

Implementatie en uitwerking van een standalone camerasysteem voor gebruik op de MOST–bus

Bouwens Stefan

Geerardyn Dave

1 mei 2007

Woord Vooraf

Dit eindwerk kon slechts tot stand komen dankzij de medewerking van enkele personen. Daarom bedanken wij nog graag even:

- ir. Jan Meel, onze hogeschoolpromotor, voor de interne afhandelingen en algemene ondersteuning.
- ir. Arnaud Darmond, onze bedrijfspromotor, voor de ondersteuning met betrekking tot de camera.
- ing. Jurgen Schoeters & ing. Jan Van Winkel voor het configureren van de Virtex FPGA met het MOST-protocol.
- ing. Danny Pauwels voor de fijne aanpassingen van de interface-PCB.
- Melexis waar we 2 weken lang stage mochten lopen.
- De Nayer Insituut waar waar we de infrastructuur mochten gebruiken.

Abstract

Melexis is een bedrijf dat gespecialiseerd is in halfgeleiders voor de auto-industrie. Voor deze masterproef moest een camera (de MLX75007 van Melexis) gebruikt worden om, in combinatie met een DSP (de ADSP-BF561 Ez-Kit Lite[®] van Analog Devices) een demo-opstelling te maken waarop beeldverwerking mogelijk was. Deze demo-opstelling moet de werking en de toepassingen van de camera illustreren.

Om dit doel te bereiken maakten we eerst een interface-PCB om de verbinding tussen de DSP en de camera te maken. Vervolgens programmeerden we de DSP om de parameters van de camera in te stellen en de beelden naar buiten te brengen via SPI, Composiet(PAL) of S-video. Hiervoor werd de video-encoder (ADV7179 van Analog Devices) gebruikt die op de ADSP-BF561 Ez-Kit Lite[®] aanwezig is.

Op het moment van schrijven is het reeds gelukt om beelden van de camera uit te lezen en naar buiten te brengen via Composiet en SPI.

Inhoudsopgave

1	Inleiding	11
1.1	Verklarende Woordenlijst	13
1.2	Situering	17
2	Specificaties	21
2.1	Algemene Specificaties	23
2.2	Specificaties voor Melexis	23
2.3	Specificaties voor MULTICARCOM/De Nayer	24
3	Uitwerking	27
3.1	Hoe is er tewerk gegaan?	29
3.1.1	Stage	29
3.1.1.1	Algoritmes: Regulator	29
3.1.1.2	Algoritmes: Lensdistortie-correctie	30
3.1.1.3	Slopes	31
3.1.1.4	Interface-PCB	33
3.1.2	Werken met Visual DSP++	35
3.1.3	Camera-aansturing	35
3.1.3.1	Programmable flags	35
3.1.3.2	Timers	36
3.1.3.3	SPI	37
3.1.4	Eén camerabeeld ontvangen en tijdelijk opslaan	39
3.1.4.1	PPI	40
3.1.4.2	Interrupts	41
3.1.4.3	DMA	42
3.1.4.4	SDRAM-geheugen	43
3.1.5	Continu beelden ontvangen en tijdelijk opslaan	43
3.1.6	Taakverdeling	43

3.1.7	Melexis: Composiet–video	45
3.1.8	Camerabeelden rechtstreeks weergeven	46
3.1.9	MULTICARCOM/De Nayer: SPI/MOST–link	49
3.1.9.1	Communicerende systemen	49
3.1.9.2	Beeldcompressie	49
3.1.9.3	SPI–slave	50
3.1.9.4	SDRAM, dataverwerking & DMA	52
3.1.9.5	LVDS	52
3.1.10	Synthese	53
3.2	Details	53
3.2.1	Camera: Melexis MLX75007	53
3.2.1.1	MLX75007	53
3.2.1.2	Hoe gebruiken wij de camera?	54
3.2.1.3	Parameters	54
3.2.2	Evaluatiebord: ADSP–BF561 EZ–KIT Lite [®]	55
3.2.2.1	Evaluatiebord: algemeen	55
3.2.2.2	BF561: programmable flags (PF)	56
3.2.2.3	BF561: timers	57
3.2.2.4	BF561: serial peripheral interface (SPI)	57
3.2.2.5	BF561: serial port (SPORT)	57
3.2.2.6	BF561: parallel peripheral interface (PPI)	58
3.2.2.7	BF561: direct memory access (DMA)	59
3.2.2.8	BF561: interrupts	59
3.2.3	Display: NW619VT	60
3.2.4	Algemene details	60
3.3	Problemen	61
4	Besluiten	67
4.1	Resultaten	69
4.1.1	Stageresultaten	69
4.1.1.1	PCB	69
4.1.1.2	Algoritmes	69
4.1.2	10 januari 2007	70
4.1.3	Huidige stand	70
4.1.3.1	Melexis: Composiet–link	70
4.1.3.2	MULTICARCOM/De Nayer: SPI/MOST–link	70

4.1.4	De Toekomst	71
4.1.4.1	Planning naar de masterproefverdediging toe	71
4.2	Besluit	72
	Lijst van figuren	74
	Lijst van tabellen	75
	Bibliografie	76
5	Bijlagen	79

Hoofdstuk 1

Inleiding

1.1 Verklarende Woordenlijst

4:2:2 4:2:2 is een subsamplingmethode (zie ook YCrCb) waarbij de luminantie (Y) voor elke pixel, maar de chrominantie (Cr,Cb) slechts om de 2 pixels wordt doorgestuurd. Om een eenvoudige decodering te doen wordt het de beeldinformatie op de volgende manier doorgestuurd: $Y_1 \text{ Cr}_1 \text{ Y}_2 \text{ Cb}_1 \text{ Y}_3 \text{ Cr}_3 \text{ Y}_4 \text{ Cb}_3 \dots$

AD Analog Devices Inc.¹

ADC Een Analog to Digital Converter zet analoge signalen om in digitale signalen.

Big-Endian Bij het opslaan (doorsturen) van data wordt de MSB eerst opgeslagen (doorgestuurd). In tegenstelling tot Little endian waar de LSB eerst wordt opgeslagen (doorgestuurd). Big Endian wordt het meest gebruikt.

Blackfin Een DSP-processor ontwikkeld door Analog Devices Inc. (zie ook DSP).

CCIR656 CCIR656 is de oude standaard om TV-beelden (met synchronisatie) digitaal voor te stellen. Zie naar ITU-R-656 voor de nieuwe standaard.

CMOS Complementary Metal Oxide Semiconductor is een technologie waarmee onder andere beeldsensoren gemaakt worden. Karakteristiek is het lagere stroomverbruik, in vergelijking met bijvoorbeeld CCD.

Composiet, Composite Composiet-video is een videostandaard voor analoge video-transmissie, dat enkel beelden bevat. Het wordt meestal via een gele RCA-connector aangesloten en stuurt Y- en C-signalen over dezelfde geleider.

(Processor-)core Het hart van een processor.

Daisy-chain De data wordt naar een IC gestuurd en komt er langs een andere poort terug uit. Dit kan gebruikt worden om te controleren of de data wel juist aangekomen is (feedback), maar het wordt hoofdzakelijk gebruikt om meerdere IC's te programmeren. Deze zijn dan als één ketting (daisy-chain) aan elkaar verbonden.

DLL Een Dynamically Linked Library is een bibliotheek van functies. Via de DLL kan een ander programma deze functies eenvoudig gebruiken.

DMA Direct Memory Access is een technologie om data rechtstreeks van/naar een geheugen te schrijven/lezen, zonder de processor te belasten. Zie voor meer uitleg ook naar hoofdstuk 3.2.2.7.

Dual-core (processor) Een processor met 2 cores.

DSP Een Digital Signal Processor is een processor die specifiek ontwikkeld is voor audio- en beeldverwerkende toepassingen.

Dutycycle Het percentage, van één periode van een bloksignaal, dat het signaal hoog is.

¹Analog Devices Inc.: <http://www.analog.com>

- EAV** End of Active Video is de synchronisatiebyte, bij de ITU-R-656 standaard (zie ITU-R-656), die aangeeft dat een beeldlijn stopt en dat de synchronisatie begint.
- FOT** Een Fiber Optic Transceiver is een elektronica-component, die data op een glasvezelkabel stuurt en van dezelfde glasvezelkabel ontvangt.
- FPGA** Een Field Programmable Gate Array is een programmeerbare chip. Deze verliest zijn data (programma) als de spanning wegvalt.
- FPS** Frames Per Second is de eenheid van framerate en staat voor het aantal videobeelden, dat per seconde (cfr. Hz) wordt weergegeven (zie ook framerate).
- Framerate** De frequentie waarmee opeenvolgende videobeelden (frames) na elkaar getoond worden (zie ook fps).
- Hysteresis** Het verschijnsel waarbij de uitgang niet meteen reageert, maar slechts nadat een bepaalde grens bereikt is. Vanaf dat deze grens overschreden is, geldt er een nieuwe (andere) grens.
- IIC, I²C** Inter-IC is een digitaal busprotocol, dat gebruikt wordt om op korte afstand tussen 2 IC's te communiceren.
- ISR** Een Interrupt Service Routine is een functie (routine) die wordt opgeroepen als er zich een bijbehorende interrupt/gebeurtenis voordoet.
- ITU-R-656** ITU-R-656 is de internationale standaard om TV-beelden (met synchronisatie) digitaal voor te stellen.
- JTAG** Joint Test Action Group is een protocol waarmee elektronische schakelingen getest kunnen worden.
- Little-endian** De MSB wordt eerst doorgestuurd of opgeslagen, de LSB laatst. Dit systeem wordt bijvoorbeeld gebruikt door Intel.
- LSB** De Least Significant Bit is de bit met het laagste waarde-aandeel.
- Luminantie** De helderheid van een pixel (te vergelijken met de intensiteit bij een beeld van grijstinten). De luminantie wordt meestal afgekort als Y. Als van een beeld enkel de luminantie weergegeven wordt, dan komt dit overeen met het beeld in grijstinten.
- LVDS** Low Voltage Differential Signaling is een protocol waarbij een signaal differentieel over 2 geleiders doorgestuurd wordt. Op de ene geleider wordt een positieve spanning gestuurd en op de andere geleider een (even grote) negatieve spanning. Het protocol wordt gebruikt wegens zijn lagere stoor gevoeligheid.
- Chrominantie** De kleurinformatie van een pixel in het YCrCb-kleurenmodel.
- MOST** Het Media Oriented Systems Transport is een protocol voor multimediatoepassingen in de wagen. Het is ontwikkeld door de MOST Coöperation².

²MOST Coöperation: <http://www.mostcooperation.com>

- MPEG** Moving Picture Experts Group is een veel gebruikte codering voor de compressie van beeld- en geluidsbestanden. Er bestaan reeds meerdere versies van.
- MSB** De Most Significant Bit is de bit met het hoogste waarde-aandeel.
- NTSC** National Television Standards Committee is de Amerikaanse standaard voor analoge videobeelden. Het wordt, ironisch bedoeld, wel eens “Never The Same Color” genoemd, omdat de weergave van de kleuren wel eens kan variëren. De standaard gebruikt 60 beelden per seconde en 525 lijnen per beeld.
- PAL** Phase Alternating Line is de Europese standaard voor analoge videobeelden. De standaard gebruikt 60 beelden per seconde en 525 lijnen per beeld. De standaard biedt ook ondersteuning voor interlacing. Dan worden afwisselend beelden doorgestuurd met enkel even of enkel oneven beeldlijnen, om een rustiger beeld te krijgen.
- PCB** Een Printed Circuit Board is een printplaat.
- PLL** Een Phase Locked Loop genereert een uitgangssignaal waarvan de frequentie een functie is van de frequentie en de fase van hetingangssignaal. De PLL gebruikt hiervoor feedback.
- PPI** De Parallel Peripheral Interface is een parallelle poort, die Analog Devices in zijn DSP's implementeert. De digitale interface is speciaal voor transfers van videodata gemaakt. De interface beschikt over speciale signalen voor videosynchronisatie en een aanpasbare busbreedte. Op de BF561 moet de PPI samen met de DMA gebruikt worden voor rechtstreekse transfers van/naar het geheugen. Zie voor meer uitleg ook naar hoofdstuk 3.2.2.6.
- PWM** Pulse Width Modulation is een techniek waarbij de dutycycle en de periode van een digitaal signaal geregeld worden om zo eender welk pulssignaal te maken.
- RCA-connector** De zogenaamde cinch-connector. De gele connector wordt meestal gebruikt voor analoge (composiet) video, de rode connector voor de rechtse luidspreker (audio in stereo) en de witte voor de linkse luidspreker (audio in stereo). Deze connector wordt ook wel eens een tulpstekker genoemd. Op figuur 3.9, op pagina 45 staan hiervan 2 voorbeelden.
- RGB** Rood Groen Blauw is een kleurmodel dat (bijna) alle kleuren kan vormen met 3 basiskleuren: rood, groen en blauw.
- SDRAM** Synchronic Dynamic Random Access Memory is een geheugentype, dat synchrono werkt. Elke bit verliest na een tijdje zijn lading en moet dan tijdig terug opgeladen worden.
- SPI** De Serial Peripheral Interface is een digitale seriële poort, voor communicatie tussen IC's. Deze technologie is de concurrent van IIC.
- SPORT** De Serial PORT is een seriële poort, die Analog Devices in zijn DSP's implementeert. Deze digitale interface is voor audiotransfers gemaakt. De SPORT beschikt over 2 seriële datakanalen om zo ondersteuning te bieden voor het sturen van data voor stereo geluid.

- SRAM** Static Random Access Memory is een geheugentype, dat zijn lading behoudt zolang de spanning op de chip blijft aangesloten. Het geheugen moet dus niet continu worden opgeladen tijdens de werking zoals bij SDRAM.
- SAV** Start of Active Video is de synchronisatiebyte, bij de ITU-R-656 standaard (zie ITU-R-656), die aangeeft dat een beeldlijn start en dat de synchronisatie eindigt.
- S-video** Separate Video is een standaard voor videotransmissie, waarbij de chrominantie en de luminantie over een afzonderlijke geleider worden doorgestuurd.
- USB** Universal Serial Bus is het digitale busprotocol, dat tegenwoordig de meest gebruikte interface vormt in computers. De eerste versie voorziet een snelheid van 12 MB/s. De tweede versie ondersteunt snelheden tot 60 MB/s.
- VGA** Video Graphics Array is de analoge standaard, om videobeelden door te sturen, die meestal gebruikt wordt bij computermonitors. De 3 kleurcomponenten (RGB) worden over 3 afzonderlijke geleiders doorgestuurd. Het voordeel van VGA ten opzichte van bijvoorbeeld Composiet is dat VGA grotere resoluties aan kan.
- YCrCb** YCrCb is een kleurenmodel dat meestal gebruikt wordt in videosystemen. Y stelt hier de luminantie voor. Cr en Cb stellen respectievelijk de rode (rood ten opzichte van groen) en de blauwe (blauw ten opzichte van groen) chrominantie voor.

1.2 Situering

Om deze masterproef te beschrijven is een duiding van het grote geheel, waartoe wordt bijgedragen, noodzakelijk. De masterproef kan gesitueerd worden in het MULTICARCOM-project [1].

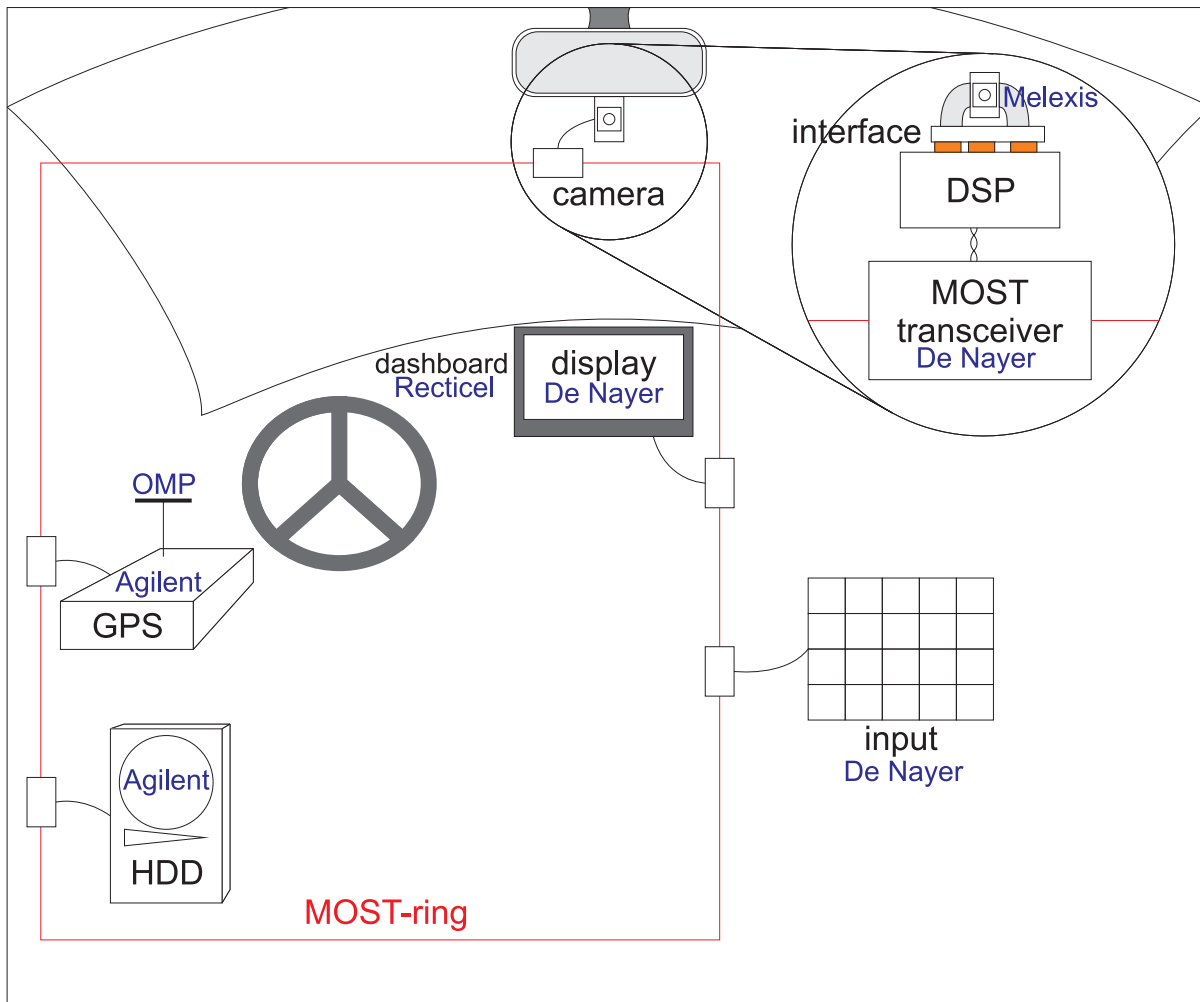
Dit clusterproject gaat uit van Flanders' DRIVE [1], een kennis- en innovatiecentrum voor de Vlaamse automobielsector, gevestigd in Lommel. Flanders' DRIVE ging officieel van start op 2 oktober 2001 en wordt gefinancierd door de Vlaamse overheid en de deelnemende bedrijven (leden). Het MULTICARCOM-project is een samenwerkingsverband tussen 5 leden van Flanders' DRIVE: Agilent Technologies Belgium, Orban Microwave Products, De Nayer Instituut, Recticel en Melexis. Deze samenwerkingsovereenkomst ging in op 1 december 2005.

MULTICARCOM is acroniem voor: "Studie en implementatie van basistechnologieën voor een geïntegreerd multimedia communicatiesysteem in voertuigen." Dit project onderzoekt het samenvoegen van verscheidene multimedia-, communicatie- en navigatiegebonden toepassingen die tot nu toe grotendeels afzonderlijk werden geïmplementeerd.

Door aparte systemen samen te voegen, kan er extra functionaliteit, meer flexibiliteit en uitgebreider comfort worden voorzien. Dit alles zal op een betrouwbaarder manier gebeuren, doordat de verschillende systemen onderling kunnen samenwerken. Bovendien kan de kostprijs gedrukt worden door besparing op onderdelen, want overeenkomstige technologieën worden gedeeld. Het is de bedoeling dat de studie tot een complete demoversie leidt waarin de systemen via de MOST-ring met elkaar communiceren (figuur 1.1).

