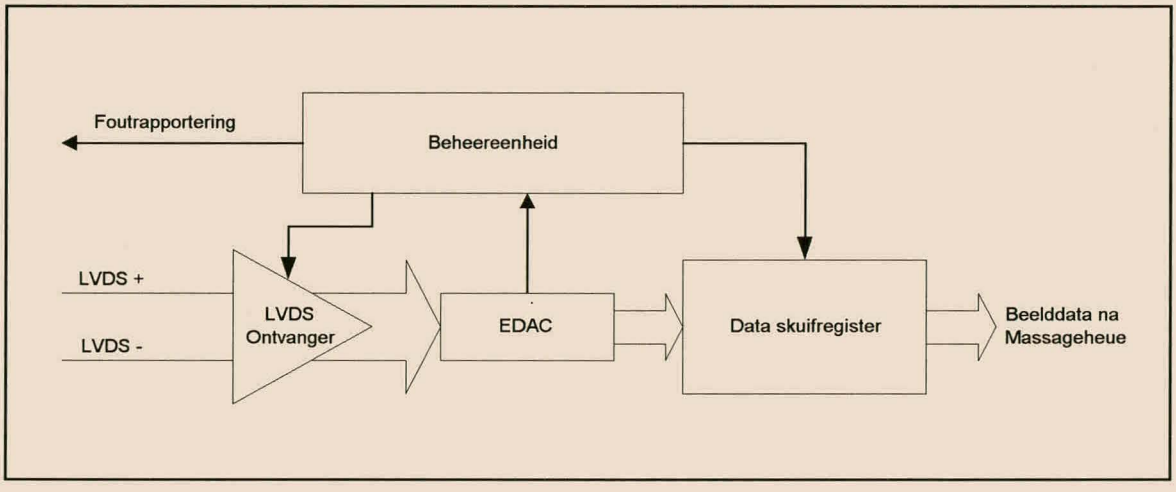
**Gevolgtrekking**

Aangesien die korrektheid van die data baie belangrik is, is die EDAC-FIFO-opsie, die beter opsie om te gebruik in 'n satelliet. Alhoewel dit ten koste van 'n wyer FIFO is, is die stelsel gewaarborg om enige enkel bisfoute te kan herstel, en enige dubbel bisfoute te kan herken.

Soos die data uit die FIFO gelees word, word dit in die data skuifregister ingeskryf en vir 'n sekere aantal data-eenhede gestoor, totdat dit by die ander punt van die pyp uitskuif. Data wat weer aangevra word, kan uit die data skuifregister verkry word. Die ontvangeenheid word rondom die opsie ontwerp.

## 4.6 Ontvangeenheid-ontwerp

Die ontvangeenheid bestaan uit die ontvang-beheereenheid en die LVDS-ontvanger. Die ontvang-beheereenheid bestaan weer uit die EDAC, data skuifregister, en 'n toestandsmasjien-beheereenheid. Figuur 4.4 toon die blokdiagram van die ontvangeenheid.




Figuur 4.4 : Blokdiagram van die ontvangeenheid

Data wat vanaf die LVDS-ontvanger ontvang word, word deur die EDAC gestuur om alle enkel bisfoute te korrigeer en dubbel bisfoute aan die toestandsmasjien-beheereenheid te rapporteer. Indien die data korrek is, word dit in 'n data skuifregister ingeskuif en die data by die ander punt uitgeskuif en na die geheue geskryf. Indien die data foutief is, word dit geïgnoreer en al die data binne die skuifregister word gewis.

Indien 'n fout voorgekom het, word dit ook aan die sendeenheid gerapporteer, sodat die foutiewe data weer gestuur kan word. Om die sendeenheid en die ontvangeenheid in sinchronisasie te hou, word 'n data skuifregister ook in die ontvangeenheid geïmplementeer.

# 5. Apparatuurontwerp

Die gedetaileerde apparatuurontwerp van die hoë spoed koppelvlak wat in die vorige afdeling voorgestel is, word in hierdie afdeling bespreek. Dit is egter belangrik om daarop te let dat die doel van hierdie apparatuurontwerp, nie is om op die volgende generasie SUNSAT te gebruik nie, maar om die LVDS en die bruikbaarheid daarvan te ondersoek. Aanpassings word dus gemaak op die stelsel wat in die vorige hoofstuk ontwerp is. Komponente is nie gekies met die doel om op 'n satelliet gebruik te word nie, alhoewel sekere komponente wel geskik is daarvoor.

## 5.1 Aanpassings

Aan die stuurkant word 'n bron van data benodig wat in staat is om groot volumes data teen 'n hoë bistempo (80 Mbps) te genereer. Aan die ontvangkant moet die data weer onttrek kan word en die ontvangde data moet met die gestuurde data vergelyk kan word. Die geskikste metode hiervoor is om van persoonlike rekenaars gebruik te maak vir die opwek en stuur van die data, en die ontvang en vergelyking daarvan. Die ISA-koppelvlak word gebruik as die skakeling tussen die persoonlike rekenaar en die stelsel.

Aangesien die ISA-koppelvlak nie data teen so 'n hoë tempo kan lewer nie, word van geheue gebruik gemaak om eers die data tydelik te stoor. Data word dus teen 'n stadiger tempo vanaf die ISA-bus na die geheue gelaai, en teen 'n hoër tempo uit die geheue gelees. So kan 'n bron met 'n hoë bistempo voorgestel word. Dieselfde gebeur aan die ontvangkant. Data word baie vinniger ontvang as wat dit deur die ISA-bus ingelees kan word. Die data word eers tydelik in die geheue gestoor waarna dit teen 'n laer bistempo deur die ISA-bus ingelees kan word.

## 5.2 Ontwerp

Die ontwerpte stelsel bestaan uit 'n groot aantal digitale logika. Die logika kan op een van die volgende drie metodes geïmplementeer word:

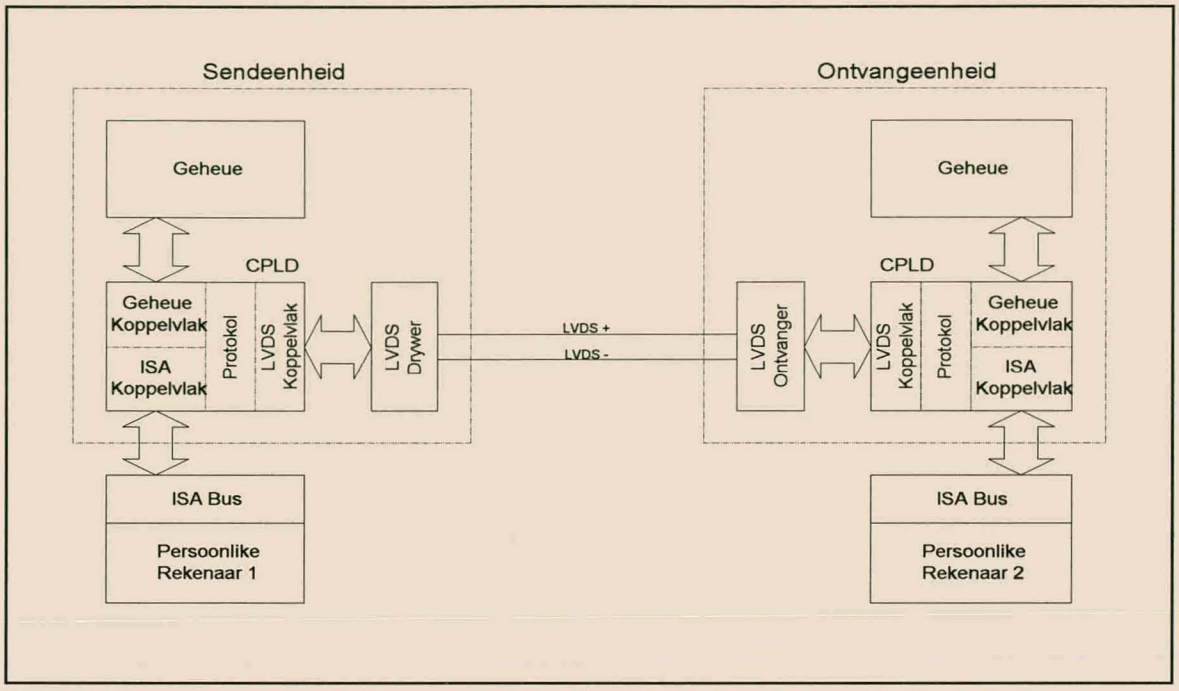
- Diskrete logika (TTL, CMOS)
- Toepassing-spesifieke geïntegreerde komponente (ASICs)
- Programmeerbare logiese eenhede (PLDs).

Tabel 5.1 toon 'n vergelyking tussen die drie metodes. PLD's kan weer in drie groepe verdeel word naamlik lae digtheid PAL/GAL-komponente, "field programmable gate arrays" (FPGA's), en komplekse PLD's (CPLD's). Die verskil tussen FPGA's en CPLD's lê in die inter konnekteringstruktuur. FPGA's maak gebruik van veelvuldige metaallyne van wissellende lengte wat deur transistors aan die logiese selle verbind is. CPLD's maak van kontinue metaallyne vir die interkonneksie gebruik, en lei tot kleiner en vinniger strukture, met voorspelbare tydvertragings.

Tabel 5.1 : Vergelyking tussen die drie metodes om logika te implementeer

| Eienskap | Diskrete logika | ASIC | PLD |
|---|---|---|---|
| Spoed | Swak | Effektief | Effektief |
| Digtheid | Swak | Effektief | Effektief |
| Koste | Swak | Effektief | Effektief |
| Ontwikkellingstyd | Gemiddeld | Swak | Effektief |
| Simulasietyd | Swak | Swak | Effektief |
| Vervaardigingstyd | Gemiddeld | Swak | Effektief |
| Bruikbaarheid | Gemiddeld | Swak | Effektief |
| Toekomstige aanpassings | Gemiddeld | Swak | Effektief |

'n CPLD word gebruik om die logika te realiseer en sodoende word die fisiese kompleksiteit van die stelsel baie vereenvoudig. Dit het 'n verdere voordeel dat die ontwerp maklik aangepas kan word sonder om veranderinge op apparatuur-vlak aan te bring. Die veelsydigheid van CPLD's laat ook toe dat die koppelvlakke na die ISA-bus, geheue en LVDS-komponente, asook die protokol daarin geïmplementeer kan word. Figuur 5.1 toon die blokdiagram van die apparatuurontwerp.




Figuur 5.1 : Blokdiagram van apparatuurontwerp

### 5.2.1 Die Hoë Spoed LVDS-komponente

LVDS-komponente kan in drie klasse ingedeel word naamlik senders en ontvangers, "Flat Panel Display Interface" (FPDI), en serialiseerders-deserialiseerders (SERDES). Die senders en ontvangers skakel slegs seine tussen TTL-vlakke en LVDS-vlakke. Die komponent bevat byvoorbeeld ag drywers. Ag TTL-bisse word by die ingange van die drywers aangelê, en dit word na ag LVDS-seine omgeskakel.

FPDI is ontwikkel om as verbinding tussen 'n persoonlike rekenaar en 'n LCD-vertooneenheid te dien. Dataserialisering en -deserialisering word deur die komponente gedoen. 'n Komponent bevat byvoorbeeld vier eenhede wat elk sewe TTL-bisse serialiseer na vier paar LVDS-lyne (dit wil sê 28:4 kompressie). Die TTL-seine word op 'n kloksein gegrendel en serieel teen sewe keer die klokfrekwensie op die LVDS-lyne uitgestuur. Die kloksein word ook op 'n paar LVDS-lyne oorgestuur. Daar is dus vier paar LVDS-datalyne en een paar LVDS-kloklyne tussen die twee komponente.

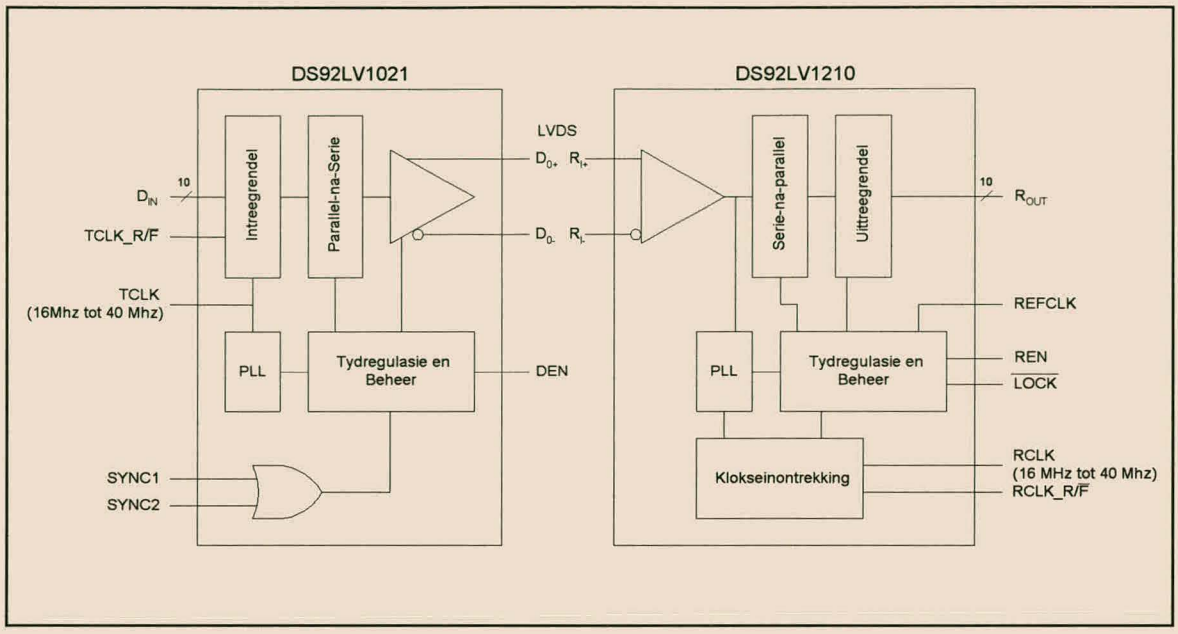
SERDES funksioneer dieselfde as FPDI, maar al die data, asook die kloksein, word oor dieselfde paar LVDS-lyne gestuur. Die stuur- en ontvangkant maak van fase-sluit lusse (PLL) gebruik om die hoër frekwensie op te wek. Die ontvanger se PLL moet eers sinchroniseer met die sender se PLL voordat data ontvang kan word. 'n SERDES het

die nadeel dat dit tyd neem om te sinchroniseer, maar die verbinding bestaan uit slegs twee geleiers en is dus meer voordelig om op 'n satelliet te gebruik.

Die LVDS-sender- en ontvangerpaar waarvan gebruik gemaak is vir die hoë spoed seriële dataskakel, is die DS92LV1021 en die DS92LV1210 vervaardig deur National Semiconductor. Die primêre kenmerke van die komponente kan soos volg opgesom word:

- Gewaarborgde verandering met elke data-oordragsiklus
- Enkele differensiële lynpaar wat multikanaal-skeefheid ("skew") elimineer
- 400 Mbps seriële LVDS-bandwydte teen 40 MHz klokfrekwensie
- 10 bis parallelle datakoppelvlak
- Sinchronisasie modus en sluit-indikator
- Programmeerbare klokrandsneller
- Hoë impedansie op ontvanger-intree indien dit van geen krag voorsien word nie
- 3,3 V tegnologie.

Die DS92LV1021 en DS92LV1210 is 'n 16 tot 40 MHz 10-bis wye bus LVDS-serialiseerder- en deserialiseerderpaar, en kan gebruik word om data oor swaar belaaide PC bord of tot 10 meter lengte kabel te stuur. Die DS92LV1021 serialiseer 'n 10-bis parallelle CMOS/TTL-bus na 'n enkele hoë spoed LVDS-datastroom met saamgevoegde klok. Die DS92LV1210 ontvang die LVDS-datastroom en deserialiseer dit om weer 'n 10-bis parallelle bus en aparte klok te lewer. Figuur 5.2 toon blokdiagramme van die DS92LV1021 en die DS92LV1210.




Figuur 5.2 : Blokdiagram van DS92LV1021 en DS92LV1210 [13]

Beide sender en ontvanger maak gebruik van PLL's. Tydens inisialisasie moet beide PLL's sinchroniseer op 'n lokale klok wat dieselfde kloksein of verskillende klokseine kan wees, maar met dieselfde frekwensie. Die klokseine word vir die sender aangelê by die TCLK-pen en vir die ontvanger by die REFCLK-pen. Daarna moet die ontvanger sinchroniseer met die sender. Dit geskied deur SYNC-patrone (sinchronisasie patrone, "0000011111") te stuur vanaf die sender. SYNC-patrone word gestuur deur die SYNC1- of SYNC2-penne aktief hoog te maak. Die ontvanger se $\overline{\text{LOCK}}$ -pen sal onaktief hoog bly tydens die proses van sinchronisasie op die klok en op die SYNC-patrone waarna dit laag gaan. Die $\overline{\text{LOCK}}$ -pen moet egter deurentyd gemonitor word in geval die ontvanger sinchronisasie verloor. Indien dit gebeur moet SYNC-patrone weer gestuur word sodat die ontvanger weer kan sinchroniseer.

Wanneer beide sender en ontvanger gesinchroniseer is kan data gestuur word. Data word vanaf DIN0-DIN9 op die rand van die kloksein TCLK ingeklok. Die rand waarop dit ingeklok word, word geselekteer deur die TCLK_R/F-pen. Indien een van die SYNC-penne hoog is vir langer as vyf TCLK-siklusse, word die data op DIN0-DIN9 geïgnoreer en word SYNC-patrone gestuur.

Die 10 databisse wat ingeklok is word geserialiseer en 'n beginbis (hoog) en 'n eindbis (laag) word bygevoeg. Die 12 bisse word dan serieel oorgestuur teen 12 maal TCLK se

frekwensie. Wanneer $\overline{\text{LOCK}}$ laag is, is die data op ROUT0-ROUT9 geldig, en word dit op die rand van RCLK uitgeklok. Die rand waarop dit uitgeklok word, word geselekteer deur RCLK_R/F-pen. Indien $\overline{\text{LOCK}}$ egter hoog is, is die uittree ROUT0-ROUT9 en RCLK in 'n hoë impedansie-toestand.