Hier doet de masterproef zijn intrede in het geheel. Het bedrijf waarvoor de masterproef loopt is Melexis, één van de deelnemers aan het MULTICARCOM-project. Melexis werd opgericht in 1989 en is actief in het ontwerpen van elektronica-componenten en volledige elektronische systemen voor de automobielsector. Het betreft een brede waaier aan producten. Deze masterproef situeert zich in de afdeling opto. Hier wordt gewerkt aan optische systemen, bedoeld voor interne en externe toepassingen in de wagen. Zoals de *FOT's* (*fiber optic transmitter*) om te communiceren over de optische MOST-ring in het kader van het MULTICARCOM-project. Melexis levert ook de camera-applicatie voor het MULTICARCOM-project en het is dit systeem dat deze masterproef behandelt (figuren 1.1 & 1.2).

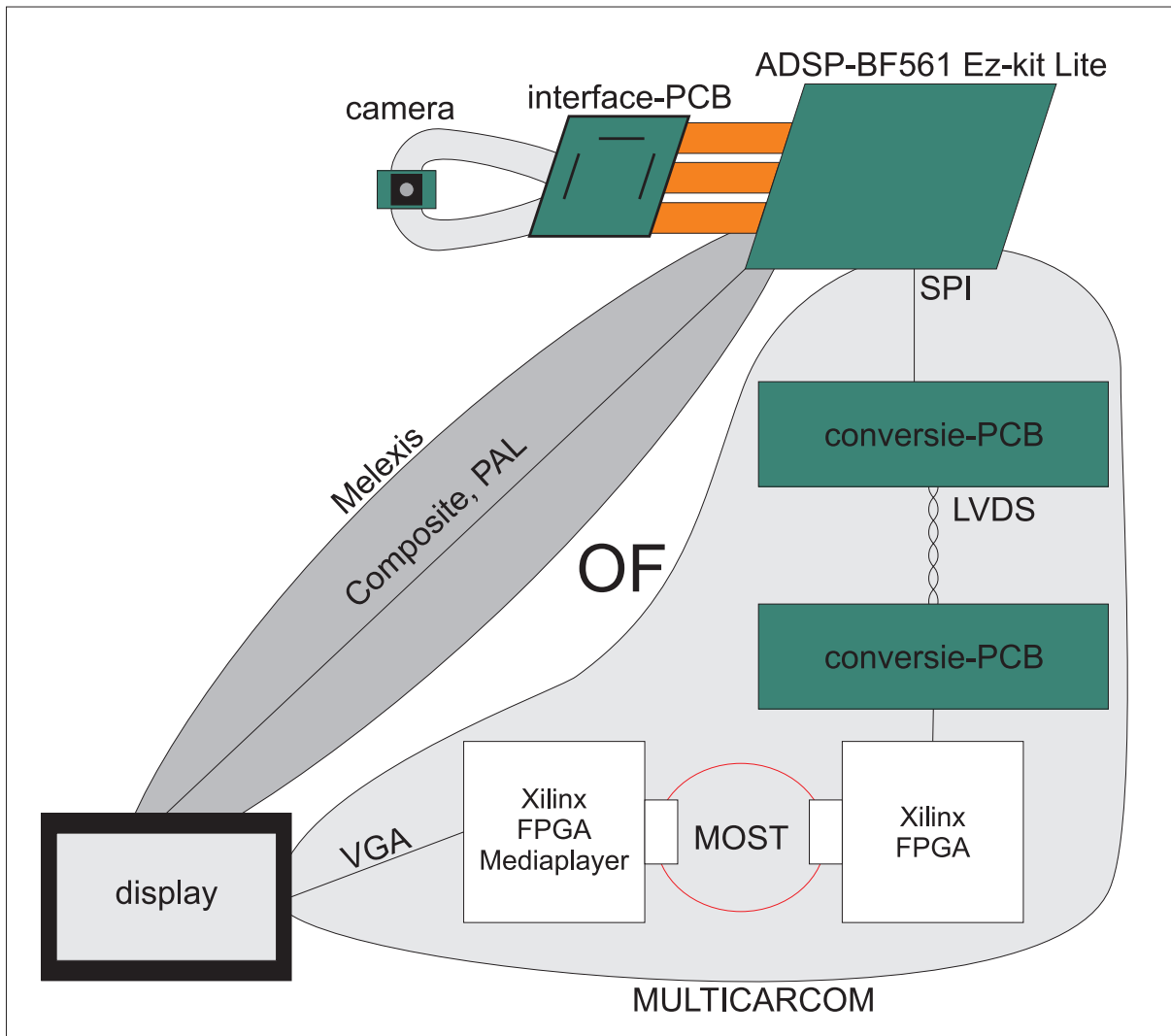
De masterproef heeft tot doel om een camerasysteem op basis van een DSP te ontwerpen dat standalone kan functioneren (figuur 1.2). Er wordt gewerkt met een camera van Melexis, de MLX75007 [2, 3]. Dit is een panoramische VGA (PVGA) camera bedoeld voor de automobielsector met een resolutie van 756×400 pixels en 9-bits grijswaarden. De camera bevat één 159-bits register, voor alle parameters, dat zich via SPI laat instellen. De camera moet aangestuurd worden en de videobeelden moeten verder verwerkt worden. Daarvoor wordt de ADSP-BF561 EZ-KIT Lite[®] [4] van Analog Devices gebruikt. Dit is een evaluatiebord dat ontworpen is voor audio- en videotoe toepassingen. Het bord bevat onder andere een video-decoder, een video-encoder, een audio-codec, 64 MB SDRAM-geheugen, 8 MB flashgeheugen en speciale interfaces geoptimaliseerd voor audio (SPORT) en video (PPI). Het demobord beschikt ook over een ruime selectie aan *I/O* en dat voor verschillende internationale standaarden van audio en video. De basis van dit demobord is een *dual core* DSP, de BF561.



Figuur 1.1: Een voorstelling van de verscheidene toepassingen, die in de wagen zullen worden geïntegreerd, als één groot mediasysteem. Hierbij is een eenvoudige taakverdeling tussen de verschillende projectpartners weergegeven. Bovenaan wordt er ingezoomd op het camerasysteem, dat het doel van deze masterproef is.

Om de camera aan het demobord te koppelen werd er een PCB ontworpen die de nodige interfacing van signalen voorziet. Het PCB-ontwerp, zowel het schema als de layout, gebeurde tijdens een 2 weken durende stage bij Melexis. Tijdens deze stage werden ook algoritmes geschreven voor een *realtime* automatische video-optimalisatie, door het continu bijregelen van de cameraparameters. Daarenboven werd er een algoritme gerealiseerd om de lensdistortie, dat is de kromming van de cameralens, te compenseren. Deze algoritmes werden gemaakt in een C++ omgeving en functioneerden op een PC. Deze algoritmes zouden ook nog in de BF561 toegepast kunnen worden.

Het programmeren van de BF561 werd volledig in Ansi-C uitgevoerd. De camerabeelden worden door de BF561 ingelezen en tijdelijk opgeslagen in een circulaire buffer, in het externe geheugen. Alle datatransfers tussen geheugen en periferie gebeuren met DMA. De opgeslagen beelden worden verwerkt en zullen vervolgens worden weergegeven. Voor het naar buiten sturen van de beelden zijn er 2 systemen ontworpen (figuur 1.2).



Figuur 1.2: Een overzicht van de 2 systemen die moeten gebouwd worden. De camera is via een interface-PCB verbonden met het evaluatiebord. Op het evaluatiebord is een beeldscherm aangesloten voor een live weergave van de camerabeelden. Dit gebeurt ofwel via Composiet-video ofwel via de MOST-ring en een VGA-interface.

Het eerste systeem (Melexis) gebruikt de video-encoder om de beelden, gecodeerd (PAL) via een standaard video-uitgang (Composiet), op een aangekoppeld display weer te geven. Ook andere videocoderingen (NTSC) en video-interfaces (S-video, Component Video) zijn mogelijk.

Het tweede systeem (MULTICARCOM/De Nayer) stuurt de video naar een *MOST-transceiver*. Dit gebeurt via een SPI die differentieel (LVDS) geïmplementeerd werd om de stoorgevoeligheid te verminderen. Het gebruikte formaat is nu *RAW-video*, voorzien van synchronisatie voor de SPI. De video wordt via een optische ring verzonden en ontvangen door een andere *MOST-transceiver*. Op dit laatste systeem draait een mediaspeler die zorgt voor weergave van de video op een display, dat via een VGA-interface is aangekoppeld. Het ontwerpen en implementeren van de *MOST-transceivers* en de optische

MOST-ring was geen onderdeel van deze masterproef. Er werd enkel gebruik van gemaakt of mee getest. De LVDS-conversie PCB werd wel getest, maar werd niet in het kader van deze masterproef ontworpen.

Voor beide systemen werd een LCD met *touchscreen* gebruikt, dat voor de automobielsector ontwikkeld is, de NW619VT [5] van Newision. Dit scherm beschikt over een VGA- en 2 Composiet-ingangen waardoor het voor de 2 besproken systemen bruikbaar was. Dit display zal bij de demoversie van het MULTICARCOM-project in het dashboard van de auto worden ingebouwd.

Deze demotoepassingen zullen, eventueel deels gewijzigd, verder gebruikt worden in de totale demoversie van het MULTICARCOM-project.

Hoofdstuk 2

Specificaties

Dit hoofdstuk beschrijft de gevraagde specificaties, waaraan zeker moet worden voldaan. Dit is dus geen beschrijving van de mogelijkheden die de uiteindelijke realisatie zal hebben. De opdracht is om een camerasysteem te bouwen dat op zichzelf kan functioneren. Dit omvat het vastleggen, vervolgens eventueel het bewerken en uiteindelijk de weergave van de beelden. Er worden 2 verschillende systemen verwacht die echter voor een groot deel gelijk zijn. De gevraagde specificaties waaraan het geheel dient te voldoen, zijn 3-ledig:

1. de algemene specificaties (hoofdstuk 2.1)
2. de specificaties van Melexis (hoofdstuk 2.2)
3. de specificaties van MULTICARCOM/De Nayer (hoofdstuk 2.3)

2.1 Algemene Specificaties

Het systeem wordt opgebouwd met de ADSP-BF561 EZ-KIT Lite[®] [4], van Analog Devices, als basis. Dit demobord, op basis van een Blackfin[®] 561 DSP [6] [7], is ontworpen voor audio- en video-toepassingen. Om de videobeelden te maken wordt een camera van Melexis gebruikt, de MLX75007 [2, 3]. Deze camera is speciaal ontworpen voor automotieve toepassingen. Het demobord beschikt over een expansie-interface waaraan de camera wordt verbonden. Hiervoor zal er een hardwarematige interface ontworpen worden. De gebruikte componenten/systemen van deze masterproef zijn voor beide projectpartners gelijk. De MOST-ring en alle systemen van het MULTICARCOM-project op het camerasysteem na, zijn geen projectonderdeel van deze masterproef (hoofdstuk 2.3).

De camera (*slave*) wordt ingesteld door de DSP (*master*), via een SPI. Dit gebeurt op basis van commando's. De camera zal monochrome beelden via een 9-bits parallelle bus naar buiten sturen. De DSP ontvangt deze beelden via een PPI. De *framerate* wordt vanuit de DSP geregeld door middel van een pulserend signaal dat als een *trigger* werkt. De ontvangen beelden dienen tijdelijk opgeslagen te worden door middel van een circulaire buffer in het externe geheugen (SDRAM) van het demobord. De overdracht van beelden van de PPI naar het geheugen moet via DMA gebeuren om de processorkernen zo veel mogelijk te ontlasten van kopieertaken. Zowel de camera als de PPI zullen op de maximum werkingsfrequentie van de camera (33 MHz) moeten functioneren. Tot hier zijn de 2 projectdelen bijna gelijk, de verdere verwerking en de weergave van de beelden zijn echter van een verschillende aard.

2.2 Specificaties voor Melexis

Melexis eist een systeem dat de mogelijkheden van de MLX75007 zo uitgebreid mogelijk benut. Dit is dus een uitbreiding op de algemene specificaties, dat beschrijft hoe de beelden op een aangekoppeld beeldscherm moeten worden weergegeven. De onderstaande beschrijving wordt best gelezen met figuur 2.1 erbij.

De maximale cameraresolutie van 756×400 pixels wordt gebruikt. Het aantal beelden per seconde moet voldoende hoog zijn zodat de gebruiker de impressie van vloeiende video

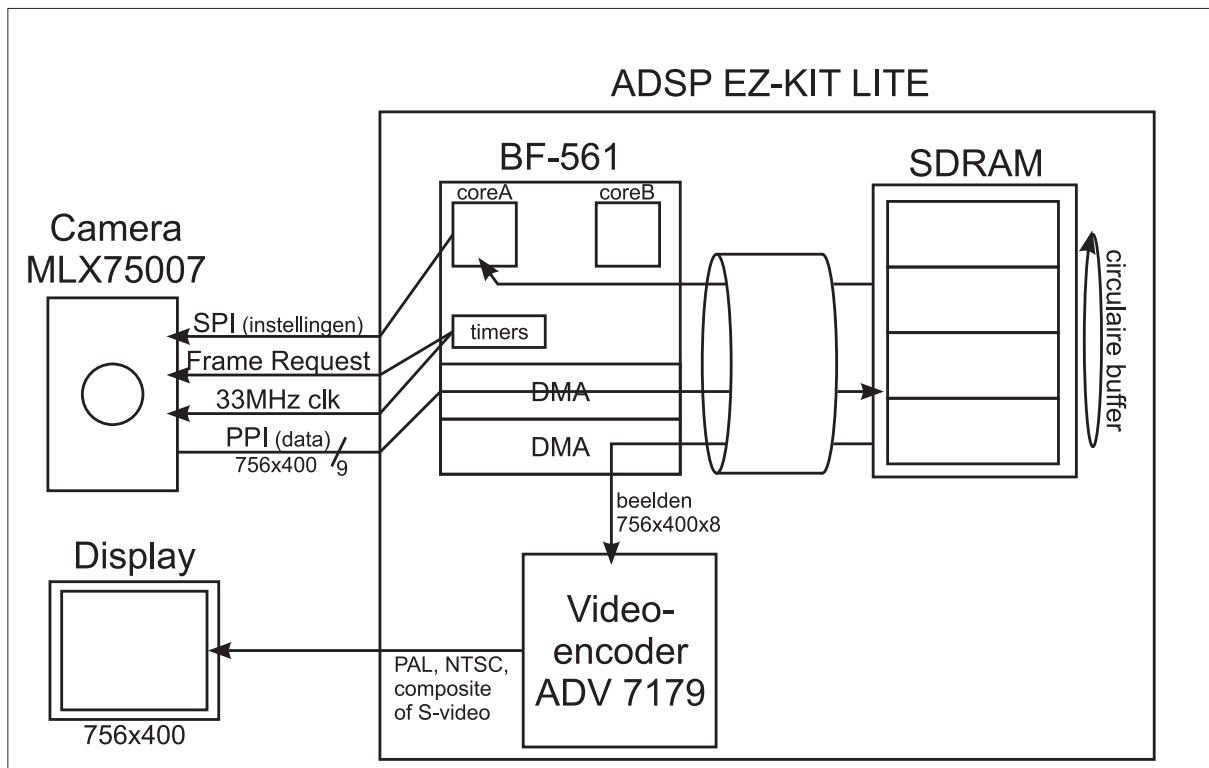
krijgt. De camera zelf werkt optimaal bij 24 fps, dus deze *framerate* is wenselijk. Hogere *framerates* mogen voorzien worden, indien deze de correcte werking van het systeem niet negatief beïnvloeden. De volgende camera van Melexis (derde generatie) zal waarschijnlijk bij een hogere *framerate* functioneren. De camera werkt op basis van 9-bits grijswaarden per pixel. Maar het systeem moet slechts 8 bits per pixel aankunnen. Het instellen van de camera gebeurt via een SPI maar de instructiecode moet zodanig worden gestructureerd dat de SPI eenvoudig door een IIC kan vervangen worden. De specificaties beschrijven dan ook dat de SPI via de SPORT wordt nagebootst. Ook dit is een voorziening voor de volgende generatie camera's van Melexis, welke door middel van IIC zal worden ingesteld.

Eens de beelden in het geheugen zijn opgeslagen, dienen deze sequentiëel verwerkt te worden. Dit omvat het aanpassen van de grijswaarden van 9 naar 8 bits per pixel en (later) het automatisch bijregelen van de camera-instellingen voor een optimale beeldweergave, zelfs bij continu variërende belichtingsomstandigheden.

De gefilmde beelden moeten op een aangekoppeld beeldscherm kunnen worden weergegeven. De aangepaste beelden zullen vanuit het geheugen, opnieuw via DMA, naar de tweede PPI worden gekopieerd. De PPI stuurt vervolgens de beelden parallel per 8 bits naar de video-encoder (ADV7179) van het demobord. Deze moet correct worden ingesteld zodat de beelden aan de video-uitgangen van het demobord beschikbaar zijn voor weergave op een beeldscherm. Het gebruikte formaat (PAL, NTSC, ... & Composiet, S-video, ...) is vrij te kiezen. Indien dit niet werkt, kan er voor gekozen worden om zelf een hardwarematige VGA-interface te ontwerpen. Het maakt dus niet echt uit met welke videocodering en via welke video-interface het te ontwerpen systeem de beelden aan zijn uitgangen beschikbaar stelt. Zolang de maximale cameraresolutie van 756×400 pixels maar gehaald wordt, met 8-bits grijswaarden per pixel en dit alles met de impressie van vloeiende video. Indien mogelijk moet alles gerealiseerd worden door slechts één van de 2 processorkernen, waarover de BF561 beschikt, te gebruiken.

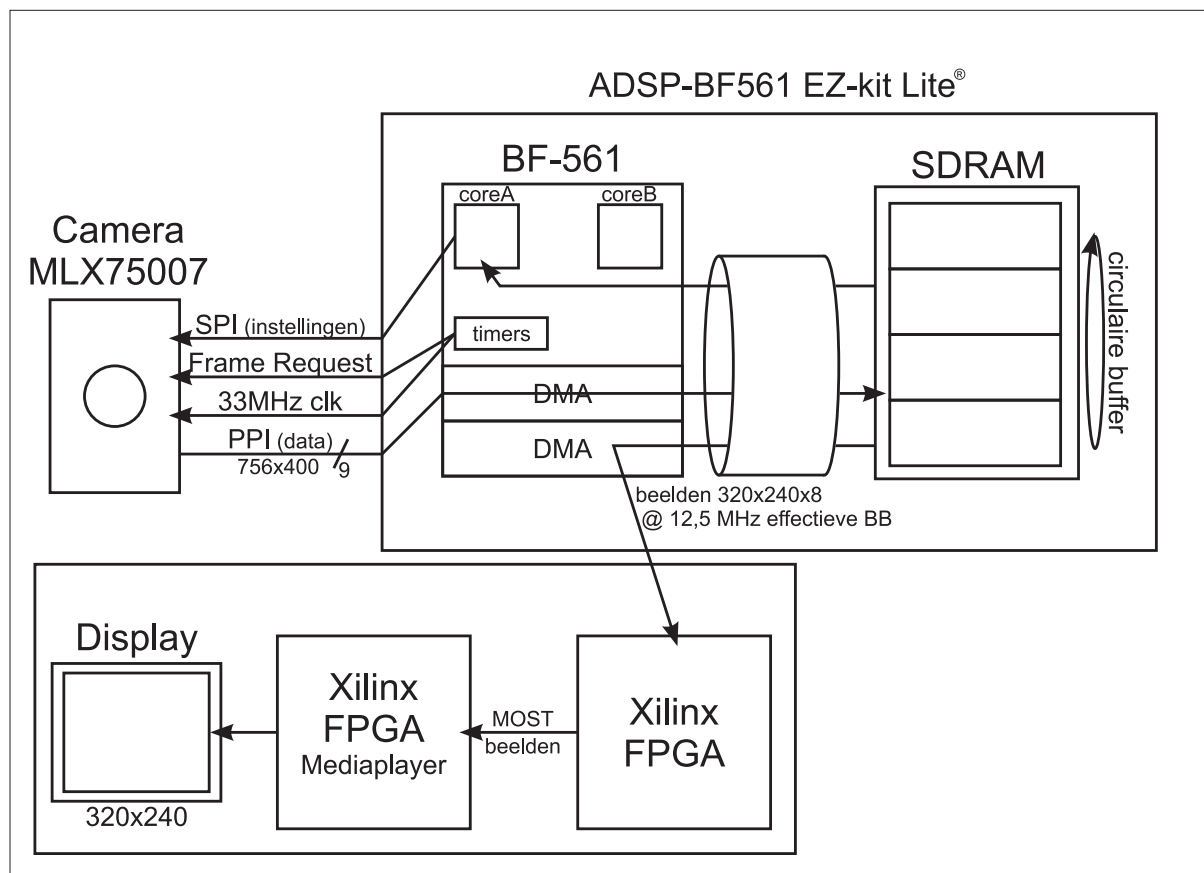
2.3 Specificaties voor MULTICARCOM/De Nayer

MULTICARCOM wil *realtime* over de gefilmde beelden kunnen beschikken, op zijn MOST-systeem, indien nodig. De specificaties voor MULTICARCOM worden in de onderstaande tekst beschreven maar zijn ook afgebeeld in figuur 2.2. De beelden zullen via SPI worden doorgestuurd naar het MOST-systeem. Deze connectie moet differentieel uitgevoerd worden. Door LVDS toe te passen neemt de storingsgevoeligheid af en kan de communicatie-afstand groter genomen worden. Het is noodzakelijk dat het demobord als *slave* fungeert. Het MOST-systeem vereist namelijk dat het als *master* kan functioneren. Het MOST-systeem bepaalt wanneer het demobord de videobeelden moet aanbieden. Voor de synchronisatie van het MOST-systeem, is het absoluut noodzakelijk dat elke byte nuttige data wordt voorafgegaan door één synchronisatiebyte, met alle bits geset ($0xFF$). De datalijn moet laag gaan, als er geen data wordt doorgestuurd. De SPI van het MOST-systeem werkt maximum op 25 MHz. Als gevolg van de toegevoegde synchronisatie zal slechts de helft van de bandbreedte (12,5 MHz) effectief bruikbaar zijn voor data. De beelden zullen bijgevolg op één of andere manier moeten gecomprimeerd worden. Deze compressie mag in het ideale geval MPEG (de versie maakt niet uit) zijn. Indien dit onmogelijk blijkt zullen er toegevingen moeten worden gedaan op de resolu-



Figuur 2.1: Een schematisch overzicht van de specificaties die door Melexis worden geëist. Het grote verschil met de vereisten van MULTICARCOM is de methode die gebruikt wordt om de beelden aan de buitenwereld te tonen. De specificaties van Melexis zijn ook veel strikter wat de gevraagde prestaties betreft.

tie, de *framerate* of het aantal grijstinten. De geëiste resolutie zal standaard reeds lager liggen dan bij Melexis. Een resolutie van 320×240 pixels is voldoende. Ook nu zal de DMA instaan voor het kopiëren van de beelden. Dit keer van het externe geheugen naar de SPI. De camera zal op zijn zijkant liggend worden ingebouwd in de wagen, het is dus noodzakelijk dat de instructiecode een bewerking voorziet om de beelden 90° te draaien. Er is geen verplichting om slechts één processorkern te gebruiken, de keuze is vrij.



Figuur 2.2: Een schematisch overzicht van de specificaties die door MULTICARCOM/De Nayer worden geëist. Het verschil met de eisen van Melexis zit vooral in de manier waarop de beelden aan de buitenwereld worden weergegeven. De specificaties van MULTICARCOM/De Nayer zijn veel strikter wat de gebruikte hardware betreft.

Hoofdstuk 3

Uitwerking

3.1 Hoe is er tewerk gegaan?

3.1.1 Stage

Tijdens onze stage werden de taken volledig verdeeld. Stefan hield zich bezig met het ontwerp van een interface-PCB, die de camera met het DSP-bord zou gaan verbinden, terwijl Dave zich bezighield met het ontwikkelen van algoritmes voor beeldverwerking en beeldoptimalisatie.

3.1.1.1 Algoritmes: Regulator

De parameters van de camera moesten manueel ingesteld worden omdat er hiervoor nog geen functie bestond die dat automatisch deed. Na een grondige studie van de camera en zijn parameters werd er tot een algoritme gekomen dat automatisch enkele parameters, zoals integratietijden en korreligheid, kon instellen.

Een eerste versie van de regulator kon enkel de integratietijd aanpassen. De regulator berekende hiervoor eerst de gemiddelde waarde van een reeks pixels en besliste dan of de integratietijd aangepast moest worden. Hoeveel pixels de regulator moest bekijken kon ingesteld worden via een ini-bestand. Hier kon men instellen hoeveel procent (zowel horizontaal als verticaal) van de pixels de regulator moest bekijken. De regulator nam steeds het midden van het beeld. Wel was het mogelijk om een aantal pixels over te slaan om zo de rekentijd te verkorten. De gewenste waarde van de integratietijd kon ook aangepast worden. Voor de instellingen werd een hysteresis gebruikt zodat de instellingen niet na elk frame aangepast zouden worden. De hysteresis was vast bepaald en kon niet ingesteld worden via het ini-bestand. Deze eerste versie werkte redelijk goed voor egale beelden (bijvoorbeeld binnenshuis, zonder vensters of TL-verlichting), maar gaf een bedroevend resultaat als er veel pixels gesatureerd waren (als er bijvoorbeeld van binnenshuis naar buiten gekeken wordt). Om het bereik te vergroten, werd de korreligheid van de frames (*frame granularity*) ook aangepast. Een verdubbeling van de korreligheid geeft ongeveer hetzelfde beeld als een halvering van de integratietijd.

Een kleine aanpassing van deze eerste regulator werkte het storende effect van flikkerende TL-verlichting weg. Hiervoor werd niet het gemiddelde bekeken van alle pixels van het huidige beeld, maar werd de mediaan genomen van de gemiddelde waarden van de laatste 30 beelden. Hierdoor verdween het storende effect van de flikkerende buislampen. Als deze regulator in de auto-industrie zou gebruikt worden, kan een dergelijke storing bijvoorbeeld voorvallen als men op een weg rijdt waar bomen aan de kant van de weg voor een flikkerende lichtinval zorgen in de lens.

De tweede versie van de regulator keek –naast het gemiddelde– ook naar het aantal gesatureerde pixels en kon de tweede integratietijd (*second slope*) aanpassen. Zo was het mogelijk om de gesatureerde pixels opnieuw te laten belichten om hier ook een duidelijk beeld te krijgen [8]. De *second slope* kan zowel ingesteld worden in de tijd als in spanning, maar de regulator verandert enkel de tijd, omdat dit anders te ingewikkeld zou worden. In de auto-industrie zou dit bijvoorbeeld kunnen voorvallen aan het einde van een tunnel, waar we zowel het verkeer binnenin de tunnel als het verkeer dat al buiten de tunnel is, goed moeten kunnen zien. In hoofdstuk 3.1.1.3 wordt de werking van de *second slope*

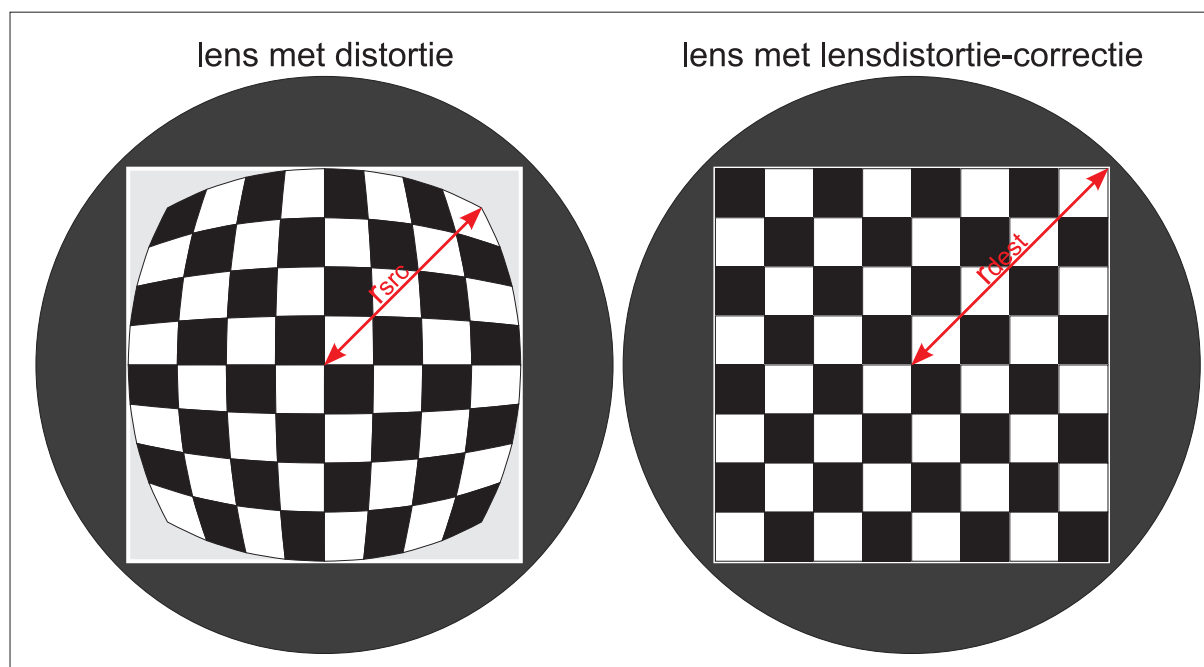
verder uitgelegd.

Een derde versie van de regulator kon eveneens de derde integratietijd (*third slope*) aanpassen, doch slechts enkel in de tijd. Dit gaf echter een slecht resultaat omdat er teveel parameters tegelijkertijd aangepast werden, waardoor de regulator voortdurend alle parameters wijzigde.

Een aanpassing, zodat niet alle parameters zich op hetzelfde moment konden aanpassen, zorgde al voor een beter resultaat. Nadien werden prioriteiten toegevoegd, zodat de regulator de parameters met de hoogste prioriteit als eerste zou aanpassen.

Een vierde versie van de regulator was een *brute-force* regulator. Deze berekende de mediaan van de 30 laatste gemiddeldes van de pixelwaarden en berekende hieruit de volgende parameters: integratietijd, korreligheid, *second slope* (tijd) en *third slope* (tijd). De spanningen van de *second slope* en *third slope* waren vast bepaald. Deze regulator gaf een opmerkelijk goed resultaat in de meeste omstandigheden.

3.1.1.2 Algoritmes: Lensdistortie-correctie



Figuur 3.1: Voorbeeld van een duidelijk zichtbare lensdistortie, die afkomstig is van de fysische eigenschappen van de lens. Op het rechtse beeld is elke pixel terug rechtgetrokken en zo is terug het correcte beeld zichtbaar.

Elke fysische lens heeft randverschijnselen, die ervoor zorgen dat de beelden ombuigen naarmate men zich verder van het fysisch middelpunt van de lens bevindt. Dit fenomeen is vooral zichtbaar bij groothoeklenzen waar de in werkelijkheid rechte lijnen, op de foto niet meer recht zijn (zie figuur 3.1). Melexis heeft groothoeklenzen beschikbaar voor zijn camera's, waardoor het nuttig leek om de lensdistortie te corrigeren. Hiertoe werd een algoritme gevonden [9] dat ons toeliet voor elke pixel (op het beeld) de originele pixel te

vinden. Dit algoritme werkt met de volgende formule:

$$r_{src} = (a * r_{dest}^3 + b * r_{dest}^2 + c * r_{dest} + d) * r_{dest}$$

Hierbij stelt r_{src} de afstand voor van het middelpunt van de lens tot de pixel en stelt r_{dest} de afstand voor van het middelpunt van de lens tot de pixel die op de plaats r_{src} zou moeten staan. De 4 parameters (a, b, c en d) kunnen ingesteld worden via het bestand *lens.ini*. De positie van het fysisch middelpunt van de lens kan hier eveneens ingesteld worden. Indien gewenst is dat de afmetingen van het beeld niet veranderen, moet voldaan zijn aan de volgende formule: $a + b + c + d = 1$. De parameter d is de schaalfactor. Als $a = b = c = 0$ en $d = 1$, dan wordt het beeld niet veranderd.

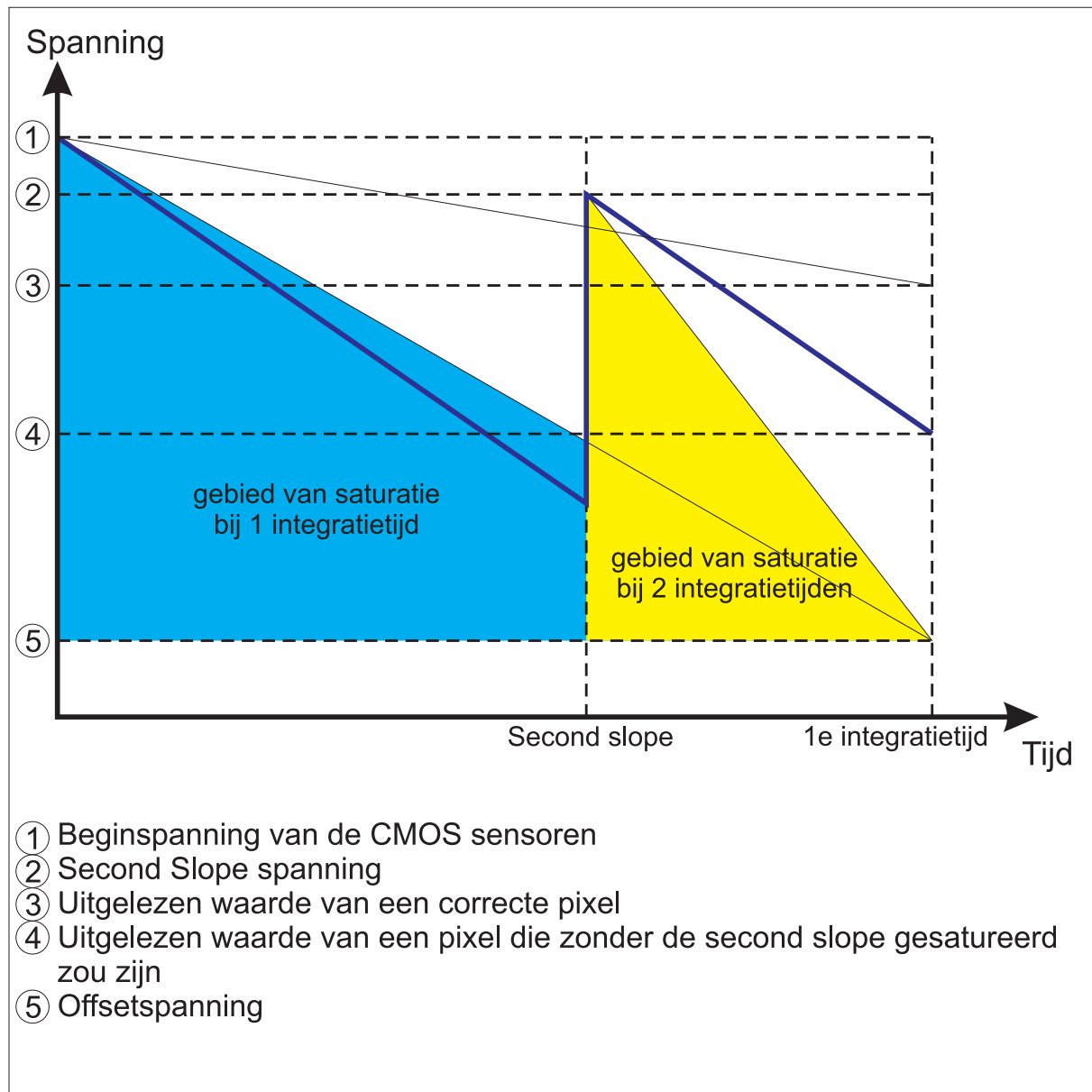
De lenzen die melexis gebruikt, hebben een schroeffitting zodat men aan de lens moet draaien om het beeld scherp te stellen. Dit zorgt ervoor dat het fysisch middelpunt van de lens zich circulair verplaatst en heeft aldus tot gevolg dat men telkens de locatie van het fysisch middelpunt moet instellen telkens men aan de lens draait. Om dit fysisch middelpunt te vinden, moet men een rechte lijn (of bijvoorbeeld een testpatroon in de vorm van een schaakbord) voor de lens verplaatsen in verticale (of horizontale) richting totdat de lijn niet meer gekromd is. Dit is de locatie van het fysisch middelpunt. Een extra functie laat toe verticale en horizontale lijnen weer te geven op de plaats waar het algoritme denkt dat het middelpunt is. Als dit niet overeen komt met het werkelijke middelpunt, dan is het mogelijk deze lijnen te verplaatsen en zo door te geven waar het nieuwe fysisch middelpunt zich bevindt.

3.1.1.3 Slopes

In figuur 3.2 wordt de werking van de *second slope* aangetoond. De camera beschikt eveneens over een *third slope*, maar aangezien de werking hiervan dezelfde is als die van de *second slope*, wordt deze hier niet verder uitgediept. Indien de *second slope* niet gebruikt wordt, moet de tijdsinstelling hiervan op 0 gezet worden.

Een pixel heeft op tijd 0 een beginspanning (op de figuur aangeduid met ①). Als er licht invalt op deze pixel zakt zijn spanning evenredig met de hoeveelheid licht die erop invalt. Indien er teveel licht invalt, zal deze pixel gesatureerd zijn bij integratietijd 1. Als er teveel pixels gesatureerd zijn hebben we een overbelicht beeld, wat niet positief is. Hiervoor biedt de *second slope* een oplossing.

De *second slope* kijkt op een bepaald moment ($g_dwSlope2$) of er pixels zijn die in het gebied van saturatie (na 1 integratietijd) zouden vallen. Indien dit zo is, zal er aan deze pixel een nieuwe spanning aangelegd worden ($g_ResLevSecondSlope$; in de figuur aangeduid met ②). Vanaf nu gaat de pixel opnieuw zakken in spanning als er licht invalt. De hoek die de pixel maakt is een maat voor de hoeveelheid licht die er invalt op de sensor. Als er veel licht invalt zal de hoek groot zijn. Na er opnieuw spanning opgezet te hebben zal normaal gezien de hoeveelheid licht nog ongeveer gelijk zijn zodat de hoek waarmee begonnen is dezelfde is als de hoek waarmee verder gegaan wordt na de *second slope*. De hoek dat deze pixel mag maken om niet gesatureerd te zijn op het einde van integratie, is nu echter veel groter dan aanvankelijk. Hierdoor kan een veel mooier beeld bekomen worden.



Figuur 3.2: De werking van de Second Slope. Voor meer uitleg, zie paragraaf 3.1.1.3

De *third slope* werkt gelijkaardig, maar er moet wel rekening mee gehouden worden dat $g_dwSlope3 \leq g_dwSlope2 < g_dwIntTime1$

3.1.1.4 Interface-PCB

Schematisch ontwerp met Orcad Capture software Om een PCB te ontwerpen was eerst een studie nodig van de interfaces en signalen die voor de camera noodzakelijk zijn, om correct te functioneren [3]. Maar ook een studie van de interfaces, die op de expansie-interface (hoofdstuk 3.2.2.1) van de ADSP-BF561 EZ-KIT Lite[®] aanwezig zijn [4].

De *programmable flags* van het evaluatiebord werden met veel aandacht gekozen, om te voorkomen dat eventueel in het verdere ontwerp nuttige pinnen/signalen/interfaces onbruikbaar zouden worden. Want bijna elke *programmable flag* is met een andere poort *gemultiplxeerd* op één pin.

De camera kan in 2 modi functioneren: een frame en een lijn geactiveerde mode. Er werd geopteerd voor de frame geactiveerde mode. Dit had als gevolg dat er absoluut 3 ingangssignalen moesten voorzien worden, de systeemklok, de *frame-request* en de SPI.

- De *frame-request*-ingang is aan één van de *timerpoorten* (*timer 3*) van de DSP gekoppeld (hoofdstuk 3.3). Deze zal door middel van PWM de benodigde pulsen genereren. In de frame geactiveerde mode handelt de camera de startpulsen van de opeenvolgende beeldlijnen zelf intern af. Hiervoor moest wel de *line-request-in* aan de *line-request-out* verbonden worden.
- De camera heeft nood aan een systeemklok (maximum 33 MHz). Op de expansie-interface van het demobord is een 27 MHz klok uitgangspin aanwezig. Toch werd op de PCB een *jumper* geplaatst waardoor ook een PWM-klok uitgang van *timer 4* kan gekozen worden. Dat heeft als voordeel dat de frequentie aanpasbaar is (hoofdstukken 3.1.3.2 & 3.2.2.3). Zowel de *MCLK-in* als aan de *ADC-CLK* van de camera gekoppeld werden aangekoppeld. De camera-masterklok en de camera-ADC-klok zijn intern niet met elkaar verbonden. De interne ADC heeft namelijk geen kloksignaal nodig wanneer een externe ADC gebruikt wordt. De ingebouwde ADC werd echter wel gebruikt. Er is ook een *MCLK-out* welke als feedback klok diest doet. Deze klok werd teruggekoppeld naar de PPI van de BF561 om als externe klok van deze interface te dienen als dit gewenst zou zijn.
- De derde cruciale poort om de camera te laten werken is de SPI. De camera zal als *slave* in het systeem moeten functioneren (hoofdstuk 2). De BF561 zal, naast het regelen en uitlezen van de camera, dienst doen voor beeldverwerking. De verwerkte beelden zullen eveneens via SPI worden doorgestuurd naar een *MOST-transceiver* (hoofdstuk 3.1.9). De BF561 beschikt over slechts één SPI en moet onvermijdelijk *master* zijn bij de communicatie met de camera. Het MOST-systeem eist dan weer dat de BF561 *slave* is in de communicatie daarmee. Er werd gelukkig reeds bij het ontwerp van de PCB gekozen om 2 selectiemogelijkheden voor cameracommunicatie te voorzien. Zo werden de *SPI-in*, de *SPI-SS* en de *SPI-CLK* van de camera aan de betreffende signalen van de BF561 verbonden. Altijd met de mogelijkheid om

via *jumpers* te kiezen tussen het betreffende SPI of SPORT signaal van de BF561. De camera beschikt over een *SPI-out*, zodat deze bruikbaar is in een *daisy chain*. Deze optie was niet nodig en werd dan ook niet gebruikt.