Beide sender en ontvanger kan in 'n lae kragverbruiktoestand geplaas word deur die $\overline{\text{PWRDN}}$- en DEN- en REN-penne laag te dryf. Tydens hierdie toestand word die PLL's gestop en al die uittreepenne in 'n hoë impedansietoestand (TRI-STATE) geplaas. Met heraktivering van die sender en ontvanger moet sinchronisasie herhaal word. Die LVDS-uittrees, DO+ en DO-, van die sender kan in 'n TRI-STATE geplaas word deur DEN onaktief laag te maak. Die uittrees ROUT0-ROUT9, RCLK en $\overline{\text{LOCK}}$ van die ontvanger kan ook in 'n TRI-STATE geplaas word deur REN onaktief laag te maak.

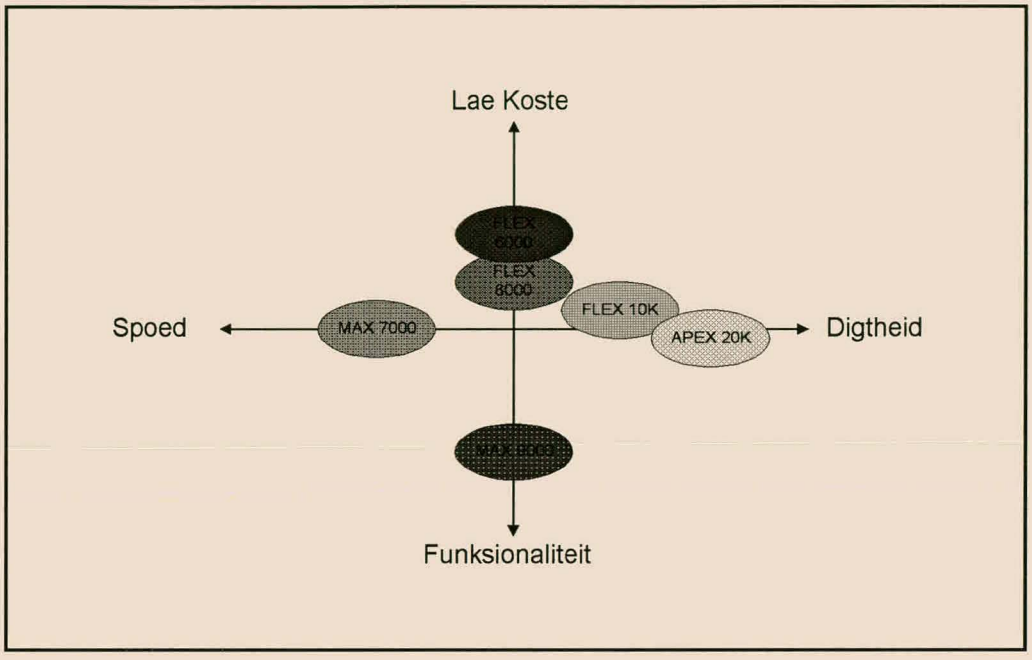
Soos reeds genoem, maak hierdie tegnologie gebruik van 'n stroomlus. Daarom word termineer-weerstande op elke end van die transmissielyn benodig. In die geval van punt-tot-punt skakel met een sender en een ontvanger, is slegs 'n weerstand by die ontvangkant nodig. 'n 100 Ω weerstand word hiervoor gebruik. As gevolg van die baie hoë bistempo is die uitleg van die PCB bord uiters belangrik. Die sender en ontvanger moet so na as moontlik aan die randkonnektor geplaas word. [13]

### 5.2.2 CPLD

Soos reeds genoem, word van 'n CPLD gebruik gemaak om die koppelvlakke na die eenhede en die protokol te implementeer. CPLD's word deur verskeie vervaardigers vervaardig. Daar is besluit om van CPLD's, vervaardig deur ALTERA, gebruik te maak. Die eerste rede hiervoor is die vertroudheid met dié CPLD's en die konfigurasie sagteware. Die tweede rede is die innoverende argitektuur en gevorderde tegnologieë waarvan die CPLD's gebruik maak.

ALTERA vervaardig die volgende sewe families van CPLD's: FLEX 10K, FLEX 8000, FLEX 6000, MAX 9000, MAX 7000, MAX 3000, en CLASSIC. Die FLEX ("Flexible Logic MatriX") reeks maak van opsoektabelle ("look-up tabLE's" – LUT's) gebruik om logiese funksies te implementeer. Die MAX- ("Multiple Array MatriX") en CLASSIC-reeks gebruik produkterme (programmeerbare-EN / vaste-OF) om logiese funksies te

implementeer. Figuur 5.3 toon die indeling van die families en die verskille word in tabel 5.2 getoon.




Figuur 5.3 : Indeling van ALTERA CPLD families[24]

Tabel 5.2 : Vergelyking tussen ALTERA CPLD families [24]

| Familie | Bruikbare I/O penne | Aantal hekke | Kern spannings-potensiaal | Herkonfigureerbare element |
|---|---|---|---|---|
| FLEX 10K | 59 tot 470 | 10000 tot 250000 | 2,5 V, 3,3 V en 5,0 V | SRAM |
| FLEX 8000 | 78 tot 208 | 2500 tot 16000 | 3,3 V en 5,0 V | SRAM |
| FLEX 6000 | 71 tot 218 | 10000 tot 24000 | 3,3 V en 5,0 V | SRAM |
| MAX 9000 | 59 tot 216 | 6000 tot 12000 | 5,0 V | EPROM |
| MAX 7000 | 36 tot 212 | 600 tot 10000 | 3,3 V en 5,0 V | EPROM |
| MAX 5000 | 16 tot 69 | 600 tot 3750 | 5,0 V | EPROM |
| CLASSIC | 22 tot 64 | 300 tot 900 | 5,0 V | EPROM |

In die vorige afdeling is genoem dat 'n FIFO deel vorm van die protokol. Om die FIFO in 'n CPLD te implementeer deur van die logiese elemente (LE's) gebruik te maak, word 'n groot deel van die CPLD in beslag geneem. Om dus groot FIFO's te implementeer, word baie groot CPLD's benodig. Die FLEX 10K familie beskik egter oor "embedded array blocks" (EABs) wat geskik is vir die implementering van geheues soos byvoorbeeld FIFO's.

Wanneer geheue in 'n FLEX 10K CPLD geïmplementeer word, word geen van die LE gebruik nie. Die geheue word gerealiseer in die EABs. Elke EAB beskik oor 2048 bisse en kan byvoorbeeld 'n 256 × 8-geheue implementeer. Die FLEX 10K familie is om hierdie rede gekies as die mees geskikte CPLD om te gebruik. Tabel 5.3 toon die vergelyking tussen FLEX 10K CPLD's.

Tabel 5.3 : Vergelyking tussen CPLD's in die FLEX 10K familie [14]

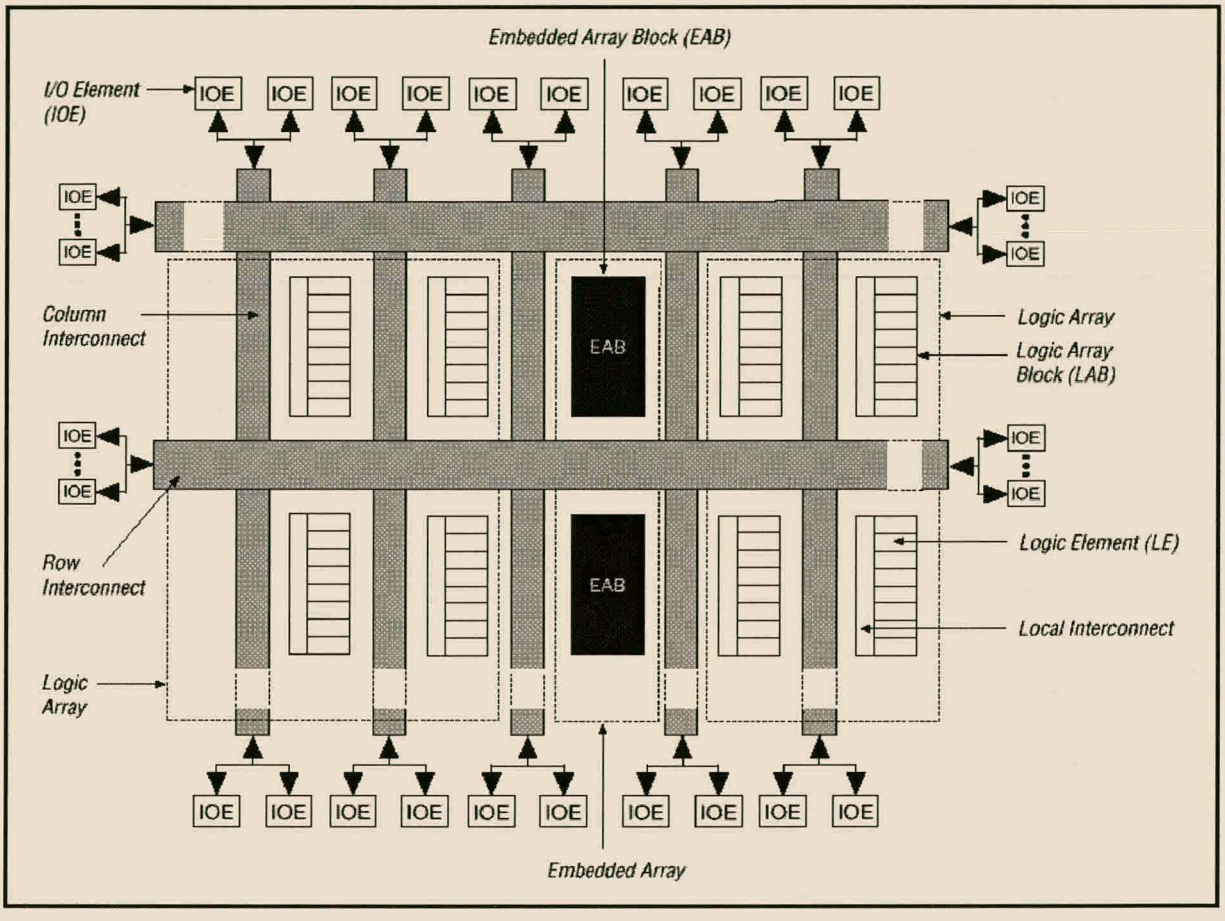
| CPLD | Aantal hekke | LE's | LAB's | EABs | Totale RAM bisse | Maksimum I/O penne |
|---|---|---|---|---|---|---|
| EPF10K10 | 10000 | 576 | 72 | 3 | 6144 | 150 |
| EPF10K20 | 20000 | 1152 | 144 | 6 | 12288 | 189 |
| EPF10K30 | 30000 | 1728 | 216 | 6 | 12288 | 246 |
| EPF10K40 | 40000 | 2304 | 288 | 8 | 16384 | 189 |
| EPF10K50 | 50000 | 2880 | 360 | 10 | 20480 | 310 |
| EPF10K70 | 70000 | 3744 | 468 | 9 | 18432 | 358 |
| EPF10K100 | 100000 | 4992 | 624 | 12 | 24576 | 406 |
| EPF10K130 | 130000 | 6656 | 832 | 16 | 32768 | 470 |
| EPF10K250 | 250000 | 12160 | 1520 | 20 | 40960 | 470 |

'n Minimum FIFO-grootte van 1 KB moet in die CPLD geïmplementeer word en 'n CPLD met minimum 8192 RAM bisse word hiervoor benodig. 'n EPLF10K20 of EPF10K30 kan hiervoor gebruik word. Die EPF10K30 CPLD is gekies bo die EPF10K20 CPLD, omdat dit oor 'n groter aantal LE's en hekke beskik.

Die belangrikste kenmerke van hierdie komponent is die volgende:

- Maak gebruik van EABs
- Baie hoë digtheid van hekke
- I/O penne kan 3,3 V seine herken
- Drie toestand I/O penne
- Lae kragverbruik
- Kontinue verbindingsstruktuur vir hoë spoed en voorspelbare vertragings
- Toegewyde oordragskakel ("Dedicated carry chain")
- Toegewyde kaskade-skakeling ("Dedicated cascade chain")
- Ses globale klokintrees.

Die interne struktuur van die CPLD word in figuur 5.4 getoon. Elke LE bestaan uit 'n LUT en 'n programmeerbare register. 'n LAB ("Logic array block") bestaan uit ag LE's wat met kort hoë spoed verbindings aan mekaar geskakel is. Elke I/O pen is verbind aan 'n IOE ("I/O Element") wat bestaan uit 'n drie-toestand bidireksionele buffer, 'n ingangregister, en uitgangregister.



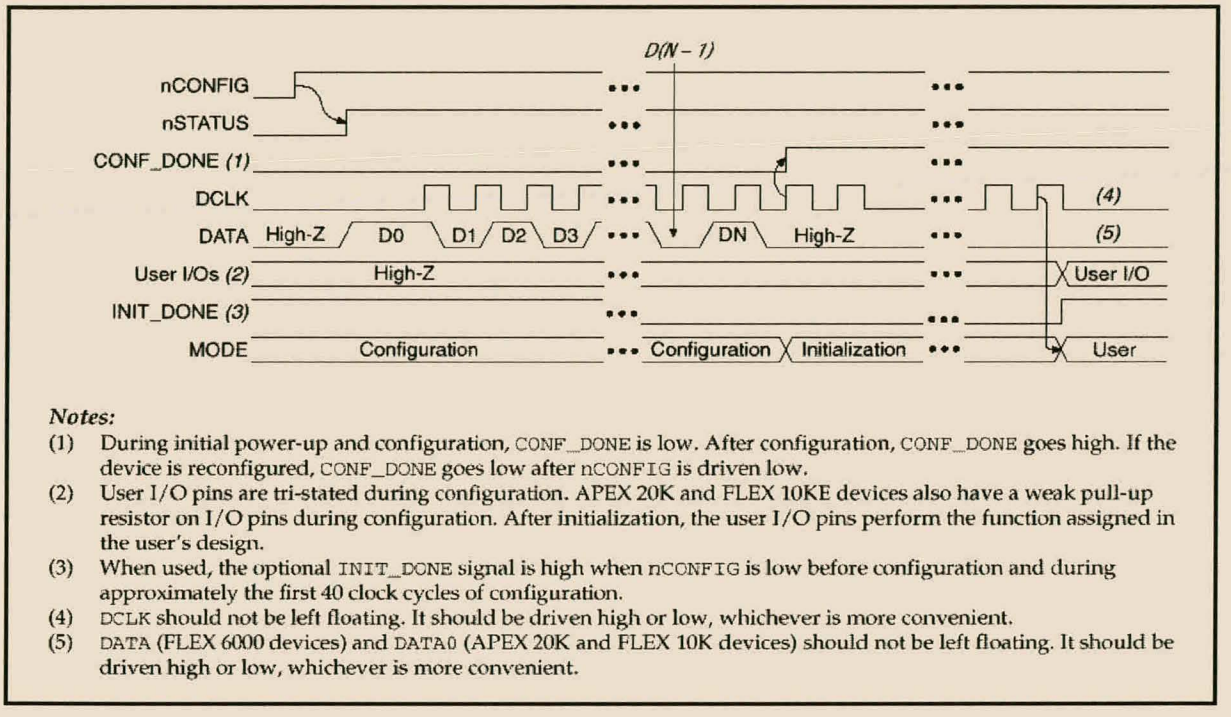

Figuur 5.4 : Blokdiagram van FLEX 10K CPLD [14]

Verskillende metodes bestaan om die CPLD te programmeer. Deur die geheue-selekteringslyne, MSEL0 en MSEL1, aan grond te verbind, word die metode om van 'n serie-PROM gebruik te maak, gekies. Dit is egter belangrik om daarop te let dat dié CPLD nie die programmeringskloksein genereer nie, dus moet of die serie-PROM of 'n eksterne bron dit genereer.

Die programmeringskloklyn, DCLK, word aan die kloklyn van die serie-PROM verbind. Die twee statuslyne, nSTATUS en CONF_DONE, word aan die uitgang

aktiveringslyn, OE, en die seleksielyn, CS, van die serie-PROM onderskeidelik verbind. Die data-intreelyn, DATA0, word aan die data-uittreelyn, DATA, van die serie-PROM verbind.

Om konfigurasie van die CPLD te begin, moet die nCONFIG-lyn hoog getrek word. Die lyn word dus met 'n optrekweerstand aan $V_{CC}$ verbind, sodat konfigurasie begin word wanneer kragtoevoer voorsien word. Figuur 5.5 toon die tyddiagram van die CPLD konfigurasiesiklus aan. [14][15]




*Notes:*

(1) During initial power-up and configuration, CONF_DONE is low. After configuration, CONF_DONE goes high. If the device is reconfigured, CONF_DONE goes low after nCONFIG is driven low.
(2) User I/O pins are tri-stated during configuration. APEX 20K and FLEX 10KE devices also have a weak pull-up resistor on I/O pins during configuration. After initialization, the user I/O pins perform the function assigned in the user's design.
(3) When used, the optional INIT_DONE signal is high when nCONFIG is low before configuration and during approximately the first 40 clock cycles of configuration.
(4) DCLK should not be left floating. It should be driven high or low, whichever is more convenient.
(5) DATA (FLEX 6000 devices) and DATA0 (APEX 20K and FLEX 10K devices) should not be left floating. It should be driven high or low, whichever is more convenient.