Een actief lage *reset* is noodzakelijk voor de camera. Als de *resetpin* niet is aangesloten dan wordt de camera in *resetmodus* gehouden via een interne *pull-down* weerstand. Op de PCB werd een *RC-resetcircuit* geplaatst, maar er is ook voor gezorgd dat de *reset* van het evaluatiebord gekozen kan worden. De minimum *resetperiode* van de camera bedraagt 300 ns. In onderstaande formule is de tijd berekend dat de camera nog door het RC-circuit in *reset-modus* gehouden wordt, na het aanleggen van de voedingsspanning.

$$T = 2 * \pi * R * C = 2 * \pi * 10 \text{ k}\Omega * 100 \text{ nF} = 6,3 \text{ ms}$$

De camera beschikt over 512 grijswaarden (9 bits) welke parallel via 9 datapinnen (D8 (MSB) tot D0 (LSB)) uitleesbaar zijn. De ADSP-BF561 EZ-KIT Lite[®] beschikt over 2 PPI's. PPI0 werd gebruikt omdat deze standaard bedoeld is als ingangsiinterface voor videobeelden. PPI1 werd vrij gehouden. PPI1 is verbonden aan de video-encoder van de ADSP-BF561 EZ-KIT Lite[®]. Er is ook voorzien dat de beide PPI's op dezelfde frequentie kunnen functioneren indien gewenst. Om die reden werd een *jumper* voorzien om de *MCLK-out* van de camera ook naar de PPI1 van de BF561 terug te koppelen. De PPI is een parallelle interface, bedoeld voor video-applicaties, en beschikt daarom over 3 synchronisatiekanalen. De *line-valid* en de *frame-valid* van de camera werden respectievelijk verbonden aan de *PPI0-synch1* en de *PPI0-synch2*. *PPI0-Synch3/Field* is niet gebruikt (hoofdstuk 3.3). Via *jumpers* is de mogelijkheid open gelaten om de *line-valid* en *deframe-valid* aan 2 *programmable flags* (PF10 en PF11) te verbinden, zodat de synchronisatie ook softwarematig kan worden afgehandeld.

De camera beschikt ook nog over een *spare3-pin*. Dit is een signaal dat ervoor zorgt dat een paar interne onderdelen van de camera periodiek een *soft-reset* ondergaan. De pin werd aan een *programmable flag* gekoppeld (PF9).

Op de expansie-interface van het evaluatiebord is 3,3 V voedingsspanning aanwezig. De camera werkt op 3,6 V waardoor deze meteen volstond en geen spanningsdeling nodig was. Alle voedingspinnen werden aangesloten en dit gebeurde tevens voor de grondpinnen.

Alle andere camera-aansluitingen zijn voor deze toepassing overbodig en werden niet aangesloten. Voor de snelste signalen werd een voorziening getroffen tegen eventuele reflecties, waardoor communicatie onmogelijk zou worden. Op deze printsporen is een *dummy-weerstand* van 1 Ω geplaatst die bij reflecties dan door een ander exemplaar, met een grotere weerstand, kan vervangen worden. Voor meer controle werd er ook nog een *dip-switch* voorzien, naast de 4 druktoesten die al op de ADSP-BF561 EZ-KIT Lite[®] aanwezig zijn. Voor de testbaarheid van het systeem zijn alle controle- en statussignalen op de PCB via een *jumper* of een testpunt beschikbaar. Hierdoor kunnen alle signalen eenvoudig worden nagemeten terwijl het systeem in werking is.

PCB Layout ontwerp met Orcad Layout software Na het schematisch ontwerp, waar veel opzoekwerk aan voorafging, moest het schema nog in een bruikbare PCB-layout verwerkt worden [10]. Eerst moest het programma ingesteld worden zodat de gewenste

lengte-eenheden (Mils) gebruikt werden. De lagen die niet gebruikt zouden worden, werden uitgeschakeld. De gewenste specificaties, zoals de breedte van de printsporen en de minimaal toegestane afstand tussen 2 naburige connecties, werden ingesteld. Vervolgens werden de *footprints* voor de gebruikte componenten opgezocht. Als de *footprint* nog niet bestond in de layout-databank, van het softwarepakket, dan moest deze getekend worden. Dit was nodig voor de TFC-145-32-F-D connectoren (90 pinnen) [11, 12], van Samtec. Hiervoor moest eerst de nummering van de connectoren bekend zijn, want deze passen maar op één manier in elkaar [13]. Er moest ook rekening gehouden worden met de absolute positionering van de expansieconnectoren ten opzichte van elkaar. Deze informatie was moeilijk te vinden. De voorkeur is uiteindelijk naar *flatcables* uitgegaan. Een goed driedimensionaal voorstellingsvermogen van het tweedimensionale gelaagde beeld is noodzakelijk om geen foutieve connecties te maken (hoofdstuk 3.3). In de bijlagen werd het schema opgenomen met het originele ontwerp van de layout (figuren 5.1 & 5.2). Gedurende de verdere masterproef zou er altijd met deze PCB verder gewerkt worden. Op de PCB werden wel handmatig aanpassingen aangebracht.

3.1.2 Werken met Visual DSP++

Na een proefversie van Visual DSP++ 4.5 gedownload en geïnstalleerd te hebben, moesten we nog met dit nieuwe pakket leren werken. Hiervoor werd de *tutorial* (die bij de help zit) geconsulteerd. Uit deze *tutorial* werd geleerd hoe het programma opgestart werd, hoe eenvoudige programmaatjes gecompileerd moesten worden, welke *breakpoints* het programma al dan niet laten stoppen, enz. Het pakket beschikt echter over zoveel functies die voor deze masterproef niet van toepassing zijn, dat die niet allemaal bekeken werden. De functies die wel gebruikt werden, zijn onder andere de functies om beelden te bekijken (*image viewer*) en om de registers uit te lezen.

3.1.3 Camera-aansturing

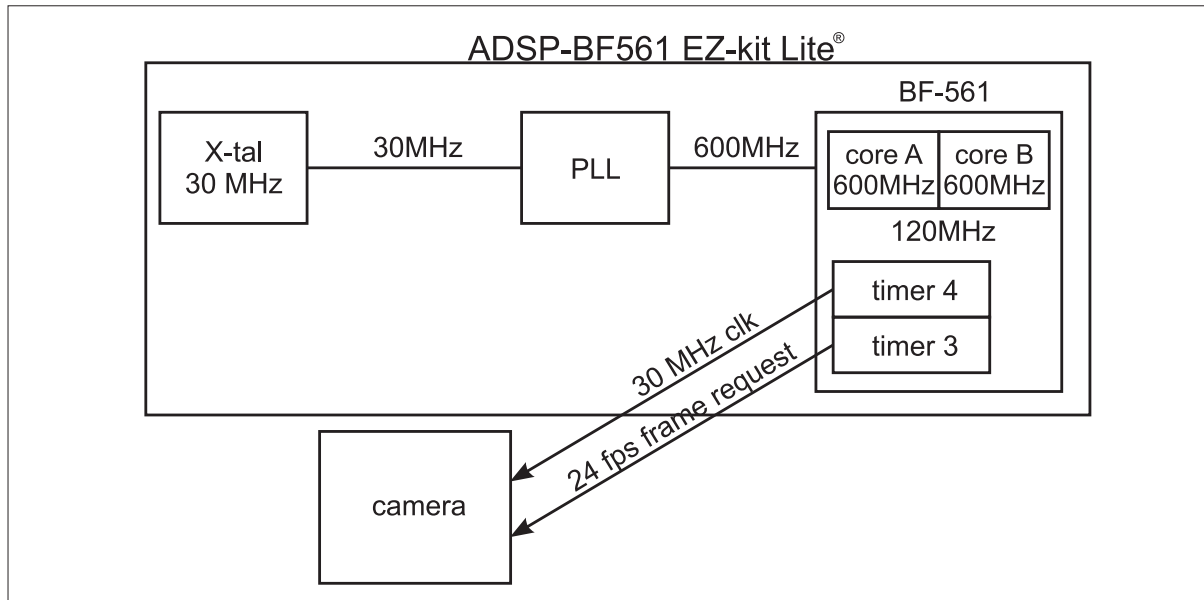
Nadat de syntax van de code was geoefend en de ADSP-BF561 EZ-KIT Lite[®] geleverd was, werd er onmiddellijk van start gegaan. De talrijke functies en instellingen van het evaluatiebord dienden zo snel mogelijk begrepen te worden. De kennis werd stapsgewijs opgebouwd door met de eenvoudigste zaken te starten.

3.1.3.1 Programmable flags

Het eenvoudigste onderdeel dat nodig was waren de *programmable flags* (hoofdstuk 3.2.2.2). Het demobord beschikt over een aantal LED's die elk aan een *programmable flag* zijn gekoppeld [4]. Bij een eerste test moesten deze zoals gepland oplichten. Het moeilijkste deel was om uit te zoeken hoe de parameterregisters [6] via de code konden beschreven en uitgelezen worden. Nadat er 2 dagen lang met *C-pointers* was geprobeerd, bleek dat er vaste functies bestonden. Er moest gewoon een extra *header-file* van Analog Devices in de code opgenomen worden om deze algoritmes bruikbaar te maken. Zoals wel vaker zal blijken is de documentatie bij de code van Analog Devices zeer beperkt en moet de ingenieur zelf de mogelijkheden uitpluizen zonder in de chaos te verdrinken. Zodra de LED's controleer-

baar waren, werden de *programmable flags* als ingangen geconfigureerd. Dit werd getest aan de hand van de druktoetsen op het evaluatiebord. Het *spare3*-testsignaal (hoofdstuk 3.1.1.4) van de camera is met een *programmable flag* gemaakt. Voorlopig wordt dit signaal continu hoog gemaakt. In een later stadium zullen er pulsen op dit signaal komen voor een verhoogde beeldkwaliteit.

3.1.3.2 Timers



Figuur 3.3: Een weergave van alle tijdsinstellingen: de PLL, de processorkernen en het systeem. Dit is de tijdsreferentie voor alle systemen van de BF561 waaronder de timers, die de systeemklok als referentie gebruiken om PWM-signalen te genereren.

Het was nu mogelijk om een *programmable flag* als ingang of als uitgang te gebruiken. Als tweede onderdeel dat absoluut noodzakelijk zou zijn, werden de *timers* (hoofdstuk 3.2.2.3) onder de loep genomen [6]. Er zouden al zeker 2 *timers* nodig zijn.

De cameraklok zou met een *timer* moeten worden gegenereerd, zodat deze op de juiste frequentie zou functioneren, doch deze werkingsfrequentie moest nog eenvoudig aanpasbaar blijven. Aangezien de cameraklok maximaal 33 MHz mag zijn, was het wenselijk om dit als standaard werkingsfrequentie te nemen. Dan zouden lagere frequenties zoals 16,5 MHz en 11 MHz, indien nodig, ook voorzien kunnen worden.

De tweede toepassing van een *timer* is nodig om de *frame-request* te genereren. Dit signaal is zeer belangrijk, omdat bij iedere puls de camera het huidige beeld vastlegt om dit vervolgens parallel naar buiten te sturen. Dit PWM-signaal vereiste, nog meer dan de werkingsklok, een eenvoudige aanpasbaarheid. Zo zou er naar hartelust tussen verschillende *framerates* gewisseld kunnen worden, bij toekomstige testen. De specificaties eisen namelijk een stroom van beelden die als vloeiend worden waargenomen door het menselijk oog (hoofdstuk 2). De camera zelf functioneert optimaal bij 24 fps maar het was niet bekend of de DSP dan nog met alle bewerkingen zou kunnen volgen.

Alvorens het echter mogelijk was om iets zinnigs met een *timer* te doen, moesten eerst de verschillende systemen en de 2 processorkernen van de DSP op de op de correcte frequentie functioneren. De *timers* gebruiken normaal de systeemklok als referentie (hoofdstuk 3.2.2.3). Deze wordt gedeeld door een geheel getal. Om dus een 33 MHz klok te maken, zou de systeemklok een veelvoud hiervan moeten zijn. De tijdsparameters van de DSP laten dit echter niet toe (hoofdstuk 3.3) en er werd uiteindelijk beslist om de 2 processorkernen te laten functioneren op 600 MHz en de andere systemen van de DSP op 120 MHz (figuur 3.3). Hierdoor is de cameraklok 30 MHz.

Een *timer* instellen bleek relatief simpel. Het volstond om 4 registers per *timer* te beschrijven [6]. De toewijzing van *timers* aan een camerafunctie was echter niet meer vrij te kiezen. Door het ontwerp van de PCB werd reeds vastgelegd aan welke *timerpoort* de klok en de *frame-request* van de camera waren doorverbonden (hoofdstuk 3.1.1.4). Beide *timers* werden ingesteld om te werken in PWM-uit modus, waardoor de bijbehorende *timerpoorten* telkens uitgangen werden (hoofdstuk 3.2.2.3).

Timer 4 is gebruikt voor de klok van de camera. De 120 MHz systeemklok moest door 4 gedeeld worden en dit met een *duty-cycle* van 50% om zodoende een perfecte 30 MHz klok voor de camera te verkrijgen (figuur 3.3).

Voor de *frame-request* werd *timer 3* genomen. De periode werd variabel ingesteld, maar initieel werd er gestart met een *frame-rate* van 24 fps (figuur 3.3). Dat betekent dat er iedere 41,7 ms een nieuw beeld gemaakt wordt. Hierdoor zou er voldoende tijd zijn om een beeld volledig in te lezen en vervolgens te verwerken. Onderstaande berekening geeft weer hoe lang één beeldtransfer duurt. Er zouden nog 31,6 ms over blijven voor verdere verwerking van de beelden door de DSP.

$$resolutie_{camerabeeld} / klokfrequentie_{camera} = t_{beeldtransfer}$$

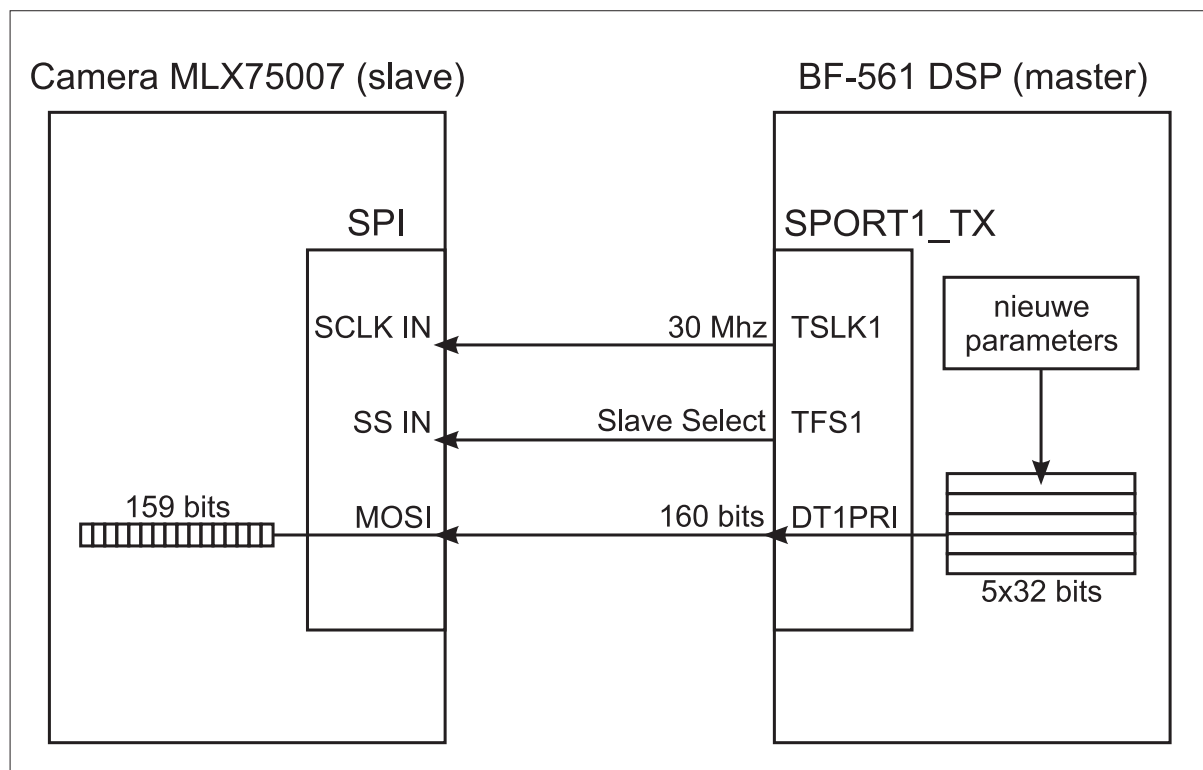
$$(756 \text{ pixels} / \text{beeldlijn} * 400 \text{ beeldlijnen}) / 30 \text{ MHz} = 10,1 \text{ ms}$$

Voor de *frame-request* was er echter geen *duty-cycle* van 50% nodig want er was slechts een korte puls gewenst. De dalende flank, bij het verdwijnen van de puls, zal namelijk het doorsturen van het nieuwe beeld starten (hoofdstuk 3.2.1.3). De pulsen mogen niet te lang zijn, want dan blijft er niet genoeg tijd tussen opeenvolgende pulsen over om een volledig beeld te transfereren. Als de pulsen echter te kort worden gemaakt dan zou het onmogelijk zijn om alle camera-instellingen gedurende de *frame-request*-puls te veranderen (hoofdstuk 3.1.3.3). Voor de lengte van de puls werd een vaste tijdsduur van 60 μ s gekozen.

Of de *timers* daadwerkelijk functioneerden zoals gewenst, werd met een oscilloscoop geverifieerd. De waargenomen problemen worden besproken in hoofdstuk 3.3.

3.1.3.3 SPI

Toen de *programmable flags* en de *timers* volledig instelbaar waren, was het mogelijk om de camera van alle nodige signalen te voorzien om deze op een onvoorspelbare manier te laten werken. Om de camera echter correct te laten functioneren, diende deze eerst met de juiste parameters ingesteld te worden.



Figuur 3.4: De camera krijgt zijn instellingen van de DSP, via de SPI. Bij deze communicatie is de DSP als master en de camera als slave ingesteld. De DSP trekt de slave-select hoog om aan te geven dat er nieuwe data is en verzendt één pakket van 160 bits. Zodra het pakket verstuurd is wordt de slave-select lijn terug laag.

De camera beschikt hiervoor over een SPI [3]. Deze interface kan alleen correct functioneren als de 2 communicerende systemen gelijkaardig zijn ingesteld. Deze SPI werkt op dezelfde frequentie als de camera. Dus als de camera op 30 MHz werkt dan zal de SPI ook op 30 MHz moeten functioneren. De camera heeft één 159-bits register dat alle cameraparameters bepaalt. Deze bits moeten in één stroom van 160 bits (MSB eerst) worden verstuurd. Dit moet 160 zijn om een veelvoud van 8 te zijn. De eerste bit is een *dummy* bit [14]. Het register in de camera verschuift de data op de stijgende klokflank en leest deze in op de dalende klokflank. Dus bij de DSP moet de data ook op een stijgende klokflank worden verschoven (figuur 3.4).

Er werd voor gekozen om als SPI niet de standaard ingebouwde hardwarematige SPI van de BF561 te gebruiken. In de plaats daarvan werd beslist om de SPI na te bootsen met de SPORT [15]. De BF561 beschikt over 2 SPORT interfaces (hoofdstuk 3.2.2.5), maar er werd geopteerd om SPORT1 te gebruiken. Dit verliep niet meteen zoals gewenst. Er moesten een aantal problemen omzeild worden (hoofdstuk 3.3). Om te bevestigen dat de camera correct was ingesteld, werden enkele van de parallelle uitgangspinnen van de camera met de oscilloscoop nagemeten. Als de camera juist ingesteld is dan moet de data gemeten aan de uitgangspinnen ook correct zijn. Vervolgens werd de lens afwisselend afgedekt of fel belicht. Bij een afgedekte lens zouden de gemeten signalen duidelijk laag moeten zijn en bij felle belichting duidelijk hoog. De SPORT werd om meerdere redenen

gebruikt.

- Melexis had liefst dat de SPI met SPORT zou nagebootst worden (hoofdstuk 2).
- De hardwarematige SPI kan maximaal functioneren bij een frequentie die 25% van de systeemklok bedraagt. Hiermee zou meteen het maximum van de mogelijkheden van deze interface benut worden, met de gebruikte werkingsfrequentie van 30 MHz. De SPORT daarentegen werkt maximaal bij een frequentie die 50% van de systeemklok bedraagt. Hierdoor kunnen er nog hogere frequenties gehaald worden, indien dit in de toekomst nodig zou zijn. Want de derde generatie camera van Melexis zal mogelijk bij 50 MHz functioneren.
- De derde generatie camera van Melexis zal IIC gebruiken in plaats van SPI om de parameters in te stellen (hoofdstukken 2 & 3.2.1.3). De BF561 heeft geen hardwarematige IIC ingebouwd, maar er is wel een standaard optie om met de SPORT een IIC na te bootsen. Een kleine aanpassing in de algoritmes volstaat om de interface als IIC te laten functioneren.

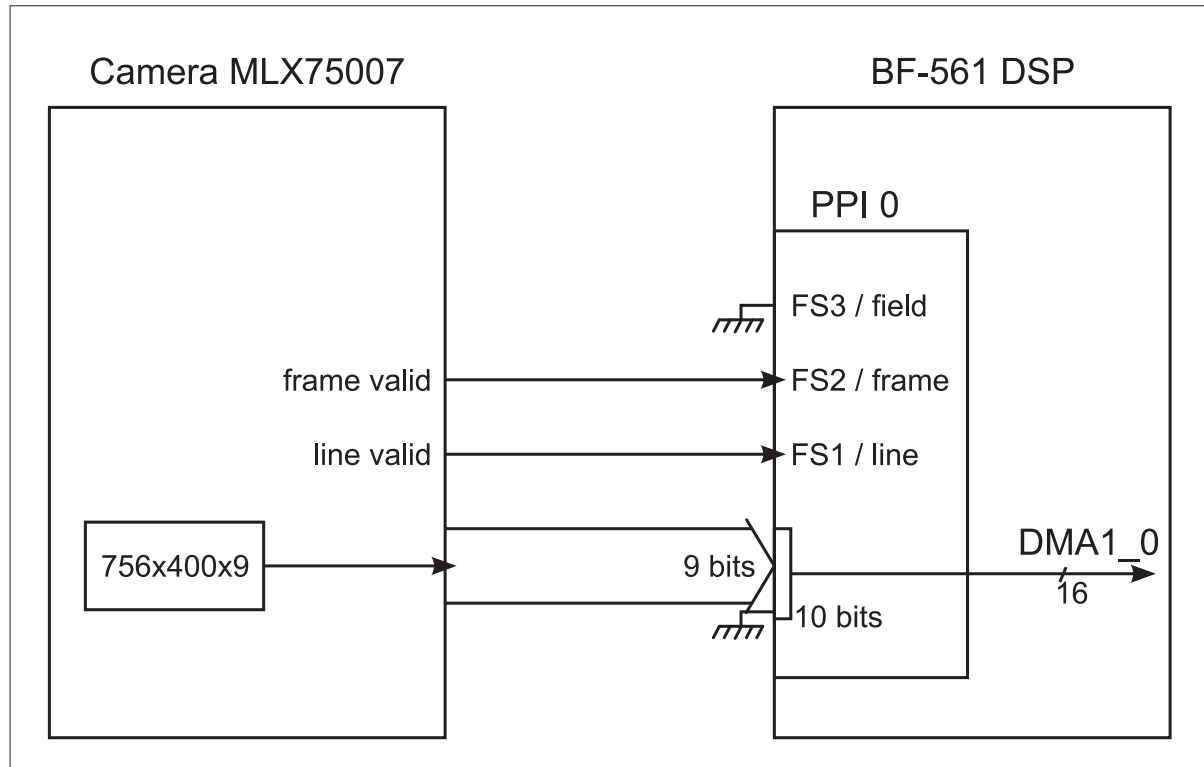
Alle opties werden wel nog open gehouden, door de PCB van de nodige *jumpers* te voorzien, zodat er in de toekomst nog tussen SPI en SPORT gewisseld zou kunnen worden (hoofdstuk 3.1.1.4).

In de instructiecode werd meteen een functie voorzien, die kan opgeroepen worden als de parameters van de camera zouden moeten gewijzigd worden. Om eender welke cameraparameter te kunnen wijzigen, moet dat tijdens de *frame-request* puls gebeuren (hoofdstuk 3.2.1.3). Indien er nieuwe parameters zijn, zal het algoritme wachten op de start van een *frame-request* puls (stijgende flank). Deze begint namelijk pas als *timer* 3 terug van 0 begint te tellen. Om dit te detecteren, werd de *timer* zodanig ingesteld, dat deze aan het eind van elke periode (*overflow*) een *interrupt* zou genereren. De *interrupt* zelf werd gemaskeerd. Er zal echter wel, iedere keer dat dit gebeurt, een bit *geset* worden in het statusregister van de desbetreffende *timer*. Door deze bit te bemonsteren weet de functie dat er een nieuwe *frame-request* puls gestart is. De functie zal de bit terug op 0 zetten en roept een tweede functie op die instaat voor het daadwerkelijk verzenden van de parameters. Het doorsturen van alle 160 bits duurt bij een frequentie van 30 MHz exact $5,33 \mu\text{s}$. De *frame-request* puls kreeg een tijdsduur van $60 \mu\text{s}$ toegewezen (hoofdstuk 3.1.3.2). Hierdoor is er voldoende tijd over om te garanderen dat de volledige datastroom, die de parameters bevat, door de camera wordt ontvangen vooraleer de *frame-request* puls gedaan is (dalende flank). Bovendien moest er voldoende tijd worden voorzien, zodat er genoeg reactietijd was. Doordat er niet echt met *interrupts* gewerkt wordt, maar enkel een bit wordt bemonsterd, is dit noodzakelijk. Want de hoofdlus zal nog andere algoritmes gaan bevatten, waardoor de gebeurtenis (nieuwe cameraparameters) met enige vertraging kan gedetecteerd worden.

3.1.4 Eén camerabeeld ontvangen en tijdelijk opslaan

De camera is nu wel regelbaar maar dit heeft weinig nut als de beelden van de camera niet kunnen worden ingelezen, tijdelijk opgeslagen en weergegeven.

3.1.4.1 PPI



Figuur 3.5: Elk camerabeeld (756×400 pixels) wordt pixel per pixel (9-bits grijswaarden) doorgestuurd van de camera naar de DSP via een 9-bits parallelle verbinding. De hardware is voorzien van 2 synchronisatiesignalen die aangeven wanneer er een beeld of een lijn van dat beeld worden doorgestuurd. De data wordt door de PPI ontvangen als 10-bits per pixel en door de DMA naar het geheugen gekopieerd als 16-bits per pixel.

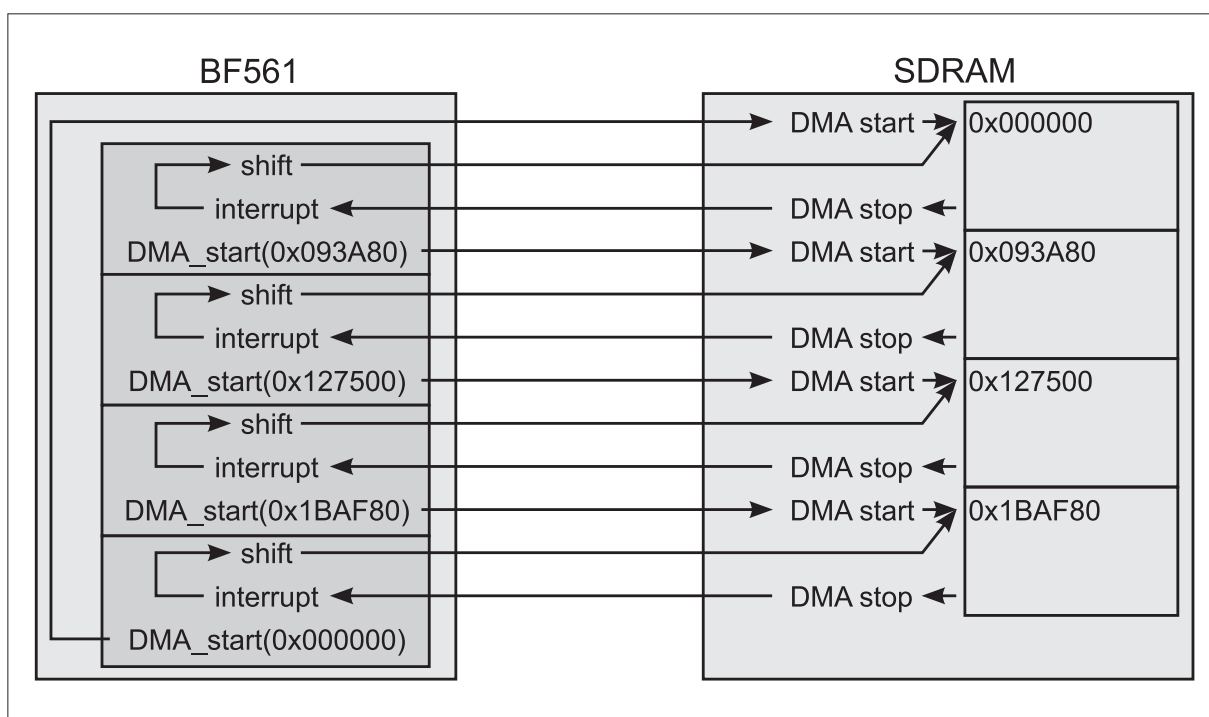
De BF561 beschikt over 2 PPI's (hoofdstuk 3.2.2.6). Er werd voor gekozen om PPI0 te gebruiken. PPI1 is namelijk verbonden aan de video-encoder [4] en deze zou later nog nodig zijn om de videobeelden naar een beeldscherm te sturen voor weergave (hoofdstuk 3.1.7). PPI0 is op zijn beurt aan de video-decoder verbonden voor het inlezen van video [4]. De video-decoder wordt echter niet gebruikt.

De videobeelden komen rechtstreeks van de camera via een 9-bits parallelle verbinding die ingesteld is op een snelheid van 30 MHz. De PPI is een component, die speciaal voor videotoeepassingen ontworpen is. Hierdoor beschikt deze over speciale synchronisatiesignalen (hoofdstukken 3.1.1.4, 3.2.2.6 & 3.3) en bovendien kan er gekozen worden uit hoeveel lijnen de parallelle verbinding moet bestaan. Aangezien de PPI geen ondersteuning voor 9 bits biedt, werd er voor 10 bits geopteerd (figuur 3.5). Toen later bleek dat 8-bits grijs-tinten voldoende zouden zijn, werd beslist om dit bij de verdere verwerking op te lossen door pixel per pixel alle beelddata 1 bit naar rechts te verschuiven. In de toekomst zou er een optimalisatie kunnen gebeuren door de parallelle verbinding te versmallen tot 8 bits (hoofdstuk 4.1.4.1).

In tegenstelling tot de meeste periferie, beschikt de PPI niet over een dataregister

(hoofdstuk 3.2.2.6), waarvan ontvangen data kan gelezen of waarin te versturen data kan geschreven worden. De gegevens worden meteen tussen PPI en geheugen (intern of extern) getransfereerd door middel van DMA (figuur 3.5). In de specificaties (hoofdstuk 2) werd reeds geëist dat de processoren zo veel mogelijk zouden ontlast worden van geheugentransacties door de DMA hiervoor in te schakelen. Er was echter gepland om eerst de data, die door de PPI was ontvangen door de processoren naar het geheugen te laten schrijven. Indien dit door één van de processoren zou gedaan moeten worden, zouden er 10,1 ms verloren gaan waarin er geen andere taken zouden kunnen worden uitgevoerd en dat enkel voor datatransfer (hoofdstuk 3.1.3.2). Zodra dit dan naar wens functioneerde, zou deze taak aan de DMA worden toegewezen. Nu was er geen enkele keuze. De PPI, de *interrupts* en de DMA, zouden tegelijkertijd moeten worden geconfigureerd.

3.1.4.2 Interrupts

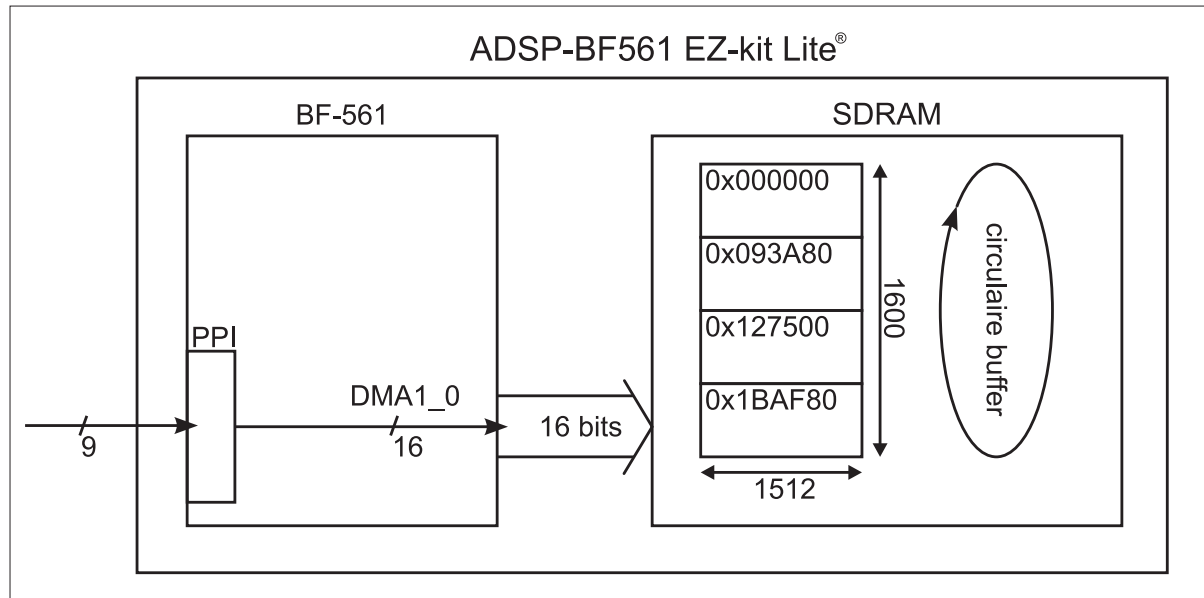


Figuur 3.6: *Elke keer er een volledig videoframe is ingelezen door de DMA, volgt er een interrupt. De ISR past het startadres aan (voor de volgende buffer) en herstart de DMA. Er wordt ook voor gezorgd dat de data in de net ingelezen buffer verwerkt wordt.*

Waar het bij de vorige onderdelen nog mogelijk was om de instelmogelijkheden eenvoudig terug te vinden in de *datasheets*, was dit bij de *interrupts* totaal niet aan de orde. Alle informatie is wel aanwezig, maar deze staat zodanig verspreid doorheen de documentatie [6], dat alleen personen die reeds geruime tijd professioneel met de BF561 bezig zijn geweest hier de rode draad in zouden terugvinden. Het grootste probleem was dat er nergens wordt vermeld, hoe aan de compiler kan worden duidelijk gemaakt dat een algoritme voor het afhandelen van een bepaalde *interrupt* bedoeld is. Er drong zich een nieuw onderzoek op van de voorbeeldcode, die door Analog Devices was voorzien.

Hieruit bleek dat er een speciale functie bestond voor het afhandelen van de *interrupts* (`EX_INTERRUPT_HANDLER`). Deze functie heeft één argument nodig, namelijk een unieke identificatie van een ISR. Om dit te doen moesten echter eerst de *interrupts* ingesteld worden (hoofdstuk 3.2.2.8). Er werd een ISR geschreven, die de *interrupt* behandelt na elke volbrachte transactie van een videoframe naar het SDRAM-geheugen (figuur 3.6).

3.1.4.3 DMA



Figuur 3.7: De video die de PPI ontvangt wordt door de DMA naar het externe SDRAM-geheugen getransfereerd. Daar wordt het tijdelijk in een circulaire buffer opgeslagen.

Vervolgens moest de DMA ingesteld worden (hoofdstuk 3.2.2.7). Dit leek een onbegonnen opdracht vanwege de overvloed aan voorbeeldcodes en de zeer uitgebreide beschrijving in de documentatie [6]. De componenten zijn tussen de *DMA-controllers* niet vrij te verwisselen. Bijgevolg lag meteen vast dat DMA1 zou gebruikt worden om de transfer van de PPI naar het geheugen te doen. PPI0 is bevestigd op kanaal 0 van de *DMA-controller* en had daardoor ook in één keer meteen de hoogste prioriteit op deze DMA-bus. Aan welk kanaal een component is toegewezen, kan gewijzigd worden. Dit werd hier echter niet gedaan.

DMA1 is een 32-bits bus, maar er werd voor gekozen om op 16-bits basis te werken. De PPI was ingesteld om 10-bits data binnen te nemen, welke dan in de 10 LSB's van een 16-bits woord worden gestopt (hoofdstuk 3.2.2.6). Dit gebeurt allemaal door de PPI (hoofdstuk 3.1.4.1). Dit 16-bit woord moet de DMA dan naar het externe SDRAM-geheugen transfereren (figuur 3.7). Omdat er beelden gekopieerd gingen worden werd er voor 2D-DMA gekozen. Hierdoor kon er met een aantal pixels per lijn en een aantal lijnen per beeld gewerkt worden, wat eenvoudiger voor te stellen is. Bovendien was dit eigenlijk een verplichting want met 1D-DMA kunnen er maximaal 65536 eenheden (afhankelijk van de instelling: 8, 16 of 32 bits) worden verwerkt per volledige DMA-transactie. Er

was echter nood aan 756×400 pixels (eenheden), dus 302400 eenheden.

De DMA werd zodanig ingesteld om na elke volledige transactie van één videoframe, een *interrupt* te genereren (figuur 3.6). Dit is nodig want de DMA werd in stop-modus gekalibreerd (hoofdstuk 3.2.2.7). Dit betekent dat na elke transactie de DMA terug wordt uitgeschakeld en dat er in de code zelf voor moest worden gezorgd dat deze tijdig terug wordt ingeschakeld. Dit wordt in de ISR gedaan, die bovendien ook instaat voor het bepalen van het volgende startadres van de geheugenbuffer (figuur 3.6). Dit dient echter te gebeuren vooraleer er een nieuwe transactie plaats heeft. Er werd namelijk een circulaire buffer gemaakt in het SDRAM-geheugen.

3.1.4.4 SDRAM-geheugen

De beelden konden nu tijdelijk in een circulaire buffer in het externe SDRAM-geheugen worden opgeslagen. Alvorens deze aan de buitenwereld te tonen, moesten deze eerst nog één bit naar rechts worden verschoven (hoofdstukken 2 & 3.1.4.1). Het geheugen aanspreken is verdacht eenvoudig. Dit bleek namelijk met gewone *C-pointers* mogelijk. Van zodra de camera naar behoren werkte, de beelden in het geheugen geschreven werden en naar 8-bits grijswaarden per pixel teruggebracht werden was het mogelijk om deze te bekijken met de *image viewer*. Dit is een handig hulpmiddel van Visual DSP++ 4.5, om het geheugen af te beelden (hoofdstuk 3.1.2).