Figuur 5.5 : Tyddiagram van die konfigurasiesiklus [15]

### 5.2.3 QS3L384 QuickSwitch

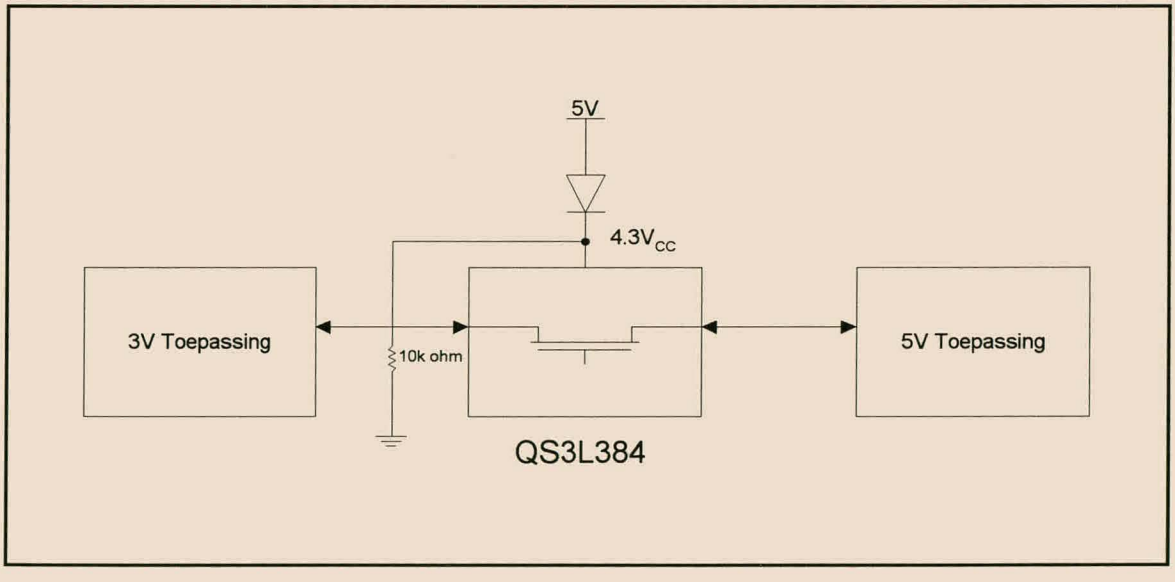
Die ISA-koppelvlak maak gebruik van 5 V tegnologie. Om daarmee te skakel word 'n 5 V CPLD gebruik. Die LVDS-komponente maak egter gebruik van 3,3 V tegnologie. Dit is dus nodig om al die intreeseine na die LVDS-komponente eers van 5 V na 3,3 V om te skakel. Dit is belangrik dat die tydvertraging wat met die omskakeling gepaard gaan, baie kort moet wees. Vir die seine vanaf die LVDS-komponente na die CPLD is 'n omskakeling nie nodig nie, aangesien die CPLD 3,3V intrees kan hanteer.

Vir die omskakeling van 5 V seine na 3,3 V seine is van 'n QS3L384 QuickSwitch gebruik gemaak, wat vervaardig word deur Quality Semiconductor. Dit is hoë spoed,

lae kragverbruik, CMOS 10-bis busskakelaars. Die belangrikste kenmerke van hierdie komponent kan soos volg opgesom word:

- Gebruik "Enhanced N channel" FET tegnologie
- 5Ω bidireksionele skakeling
- Zero deurskakelvertraging
- Lae kragverbruik.

Die komponent bestaan uit twee pare van vyf bidireksionele skakelaars wat geskakel word deur die twee penne $\overline{BEA}$ en $\overline{BEB}$. Die koppeling van die kragpen, VCC, is gedoen volgens die ontwerpsnota. Dit is gekoppel via 'n diode, met 0,7 V spanningsval, aan 5 V, en met 'n 10 kΩ bloeiweerstand na grond. Die spanning van die VCC pen is dus 4,3 V. Volgens die datavel sal die uittree vir 'n 5 V intree, altyd 1 V minder as die spanning van VCC wees, en dus gelyk aan 3,3 V. Figuur 5.6 toon die opstelling hiervan. [16]

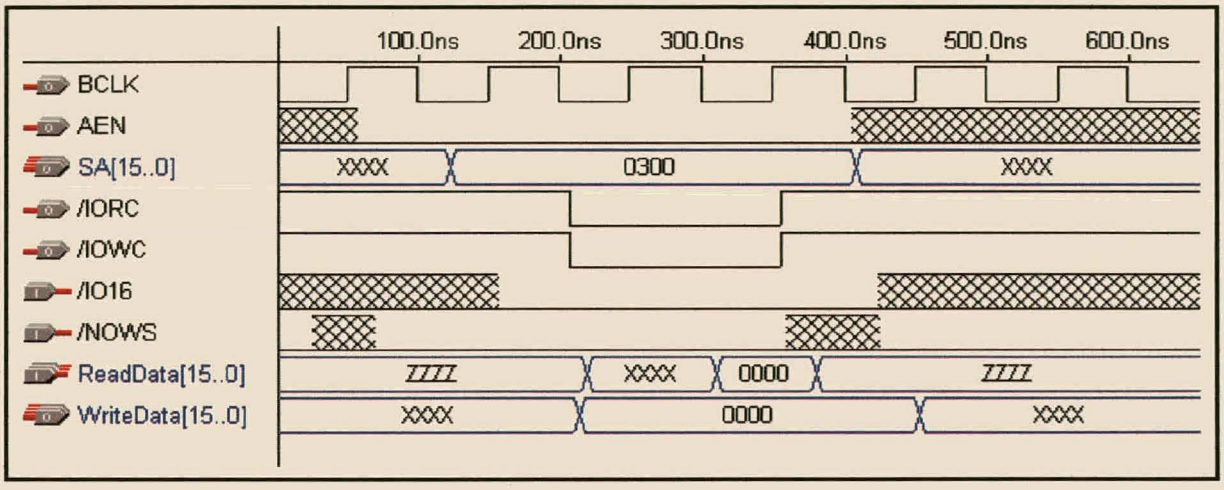



Figuur 5.6 : Opstelling van QS3L384 vir omskakeling van 5V na 3V seine [17]

### 5.2.4 ISA-koppelvlak

Daar is van die ISA-koppelvlak gebruik gemaak vir die skakeling tussen die persoonlike rekenaar en die geïmplementeerde stelsel. Die rekenaar dien dus as beide bron en ontvanger van die data. Die korrektheid van die data kan so effektief getoets word. 5 V krag word ook verkry vanaf die ISA-bus.

Die standaard drie-busklok 16-bis I/O toegang tot die ISA-bus is gebruik. Hierdie opstelling lewer 'n vier keer vinniger toegang as die standaard 8-bis koppeling. Dit maak nie net gebruik van dubbel die buswydte nie, maar gebruik ook net helfte die aantal busklok-siklusse per data-oordrag. As die frekwensie van die busklok geneem word as 10 MHz, is die teoretiese maksimum oordragtempo 6 MB/s. Figuur 5.7 toon die tyddiagram van 'n standaard 16-bis I/O toegang aan.




Figuur 5.7 : Tyddiagram van standaard 16-bis I/O ISA-toegang [18]

Die adreslyne, SA0-SA9, is direk aan die CPLD verbind en word vir die adres-dekodering gebruik. Die beheerlyne vanaf die ISA-bus, AEN, RESETDRV, $\overline{\text{IORC}}$ en $\overline{\text{IOWC}}$, asook die busklok, BCLK, word ook direk aan die CPLD verbind. Die beheerlyne na die ISA-bus, $\overline{\text{NOWS}}$ en $\overline{\text{IRQ}}$'s, word deur die ISA-bus hoog getrek. Om hierdie rede word oop kollektor, omkerende buffers (74HC05) gebruik om die lyne laag te trek, wanneer die sein aktief is. Omdat die beheerlyne ook nie aktief moet wees gedurende die tyd wat die CPLD konfigureer nie, is van aftrekweerstande by die buffer-intrees gebruik gemaak wat die uittrees in 'n hoë impedansietoestand plaas. Dieselfde geld vir die 16-bis toegangsbeheerlyn, $\overline{\text{IO16}}$. Wanneer hierdie lyn aktief laag getrek word, word 'n 16-bis I/O toegang forseer.

Die 16-bis databus word van die CPLD databus geskei deur twee bidireksionele drie-toestandbuffers (74AC245). Twee redes bestaan hiervoor. Eerstens is dit om te verhoed dat data op die ISA-bus geplaas word tydens CPLD-konfigurasie, en tweedens

word die CPLD-bus vir beide geheuetoegang en ISA-toegang gebruik. Tydens geheuetoegang moet verhoed word dat data op die ISA-bus geskryf word. Beide buffers word saam direk beheer vanaf die CPLD. Die aktiveringslyne ($\overline{G}$) word egter deur 'n optrekweerstand onaktief hoog getrek indien dit nie deur die CPLD gedryf word nie en plaas sodoende die buffers se I/O's in 'n hoë impedansietoestand. [18]

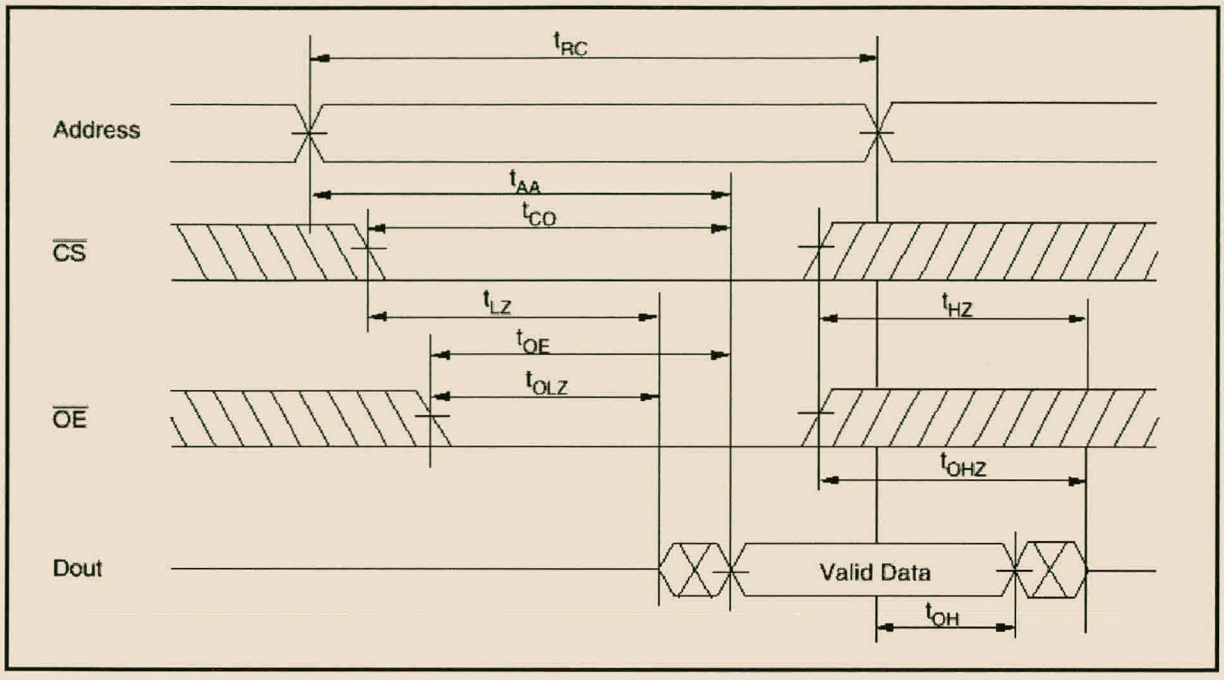
### 5.2.5 HM628512 geheue

Soos hierbo genoem, is die ISA-bus te stadig om 'n 80 Mbps bron na te boots. Om hierdie rede word die data eers in geheue gestoor waarvandaan toegang vinnig genoeg verkry kan word. Die geheue waarvan gebruik gemaak word, is die 4M SRAM HM628512 geheue vervaardig deur HITACHI. Dit is 'n 4 Mbis statiese RAM geheue, verdeel in 512 k × 8-bis posisies. Die hoofkenmerke van die komponent kan soos volg opgesom word:

- Toegangstyd = 70ns
- Lae kragverbruik – Aktief : 50mW/MHz
  – Bystand: 10µW
- Totaal statiese geheue – geen verfrissing benodig
- Drie-toestanduittrees.

Twee van hierdie geheues is gebruik om 1 megagreep geheue te realiseer. Die 8-bis I/O lyne, I/O0-I/O7, van die een geheue is verbind aan die onderste greep van die CPLD-databus en die I/O lyne van die tweede geheue is verbind aan die boonste greep van die CPLD-databus. Die geheues is dus in parallel gekoppel sodat 'n 512k × 16-bis konfigurasie verkry is. Die adreslyne A0-A18 van beide geheues is direk aan die CPLD verbind en beheer. Die beheerlyne, $\overline{CS}$, $\overline{OE}$ en $\overline{WE}$, van die twee geheues word afsonderlik deur die CPLD beheer, sodat 'n lees of skryf na slegs een geheue moontlik is.

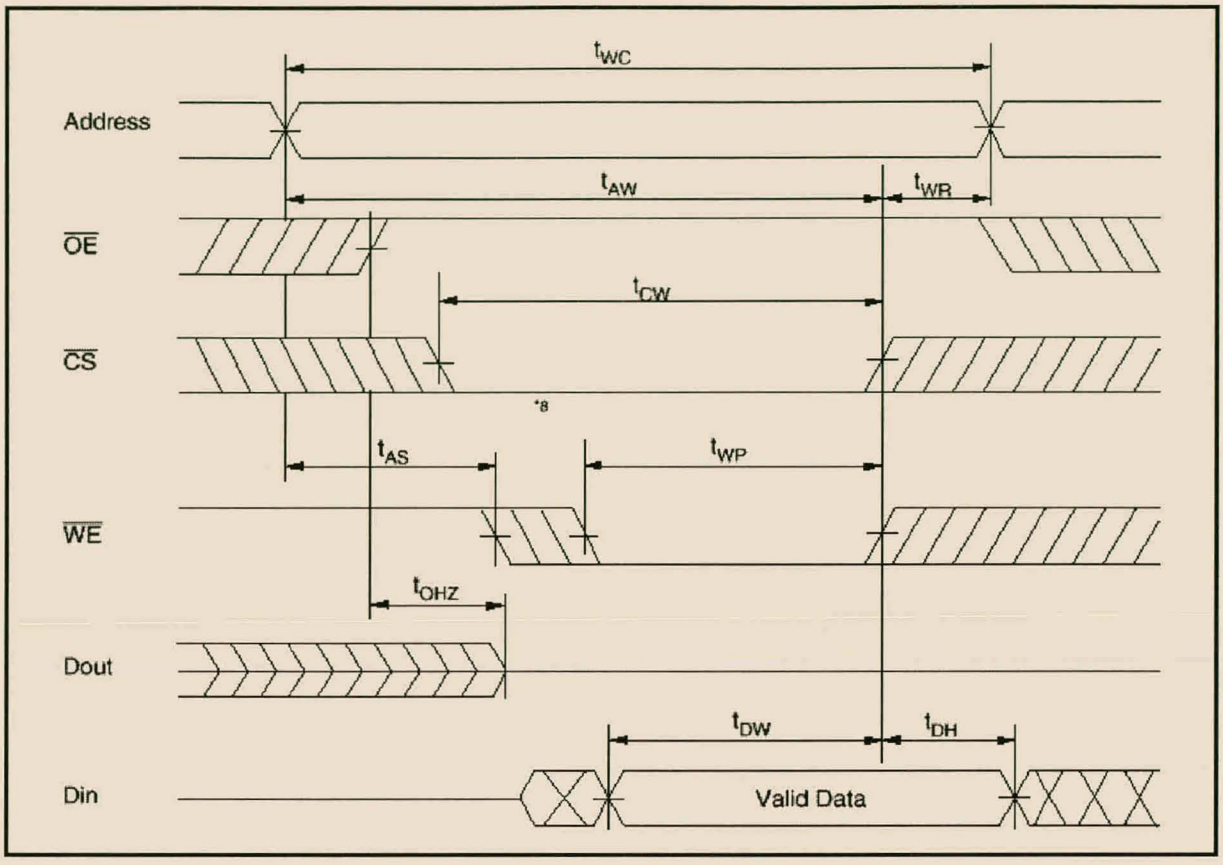
Die belangrikste kenmerk van die geheues tydens 'n lees- en skryfsiklus is dat die vinnigste toegangstyd 70 ns is. Figuur 5.8 en figuur 5.9 toon die tyddiagramme tydens 'n leessiklus en 'n skryfsiklus. Tabel 5.4 en tabel 5.5 toon die tydlengte wat op figure 5.8 en 5.9 aangedui word. [19]




Figuur 5.8 : Tyddiagram van geheue leessiklus [19]

Tabel 5.4 : Tydlengtes op figuur 5.8 aangetoon [19]

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Read cycle time | $t_{RC}$ | 70 | - | ns |
| Address access time | $t_{AA}$ | - | 70 | ns |
| Chip select access time | $t_{CO}$ | - | 70 | ns |
| Output enable to output valid | $t_{OE}$ | - | 35 | ns |
| Chip selection to output in low-Z | $t_{LZ}$ | 10 | - | ns |
| Output enable to output in low-Z | $t_{OLZ}$ | 5 | - | ns |
| Chip deselection to output in high-Z | $t_{HZ}$ | 0 | 25 | ns |
| Output disable to output in low-Z | $t_{OHZ}$ | 0 | 25 | ns |
| Output hold from address change | $t_{OH}$ | 10 | - | ns |




Figuur 5.9 : Tyddiagram van geheue skryfsiklus [19]

Tabel 5.5 : Tydlengtes op figuur 5.9 aangetoon [19]

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Write cycle time | $t_{WC}$ | 70 | - | ns |
| Chip selection to end of write | $t_{CW}$ | 60 | - | ns |
| Address setup time | $t_{AS}$ | 0 | - | ns |
| Address valid to end of write | $t_{AW}$ | 60 | - | ns |
| Write pulse width | $t_{WP}$ | 50 | - | ns |
| Write recovery time | $t_{WR}$ | 0 | - | ns |
| $\overline{WE}$ to output in high-Z | $t_{WHZ}$ | 0 | 25 | ns |
| Data to write time overlap | $t_{DW}$ | 30 | - | ns |
| Data hold from write time | $t_{DH}$ | 0 | - | ns |
| Output active from output in high-Z | $t_{OW}$ | 5 | - | ns |
| Output disable to output in high-Z | $t_{OHZ}$ | 0 | 25 | ns |