3.1.5 Continu beelden ontvangen en tijdelijk opslaan

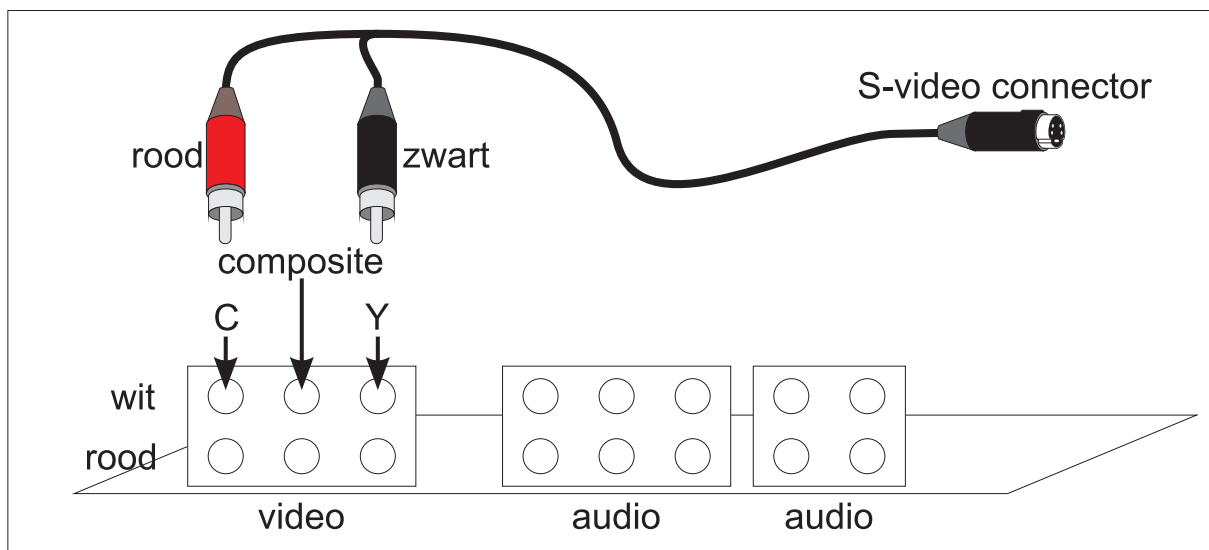
Tot hiertoe waren, voor het testen, de algoritmes zodanig gemaakt, dat er slechts één beeld ingelezen werd (hoofdstuk 4.1.2). Nadat dit beeld was ontvangen, werd de DMA niet opnieuw door de ISR geactiveerd, waardoor er geen nieuwe beelden naar het geheugen werden gestuurd, en werd de instructiecode gestopt. Alle algoritmes waren meteen zo ontworpen dat het continu opslaan en verwerken ervan mogelijk zou zijn. Om te testen was het eenvoudiger om dit eerst slechts voor één beeld te doen. Zodra dat goed functioneerde, werd er een hoofdflus in de code voorzien, zodat een continue verwerking van achtereenvolgende videoframes mogelijk werd. De DMA werd ook na elk volledig ingelezen beeld, terug geactiveerd voor het volgende. Dit was normaal een eenvoudige volgende stap en er werden geen problemen verwacht. In hoofdstuk 3.3 wordt gedetailleerd besproken wat er echter toch allemaal fout liep, waardoor het nonstop opslaan en verwerken van beelden helaas niet functioneerde. Er is veel tijd nodig geweest om dit op te lossen maar het is uiteindelijk wel gelukt (hoofdstuk 4.1.3). In figuur 3.8 wordt een afbeelding van de circulaire buffer weergegeven. De afbeelding werd gemaakt met de *image viewer* van Visual DSP++ 4.5 nadat de uitvoering van de instructiecode manueel werd gestopt.

3.1.6 Taakverdeling

Omdat het verdere deel van deze masterproef uit 2 totaal verschillende delen bestond werd besloten om hiervoor de taken te verdelen: Stefan zou het deel voor MULTICARCOM verder uitwerken, terwijl Dave het deel voor Melexis verder zou uitwerken. De uitwerking van deze beide delen is terug te vinden in de volgende paragrafen.



Figuur 3.8: Dit is de data die in de geheugenbuffer aanwezig was nadat de programma-uitvoering op een willekeurig moment werd gestopt. De afbeelding werd met de image viewer van Visual DSP++ 4.5 gemaakt. De circulaire buffer, die tijdelijk in het externe geheugen zit opgeslagen, is duidelijk zichtbaar als 4 opeenvolgende videoframes onder elkaar. De tweede afbeelding in de buffer van bovenaan te beginnen, werd laatst gemaakt. Dat is zichtbaar doordat de grijstinten nog niet naar 8-bits per pixel werden gecorrigeerd.



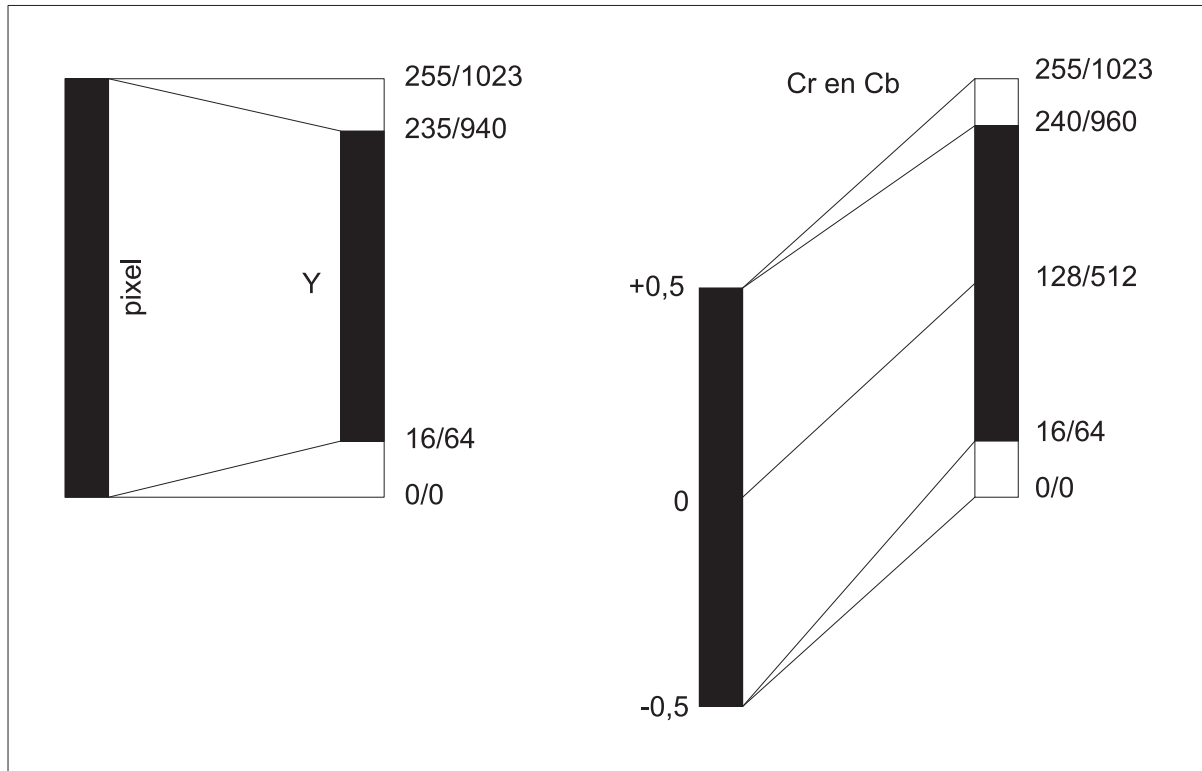
Figuur 3.9: *Connectoren op de BF EZ-kit LITE*

3.1.7 Melexis: Composiet–video

Voor Melexis is het belangrijk dat ze met deze demo–opstelling kunnen aantonen dat hun product (de camera) werkt en te gebruiken is voor de auto–industrie. Hiervoor moesten de beelden van de camera op een scherm weergegeven worden. Hiervoor werd hetzelfde scherm gebruikt als datgene dat gebruikt werd voor het MULTICARCOM–project.

Om de beelden naar buiten te brengen op het ADSP–evaluatiebord vroeg Melexis gebruik te maken van de S–video uitgang via de video–encoder. De video–encoder die op het evaluatiebord aanwezig is, is een ADV1719 [4]. Deze video–encoder kan beelden genereren in verschillende formaten [16]. Enkele van deze formaten zijn PAL, NTSC en RGB. De signalen komen dan op de RCA–connectoren op het evaluatiebord naar buiten (zie figuur 3.9). De video–encoder is momenteel ingesteld zodat er op de middenste RCA–connector een composietsignaal naar buiten komt, terwijl er op hetzelfde moment op de 2 buitenste RCA–connectoren de luminantie en de chrominantie naar buiten komen voor een S–video signaal te creëren. Hiervoor werd een kabel gemaakt om de omzetting van de connectoren te maken.

Om een videosignaal uit de video–encoder te krijgen moet de video–encoder de juiste data krijgen. De data moet voldoen aan de CCIR656–standaard. Hiervoor moeten de beelden worden omgezet naar het YCrCb 4:2:2–formaat. De pixelwaarden voor de luminantie moeten gelegen zijn tussen 16 en 235 [19]. Voor de chrominantie moeten de pixelwaarden liggen tussen 16 en 240, waarbij 128 zwart voorstelt (zie figuur 3.10). Hiervoor moest een herschaling gebeuren door de processor. Dit nam teveel tijd in beslag, om vloeiende beelden weer te geven, omdat elke pixel moet gelezen worden uit het (trage) SDRAM–geheugen. Om aan dit ongemak tegemoet te komen werd er een DMA–controller ingesteld om in 10 blokken de beelden eerst naar het snellere (maar kleinere) SRAM–geheugen te schrijven waarna de processor zijn bewerkingen kan uitvoeren op ‘snel’ geheugen. Nadien maken we alweer gebruik van de DMA–controller om de data terug te schrijven naar het



Figuur 3.10: Omzetten van pixels naar YCrCb [17, 18]

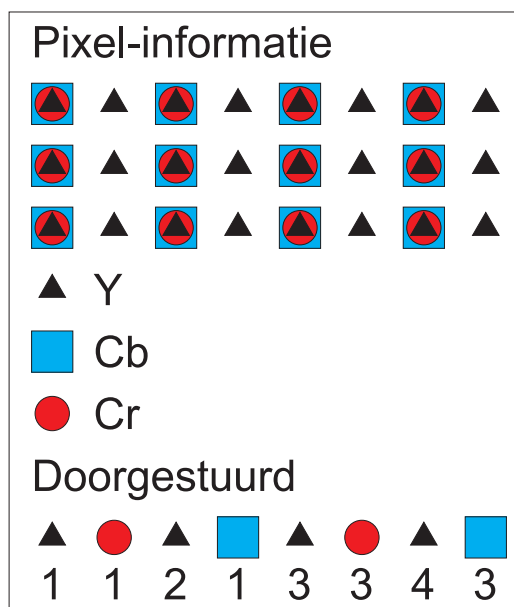
beeldgeheugen.

Een ander nadeel van de CCIR656-standaard is dat de beelden in 2 frames moeten gestuurd worden: 1 voor de even lijnen en 1 voor de oneven lijnen (zie figuur 3.12). Daarboven moeten de synchronisatie-signalen mee worden doorgegeven in de datastroom. Elke lijn begint met een horizontale synchronisatie. Deze synchronisatie begint met een EAV (End of Active Video) en eindigt met een SAV (Start of Active Video). Zie hiervoor ook figuur 3.13.

De SAV- en EAV- signalen geven tevens aan waar men zich bevindt (zie tabel 3.1 op pagina 48). Elk beeld begint met een verticale synchronisatie.

3.1.8 Camerabeelden rechtstreeks weergeven

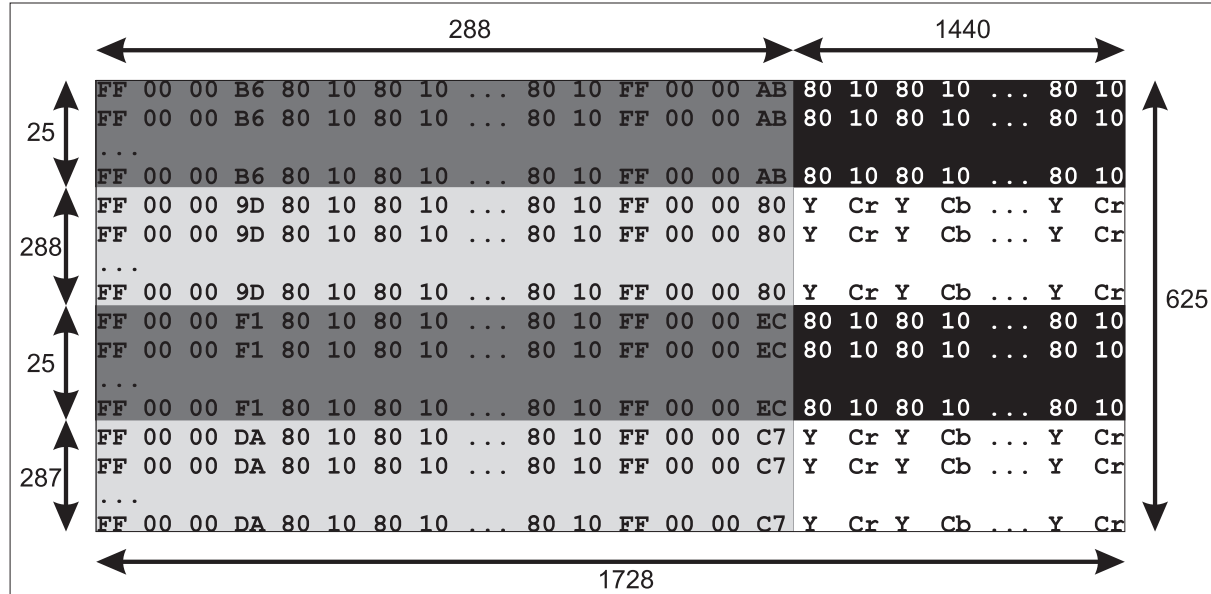
Nadat het gelukt was om enkelvoudige (al dan niet zelf gegenereerde) beelden weer te geven op het scherm, was het de bedoeling om bewegende beelden weer te geven. Dit verliep echter niet van een leien dakje aangezien dan alle verschillende blokken met elkaar moeten gesynchroniseerd zijn. Omdat er maar 2 databussen zijn en omdat de beide gebruikte DMA's (camera-SDRAM en SDRAM-beeld) op dezelfde bus staan en niet van bus kunnen verwisselen, was een grondige controle nodig van welke DMA op welk moment de bus gebruikt. Elke DMA-controller heeft een aantal periferie-elementen ter beschikking die hij kan gebruiken. Deze elementen kunnen echter niet van controller verwisseld worden wat het grote probleem is aangezien de beide parallelle interfaces (PPI0



Figuur 3.11: De YCrCb 4:2:2-standaard



Figuur 3.12: Framing en synchronisatie van de CCIR656-standaard



Figuur 3.13: *Pixelvoorbeeld van de CCIR656-standaard*

Waar	SAV/EAV	Waarde
Vsync1	EAV	FF 00 00 B6
	SAV	FF 00 00 AB
Video oneven	EAV	FF 00 00 9D
	SAV	FF 00 00 80
Vsync2	EAV	FF 00 00 F1
	SAV	FF 00 00 EC
Video even	EAV	FF 00 00 DA
	SAV	FF 00 00 C7

Tabel 3.1: EAV- en SAV-signalen voor de verschillende plaatsen

en PPI1) op dezelfde controller staan aangesloten. De prioriteiten van de DMA's speelden hier ook een grote rol om de timing juist te krijgen.

3.1.9 MULTICARCOM/De Nayer: SPI/MOST-link

De camera is regelbaar en de beelden worden door de DSP ingelezen en in het externe geheugen bewaard. Het kan echter niet de bedoeling zijn dat de gebruiker de programma-uitvoering steeds manueel moet stoppen om dan de laatste beelden in de buffer statisch weer te geven met de *image viewer* (hoofdstuk 3.1.2). De beelden moeten als quasi live video aan de buitenwereld getoond worden (hoofdstuk 4.1.3.2).

3.1.9.1 Communicerende systemen

Het systeem waarmee wordt gecommuniceerd is een Xilinx Virtex-II Pro platform. Dit platform stuurt de beelden die het van de ADSP-BF561 EZ-KIT Lite[®] ontvangt via een optische fiber met het MOST-protocol naar een tweede Virtex-II Pro platform. Op het tweede platform draait een Mediaplayer die de video afspeelt op een apart aangesloten beeldscherm. Voor de Mediaplayer moeten de beelden MPEG (de versie maakt niet uit) gecodeerd zijn (hoofdstuk 2.3). De tijd liet het echter niet toe om zelf een MPEG-encoder algoritme voor de BF561 te schrijven. Op de site van Analog Devices werd er MPEG-code (MPEG2 & MPEG4) gevonden, speciaal voor implementatie op de BF561. De code was helaas niet compleet. Het waren slechts een paar algoritmes ter illustratie, die slecht een klein onderdeel zijn van een complete MPEG-encoder. Er werden ook nog andere bronnen op het internet gezocht. Een MPEG-encoder specifiek voor de BF561 is wel degelijk te vinden, maar er moet voor betaald worden.

3.1.9.2 Beeldcompressie

De communicatie met de camera zou via een 25 MHz SPI gebeuren. Deze bus is niet bedoeld voor transfers van videodata met deze proporties, door de beperkte bandbreedte in combinatie met een seriële werking. Bovendien wordt de reeds beperkte bandbreedte nog eens gehalveerd. In de specificaties (hoofdstuk 2.3) staat beschreven dat per byte bruikbare data een synchronisatiebyte, met alle bits op één (*0xFF*), nodig is. Dit resulteert dus in een bruikbare bandbreedte van 12,5 MHz. Zonder een MPEG-encoder zou er dus een andere vorm van beeldcompressie nodig zijn. De enige mogelijkheden die op zo'n moment overblijven zijn een kleinere resolutie, een lagere *framerate*, minder grijsstinten of een combinatie ervan. In alle gevallen verliest de video aan kwaliteit, maar er werd beslist om de *framerate* naar 12 fps te verlagen. Het aantal grijswaarden werd op 256 niveau's (1 byte) en de resolutie op 320×240 pixels gehouden.

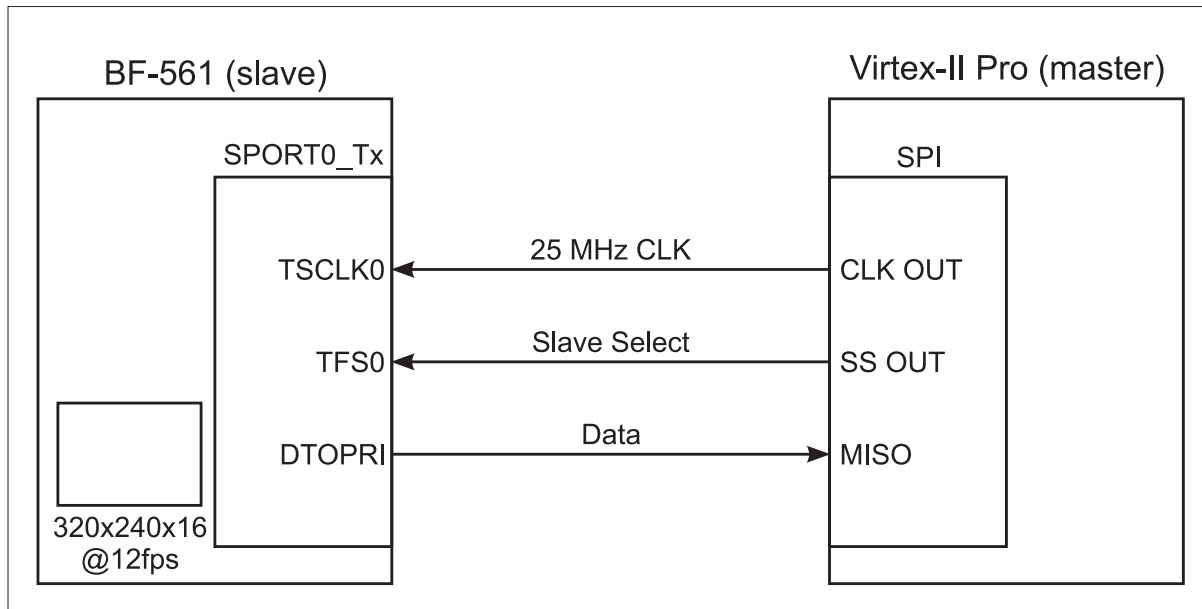
$$resolutie_{beeld} * grijsbits_{beeld} / klokfrequentie_{SPI} = t_{beeldtransfer}$$

$$((320 \text{ pixels} / \text{beeldlijn} * 240 \text{ beeldlijnen}) * 8 \text{ bits} / \text{pixel}) / 12,5 \text{ MHz} = 49,2 \text{ ms}$$

Het duurt dus 49,2 ms om elk beeld door te zenden. Door de *framerate* van 12 fps is er elke 83,3 ms een nieuw beeld. Op deze manier wordt er aan de specificaties (hoofdstuk 2.3)

voldaan en er blijven nog $83,3 \text{ ms} - 49,2 \text{ ms} = 34,1 \text{ ms}$ voor de Virtex-II Pro over om ieder videobeeld te verwerken. Een gebruiker zal helaas niet meer de impressie van vloeiende video hebben maar een demo-opstelling met deze instellingen toont wel aan dat het principe werkt.

3.1.9.3 SPI-slave

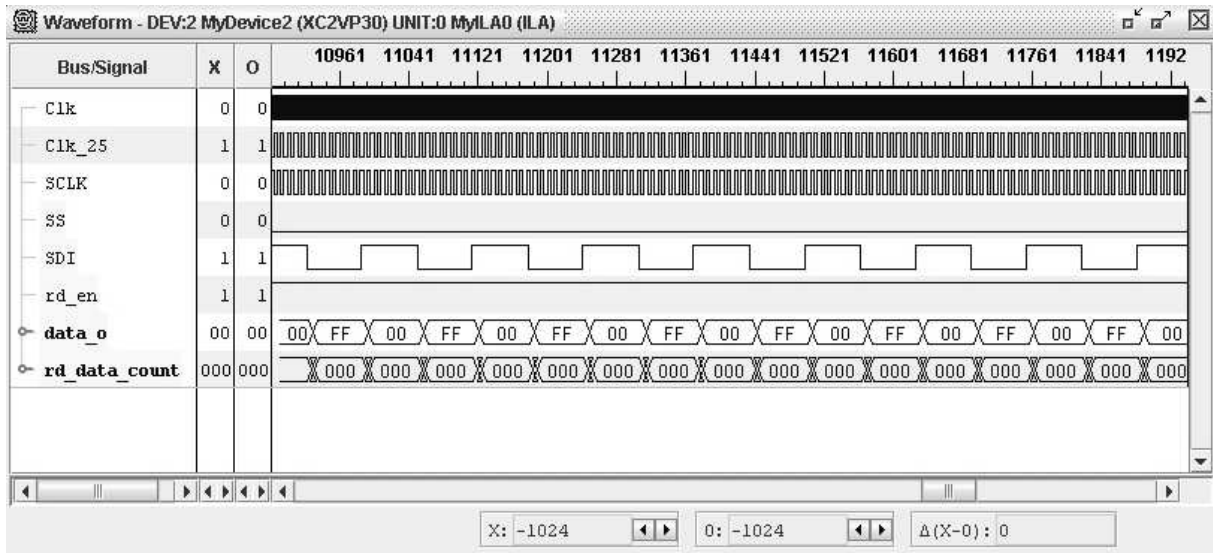


Figuur 3.14: De videotransfer van de BF561 naar de Virtex-II Pro gebeurt via een SPI. Deze functioneert op 25 MHz maar door de gebruikte synchronisatie is de nuttige bandbreedte gehalveerd. Wanneer de Virtex-II Pro de slave-select laag trekt, begint de BF561 videoframes door te sturen. Dit is een continue stroom van 153600 bytes per beeld (synchronisatie inbegrepen). Zolang de slave-select laag is wordt elk nieuw videoframe verstuurd aan een snelheid van 12 fps. Dit gaat zo door tot de slave-select terug hoog wordt.

Het was de bedoeling om de hardwarematige SPI (hoofdstuk 3.2.2.4) van de BF561 voor deze taak te gebruiken. Onder andere om deze reden werd deze niet ingezet voor communicatie met de camera (hoofdstuk 3.1.3.3). Er zijn pogingen ondernomen om de SPI in te stellen, maar wat er ook werd gedaan, deze component weigerde om de data correct over te brengen (zie hoofdstuk 3.3).

Er werd dan maar voor geopteerd om nogmaals de SPI met een SPORT na te bootsten [15]. Dit was tenslotte ook al voor de camera-aansturing zo gedaan (hoofdstuk 3.1.3.3) en er was nog één SPORT vrij. De SPORT0 was nog vrij maar omdat dit niet vooraf gepland was zouden er modificaties aan de interface-PCB nodig zijn. Alvorens deze aanpassingen aan te brengen moest er verzekerd worden dat de SPORT wel voor deze toepassing geschikt is. Om de configuratie van de SPORT te testen werd er een algoritme geschreven dat een transfer van een gekende datasequentie uitvoerde. De camera werd in dit testprogramma niet gebruikt. Hierdoor kon de SPORT1 gebruikt worden want de cameraparameters hoefden niet ingesteld te worden. Het voordeel hiervan is dat SPORT1

ook beschikbaar is op een aparte connector van de ADSP-BF561 EZ-KIT Lite[®] en dat er getest kon worden zonder ook maar één modificatie aan de interface-PCB te maken. De ADSP-BF561 EZ-KIT Lite[®] is de *slave* in deze SPI-communicatie. Om te testen moest het systeem dan ook van de nodige signalen (SPI-CLK, *slave-select*) worden voorzien. Dit gebeurde met een Virtex-II Pro platform dat een SPI-*master* emuleerde en voor weergave van de ontvangen data zorgde (figuur 3.15). Van zodra de SPORT1 correct functioneerde werden de interface-PCB en de algoritmes aangepast om SPORT0 te gebruiken (figuren 5.5 & 5.6). Deze benadering bleek een goed idee, maar er was toch vertraging door een onverwacht probleem (hoofdstuk 3.3) ten gevolge van deze configuratiemethode.



Figuur 3.15: Een weergave van de ontvangen datastroom zoals de analyzer deze weergeeft. De ontvangen sequentie is `0xFF00` dat continu herhaald wordt. Dit beeld werd gemaakt van zodra de LVDS-implementatie correct functioneerde, de ontvangen data is dan ook gelijk aan de verzonden testsequentie.

De SPORT is geconfigureerd voor externe laag actieve late synchronisatiepulslen [6]. In deze modus fungeert de BF561 als *slave* en heeft de synchronisatie dezelfde kenmerken als de SPI-SS (*slave-select*), wat een noodzaak is voor een correcte nabootsing van de SPI. Deze verbinding gebruikt de *master* namelijk om aan te geven wanneer de *slave* data mag verzenden. De data wordt gelezen/geschreven op de stijgende klokflank en bitgewijs verschoven op de dalende klokflank. Deze instellingen zijn zeer belangrijk want 2 communicerende systemen kunnen elkaar alleen begrijpen als het gebruikte protocol volledig gelijk is. De datawoordlengte werd op 16-bits (MSB eerst) ingesteld, hierdoor kunnen telkens één synchronisatiebyte en één databyte samen worden verzonden. Als de *master* (Virtex-II Pro platform) aangeeft dat er data nodig is dan wordt er bij ieder nieuw beeld een stroom (pakket) van $320 * 240 * 16 = 1228800$ bits verstuurd (synchronisatie inbegrepen). Er worden continue nieuwe camerabeelden gemaakt (12 fps), in de geheugenbuffer geplaatst en verwerkt maar alleen als het Virtex-II Pro platform data vraagt (SPI-SS laag) worden er beelden verzonden. De opbouw van het systeem wordt schematisch weergegeven in figuur 3.14.

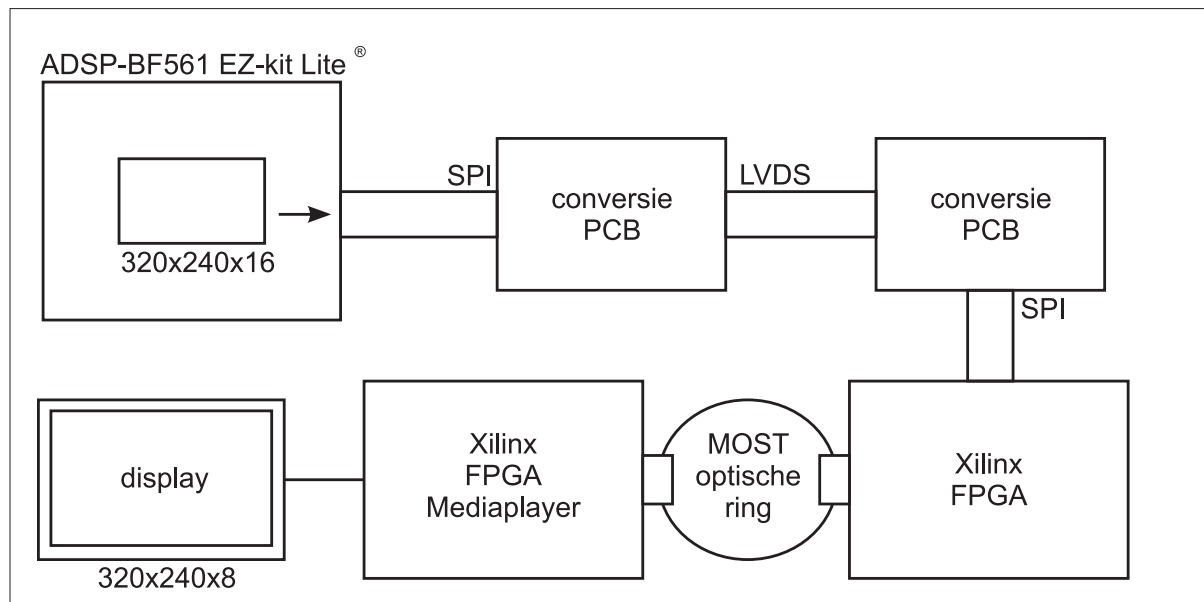
3.1.9.4 SDRAM, dataverwerking & DMA

De beelden werden toch reeds in een formaat van 2 bytes per pixel opgeslagen in het externe SDRAM-geheugen, waarvan slechts 9 bits nuttige data voorstellen (hoofdstuk 3.1.4.3). Bij de verwerking is er nu naast de reeds geïmplementeerde verschuiving (hoofdstuk 3.1.4.4) ook een bitgewijze "AND" geïntegreerd. Dit resulteert terug in 2 bytes per pixel maar nu met de eerste byte voor de synchronisatie ($0xFF$) en de laatste byte voor de grijswaarde van de pixel.

De data wordt vanuit het geheugen uitgelezen en naar de SPORT0 getransfereerd door DMA2. Aangezien deze *DMA-controller* in deze toepassing nog voor niets anders gebruikt is zijn er geen problemen met de prioriteit of complicaties met andere taken die de prestaties zouden kunnen ondermijnen mogelijk. Om zeker te zijn werd de SPORT0 op kanaal 0 verbonden zodat deze sowieso de hoogste prioriteit heeft op deze DMA-bus. De DMA is ingesteld in 2D modus met 16-bits transfers. Er worden geen *interrupts* gegenereerd.

3.1.9.5 LVDS

Zodat de communicatie minder storingsgevoelig zou zijn werden alle SPI-verbindingen tussen de BF561 en de Virtex-II Pro differentieel geïmplementeerd (figuur 3.16). Deze toevoeging werd gemaakt van zodra de SPI-communicatie correct functioneerde. De 2 communicerende systemen werden van een conversie-PCB voorzien. Het ontwerp van deze PCB hoort niet tot deze masterproef, maar de SPI-link werd ook met LVDS getest.



Figuur 3.16: De volledige route die de videodata aflegt, van het evaluatiebord tot het beeldscherm, wordt weergegeven. Om de transmissie minder stoorgevoelig te maken is de SPI differentieel uitgevoerd.

3.1.10 Synthese

Om alles correct te laten functioneren was eerst een volledige assemblage van de onderdelen nodig. De PCB moest van de nodige componenten voorzien worden. Dit gebeurde allemaal manueel. Als er één zaak werd geleerd, dan is het wel dat er bij het kiezen van componenten rekening moet gehouden worden met de beschikbaarheid. De 3 connectoren van de expansie-interface waren echter niet zelf te kiezen. Bij de ADSP-BF561 EZ-KIT Lite[®] zijn dat SFC-145-T2-F-D-A connectoren [20], van Samtec. De connectoren van Samtec (TFC-145-32-F-D [12]), die compatibel zijn met die van het evaluatiebord, werden met moeite verkregen door het aanvragen van *samples*. Er werd geen leverancier gevonden in België die deze connectoren kon leveren. En voor een rechtstreekse bestelling bij Samtec, moesten er minstens 125 stuks besteld worden. Nog moeilijker was het om de bijbehorende *flatcables* te verkrijgen. Omdat deze meer kosten geeft Samtec hoogstens 2 *samples* per aanvraag. Bovendien is de versie met 90 contactpinnen geen standaard model. Er werd dan maar voor de versie met 100 contactpinnen gekozen (TFMDL-50-T-10.00 [21]). De connectoren op deze *flatcables* werden dan handmatig bijgeslepen tot 90 pinnen.

Aan de PCB werden ook nog een aantal modificaties aangebracht. Deze werden met fijne draadjes op de PCB gemaakt, zodat er kon worden verder gewerkt zonder een nieuwe layout te moeten ontwerpen. Er werden wel nieuwe schema's getekend voor fouten die tijdens het project aan het licht kwamen. Er zijn ook schema's getekend voor toekomstige ontwerpen (figuren 5.7 tot 5.10) waarin het systeem wordt geoptimaliseerd (bijvoorbeeld: 8-bits databus tussen de camera en de DSP) en overbodige hardware wordt weggelaten.

De conversie-PCB voor omzetting naar LVDS (hoofdstuk 3.1.9.5) werd eveneens manueel van alle componenten voorzien en er werd zelf een twisted-pair verbinding voorzien. Deze PCB behoort niet tot deze masterproef.

De verbinding met een beeldscherm bij het systeem voor Melexis (hoofdstuk 3.1.7), zou eerst gebruik maken van S-video in plaats van Composiet-video. Daar werd een videokabel voor gemaakt, maar deze werd uiteindelijk niet gebruikt.

3.2 Details

In deze paragrafen bespreken we de details van de masterproef.

3.2.1 Camera: Melexis MLX75007

3.2.1.1 MLX75007

De camera die we gebruikt hebben werd geleverd door Melexis en is de MLX75007 [3]. Deze CMOS camera is een camera van tweede generatie en is ontworpen voor de auto-industrie. Hij beschikt over een maximale resolutie van 750 bij 400 pixels [2] en de pixels kunnen zowel analoog als digitaal uitgelezen worden. De digitale uitlezing gebeurt door middel van een on-chip 9-bits ADC. Dit geeft 512 mogelijke grijstinten. De pixels kunnen ook onchip gecorrigeerd worden. Of dit al dan niet gebeurt is instelbaar door de

verschillende registers van de camera. De registers van de camera worden ingesteld via SPI.

De camera werkt op een spanning van 3.6 V en heeft een temperatuurbereik van -40°C tot 105°C . De klokfrequentie mag maximaal 33 MHz zijn.

3.2.1.2 Hoe gebruiken wij de camera?

De 9 bits die via PPI van de camera naar de DSP gaan, worden big-endian in het geheugen opgeslagen. Aangezien de DSP geen 9 bits datawoorden aankan, werd gewerkt met 10 bits datawoorden om geen dataverlies te hebben. De 10e bit werd opgevuld met een logische 1. Dit is ook goed voor de camera's van derde generatie aangezien deze camera's zullen werken met een ADC met 10 bits nauwkeurigheid. Voor Melexis werd met een *framerate* van 24 frames/s en een resolutie van 756 bij 400 pixels gewerkt. Het is mogelijk om te werken met 756 pixels, ook al is de maximale resolutie slechts 750 pixels, omdat er ook nog 5 dummybits worden meegestuurd. Het aantal pixels is best een veelvoud van 2, vandaar dat er 756 pixels in plaats van 755 pixels gebruikt werden. De lijnen worden horizontaal gespiegeld doorgestuurd [3] zodat het lijkt alsof de kijker achter de camera staat.

3.2.1.3 Parameters

De camera beschikt over 44 parameters (die vervat zitten in 159 bits) die big-endian via de SPI kunnen ingesteld worden [3]. Omdat dit geen veelvoud van 8 is (8 bits per byte), moet er een bit worden toegevoegd. Deze bit moet vooraan toegevoegd worden [14] omdat de SPI van de camera met een daisy-chain werkt. De extra bit gaat er als eerste door en komt er als eerste terug uit.

Het instellen van de parameters gebeurt best tijdens de *frame-request-puls* omdat op dit moment alle parameters kunnen aangepast worden. Een aantal parameters kan ook op andere tijdstippen aangepast worden, maar om fouten hieromtrent te vermijden stellen we de parameters steeds in tijdens de *frame-request-puls*.

Voorlopig werden de registers nog ingesteld via SPI, maar in de volgende generatie camera's (derde generatie) zullen de registers ingesteld moeten worden via IIC. Hiervoor waren reeds functies ontworpen om de video-encoder in te stellen en deze functies zullen opnieuw gebruikt kunnen worden mits een kleine aanpassing. Dat de camera op dit moment via de SPORT (die op zijn beurt een SPI emuleerde) ingesteld wordt is zeer interessant aangezien deze sport eveneens IIC kan emuleren.

De 44 parameters worden via 5 32-bits registers ingesteld. Hier zullen de instellingen besproken worden die een goed resultaat gaven. We gebruiken dezelfde namen in onze code als in de voorbeeldcode van het 'simple-programma'. Het programma 'simple' wordt geleverd door Melexis en kan gebruikt worden om de camerabeelden in te lezen met een demobord van Melexis (via USB). In dit programma kunnen alle parameters manueel ingesteld worden, wat zeer handig is om te leren werken met de parameters en de camera.

- Een eerste belangrijke parameter is de *g_bEnabADC*, wat staat voor *Enable ADC*. Deze variabele bestaat uit 1 bit en staat op positie 4 in de bitstream. Deze variabele moet op '1' staan om digitale data te kunnen ontvangen van de camera.

- Een tweede belangrijke parameter is de $g_bFrameGran$, wat staat voor *Frame sequencer clock granularity*. Deze variabele bestaat uit 4 bits en staat op positie 32 tot 35. Hij bepaalt samen met $g_dwIntTime1$ de integratietijd. Een goede start is deze parameter op $0x0A$ instellen. Een verdubbeling van de korreligheid of een halvering van de integratietijd geeft hetzelfde resultaat. Op deze manier kan het bereik van de integratietijd vergroot worden.
- Een volgende belangrijke parameter is de $g_dwIntTime1$, wat staat voor *Integration Time 1*. Deze variabele bestaat uit 10 bits en staat op positie 36 tot 45. Deze parameter is een maat voor de eerste integratietijd. Om de werkelijke integratietijd te berekenen zijn ook nog de parameters $g_dwDelay$, $g_bFrameGran$, $g_bRowGran$, de masterklok (*MCLK*) en enkele constanten nodig. Deze parameter op $0x02BC$ instellen geeft een goed resultaat binnenshuis.
- De variabelen $g_dwSlope2$ en $g_dwSlope3$ zijn de tijden van de tweede en derde integratietijd. Indien er van deze 2 extra integratietijden geen gebruik gemaakt wordt, moeten deze beide ingesteld worden op 0. Beide variabelen zijn vervat in 10 bits. De tweede integratietijd staat op positie 56 tot 65 en de derde volgt hier meteen op: 66 tot 75.
- De variabelen $g_bResLevSecondSlope$ en $g_bResLevThirdSlope$ zijn de beginspanningen van de tweede en derde integratietijd. Beide variabelen zijn vervat in 5 bits. De resetwaarde van de tweede integratietijd staat op positie 76 tot 80 en die van de derde volgt hier meteen op: 81 tot 85.
- De laatste 2 parameters die hier verder besproken worden zijn $g_dwXWidth$ en $g_dwYHeight$. Deze stellen respectievelijk het aantal pixels en het aantal lijnen in. Deze moeten voor beide opstellingen (Melexis en Multicarcom) op $0x02F4$ en $0x0190$ staan. Dit wordt gedaan door respectievelijk de bits 103 tot 112 en 122 tot 130 in te stellen.

3.2.2 Evaluatiebord: ADSP-BF561 EZ-KIT Lite[®]

Dit hoofdstuk geeft een uitgebreide uitleg van het gebruikte demobord. Enkel de gebruikte voorzieningen worden besproken. De instelmogelijkheden worden toegelicht maar er wordt geen overzicht van de gebruikte registers gegeven. Voor informatie betreffende niet gebruikte onderdelen en meer details over gebruikte onderdelen dient de documentatie van Analog Devices geraadpleegd te worden. De documentatie omvat een algemene beschrijving van het evaluatiebord [4], een hardwarematige beschrijving van de DSP [6] en een gedetailleerde beschrijving van de parameterregisters en alle instelmogelijkheden van de DSP [7].

3.2.2.1 Evaluatiebord: algemeen

De ADSP-BF561 EZ-KIT Lite[®] is een evaluatiebord van Analog Devices dat speciaal voor video-toepassingen ontworpen werd. Alle componenten op het demobord zijn voor video geoptimaliseerd. Het bord bevat onder andere een video-decoder, een video-encoder,

een audio-codec, een extern SDRAM-geheugen van 64 MB en speciale interfaces geoptimaliseerd voor audio (SPORT) en video (PPI). Het evaluatiebord beschikt ook over een ruime selectie aan *I/O* en dat voor verschillende internationale standaarden van audio en video.