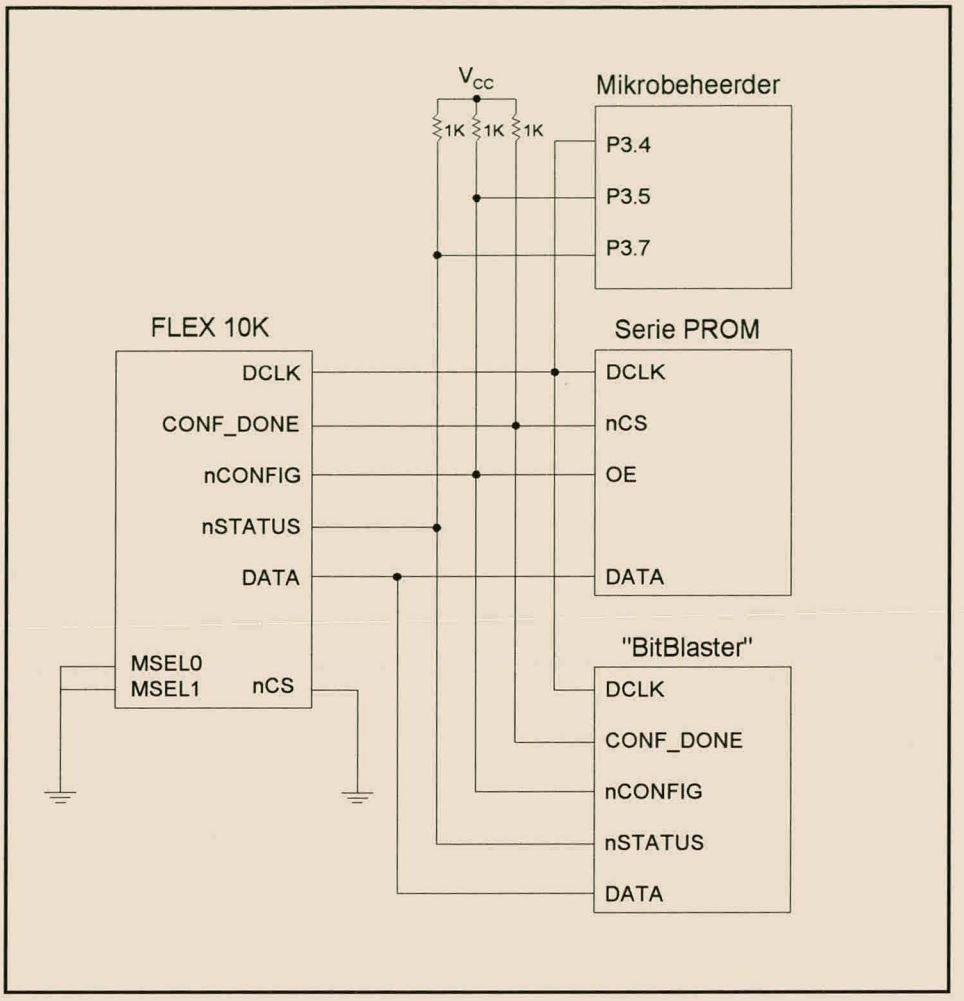
### 5.2.6 AT17Cxxx Serie-PROM

Om die CPLD te konfigureer word van 'n serie-PROM gebruik gemaak. Die konfigurasiedata word serieel, sinchroon met 'n kloksein in die CPLD ingeklok. Die FLEX10K CPLD-familie besit egter nie 'n interne klokgenerator vir die opwek van die kloksein nie.

Die serie-PROM wat aanbeveel is om saam met dié CPLD te gebruik is die EPC1 of die EPC1441, vervaardig deur ALTERA. Die serie-PROM's besit 'n interne klokgenerator wat die data in die CPLD kan inklok. Die nadeel is egter dat dit slegs een maal programmeerbaar is. 'n Alternatief is die AT17Cxxx reeks vervaardig deur ATMEL. Hierdie serie-PROM's is herprogrammeerbaar, maar slegs die AT17C512 en AT17C 010 beskik oor 'n interne ossillator vir die opwek van 'n kloksein. [21]

Vir die ontwerp is voorsiening gemaak vir 'n AT17C128 serie-PROM met 'n konfigurasie van 128 k × 1. 'n Kloksein moes dus ekstern opgewek word vir die uitklok van data uit die serie-PROM en die inklok daarvan in die CPLD. Hiervoor is van 'n AT98C2051 mikrobeheerder gebruik gemaak.

Die DATA-pen is aan die data-intree van die CPLD verbind. Die DCLK-penne van beide die serie-PROM en CPLD is aan die mikrobeheerder verbind. Die beheerpenne, OE en nCS, is verbind aan die beheerpenne, nSTATUS en CONF_DONE, van die CPLD. Beide die seine is van optrekweerstande voorsien. Die opstelling om die CPLD te konfigureer word in figuur 5.10 getoon. [20]




Figuur 5.10 : Opstelling vir die konfigurasie van die CPLD

### 5.2.7 AT89C2051 Mikrobeheerder

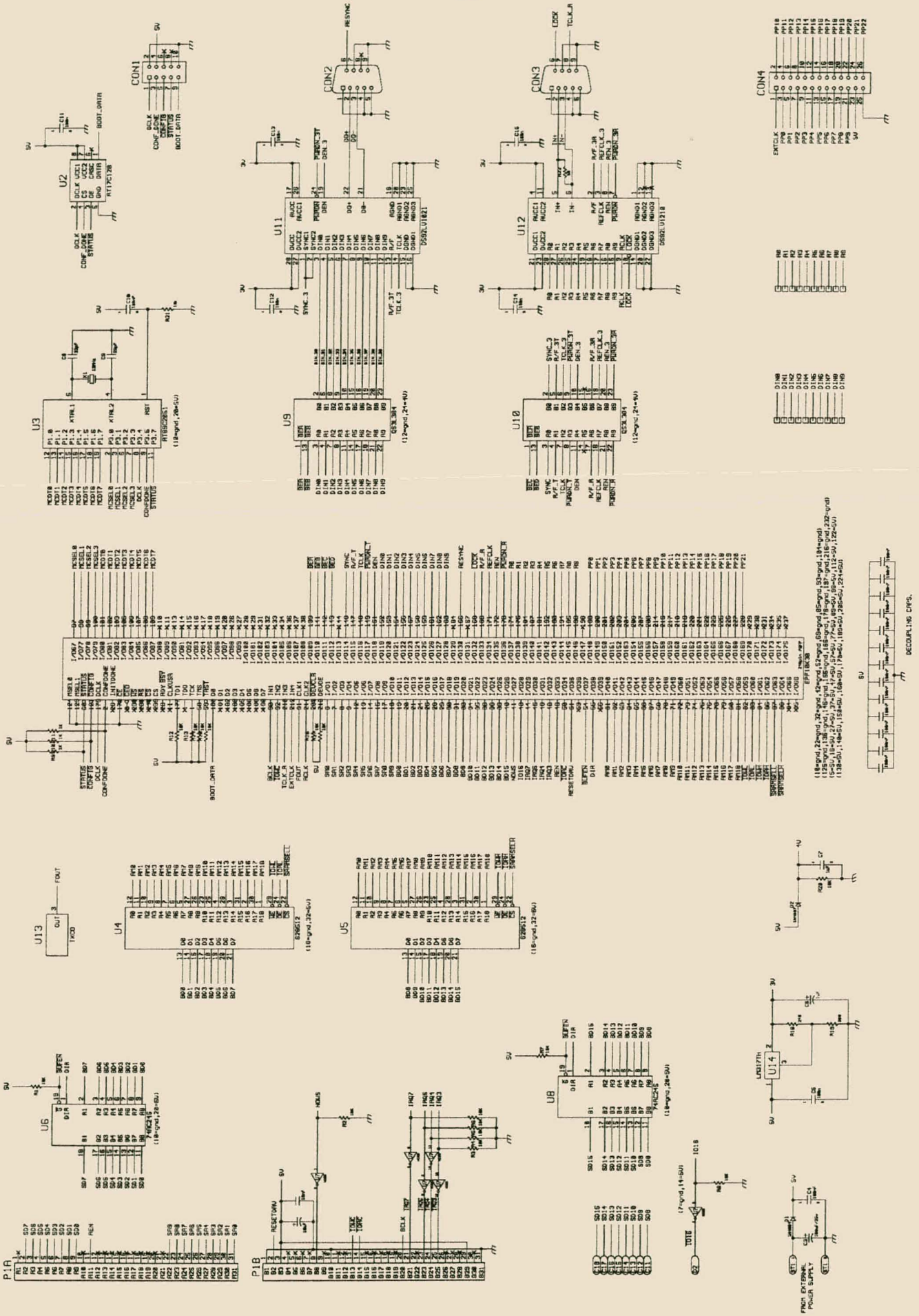
Soos reeds genoem kan die FLEX10K reeks CPLD's serieel gekonfigureer word deur 'n serie-PROM, maar dit kan egter nie die kloksein genereer waarop die data geklok moet word nie. Die spesifieke serie-PROM, AT17C128, waarvoor voorsiening gemaak is, het ook nie 'n klokgenerator ingebou nie. Om dus die kloksein op te wek word van 'n AT89C2051 mikrobeheerder van ATMEL gebruik gemaak. Hierdie mikrobeheerder beskik oor 2KB interne programgeheue en 128 × 8-bis interne RAM.

Die beheerseine, nSTATUS en CONF_DONE, vanaf die CPLD is verbind aan P3.7 en P3.6. Afhangende van die toestand van die seine, word die kloksein opgewek en op pen P3.4 uitgeklok. Die pen is aan DCLK, op die CPLD en serie-PROM verbind. Bylae A bevat die programkode wat in die AT89C2051 gelaai is. Die mikrobeheerder se orige I/O penne is direk aan die CPLD verbind. [22]

### 5.2.8 Eksterne klokgenerator

Die hoofklok op die ontwerp is afkomstig van 'n kristal-ossillator pakkie. Die opstelling is so ontwerp dat pakkies van verskillende frekwensies getoets kan word. Die uittree daarvan is direk verbind aan 'n globale klok-intreepen op die CPLD. Die CPLD kan die kloksein afdeel en voorsien al die ander klokseine op die bord.

Die skematiese diagram van die totale apparatuur uitleg, word in figuur 5.11 getoon.



Figuur 5.11 : Skematiese diagram van die totale apparatuurontwerp

# 6. Digitale ontwerp

Vir hierdie ontwerp word aangeneem dat data teen ongeveer 80 Mbps gestuur moet word. In die vorige hoofstuk is 'n LVDS-sender met serialiseerder en ontvanger met deserialiseerder gekies, wat data teen 400 Mbps oor 10 meter kabel kan stuur. Die data word 10 bisse op 'n slag, parallel teen 40 MHz sinchroon met 'n kloksein ingeklok en aan die ontvangkant weer uitgeklok. Foute wat meestal gepaard gaan met die verloor van sinchronisasie, kan egter voorkom. Om hierdie rede moet 'n protokol ontwikkel word om te verhoed dat data verlore gaan. So 'n voorgestelde protokol is reeds in hoofstuk 4 bespreek.

Deur die koppelvlakke in 'n CPLD te implementeer word die ontwerp daarvan vanaf vaste hardeware na herkonfigureerbare hardeware verskuif. Dit het die voordeel dat die ontwerp maklik aangepas kan word, en die kompleksiteit op fisiese vlak baie vereenvoudig word. 'n Protokol vir die stelsel kan ook in die CPLD geïmplementeer word sonder om aan die fisiese uitleg daarvan te verander.

Sekere van die komplekse eenhede wat vir die protokol benodig word, bestaan reeds as boublokke wat direk in die ontwerp gebruik kan word. ALTERA se Megafunctions komponente en SYNOPSYS se DesignWare komponente is vir die doel ondersoek. ALTERA-komponente is ondersoek omdat 'n ALTERA vervaardigde CPLD gebruik is. Die programmeringssagteware, MaxPlus II, is dus in staat om die komponent vir die spesifieke CPLD te optimaliseer. Die SYNOPSYS-komponente is ondersoek omdat die CPLD waarin dit geïmplementeer gaan word, gespesifiseer kan word en die komponent daarvoor geoptimaliseer kan word.

**Synopsys DesignWare-Komponente**

Synopsys se DesignWare-komponente is 'n versameling van komponent boublokke wat direk in ontwerpe gebruik kan word. Dit is vooraf ontwerp en getoets, en is nou met die ontwikkellingsomgewing van Synopsys geïntegreer. Dit laat Synopsys toe om 'n hoë vlak van optimalisering daarop te kan toepas. Die komponente is tegnologie-onafhanklik, en die veranderbare parameters van die komponent maak dit maklik

aanpasbaar vir verskillende ontwerpe. DesignWare-komponente kan in die volgende biblioteke verdeel word:

- *Standard Library* – Standaard algoritmiese en logiese funksies
- *Foundation Library* – Meer gevorderde algoritmiese en sekwensiële logiese funksies
  - ALU komponente
  - Gevorderde wiskundige komponente
  - Basiese sekwensiële komponente
  - Databetroubaarheids komponente
  - Ontfoutings en toets komponente
- *DSP Library* – Digitale filters
- *GTECH Library* – Tegnologie-onafhanklike hekvlak biblioteek.

Om die DesignWare-komponente te gebruik in 'n ontwerp, moet dit eers in 'n VHDL-komponent ingetrek word. Die VHDL-komponent word dan in SYNOPSYS se Design Analyzer ingelaai, en vir 'n spesifieke CPLD, plek- of spoedsgewys, geoptimeer. Die geoptimeerde komponent word dan in 'n EDIF-formaat gestoor om in die ontwerp gebruik te word. In die ontwerp word die EDIF-lêer as 'n komponent ingetrek, deur die CPLD-sagteware geplaas, en uitgelê. [23]

**ALTERA Megafunctions-komponente**

ALTERA Megafunctions is hoë vlak boublokke wat in CPLD-ontwerpe gebruik kan word, en deur die MaxPlus II sagteware geïmplementeer word.. 'n Biblioteek van komponente, wat sekere LPM-komponente ("Library of Paramaterized Modules") insluit, vorm deel van die MaxPlus II programmerings sagteware. Die komponente is tegnologie onafhanklik en kan vir 'n ontwerp aangepas word deur die veranderbare parameters op te stel.

Die Megafunction-komponente kan in die volgende groepe ingedeel word:

- *Algoritmiese komponente* – Basiese algoritmiese funksies, tellers en vergelykers
- *Hekvlak komponente* – Bestaan uit basiese logiese hekke
- *Geheue komponente* – Bestaan uit die basiese geheue-komponente soos wipkringe en registers, asook FIFO's, RAM's en ROM's.

Ekstra Megafunctions word ook van Altera Megafunction Partners Program (AMPP) en Altera MegaCore ontwikkel, maar moet ekstra aangekoop word.

Die Megafunctions kan geïmplementeer word deur van 'n MaxPlus II-funksie, die MegaWizard, gebruik te maak. Die komponent word gekies en die veranderbare parameters word opgestel. 'n AHDL, VHDL, of Verilog HDL-komponent word dan geskep wat in die ontwerp ingetrek kan word. [24]

## 6.1 Implementering van ISA-koppelvlak

Daar word van die ISA-bus gebruik gemaak om met die persoonlike rekenaar te kommunikeer. 'n ISA-koppelvlak moet dus geïmplementeer word. Slegs die implementering van die koppelvlak word in hierdie afdeling bespreek. Die implementering van die protokol-eenhede word in volgende afdelings bespreek.

ISA-adresse is vir die volgende drie aksies benodig en 'n onbesette adres is aan elk toegeken:

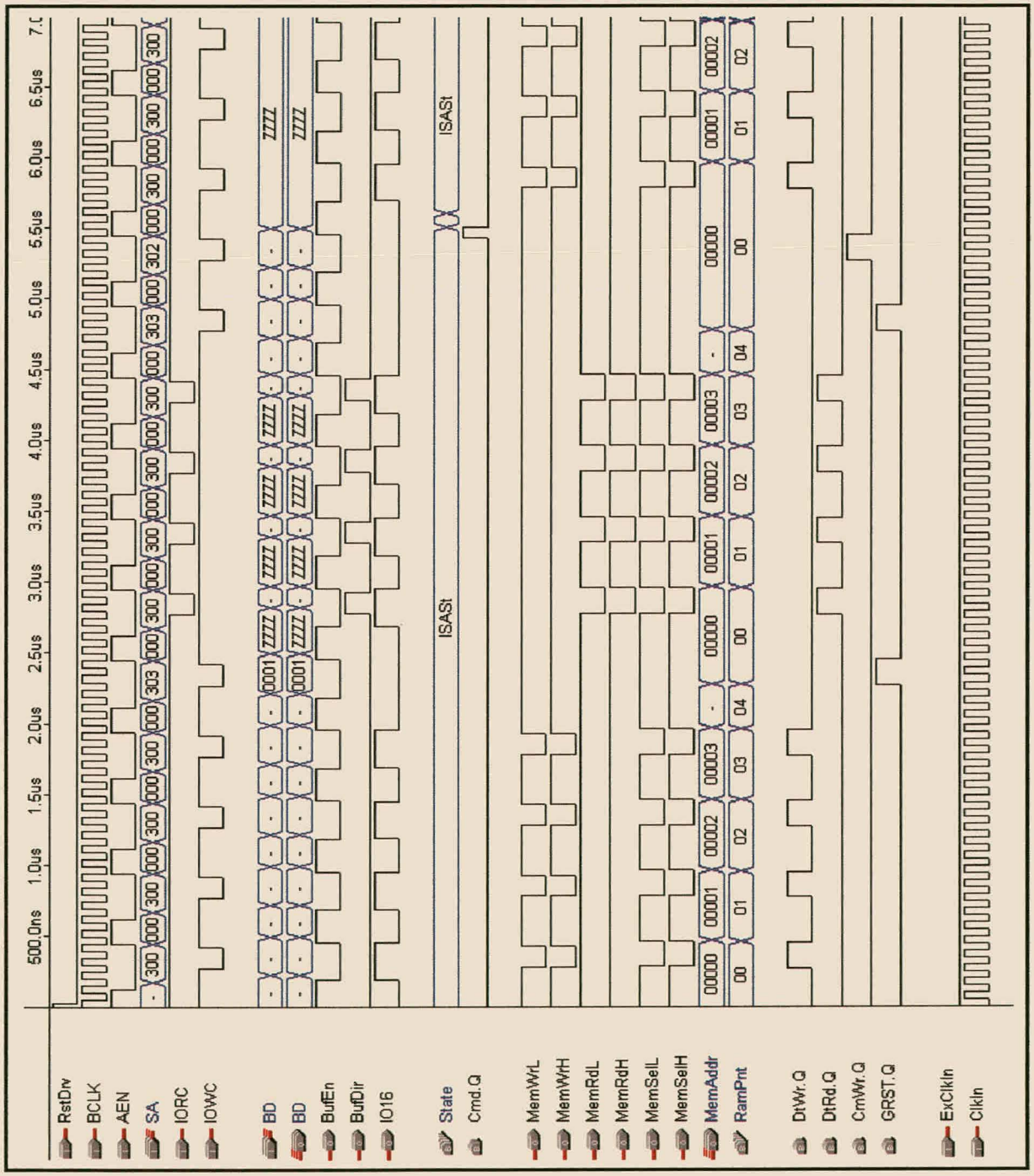
- Vir die lees en skryf van data – 300H
- Vir die skryf van 'n bevel – 301H
- Vir 'n globale herstelling van alle registers– 302H.

Met die lees en skryf van data, word 'n 16-bis ISA I/O-oordrag uitgevoer. 16-Bis data word van die geheue op die stelsel gelees, of daarheen geskryf. Alle beheer van die geheue word outomaties uitgevoer, en die geheue adresregister word herstel deur 'n globale herstelbevel. 'n Skryf van 'n bevel, beveel die stelsel om 'n sekere aksie uit te voer. Dit word gedoen deur 'n bevelwaarde na adres 301H te skryf. Met die skryf van enige waarde na adres 302H, word 'n globale herstel van alle toestande en registers gedoen.

Daar is van 'n toestandsmasjien gebruik gemaak om beheer oor die koppelvlak uit te oefen. Die begintoestand van die toestandsmasjien, is die "ISASt" waartydens toegang vanaf die persoonlike rekenaar tot die stelsel verkry kan word. Voorsiening is vir bykomende toestande gemaak. Die toestand waarin die toestandsmasjien verkeer, word

deur die bevelregister bepaal, en kan deur die ISA-herstellyn of globale herstelbevel in die begintoestand geplaas word.

Die VHDL-kode word in bylae B getoon. Figuur 6.1 toon tydsimulasies van die ISA-koppelvlak aan wat in MaxPlus II verkry is.



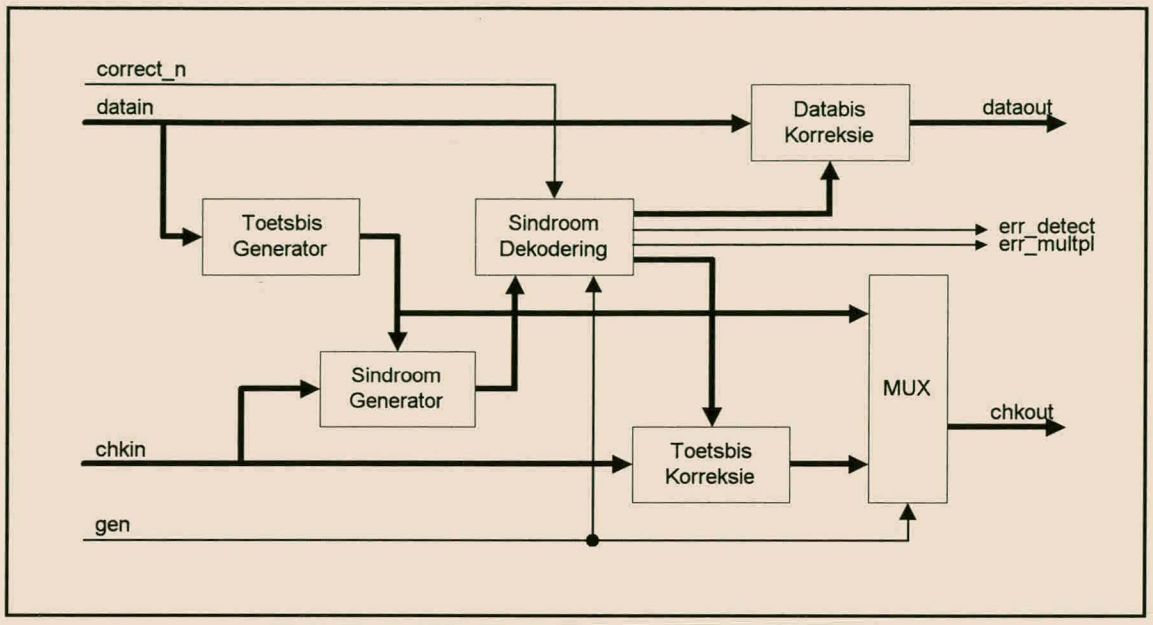

Figuur 6.1 : Tydsimulasie van ISA-koppelvlak

## 6.2 Implementering van EDAC

'n EDAC word aan die stuur- en ontvangkant geïmplementeer om moontlike enkel bisfoute wat tydens die tydelike storing en transmissie mag plaasvind, te identifiseer en korrigeer, sonder dat dit nodig is om die data te herversend. Drie moontlikhede vir die implementering van die EDAC bestaan.

Die eerste opsie is om die EDAC self te ontwikkel. Die nadeel hiervan egter is die tyd verbonde aan die ontwerp en ontwikkeling van iets wat reeds bestaan. Die tweede opsie is om van 'n ALTERA ontwerpte Megafunctions-komponent gebruik te maak. So 'n komponent is egter nie as deel van MaxPlus II beskikbaar nie, en sou aangekoop moes word. Die derde, en mees geskikte opsie, was om van 'n SYNOPSYS-ontwerpte DesignWare-komponent gebruik te maak.

Daar is van SYNOPSYS se DesignWare DW_ecc-komponent gebruik gemaak. Die wydte van die data (WIDTH) en die toetsbisse (CHKBITS) is twee statiese parameters wat tydens die ontwerp gespesifiseer moet word. Daar moet aan 'n minimum wydte van toetsbisse ten opsigte van die datawydte voldoen word. 'n Derde statiese parameter (SYND_SEL) wat bepaal of tydens 'n leessiklus die toetsbisse moet verskyn, en of die fout sindroom moet verskyn, word gespesifiseer. Vir al die implementasies wat volg, moet die toetsbisse verskyn (SYND_SEL = '0'). Figuur 6.2 toon die blokdiagram van die komponent.




Figuur 6.2 : Blokdiagram van die SYNOPSYS DW_ecc-komponent [25]

Die intree GEN plaas die EDAC in 'n skryf- (GEN = HOOG) of leesmodus (GEN = LAAG). In skryfmodus word toetsbisse gegenereer, en in leesmodus word die data en toetsbisse ingelees en gekorrigeer. Die korrigering van data word deur die intree, CORRECT_N, in staat gestel. Die intrees DATAIN (wydte WIDTH) en CHKIN (wydte CHKBITS) is die intrees van die data- en toetsbisse.

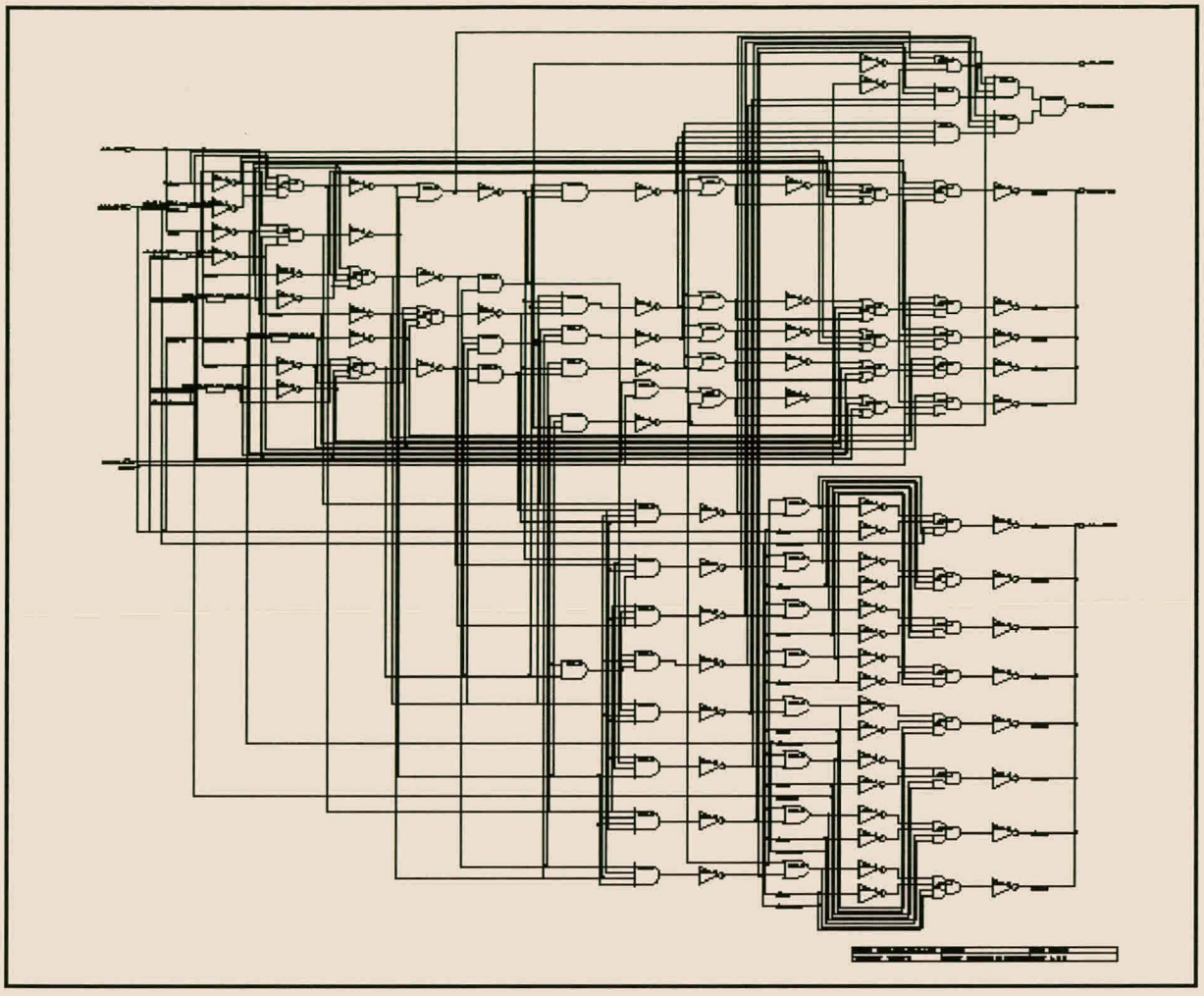
Daar word van twee uittrees gebruik gemaak om korrigeerbare enkel bisfoute, en onkorrigeerbare veelvuldige bisfoute aan te toon, naamlik ERR_DETECT en ERR_MULTPL. Die uittree DATAOUT (wydte WIDTH) is die uittree vir die gekorrigeerde data, indien CORRECT_N aktief is, en 'n korrigeerbare fout voorkom. Die uittree CHKOUT is die uittree van die berekende toetsbisse in leesmodus, en die gekorrigeerde toetsbisse in skryfmodus. Tabel 6.1 toon 'n waarheidstabel van die komponent. [25]

Tabel 6.1 : Waarheidstabel van SYNOPSYS se DW_ecc-komponent [25]

| GEN | CORRECT_N | ERR_DETECT | ERR_MULTPL | Foutstatus, aksie |
|---|---|---|---|---|
| 1 | X | 0 | 0 | Geen deteksie en korreksie |
| 0 | 0 | 0 | 0 | Geen fout |
| 0 | 0 | 1 | 0 | Enkelfout, gekorrigeer |
| 0 | 0 | 1 | 1 | Veelvuldige fout, geen korreksie |
| 0 | 1 | 0 | 0 | Geen fout |
| 0 | 1 | 1 | 0 | Enkelfout, geen korreksie |
| 0 | 1 | 1 | 1 | Veelvuldige fout, geen korreksie |

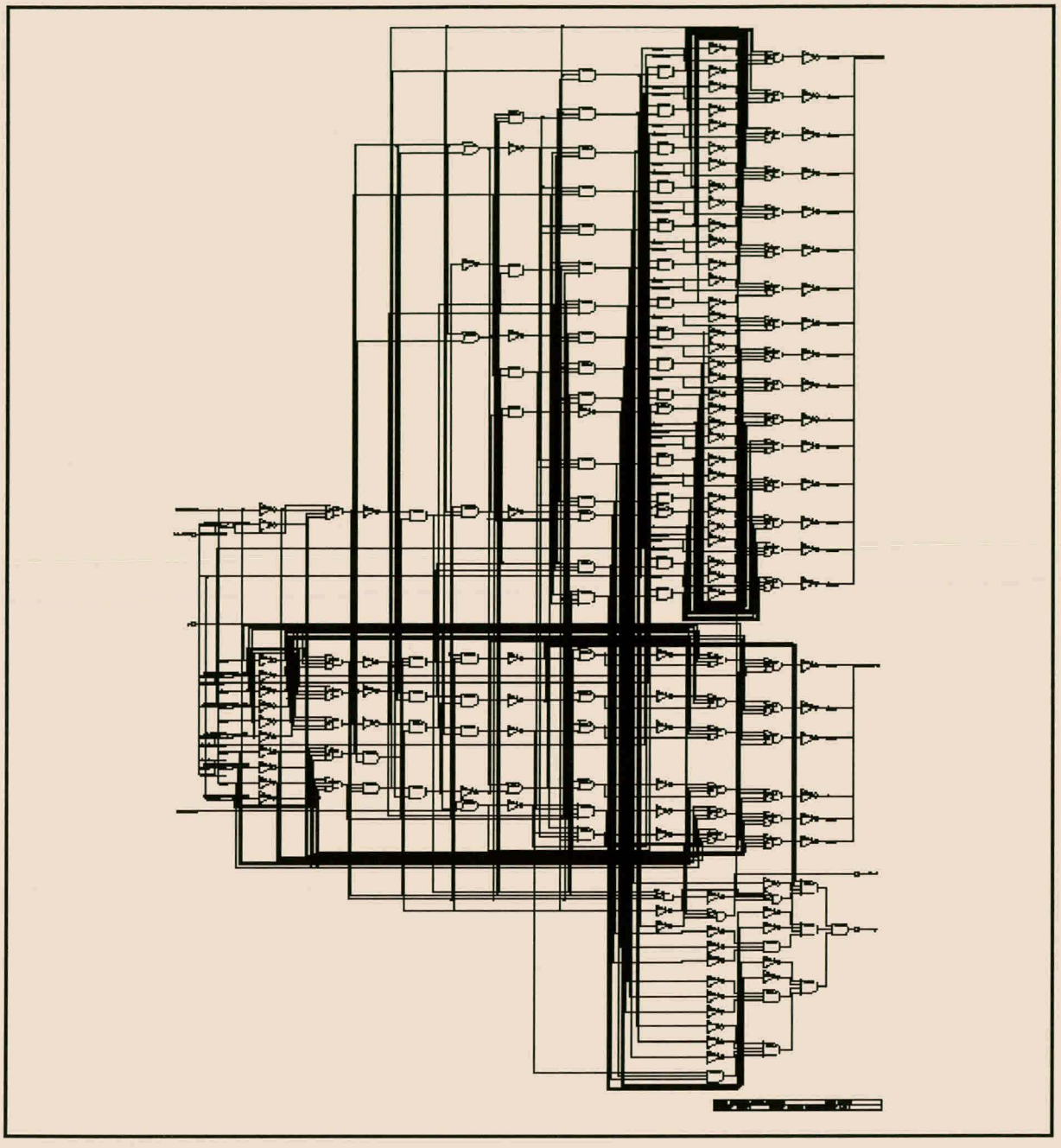
Twee kombinasies van die komponent met verskillende data- en toetsbiswydtes is geëvalueer. VHDL-kode is geskryf om die twee verskillende kombinasies van die DW_ecc komponent in SYSNOPSYS se DESIGN ANALYZER te laai en vir die FLEX 10K CPLD-familie te optimaliseer. Bylae C bevat die VHDL-kode wat hiervoor geskryf is. Elk word dan as 'n nuwe EDIF-komponent gestoor wat weer in ALTERA se MaxPlus II ingelaai kan word.

Die eerste komponent-implementasie is 'n 8-bis data-intree met 5 toetsbisse. Die effektiwiteit van so 'n opstelling is nie baie goed nie, aangesien slegs 8 van die uiteindelike 13 bisse, data voorstel (effektiwiteit = 8/13 = 61,5%). Die opstelling gebruik egter die minste logiese selle (LC) en is ook die vinnigste. Figuur 6.3 toon die skematiese diagram verkry deur SYSNOPSYS se DESIGN ANALYZER.



Figuur 6.3 : 'n 8-bisdata-5-toetsbis-EDAC

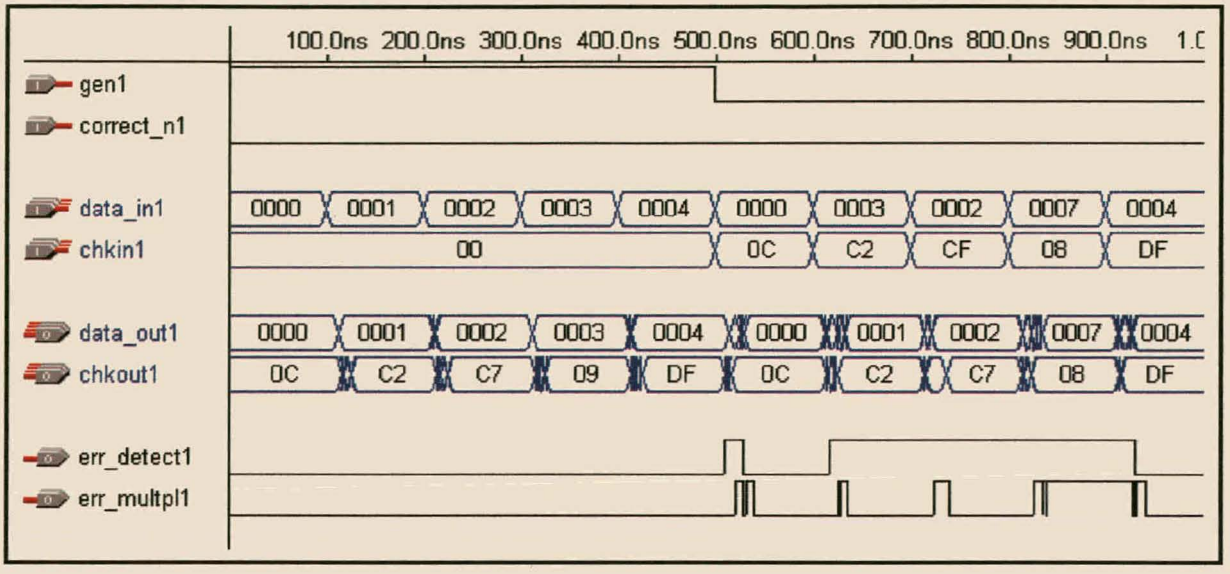
Die tweede komponent-implementasie is 'n 16-bis data-intree met 6 toetsbisse. Die opstelling se effektiwiteit is heelwat beter met 16 bisse data uit 'n totaal van 22 bisse (effektiwiteit = 16/22 = 72,7%), maar heelwat meer logiese selle word gebruik en dit is stadiger. Figuur 6.4 toon die skematiese diagram verkry deur SYSNOPSYS se DESIGN ANALYZER.



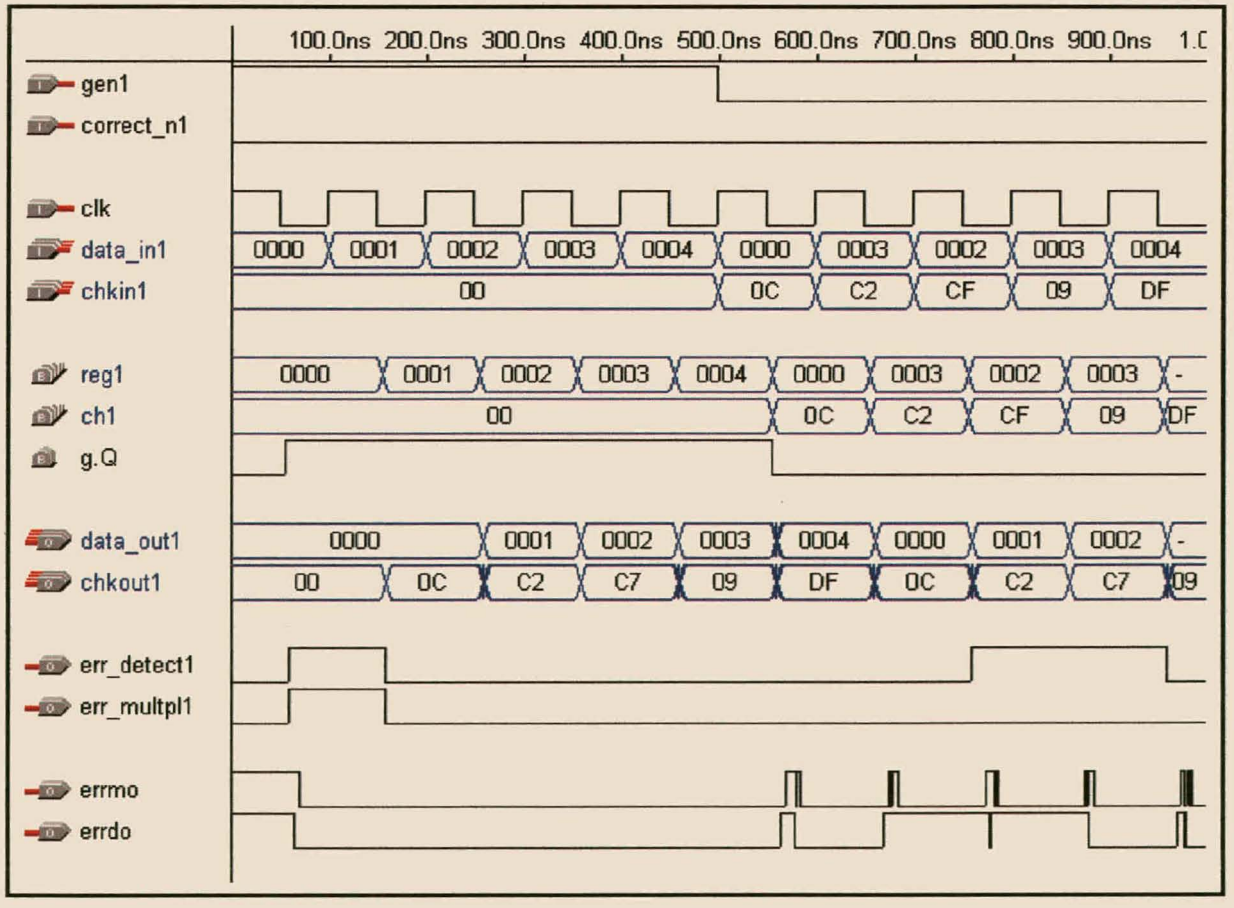
Figuur 6.4 : 'n 16-bis-data-6-toetsbis-EDAC

Twee kombinasies van VHDL-kode is geskryf om die geskepte EDIF-komponente in 'n CPLD te implementeer en evalueer. Die eerste was om die komponente asinchroon te toets, dit wil sê die tydvertraging van data deur slegs die komponent. Die tweede was om die komponent sinchroon saam met 'n kloksein te toets en te ondersoek wat die maksimum klokfrekwensie is wat verkry kan word. Die data- en beheerlyne word voor en na die EDAC in registers geklok. Die VHDL-kode vir die twee implementasies word ook in bylae C getoon.

Figuur 6.5 en figuur 6.6 toon die werking van die asinchrone en sinchrone implementasie.




Figuur 6.5 : Werking van die asinchrone EDAC implementasie




Figuur 6.6 : Werking van sinchrone EDAC implementasie

Vir elk van die twee EDAC-komponente hierbo is 'n asinchrone ontwerp en 'n sinchrone ontwerp gedoen. Elke ontwerp is in ALTERA se MaxPlus II gewoon geoptimaliseer, asook vir spoed geoptimaliseer. Tabel 6.2 toon 'n vergelyking van die komponente vir die asinchrone ontwerpe en tabel 6.3 toon 'n vergelyking vir die sinchrone ontwerpe.

Tabel 6.2 : Vergelyking van asinchrone ontwerpe

| Komponent | Grootte van gewone optimalisering | Grootte van spoedoptimalisering |
|---|---|---|
| edac8_5 | 57 LCs (3%) | 59 LCs (3 %) |
| edac16_6 | 89 LCs (5 %) | 101 LCs ( 5%) |

Tabel 6.3 : Vergelyking van sinchrone ontwerpe

| Komponent | Grootte van gewone optimalisering | Maksimum frekwensie van gewone optimalisering | Grootte van spoed-optimalisering | Maksimum frekwensie van spoed-optimalisering |
|---|---|---|---|---|
| edac8_5 | 27,62 MHz | 72 LCs (4 %) | 35,46 MHz | 75 LCs (4 %) |
| edac16_6 | 22.12 MHZ | 113 LCs (6 %) | 29,94 MHz | 120 LC (6 %) |

**Gevolgtrekking**

Die 8-databis-5-toetsbis opstelling funksioneer vinniger, en gebruik minder spasie as die 16-databis-6-toetsbis opstelling. Dit is egter belangrik om daarop te let dat die 16-databis-6-toetsbis opstelling eerstens 'n hoër effektiwiteit lewer, en tweedens dubbeld die hoeveelheid data verwerk (dit wil sê dubbel die oordragtempo van die 8-databis-5-toetsbis opstelling). Die nadeel is dit maak egter van ±50 % meer LEs gebruik.

## 6.3 Implementering van FIFO

Aan die stuurkant van die stelsel word 'n FIFO geïmplementeer, sodat die kamera nie onderbreek word indien die data nie onmiddellik gestuur kan word nie. Dieselfde drie moontlikhede as vir die EDAC bestaan vir die implementering van die FIFO, naamlik om dit self te skryf, van 'n ALTERA Megafunction-komponent gebruik te maak, of om 'n SYNOPSYS-komponent te gebruik.

Om so 'n FIFO self te implementeer sou tydrowend wees, en dieselfde mate van optimalisering sal nie verkry kan word nie. Beide sinchrone en asinchrone FIFO's is

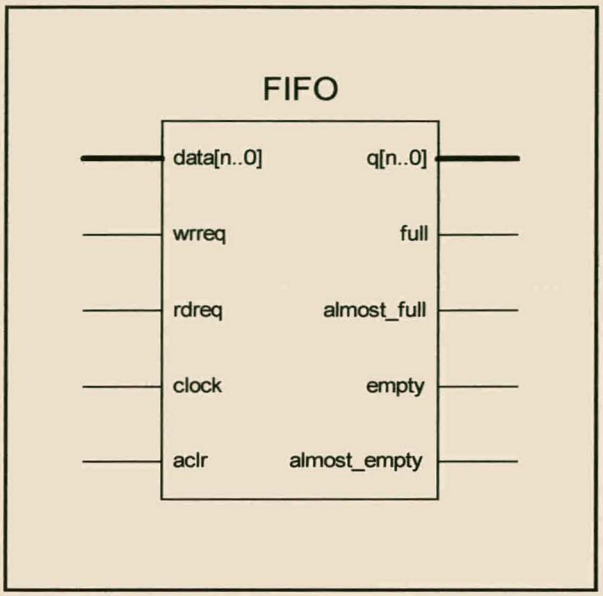
verkrygbaar as SYNOPSYS DesignWare-komponent. [27] Die bruikbaarheid van die komponente is egter beperk deur die volgende twee faktore:

- Eerstens is die maksimum diepte van die FIFO's 256 bisse. Indien groter FIFO's geïmplementeer wil word, moet van 'n DesignWare FIFO-beheerder saam met 'n eksterne dubbelpoort-geheue gebruik gemaak word.
- Alhoewel die FIFO vir die FLEX 10K CPLD-familie geoptimaliseer kan word, maak dit nie gebruik van die spesiale EAB van dié CPLD-familie nie. Dus word 'n baie groot CPLD met 'n groot aantal LEs, benodig.

Daar is van die ALTERA FIFO Megafunction gebruik gemaak vir die implementering van die FIFO. Statiese parameters wat tydens die opstelling van die komponent gespesifiseer moet word is die wydte en die diepte van die FIFO, asook 'n sinchrone, of 'n asinchrone FIFO wat geïmplementeer moet word. Die volgende vier vlaggies wat die toestand van die FIFO aandui moet ook gespesifiseer word:

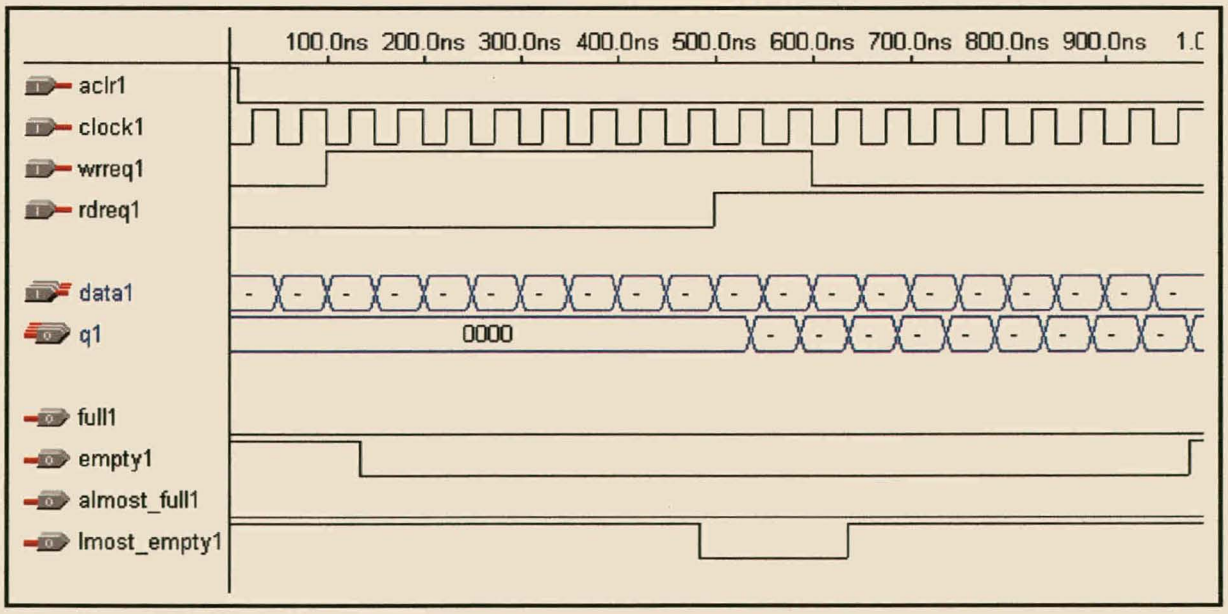
- Vol vlaggie ("full") – Aktief as alle geheueposisies gebruik is
- Leë vlaggie ("empty") – Aktief as geen van die geheue posisies gebruik is nie
- Amper-vol vlaggie ("almost_full") – Aktief as meer as 'n gespesifiseerde hoeveelheid geheue gebruik is
- Amper-leë vlaggie ("almost_empty") – Aktief as minder as 'n gespesifiseerde hoeveelheid geheue gebruik is

Daar is geselekteer om van 'n sinchrone FIFO gebruik te maak, omdat die FIFO dan in die EAB's geïmplementeer word. Die asinchrone FIFO is te kompleks en moet in LEs geïmplementeer word, wat weereens lei tot die nodigheid van 'n baie groter CPLD. Figuur 6.7 toon die blokdiagram van die geïmplementeerde FIFO.




Figuur 6.7 : Blokdiagram van die ALTERA Megafunction FIFO

Indien die skryfversoeklyn, WRREQ, aktief is, word die data sinchroon op die opgaande rand van die kloksein, CLOCK, by die ingang, DATA, onder in die FIFO ingeskryf. Indien die leesversoeklyn, RDREQ, aktief is, word data by die uitgang, Q, sinchroon op die opgaande rand van CLOCK, bo uit die FIFO gelees. Die VHDL-kode vir 'n 512 × 16 FIFO-komponent, wat deur die Megafunction Wizard geskep is, word in bylae D getoon. Figuur 6.8 toon 'n tyddiagram van die werking van die FIFO-komponent.




Figuur 6.8 : Tyddiagram om die werking van die FIFO-komponent aan te toon

VHDL-kode is geskryf om verskillende kombinasiegroottes van die FIFO te ondersoek, en word ook in bylae D getoon. Die volgende ses kombinasies is ondersoek:

- 245-Greep FIFOs – 256 × 8 en 128 × 16
- 512-Greep FIFOs – 512 × 8 en 256 × 16
- 1024-Greep FIFOs – 1024 × 8 en 512 × 16.

Elk van die ses kombinasies is in MaxPlus geïmplementeer en eers gewoon geoptimaliseer, en daarna vir spoed geoptimaliseer. Tabel 6.4 toon die vergelyking in grootte en spoed tussen die ses FIFO's.

Tabel 6.4 : Vergelyking van verskillende FIFO's ten opsigte van grootte en spoed

| FIFO | Grootte | Spoed | Grootte | Spoed | Geheue grootte |
|---|---|---|---|---|---|
| | Gewoon | geoptimaliseer | Spoed | geoptimaliseer | |
| 256 × 8 | 276 LEs | 26,73 MHz | 273 LEs | 44,84 MHz | 2048 bisse |
| 512 × 8 | 299 LEs | 27,70 MHz | 282 LEs | 40,48 MHz | 4096 bisse |
| 1024 × 8 | 307 LEs | 22,77 MHz | 293 LEs | 38,61 MHz | 8192 bisse |
| 128 × 16 | 384 LEs | 31,05 MHz | 386 LEs | 43,10 MHz | 2048 bisse |
| 256 × 16 | 396 LEs | 25,12 MHz | 393 LEs | 36,10 MHz | 4096 bisse |
| 512 × 16 | 418 LEs | 25,57 MHz | 402 LEs | 41,66 MHz | 8192 bisse |

**Gevolgtrekking**

Kleiner geïmplementeerde FIFOs neem minder LEs in beslag, en funksioneer in die algemeen vinniger. Hierdie parameters is natuurlik baie afhanklik van die plasing en konneksie wat binne die CPLD gemaak word. Dit is baie belangrik om daarop te let dat die 16-bis wye FIFOs dubbel die hoeveelheid data op 'n keer oordra. As byvoorbeeld gekyk word na die twee 8192-bis FIFOs, kan dubbel die bisoordragtempo met die 512 × 16 FIFO verkry word as met die 1024 × 8 FIFO, maar ten koste van ±100 LEs (33 %) meer.

# 7. Gevolgtrekkings en Aanbevelings

## 7.1 Gevolgtrekkings

In hoofstuk 2 is die huidige implementasie van die hoë spoed koppelvlak tussen die hoofkamera en die massageheue beskryf. Die beelddata word analoog parallel tot by die massageheue oorgedra, waar dit versyfer word en in die massageheue ingeskryf word. Komplekse kabel opstelling gaan hiermee gepaard, en die invloed van ruis daarop is hoog.

Die volgende generasie kamera gaan as 'n funksionele eenheid ontwikkel word. Die eenheid gaan al die nodige klokseine opwek, kamerabeheer behartig, en die beelddata versyfer. 'n Groot hoeveelheid data gaan dus teen 'n hoë tempo gegenereer word, en moet na die massageheue oorgedra word sodat dit gestoor kan word.

Die doel van die tesis, soos in hoofstuk 3 genoem, is om 'n geskikte tegnologie te vind waarmee 'n hoë spoed digitale koppelvlak geïmplementeer kan word, asook die ontwikkeling van die koppelvlak. Die belangrikste vereistes waaraan die koppelvlak moet voldoen is die volgende:

- Dit moet 'n hoë bandwydte kan handhaaf
- Die kragverbruik daarvan moet laag wees
- Die invloed van ruis moet laag wees
- Die elektromagnetiese uitstraling daarvan moet laag wees
- Dit moet min plek in beslag neem.

LVDS voldoen aan al hierdie vereistes. Dit maak gebruik van klein differensiële spannings (±350 mV) vir die transmissie van data oor twee geleiers. Om hierdie rede kan hoë bandwydtes (655 Mbps) en lae kragverbruik (≈1,2 mW) verkry word.

Die LVDS-komponent wat vir die implementasie gebruik is, kan volgens spesifikasies data teen 400 Mbps oor tot 10 m lengte kabel versend. 'n Eenvoudige opstelling is gemaak waarmee die maksimum bandwydte en lengte bepaal kan word, voordat foutiewe data ontvang word, of die ontvanger sinchronisasie verloor.

Daar is begin deur data teen 160 Mbps oor 2 m gedraaide-paar-kabel te stuur. Geen fout is waargeneem nie. Die oordragtempo is tot by 400 Mbps verhoog sonder dat foute voorgekom het. Daarna is die kabel verleng tot 10 m, en steeds het geen foute voorgekom nie. Daar is dus gevind dat data baie suksesvol teen 400 Mbps oor 10 m kabel gestuur kan word deur van hierdie komponent gebruik te maak.

## 7.2 Aanbevelings

Sekere veranderinge en verbeteringe kan ten opsigte van die ontwerpte stelsel aanbeveel word. Hierdie aanbevelings spruit nie net voort uit kennis wat tydens die ontwikkeling van die stelsel opgedoen is nie, maar ook as gevolg van nuwe implementering van tegnologieë.

Die veranderinge wat aanbeveel word, is nodig sodat die fisiese kompleksiteit van die stelsel sal afneem. Die verbeteringe word aanbeveel om die betroubaarheid van die stelsel te verhoog. Daar word ook sekere aanbevelings gemaak ten opsigte van die uitbreibaarheid van die stelsel.

**Implementering van slegs 'n koppelvlak in die CPLD**

Die implementering van die FIFO in die CPLD beperk die grootte en tipe FIFO wat gebruik kan word. Dit verlaag ook die maksimum frekwensie waarteen die stelsel kan funksioneer. 'n Moontlikheid wat in die toekoms na gekyk kan word is om van 'n FIFO ASIC gebruik te maak. Die CPLD is wel geskik vir die beheer van die stelsel en vir die implementering van die EDAC. Slegs die nodige koppelvlakke na die eenhede, asook die fouthantering sal in die CPLD geïmplementeer word.

**Inkorporering van koppelvlakeenhede by substelseleenhede**

Die volgende generasie kamera is in hoofstuk 2 bespreek. Daar word beplan om die kamera as 'n funksionele eenheid te ontwikkel wat al die nodige klokseine self sal genereer, en die analoog-na-syfer-omskakeling sal doen. Die funksionele eenheid sal van 'n CPLD gebruik maak vir die beheer van die kamera en data. Dit is dus nie nodig om 'n tweede CPLD te gebruik vir die implementering van die hoë spoed koppelvlak nie. Die hoë spoed koppelvlak kan direk in die CPLD op die kamerabord geïmplementeer word.

**Gebruik van CPLD met LVDS en PLL's ingebou**

Na voltooiing van die ondersoek het ALTERA sy APEX 20K familie CPLD's bekend gestel. Dit is nie net ALTERA se grootste en vinnigste CPLD nie, maar maak ook gebruik van werkverrigting-verhogingseenhede soos veelvuldige PLL's. Die familie CPLDs ondersteun ook hoë bandwydte, lae spanning I/O standaarde soos LVTTL, LVCMOS, GTL+, SSTL, HSTL, AGP, CTT, asook LVDS met 'n oordragtempo van tot 622 Mbps.

Die bruikbaarheid van hierdie familie CPLD's kan ook in die toekoms ondersoek word. Dit sal die gebruik van die addisionele LVDS senders en onvangers uitskakel, maar sal 'n groot toename in koste tot gevolg hê. [26]

**Gebruik van parallelle LVDS-verbindings**

Die vereistes wat aan die hoë spoed koppelvlak gestel is, en waaraan dit voldoen, is dat dit groot volumes data teen 'n hoë tempo moet oordra. Die kragverbruik wat hiermee gepaard gaan moet laag wees en die area wat in beslag geneem word min. Alle transmissiefoute sal deur die protokol herstel word.

Vir volgende generasie satelliete kan die gebruik van FPDI-komponente ondersoek word. Elke kleur kan byvoorbeeld deur sy eie LVDS verbinding tot by die massageheue vervoer word. Die voordeel daarvan is dat indien een van die fisiese verbindings onklaar sou raak, die kamera nie onbruikbaar sal wees nie. Dit is egter ten koste van meer komplekse verbindings en meer area wat in beslag geneem sal word.

**Implementering van IEEE 1394 (Firewire)**

Die laaste aanbeveling word gemaak ten opsigte van derde en vierde generasie SUNSAT mikrosatelliete. Op die huidige en volgende generasie mikrosatelliet word 'n algemene hoë spoed netwerk nog nie benodig nie, aangesien al die hoë spoed verbindings punt-tot-punt verbindings tussen twee spesifieke stelsels is.

Die spoed en hoeveelheid data wat tussen substelsels oorgedra moet word, gaan egter toeneem vir generasies om te kom. 'n Behoefte gaan ontstaan vir 'n algemene hoë

spoed netwerk, en sal die medium spoed netwerk kan vervang. Die IEEE 1394 standaard is die mees geskikte tegnologie om te gebruik.

IEEE 1394 beskik reeds oor 'n goed gedefinieerde fouthanteringsprotokol, en 'n gewaarborgde data-oordragtempo. Die nadele van die tegnologie, soos in hoofstuk 3 genoem, is die kompleksiteit, kragverbruik en koste daarvan, asook dat dit 'n boomnetwerk-topologie implementeer. Soos die tegnologie ontwikkel, sal die kompleksiteit, kragverbruik en koste daarvan verlaag. Deur van intelligente koppeling van die eenhede gebruik te maak, kan oortolligheid in die netwerk ingebou word wat sodoende die nadele van die boomstruktuur uitskakel.

# Verwysings


[1] Goosen N., M-Tesis: "Die Ontwerp en Evaulering van die Sekondêre Rekenaar in SUNSAT", 1996.

[2] Badenhorst., P.J., M-Thesis: "The Development of a memory System for the Second Generation SUNSAT Micro-Satellite", 1997.

[3] Du Plessis., F.E., M-Tesis: "Elektro-Optiese Analise en Ontwikkeling van 'n Hoë-Resolusie Beeldskandeerder vir die SUNSAT Mikrosatelliet", 1997.

[4] Cronje M.L., *Manual: Imager Interface, Issue 1*. Internal Report, Department Electrical and Electronic Engineering, University of Stellenbosch, June 1999.

[5] National Semiconductor, *Summary of Well Known Interface Standards* Application Note 216, July 1998.

[6] Stallings W., *Data and Computer Communications*, Fifth Edition, Prentice Hall, New Jersey, 1997.

[7] Halsall F., Data Communication, *Computer Networks and Open Systems*, Third Edition, Addison-Wesley, Massachusetts, 1993.

[8] National Semiconductor, *LVDS Owner's Manual*, Design Guide, Spring 1997.

[9] National Semiconductor, *LVDS Quad Dynamic $I_{cc}$ vs Frequency,* Application Note 1110, May 1998.

[10] Texas Instruments, *High Speed Data Transmission* Part 3 Lit # SLLBE∅4A, Brochure, March 1998.

[11] Shanley T., Anderson D., *PCI System Architecture*, Third Edition, Addison-Wesley, Massachusetts, 1995.

[12] TIA/EIA Standard, *Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits*, Telecommunications Industry Association, March 1996.

[13] National Semiconductor, *DS92LV1021 and DS92LV1210 16-40MHz 10 Bit Bus LVDS Serializer and Deserializer*, Data Sheet, March, 1999.

[14] ALTERA, *FLEX 10K, Embedded Programmable Logic Family*, Data Sheet, June 1999.

[15] ALTERA, *Configuring FLEX 10K Devices*, Application Note, January, 1999.

[16] Quality Semiconductor, Inc., *High-Speed, Low Power CMOS 10-Bit Bus Switches*, Data Sheet, December 3, 1998.

[17] Quality Semiconductor, Inc., *5V and 3V Logic Conversion with Zero Delay*, Application Note AN-11ALTERA, August 3, 1998.

[18] *PC Instrumentation for the 90's*, Fourth Edition, Boston Technical Books, Boston, 1994.

[19] Hitachi, *HM628512B Series*, Data Sheet, January 13, 1999.

[20] ATMEL, *FPGA Configuration EEPROM*, Data Sheet, July 1998.

[21] ATMEL, *AT 17CXXX FPGA Serial Configuration $E^2$PROM*, Application Note, May 1998.

[22] ATMEL, *8-Bit Microcontroller with 2K Bytes Flash*, Data Sheet, December 1997.

[23] Synopsys, Inc., *Overview of DesignWare Components*, Reference Manual, 1998.

[24] ALTERA, Introduction, Programmable Logic & ASICs, General Note, January 1998.

[25] Synopsys, Inc., *DW_ecc, Error Checking and Correction*, Reference Manual, 1998.

[26] ALTERA, *APEX Devices*, Device Brochure, August 1999.

[27] Synopsys, Inc., *Basic Sequential Family, FIFO Overview*, Reference Manual, 1998.

[28] Synopsys, Inc., *DW04_crc32, 32-Bit CRC Polynomial Generator/Checker*, Reference Manual, 1998.

[29] Texas Instruments, *Mixed-Signal and Analog Products*, Designer's Guide and Reference, January 1999.

[30] Universal Serial Bus Specification 1.00 Final Draft Revision, Chapter 1.

# 8. Bylae A

## 8.1 Programkode om mikrobeheerder te programmeer

```
;***************************************************************************
;
; PROGRAM:      Atmel
; AUTHOR:       H. Strydom
; DESCRIPTION: Genereer klok om FPGA te programeer.
;
;************************ EQUATES ***************************************

        DCLK            EQU 0B4H                ; P3.4
        Conf_Done       EQU 0B5H                ; P3.5
        nStatus EQU 0B7H                ; P3.7

;********************** RESET VECTOR ***********************************

        SJMP Main                               ; Jump to main program

;********************** MAIN PROGRAM ***********************************

        ORG   030H

Main:
        MOV   SP, #050H                         ; Stack pointer to scratch area at 50H

Init:
        CLR   DCLK                              ; Clear P3.4
        SETB  Conf_Done                         ; Set P3.5
        MOV   TMOD, #02H                        ; Time 0, Mode 2
        MOV   TH0, #-50                         ; Load counter with -50
        SETB  TR0                               ; Start Timer 0
        MOV   R0, #0FFH                         ; Load 255 in R0

Loop:
        JNB   TF0, Loop                         ; Loop till couter overflow
        CLR   TF0                               ; Clear flag
        DJNZ  R0, Loop                  ; Loop till R0 = 0
        CLR   TR0                               ; Stop Timer 0

DClock:
        SETB  DCLK                              ; DClk high
        CLR   DCLK                              ; DClk low
        JNB   Conf_Done, DClock                 ; If Conf_Done = '0'

        MOV   R0, #11                   ; Clock counter

Loop2:
        SETB  DCLK                              ; DCLK high
        CLR   DCLK                              ; DCLK low
        DJNZ  R0, Loop2                         ; Count 10 counts

        CLR   DCLK                              ; DCLK low

Wait:
        JMP   Wait                              ; Endless loop

;************************ THE END ***************************************

        END
```

# 9. Bylae B

## 9.1 VHDL-kode van ISA-koppelvlak

```
--------------------------------------------------------------------------------
--  Tesis
--  ISA-Koppelvlak
--  Hendrik Strydom
--------------------------------------------------------------------------------

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_unsigned.all;

entity ISAKv is
  port (

        RstDrv  : in std_logic;                                 -- ISA Reset
        BCLK    : in std_logic;                                 -- ISA bus clock
        IORC    : in std_logic;                                 -- ISA IO read line
        IOWC    : in std_logic;                                 -- ISA IO write line
        AEN     : in std_logic;                                 -- ISA Address Enable
        SA      : in std_logic_vector (9 downto 0);             -- ISA address lines
        BD      : inout std_logic_vector (15 downto 0);         -- ISA data lines buffered
        NOWS    : out std_logic;                                -- ISA No Wait Status line
        IRQ3    : out std_logic;                                -- ISA IRQ line
        IRQ4    : out std_logic;                                -- ISA IRQ line
        IRQ5    : out std_logic;                                -- ISA IRQ line
        IRQ7    : out std_logic;                                -- ISA IRQ line
        IO16    : out std_logic;                                -- ISA 16-bit enable
        BufDir  : out std_logic;                                -- 245 direction line
        BufEn   : out std_logic;                                -- 245 Enable line

        ClkIn   : in std_logic;                                 -- 4 MHz input clock
        ExClkIn : in std_logic;                                 -- External clock

        MemAddr : out std_logic_vector (18 downto 0);           -- SRAM Address
        MemRdL  : out std_logic;                                -- SRAM read line
        MemWrL  : out std_logic;                                -- SRAM write line
        MemSelL : out std_logic;                                -- SRAM select
        MemRdH  : out std_logic;                                -- SRAM read line
        MemWrH  : out std_logic;                                        -- SRAM write line
        MemSelH : out std_logic                                 -- SRAM select

        );
end ISAKv;

--------------------------------------------------------------------------------

architecture BEHAVIORAL of ISAKv is

  signal        nAddrOK                 : std_logic;

  signal        DtWr, DtRd,
                        CmWr,
                        GRST            : std_logic;

  type mem_state_values is (ISASt, OtherSt);
  signal        State                   : mem_state_values;

  signal        RamPnt                  : std_logic_vector (7 downto 0);

  signal        MemClk                  : std_logic;

  signal        Cmd                     : std_logic;

--------------------------------------------------------------------------------

begin
```

```
  IRQ3 <= '0';
  IRQ4 <= '0';
  IRQ5 <= '0';
  IRQ7 <= '0';
  NOWS <= '0';

--------------------------------------------------------------------------------
-- nAddrOK --

  process (AEN, SA)
  begin
    if (SA(9 downto 2) = "11000000") and (AEN = '0') then
      nAddrOK <= '0';
    else
      nAddrOK <= '1';
    end if;
  end process;

--------------------------------------------------------------------------------
-- Assignment of new states --
-- Read cycle --

  process (nAddrOK, IORC)
  variable Temp  : std_logic_vector (2 downto 0);
  begin
    case SA(1 downto 0) is
      when "00" => Temp := "001";
      when "01" => Temp := "001";
      when "10" => Temp := "010";
      when "11" => Temp := "100";
      when others => Temp := "000";
    end case;
    if nAddrOK = '1' then
      DtRd <= '0';
    elsif IORC'event and IORC = '0' then
      DtRd <= Temp(0);
    end if;
  end process;

-- Write cycle --

  process (nAddrOK, IOWC)
  variable Temp  : std_logic_vector (2 downto 0);
  begin
    case SA(1 downto 0) is
      when "00" => Temp := "001";
      when "01" => Temp := "001";
      when "10" => Temp := "010";
      when "11" => Temp := "100";
      when others => Temp := "000";
    end case;
    if nAddrOK = '1' then
      DtWr <= '0';
      CmWr <= '0';
      GRST <= '0';
    elsif IOWC'event and IOWC = '0' then
      DtWr <= Temp(0);
      CmWr <= Temp(1);
      GRST <= Temp(2);
    end if;
  end process;

--------------------------------------------------------------------------------
-- Assignment IO16 --

  process (AEN, SA, State, nAddrOK)
  begin
    if AEN = '1' then
      IO16 <= '0';
    elsif nAddrOK'event and nAddrOK = '0' then
      if State = ISASt then
        IO16 <= not(SA(1));
      end if;
    end if;
  end process;
```

```
--------------------------------------------------------------------------------
-- Assignment BufEn --

  process (State, nAddrOK)
  begin
    if State = ISASt then
      BufEn <= nAddrOK;
    else
      BufEn <= '1';
    end if;
  end process;

--------------------------------------------------------------------------------
-- BufDir --

  BufDir <= not (IORC);

--------------------------------------------------------------------------------
-- Command Register --

  WrProc : process (State, CmWr, IOWC, BD)
  variable Temp : std_logic_vector (3 downto 0);
  begin
    case BD (7 downto 0) is
      when "00000001"  => Temp := "0001";
      when others      => Temp := "0000";
    end case;
    if not(State = ISASt) then
      Cmd <= '0';
    elsif IOWC'event and IOWC = '1' then
      if CmWr = '1' then
        Cmd <= Temp (0);
      end if;
       end if;
  end process WrProc;

--------------------------------------------------------------------------------
-- State --

  SM : process (RstDrv, GRST, BCLK)
  begin
    if RstDrv = '1' or GRST = '1' then
      State <= ISASt;
    elsif BCLK'event and BCLK = '1' then
      case State is
        when ISASt     =>      if Cmd = '1' then
                                 State <= OtherSt;
                               else
                                 State <= ISASt;
                               end if;
        when others => State <= ISASt;
      end case;
    end if;
  end process SM;

--------------------------------------------------------------------------------
-- Assignment of BD --

BD <= "ZZZZZZZZZZZZZZZZ";

--------------------------------------------------------------------------------
-- MemRd --

  process (State, DtRd)
  begin
    case State is
      when ISASt       => MemRdL <= not DtRd;
                          MemRdH <= not DtRd;
      when others      => MemRdL <= '1';
                          MemRdH <= '1';
    end case;
  end process;

--------------------------------------------------------------------------------
-- MemWr --

  process (State, DtWr, IOWC)
```

```
  begin
    case State is
      when ISASt       => if DtWr = '1' then
                            MemWrL <= IOWC;
                            MemWrH <= IOWC;
                          else
                            MemWrL <= '1';
                            MemWrH <= '1';
                          end if;
      when others      => MemWrL <= '1';
                          MemWrH <= '1';
    end case;
  end process;

-------------------------------------------------------------------------------
-- MemSel --

  process (State, DtRd, DtWr)
  begin
    case State is
      when ISASt                => if DtRd = '1' or DtWr = '1' then
                                     MemSelL <= '0';
                                     MemSelH <= '0';
                                   else
                                     MemSelL <= '1';
                                     MemSelH <= '1';
                                   end if;
      when others               => MemSelL <= '1';
                                   MemSelH <= '1';
    end case;
  end process;

-------------------------------------------------------------------------------
-- RamPnt --

  MemClk <= DtRd or DtWr;

  process (GRST, MemClk)
  begin
    if GRST = '1' then
      RamPnt <= "00000000";
    elsif MemClk'event and MemClk = '0' then
        if State = ISASt then
          RamPnt <= RamPnt + 1;
        end if;
    end if;
  end process;

-------------------------------------------------------------------------------
-- MemAddr --

  MemAddr <= "00000000000" & RamPnt;

-------------------------------------------------------------------------------

end BEHAVIORAL;

--===========================================================================--
```

# 10. Bylae C

## 10.1 VHDL-kode om 8-databisse-5-toetsbisse-EDAC te toets

```
library IEEE,DW04;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_arith.all;
   use DW04.DW04_components.all;

entity edac8_5a is

      generic (
                     data_width: integer := 8;
                     cb_width: integer := 5;
                     synd: integer := 0);

      port (
               gen                    : in std_logic;
               correct_n              : in std_logic;
               data_in                : in std_logic_vector (data_width-1 downto 0);
               chkin                  : in std_logic_vector (cb_width-1 downto 0);
               data_out               : out std_logic_vector (data_width-1 downto 0);
               chkout                 : out std_logic_vector (cb_width-1 downto 0);
               err_detect             : out std_logic;
               err_multpl             : out std_logic);

end edac8_5a;

architecture BEHAVIORAL of edac8_5a is

begin

  U1 : DW_ecc

       generic map(
                     width          => data_width,
                     chkbits        => cb_width,
                     synd_sel       => synd )

       port map (
                     chkout         => chkout,
                     datain         => data_in,
                     gen            => gen,
                     correct_n      => correct_n,
                     chkin          => chkin,
                     dataout        => data_out,
                     err_multpl     => err_multpl,
                     err_detect     => err_detect );

end BEHAVIORAL;
```

## 10.2 VHDL-kode om 16-databisse-6-toetsbisse-EDAC te toets

```
library IEEE,DW04;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_arith.all;
   use DW04.DW04_components.all;

entity edac16_6a is

      generic (
                     data_width: integer := 16;
                     cb_width: integer := 6;
                     synd: integer := 0);

      port (
               gen                    : in std_logic;
               correct_n              : in std_logic;
               data_in                : in std_logic_vector (data_width-1 downto 0);
               chkin                  : in std_logic_vector (cb_width-1 downto 0);
               data_out               : out std_logic_vector (data_width-1 downto 0);
               chkout                 : out std_logic_vector (cb_width-1 downto 0);
               err_detect             : out std_logic;
               err_multpl             : out std_logic);

end edac16_6a;

architecture BEHAVIORAL of edac16_6a is

begin

  U1 : DW_ecc

       generic map(
                     width          => data_width,
                     chkbits        => cb_width,
                     synd_sel       => synd )

       port map (
                     chkout         => chkout,
                     datain         => data_in,
                     gen            => gen,
                     correct_n      => correct_n,
                     chkin          => chkin,
                     dataout        => data_out,
                     err_multpl     => err_multpl,
                     err_detect     => err_detect );

end BEHAVIORAL;
```

## 10.3 VHDL-kode om EDAC-komponent asinchroon te toets

```
LIBRARY IEEE;
 use IEEE.std_logic_1164.all;
 use IEEE.std_logic_arith.all;

ENTITY Edac IS
        PORT (

                gen1                    : in std_logic;
                correct_n1              : in std_logic;
                data_in1                : in std_logic_vector (15 downto 0);
                chkin1                  : in std_logic_vector (7 downto 0);
                data_out1               : out std_logic_vector (15 downto 0);
                chkout1                 : out std_logic_vector (7 downto 0);
                err_detect1             : out std_logic;
                err_multpl1             : out std_logic);

END Edac;

ARCHITECTURE Behavioral OF Edac IS
-------------------------------------------------------------------
COMPONENT edac8_5a
        PORT (

                gen                     : in std_logic;
                correct_n               : in std_logic;
                data_in                 : in std_logic_vector (15 downto 0);
                chkin                   : in std_logic_vector (7 downto 0);
                data_out                : out std_logic_vector (15 downto 0);
                chkout                  : out std_logic_vector (7 downto 0);
                err_detect              : out std_logic;
                err_multpl              : out std_logic);

END COMPONENT;
-------------------------------------------------------------------

BEGIN

-------------------------------------------------------------------

U1: edac8_5a
PORT MAP
        (
                gen1,
                correct_n1,
                Data_in1,
                chkin1,
                Data_out1,
                chkout1,
                err_detect1,
                err_multpl1
        );

-------------------------------------------------------------------

END Behavioral;

-------------------------------------------------------------------
```

## 10.4 VHDL-kode om EDAC-komponente sinchroon te toets

```
LIBRARY IEEE;
 use IEEE.std_logic_1164.all;
 use IEEE.std_logic_arith.all;

ENTITY Edac IS
        PORT (

                errdo,
                errmo                   : out std_logic;

                clk                     : in std_logic;

                gen1                    : in std_logic;
                correct_n1              : in std_logic;
                data_in1                : in std_logic_vector (7 downto 0);
                chkin1                  : in std_logic_vector (4 downto 0);
                data_out1               : out std_logic_vector (7 downto 0);
                chkout1                 : out std_logic_vector (4 downto 0);
                err_detect1             : out std_logic;
                err_multpl1             : out std_logic);

END Edac;

ARCHITECTURE Behavioral OF Edac IS
-------------------------------------------------------------------------
COMPONENT edac8_5a
        PORT (

                gen                     : in std_logic;
                correct_n               : in std_logic;
                data_in                 : in std_logic_vector (7 downto 0);
                chkin                   : in std_logic_vector (4 downto 0);
                data_out                : out std_logic_vector (7 downto 0);
                chkout                  : out std_logic_vector (4 downto 0);
                err_detect              : out std_logic;
                err_multpl              : out std_logic);

END COMPONENT;
-------------------------------------------------------------------------

  signal  reg1, reg2            : std_logic_vector (7 downto 0);
  signal  ch1, ch2                      : std_logic_vector (4 downto 0);
  signal  errd, errm            : std_logic;
  signal  g                                     : std_logic;

-------------------------------------------------------------------------

BEGIN

  errdo <= errd;
  errmo <= errm;

-------------------------------------------------------------------------

U1: edac8_5a
PORT MAP
        (
                g,
                correct_n1,
                reg1,
                ch1,
                reg2,
                ch2,
                errd,
                errm
        );

-------------------------------------------------------------------------

  process (clk)
  begin
    if clk'event and clk = '0' then
```

```
      g <= gen1;
      reg1 <= Data_in1;
      ch1 <= chkin1;

      Data_out1 <= reg2;
      chkout1 <= ch2;

      err_detect1 <= errd;
      err_multpl1 <= errm;

    end if;
  end process;

-------------------------------------------------------------------------

END Behavioral;

-------------------------------------------------------------------------
```

# 11. Bylae D

## 11.1 VHDL-kode van FIFO-komponent wat deur Megafuction Wizard geskep is

```
-- megafunction wizard: %FIFO%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: SCFIFO

-- ============================================================
-- File Name: FifoCmp.vhd
-- Megafunction Name(s):
--                      SCFIFO
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD GENERATED FILE. DO NOT EDIT THIS FILE!
-- ************************************************************


--      Copyright (C) 1988-1998 Altera Corporation

--      Any megafunction design, and related net list (encrypted or decrypted),
--      support information, device programming or simulation file, and any other
--      associated documentation or information provided by Altera or a partner
--      under Altera's Megafunction Partnership Program may be used only to
--      program PLD devices (but not masked PLD devices) from Altera.  Any other
--      use of such megafunction design, net list, support information, device
--      programming or simulation file, or any other related documentation or
--      information is prohibited for any other purpose, including, but not
--      limited to modification, reverse engineering, de-compiling, or use with
--      any other silicon devices, unless such use is explicitly licensed under
--      a separate agreement with Altera or a megafunction partner.  Title to
--      the intellectual property, including patents, copyrights, trademarks,
--      trade secrets, or maskworks, embodied in any such megafunction design,
--      net list, support information, device programming or simulation file, or
--      any other related documentation or information provided by Altera or a
--      megafunction partner, remains with Altera, the megafunction partner, or
--      their respective licensors.  No other licenses, including any licenses
--      needed under any third party's intellectual property, are provided herein.

LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.all;

ENTITY FifoCmp IS
        PORT
        (
                data            : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                wrreq           : IN STD_LOGIC ;
                rdreq           : IN STD_LOGIC ;
                clock           : IN STD_LOGIC ;
                aclr            : IN STD_LOGIC ;
                q               : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
                full            : OUT STD_LOGIC ;
                empty           : OUT STD_LOGIC ;
                almost_full     : OUT STD_LOGIC ;
                almost_empty    : OUT STD_LOGIC
        );
END FifoCmp;


ARCHITECTURE SYN OF FifoCmp IS

        SIGNAL sub_wire0        : STD_LOGIC ;
        SIGNAL sub_wire1        : STD_LOGIC ;
        SIGNAL sub_wire2        : STD_LOGIC ;
        SIGNAL sub_wire3        : STD_LOGIC_VECTOR (15 DOWNTO 0);
```

```
        SIGNAL sub_wire4       : STD_LOGIC ;


        COMPONENT SCFIFO
        GENERIC (
                LPM_WIDTH               : POSITIVE;
                LPM_NUMWORDS            : POSITIVE;
                ALMOST_FULL_VALUE       : POSITIVE;
                ALMOST_EMPTY_VALUE      : POSITIVE;
                LPM_SHOWAHEAD           : STRING;
                USE_EAB                 : STRING;
                MAXIMIZE_SPEED          : POSITIVE
        );
        PORT (
                        almost_full     : OUT STD_LOGIC ;
                        rdreq           : IN STD_LOGIC ;
                        empty           : OUT STD_LOGIC ;
                        aclr            : IN STD_LOGIC ;
                        almost_empty    : OUT STD_LOGIC ;
                        clock           : IN STD_LOGIC ;
                        q               : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
                        wrreq           : IN STD_LOGIC ;
                        data            : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                        full            : OUT STD_LOGIC
        );
        END COMPONENT;

BEGIN
        almost_full     <= sub_wire0;
        empty           <= sub_wire1;
        almost_empty    <= sub_wire2;
        q               <= sub_wire3(15 DOWNTO 0);
        full            <= sub_wire4;

        SCFIFO_component : SCFIFO
        GENERIC MAP (
                LPM_WIDTH => 16,
                LPM_NUMWORDS => 512,
                ALMOST_FULL_VALUE => 256,
                ALMOST_EMPTY_VALUE => 8,
                LPM_SHOWAHEAD => "OFF",
                USE_EAB => "ON",
                MAXIMIZE_SPEED => 7
        )
        PORT MAP (
                rdreq => rdreq,
                aclr => aclr,
                clock => clock,
                wrreq => wrreq,
                data => data,
                almost_full => sub_wire0,
                empty => sub_wire1,
                almost_empty => sub_wire2,
                q => sub_wire3,
                full => sub_wire4
        );


END SYN;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
-- Retrieval info: PRIVATE: Width NUMERIC "16"
-- Retrieval info: PRIVATE: Depth NUMERIC "512"
-- Retrieval info: PRIVATE: Clock NUMERIC "0"
-- Retrieval info: PRIVATE: Full NUMERIC "1"
-- Retrieval info: PRIVATE: Empty NUMERIC "1"
-- Retrieval info: PRIVATE: UsedW NUMERIC "0"
-- Retrieval info: PRIVATE: AlmostFull NUMERIC "1"
-- Retrieval info: PRIVATE: AlmostEmpty NUMERIC "1"
-- Retrieval info: PRIVATE: AlmostFullThr NUMERIC "256"
-- Retrieval info: PRIVATE: AlmostEmptyThr NUMERIC "8"
-- Retrieval info: PRIVATE: sc_aclr NUMERIC "1"
-- Retrieval info: PRIVATE: sc_sclr NUMERIC "0"
```

```
-- Retrieval info: PRIVATE: rsFull NUMERIC "1"
-- Retrieval info: PRIVATE: rsEmpty NUMERIC "1"
-- Retrieval info: PRIVATE: rsUsedW NUMERIC "0"
-- Retrieval info: PRIVATE: wsFull NUMERIC "1"
-- Retrieval info: PRIVATE: wsEmpty NUMERIC "0"
-- Retrieval info: PRIVATE: wsUsedW NUMERIC "0"
-- Retrieval info: PRIVATE: dc_aclr NUMERIC "1"
-- Retrieval info: PRIVATE: LegacyRREQ NUMERIC "1"
-- Retrieval info: PRIVATE: LE_BasedFIFO NUMERIC "0"
-- Retrieval info: PRIVATE: Optimize NUMERIC "1"
-- Retrieval info: CONSTANT: LPM_WIDTH NUMERIC "16"
-- Retrieval info: CONSTANT: LPM_NUMWORDS NUMERIC "512"
-- Retrieval info: CONSTANT: ALMOST_FULL_VALUE NUMERIC "256"
-- Retrieval info: CONSTANT: ALMOST_EMPTY_VALUE NUMERIC "8"
-- Retrieval info: CONSTANT: LPM_SHOWAHEAD STRING "OFF"
-- Retrieval info: CONSTANT: USE_EAB STRING "ON"
-- Retrieval info: CONSTANT: MAXIMIZE_SPEED NUMERIC "7"
-- Retrieval info: USED_PORT: data 0 0 16 0 INPUT NODEFVAL data[15..0]
-- Retrieval info: USED_PORT: q 0 0 16 0 OUTPUT NODEFVAL q[15..0]
-- Retrieval info: USED_PORT: wrreq 0 0 0 0 INPUT NODEFVAL wrreq
-- Retrieval info: USED_PORT: rdreq 0 0 0 0 INPUT NODEFVAL rdreq
-- Retrieval info: USED_PORT: clock 0 0 0 0 INPUT NODEFVAL clock
-- Retrieval info: USED_PORT: full 0 0 0 0 OUTPUT NODEFVAL full
-- Retrieval info: USED_PORT: empty 0 0 0 0 OUTPUT NODEFVAL empty
-- Retrieval info: USED_PORT: almost_full 0 0 0 0 OUTPUT NODEFVAL almost_full
-- Retrieval info: USED_PORT: almost_empty 0 0 0 0 OUTPUT NODEFVAL almost_empty
-- Retrieval info: USED_PORT: aclr 0 0 0 0 INPUT NODEFVAL aclr
-- Retrieval info: CONNECT: @data 0 0 16 0 data 0 0 16 0
-- Retrieval info: CONNECT: q 0 0 16 0 @q 0 0 16 0
-- Retrieval info: CONNECT: @wrreq 0 0 0 0 wrreq 0 0 0 0
-- Retrieval info: CONNECT: @rdreq 0 0 0 0 rdreq 0 0 0 0
-- Retrieval info: CONNECT: @clock 0 0 0 0 clock 0 0 0 0
-- Retrieval info: CONNECT: full 0 0 0 0 @full 0 0 0 0
-- Retrieval info: CONNECT: empty 0 0 0 0 @empty 0 0 0 0
-- Retrieval info: CONNECT: almost_full 0 0 0 0 @almost_full 0 0 0 0
-- Retrieval info: CONNECT: almost_empty 0 0 0 0 @almost_empty 0 0 0 0
-- Retrieval info: CONNECT: @aclr 0 0 0 0 aclr 0 0 0 0
```

## 11.2 VHDL-kode om verskillende kombinasies FIFO's te ondersoek

```
LIBRARY IEEE;
 use IEEE.std_logic_1164.all;
 use IEEE.std_logic_arith.all;

ENTITY Fifo1 IS
        PORT
        (
                data1                   : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                wrreq1                  : IN STD_LOGIC ;
                rdreq1                  : IN STD_LOGIC ;
                clock1                  : IN STD_LOGIC ;
                aclr1                   : IN STD_LOGIC ;
                q1                              : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
                full1                   : OUT STD_LOGIC;
                empty1                  : OUT STD_LOGIC;
                almost_full1            : OUT STD_LOGIC;
                almost_empty1           : OUT STD_LOGIC
        );
END Fifo1;

ARCHITECTURE Behavioral OF Fifo1 IS
-----------------------------------------------------------------------
COMPONENT FifoCmp
        PORT
        (
                data                    : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                wrreq                   : IN STD_LOGIC ;
                rdreq                   : IN STD_LOGIC ;
                clock                   : IN STD_LOGIC ;
                aclr                    : IN STD_LOGIC ;
                q                               : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
                full                    : OUT STD_LOGIC ;
                empty                   : OUT STD_LOGIC ;
                almost_full             : OUT STD_LOGIC ;
                almost_empty    : OUT STD_LOGIC
        );
END COMPONENT;

-----------------------------------------------------------------------

BEGIN
U1: FifoCmp
PORT MAP
        (
                data1,
                wrreq1,
                rdreq1,
                clock1,
                aclr1,
                q1,
                full1,
                empty1,
                almost_full1,
                almost_empty1
        );

-----------------------------------------------------------------------

END Behavioral;

--=======================================================================--
```