De basis van dit demobord is de Blackfin[®]561, een DSP met

2 processorkernen (dual core). De 2 processorkernen van deze DSP functioneren bij een frequentie die maximum 600 MHz (CCLK) bedraagt. De andere componenten van het systeem functioneren bij een frequentie van maximum 133 MHz (SCLK). Op het demobord is een PLL voorzien om de referentieklok te regelen. In hoofdstuk 3.3 worden problemen met het instellen van de DSP-tijdparameters besproken. De BF561 bevat een intern SRAM-geheugen van niveau 1 en niveau 2. Het geheugen van niveau 1 is voor elke processorkern apart en werkt op de CCLK-frequentie. Dit geheugen omvat zowel instructie- als datageheugen. Het geheugen van niveau 2 is gedeeld tussen de 2 processorkernen. Dit geheugen werkt op de helft van de CCLK-frequentie en heeft een grootte van 128 kB.

Het evaluatiebord kan via USB aan een PC verbonden worden. Er is een *debugger* van Analog Devices (Visual DSP++) om algoritmes te schrijven, te compileren en in de DSP te laden voor uitvoering. Er is ook een JTAG-interface aanwezig voor diagnostische testen. Naast 64 MB SDRAM-geheugen werd er ook 8 MB flashgeheugen geplaatst. Hierin blijft de data ook bewaard als het demobord uitgeschakeld is. Dit geheugen kan gebruikt worden om instructiecode in op te slaan, zodat er geen USB-connectie meer nodig is en het demobord onafhankelijk kan opstarten en functioneren. Visual DSP++ 4.5 bevat hiervoor een *Flash-programmer*, een hulpmiddel om de code in het flashgeheugen te schrijven. Hiervan werd op het moment van schrijven nog geen gebruik gemaakt.

Tenslotte is er op de ADSP-BF561 EZ-KIT Lite[®] een expansie-interface voorzien. Deze hardwarematige interface bestaat uit 3 connectoren met elk 90 aansluitingen. Op deze connectoren zijn de belangrijkste interfaces van de BF561 aangewezen: SPI, SPORT's, PPI's, *programmable flags*, UART, *reset*, video-controle, externe businterface, 27 MHz klok, grond, 3,3 V en 5 V. De expansie-interface kan gebruikt worden om het evaluatiebord uit te breiden met extra hardware (hoofdstuk 3.1.1.4).

3.2.2.2 BF561: programmable flags (PF)

De BF561 beschikt over 48 *programmable flags*. Dit zijn *I/O*-pinnen voor multifunctioneel gebruik, die stuk voor stuk op de expansie-interface van de ADSP-BF561 EZ-KIT Lite[®] aanwezig zijn, al dan niet *gemultiplexeerd* met een andere poort. Op het demobord zijn ook 12 LED's en 4 druktoetsen voorzien die via de *programmable flags* met de BF561 verbonden zijn. Om in zoveel mogelijk toepassingen te kunnen voorzien is elke *programmable flag* apart instelbaar. Een *programmable flag* kan als ingang of als uitgang gebruikt worden. Indien een *programmable flags* als ingang ingesteld is, kan deze niveau- of flankgevoelig zijn (alleen dalende flanken, alleen stijgende flanken of zowel stijgende als dalende flanken).

3.2.2.3 BF561: timers

De BF561 bevat 12 *timers* voor multifunctioneel gebruik. Deze *timers* zijn allemaal identiek maar alleen de eerste 7 *timers* hebben een poort. Deze poort kan als ingang en uitgang voor PWM-signalen gebruikt worden. Als de poort als uitgang is ingesteld, dan kan de *timer* voor het genereren van een klok of een ander PWM-signaal gebruikt worden. Indien deze poort als ingang is ingesteld, dan kan de *timer* gebruikt worden om de periode of pulsbreedte van het PWM-signaal te meten. De referentieklok van de *timers* is de systeemklok (SCLK). Het is ook mogelijk om een PWM-signaal als referentieklok voor een *timer* te gebruiken. In deze situatie is de *timerpoort* een ingang. De polariteit en de fase ten opzichte van de referentieklok is aanpasbaar. Als de *timers* dezelfde referentieklok gebruiken, dan wordt bijgevolg ook het faseverschil tussen de verschillende *timers* instelbaar. *Timers* 8 tot 11 kunnen ook gebruikt worden om de interne synchronisatie van de PPI's te genereren (hoofdstuk 3.2.2.6). Men kan ook nog kiezen of een *timer* bij emulatie werkt of niet. Waarom deze optie voorzien werd is onduidelijk (hoofdstuk 3.3).

3.2.2.4 BF561: serial peripheral interface (SPI)

De BF561 voorziet één ingebouwde SPI (*Serial Peripheral Interface*). Dit is een seriële interface voor communicatie tussen verschillende componenten van een systeem. De SPI is aanwezig op de expansie-interface en een SPI-connector. De maximale frequentie bedraagt 25% van de systeemklok (SCLK). De SPI beschikt over de standaard verbindingen, een SPI-klok (SCK), een *slave-select* (SS) een MOSI (*Master Out - Slave In*) en een MISO (*Master In - Slave Out*). Wanneer de SPI als *master* is ingesteld, voorziet deze maximaal 7 SS-uitgangen voor het aanspreken van *slaves*. Er is ook een SS-ingang voor wanneer de SPI als *slave* wordt gebruikt. Deze in- en uitgangen zijn stuk voor stuk *gemultiplxeerd* met *programmable flags*, die dus onbruikbaar worden als een SS voor de SPI gebruikt wordt. Om ervoor te zorgen dat 2 communicerende systemen hetzelfde protocol gebruiken zijn er een aantal instellingen voorzien, onder andere de klokpolariteit, de klokfase, de bitvolgorde en de datawoordlengte (maximum 2 bytes). De SPI kan zowel voor het verzenden als ontvangen van data gebruikt worden. De data wordt in een verzendingsregister geschreven of uit een ontvangregister gelezen. Dit kan ook gebeuren met DMA, maar dan is het niet mogelijk simultaan data te verzenden en ontvangen.

3.2.2.5 BF561: serial port (SPORT)

Er zijn op de BF561 2 SPORT's (*Serial PORT*) aanwezig. Beide SPORT's zijn aanwezig op de expansie-interface en SPORT1 ook op een aparte SPORT-connector. Een SPORT is een seriële interface die speciaal ontworpen werd voor de transfer van audiosignalen. Daarom is SPORT0 verbonden aan de audio-codec van de ADSP-BF561 EZ-KIT Lite[®]. Ook aan het ontwerp van deze poort is te zien dat deze voor audiodata gemaakt is. Elke poort heeft één klok-, één framesynchronisatie-, en 2 dataconnecties. De 2 dataconnecties werken synchroon maar de doorgestuurde data kan verschillen. Dit is een voorziening voor stereogeluid. Het tweede datakanaal kan uitgeschakeld worden.

In tegenstelling tot de SPI waar de hardware voor zenden en ontvangen één geheel vormt, zijn deze bij elke SPORT afzonderlijk uitgevoerd. Dit betekent dat er aparte

registers zijn voorzien om data te verzenden en te ontvangen. In deze masterproef is er enkel gebruikt gemaakt van de zendmogelijkheden van de SPORT; daarom wordt er systematisch over SPORT0 en SPORT1 gesproken. In al deze gevallen gaat het over SPORT0TX en SPORT1TX. Voor de afzonderlijke signalen werd de benaming van de SPORT-zender gebruikt.

De SPORT functioneert bij een frequentie die maximaal 50% van de systeemklok (SCLK) bedraagt. De SPORT kan als *slave* of als *master* ingesteld worden. In de documentatie wordt hier respectievelijk naar verwezen als externe en interne synchronisatie en klok. Om zeker te zijn dat beide systemen hetzelfde protocol gebruiken kunnen onder andere de bitvolgorde, de datawoordlengte (maximum 4 bytes), de klokpolariteit, en het gebruikte synchronisatieformaat worden ingesteld. Het is zelfs mogelijk om data in een gecodeerde vorm door te sturen. Met de SPORT kunnen makkelijk andere interfaces nagebootst worden zoals SPI (hoofdstuk 3.3) [15] en IIC. Elke SPORT werkt met een databuffer, één voor verzending en één voor ontvangst. Het is eveneens mogelijk om een SPORT met DMA te gebruiken. Zenden en ontvangen van data is wel simultaan mogelijk in DMA-modus omdat de zender en de ontvanger gescheiden zijn uitgevoerd; met aparte parameterregisters.

3.2.2.6 BF561: parallel peripheral interface (PPI)

Net zoals de SPORT er is voor audio, is er ook een interface voor video. Dit zijn de 2 PPI's (*Parallel Peripheral Interface*) van de BF561. De PPI's zijn aanwezig op de expansie-interface van het evaluatiebord. PPI0 is verbonden met de video-decoder van het demobord, voor het ontvangen van video. PPI1 is op zijn beurt verbonden met de video-encoder, voor het versturen van video. Een PPI is een parallelle interface, met een aanpasbare busbreedte waardoor alle waarden tussen 8-bits en 16-bits (geen 9-bits) mogelijk zijn. Door meer dan 8-bits busbreedte te kiezen, zijn er een aantal *programmable flags* niet langer beschikbaar omdat deze stuk voor stuk met één van de datalijnen van de PPI *gemultiplexeerd* zijn. Een PPI beschikt daarenboven over 3 signalen voor videosynchronisatie: de lijn-, de frame- en de *field*-synchronisatie. Er is ook een kloksignaal voorzien. Omdat de PPI zelf geen klok kan maken, moet deze steeds van een externe bron komen. Dit kan het 27 MHz kristal, de klok van de video-decoder of een klok van een bron op de expansie-interface zijn.

Er zijn verschillende parameters om te zorgen dat hetzelfde protocol gebruikt wordt door 2 communicerende systemen. De klokpolariteit, de polariteit van de synchronisatiesignalen en het aantal gebruikte synchronisatiesignalen (0 tot 3) is instelbaar. Als er 2 synchronisatiesignalen worden gebruikt, moet de optie voor 3 gekozen worden en moet de *field*-synchronisatie op de grond worden aangesloten (hoofdstuk 3.3). Een PPI kan als *master* en als *slave* functioneren. In de documentatie wordt hiernaar verwezen als respectievelijk interne en externe synchronisatie. In het geval van interne synchronisatie, wordt deze door de multifunctionele *timers* van de BF561 gemaakt (hoofdstuk 3.2.2.3). Er zijn ook parameters die bepalen hoe de data moet worden verpakt of uitgepakt, respectievelijk voor of na transfer door de DMA. Er kan ook worden gekozen om de PPI de volledige 32-bits van de DMA-bus te laten gebruiken, ook al is de data maximaal 16-bits per PPI-woord. Deze instellingen laten toe om de DMA efficiënter te gebruiken.

Een PPI werkt alleen met datatransfers van of naar het geheugen door middel van DMA. Er is geen enkel dataregister aanwezig. Een PPI kan ook slechts enkel als zender of ontvanger fungeren, niet de beiden tegelijk. De PPI's bieden ondersteuning voor ITU-R 656, de norm om analoge TV-beelden digitaal voor te stellen. De PPI kan de horizontale en verticale *blanking* herkennen. Dit is slechts uitsluitend voor het ontvangen van TV-signalen terwijl in deze masterproef er enkel TV-signalen verzonden werden. Daarvoor biedt de PPI jammer genoeg geen automatische ondersteuning (hoofdstuk 3.1.7). In dat geval moet de horizontale en verticale synchronisatie zelf gemaakt worden.

3.2.2.7 BF561: direct memory access (DMA)

De BF561 beschikt over 3 *DMA-controllers* (*Direct Memory Access*). DMA is een technologie die de processorkernen moet ontlasten van kopieertaken en transfers van en naar geheugen. Elke *DMA-controller* controleert één DMA-bus. Voor datatransfers tussen 2 interne geheugens van de BF561 is er de IMDMA (*Internal Memory DMA*). DMA1 en DMA2 zijn dan weer bedoeld voor datatransfers met periferie-onderdelen of extern geheugen. DMA1 kan enkel gebruikt worden voor datatransfers tussen de PPI's en het geheugen en voorziet een 32-bits bus. DMA2 heeft slechts een 16-bits bus en bevat onder andere de SPORT's en de SPI. De componenten kunnen niet gewisseld worden tussen de verschillende DMA-bussen.

Elke *DMA-controller* heeft 16 kanalen en elke component hangt standaard op één bepaald kanaal van een *DMA-controller*. Deze toewijzing kan wel gewijzigd worden en is bepalend voor de prioriteit op de DMA-bus. Hoe lager de nummer van het toegewezen kanaal, hoe hoger de prioriteit. Elk kanaal van elke *DMA-controller* is apart instelbaar. De instellingen mogen nooit in conflict zijn met de verbonden component, zo moet bijvoorbeeld de datawoordlengte altijd gelijk zijn. Speciaal voor beelden is er tweedimensionale DMA, dit werkt handiger met beelden en laat grotere datatransfers toe.

Er zijn verschillende werkingsmodi: *stop*, *autobuffer*, *descriptor array* en *descriptor list* (klein en groot). Bij de eerste 2 modi, de zogenaamde registermodi, wordt de DMA gestart door het instellen van de parameterregisters. De instellingen blijven hetzelfde zolang er niet vanuit de instructiecode wordt ingegrepen. In stopmodus moet de DMA na elke volbrachte transfer terug vanuit de instructiecode geactiveerd worden en in autobuffermodus gebeurt dit automatisch na elke volbrachte transfer. Deze 2 modi zijn het best qua geleverde prestatie. Bij de *descriptor*-modi initialiseert elke volbrachte transfer automatisch een volgende transfer die echter volledig andere instellingen kan hebben. De instellingen van de alle transfers en een verwijzing van elke transfer naar de instellingen van de volgende transfer worden in een *descriptor* in het geheugen bewaard. Er is dus geen interventie vanuit het programma nodig. Deze 3 modi bieden veel meer flexibiliteit, er is echter wel een verlies aan prestatie want bij elke nieuwe transfer moeten de volgende instellingen uit het geheugen worden gehaald.

3.2.2.8 BF561: interrupts

Alle besproken onderdelen kunnen *interrupts* genereren. Normaal zijn *interrupts* gemaskeerd. Om een *interrupt* te kunnen gebruiken moet het masker in de processorkern

verwijderd worden zodat de *interruptcontroller* de *interrupt* kan verwerken. De *interrupt* moet bij de component, die deze genereert, ook geactiveerd worden voor gebruik. Soms moet er bij de component bovendien nog een masker verwijderd worden (bijvoorbeeld: de *timers*). Elke component die een *interrupt* kan genereren zit op één van de 64 *interruptkanalen*. Er zit exact één component op elk kanaal volgens een vaste toewijzing. Elk *interruptkanaal* is standaard toegewezen aan één van de *interruptvectoren* voor algemeen gebruik (IVG7 tot IVG15). Er zitten 8 *interruptkanalen* op elke *interruptvector*. Aan welke interruptvector een interruptkanaal is toegewezen is wel aanpasbaar en bepaalt bovendien de prioriteit.

3.2.3 Display: NW619VT

Dit beeldscherm werd in de 2 gerealiseerde systemen gebruikt om de camerabeelden weer te geven. Het was meteen ter beschikking en het zal ook in de wagen toegepast worden bij het MULTICARCOM-project. Daarom wordt er hier een kleine toelichting bij het beeldscherm gegeven, al is het geen onderdeel van deze masterproef.

De NW619VT [5] van Newision is een 7 inch beeldscherm dat speciaal voor toepassingen in de wagen ontworpen is. Het scherm is van een touchscreen voorzien. Het beeldscherm beschikt over een VGA-ingang en 2 Composiet-ingangen. Hierdoor kon het beeldscherm voor de 2 systemen gebruikt worden, waar anders een computermonitor met een VGA-ingang en een TV met een Composiet-ingang nodig waren. Er is bovendien nog een audio-ingang voorzien, voor de ingebouwde luidsprekers. Hiervan werd geen gebruik gemaakt. Het beeldscherm heeft een beeldresolutie van 1024×768 pixels. Er is ondersteuning voor PAL, NTSC en SECAM met automatische schermaanpassing, zodat het volledige beeld gebruikt wordt. Er kan tussen de verschillende ingangen gewisseld worden via een schermmenu en er is een afstandsbediening bijgeleverd.

3.2.4 Algemene details

- Werkingsfrequentie camera: 30 MHz
- Werkingsfrequentie DSP-cores: 600 MHz
- Werkingsfrequentie DSP-systemen: 120 MHz
- Cameraresolutie: $756 \times 400 \times 9$ bits
- SPI-connectie DSP \rightarrow camera: 160-bits commando's @ 30 MHz
- PPI-connectie camera \rightarrow DSP: $756 \times 400 \times 9$ bits @ 30 MHz
- Busbreedte Camera \rightarrow DSP: 9 bits
- Synchronisatie camera \rightarrow DSP: lijn & frame
- *Framerate* camera: 24 fps (*frame-request*)
- PPI-connectie DSP \rightarrow video-encoder: $1728 \times 625 \times 8$ bits @ 27 MHz

- Busbreedte DSP → video-encoder: 8 bits
- Synchronisatie DSP → video-encoder: lijn, frame & field
- *Framerate* video-encoder: 50 fps
- Codering & interface video-encoder → beeldscherm: PAL & Composiet
- SPI-connectie DSP → *MOST-transceiver*: $320 \times 240 \times 16$ bits @ 25 MHz
- *Framerate* DSP → *MOST-transceiver*: 12 fps

3.3 Problemen

Dit hoofdstuk geeft een toelichting bij alle problemen die tijdens deze masterproef zijn tegengekomen en vertelt iets meer over de manier waarop ze werden opgelost.

Stage: Matlab Tijdens de stageperiode was het aanvankelijk de bedoeling om via Matlab de camera aan te sturen met een DLL die werd geleverd door Melexis. Het voordeel van Matlab is dat het zeer eenvoudig is om algoritmes te maken en te testen. Matlab kan werken met DLL's, maar de DLL die werd geleverd maakte gebruik van *pointers*. Hier zat het probleem: Matlab kan werken met *pointers*, maar dit is eigenlijk slechts een emulatie van *pointers* [22]. Het geeft geen adres door, maar maakt een kopie. Dit had tot gevolg dat Matlab niet gebruikt kon worden met de bestaande DLL. Hiervoor waren 3 mogelijke oplossingen: de DLL aanpassen, een DLL aanmaken die als software-interface functioneerde of overschakelen op een ander pakket. De DLL aanpassen ging niet aangezien er niet over de volledige code kon beschikt worden en aldus eigenlijk de volledige DLL zou moeten worden herschreven. Een interface-DLL ontwerpen zou teveel tijd in beslag nemen (er waren maar 2 weken stage voorzien) omdat de volledige werking van DLL's onderzocht zou moeten worden. Er werd gekozen om over te schakelen naar een andere programmeeromgeving. Het '*Simple*'-programma van Melexis is geschreven in C++ en daarom werd besloten om de algoritmes eveneens in C++ te maken. Hiervoor kon gebruik gemaakt worden van de broncode van '*Simple*'. Dat de algoritmes in C++ geschreven werden had ook ineens als voordeel dat ze gemakkelijker implementeerbaar waren in de DSP.

Stage: Interface-PCB Zodra de interface-PCB geleverd was kwam er een grote fout aan het licht. De 2 connectoren om de camera aan te sluiten werden fout gepositioneerd ten opzichte van elkaar. Hier was ook niet veel aandacht aan besteed want de onderlinge positionering van deze connectoren was al door Melexis gedaan om zeker te zijn dat dit foutloos zou gebeuren. De connectie van de afzonderlijke pinnen, van elke connector werd wel correct uitgevoerd. Door een camera met *flatcables* te gebruiken werd het probleem eenvoudig opgelost. De *flatcables* maken het geheel wel stoorgevoeliger, wat de beeldkwaliteit vermindert. Deze kwaliteitsafname is echter gering. Er werd ook ontdekt dat er geen printspoor was voorzien voor de *frame-request*, ten gevolge van een communicatiefout bij het maken van onderlinge

afspraken. De *frame-request* is echter een zeer belangrijk signaal. Omdat het hier over een laagfrequent signaal gaat was het probleem eenvoudig te verhelpen door met een draadje een verbinding te solderen. Er werd ook een nieuwe layout ontworpen (figuren 5.3 & 5.4 op pagina's 83 en 83). Met deze layout werd ook een PCB geproduceerd, maar deze is nooit gebruikt.

DSP-timings Om alle functies correct te laten werken is een juiste tijdreferentie noodzakelijk [7]. De BF561 bestaat in een versie met maximum 500 MHz en een versie met maximum 600 MHz als klokfrequentie voor de processorkernen (CCLK). De gebruikte versie heeft een 600 MHz CCLK. In beide versies mag de systeemklok (SCLK) niet meer dan 133 MHz zijn. De camera kan maximum werken op een 33 MHz klok dus het is wenselijk deze frequentie te gebruiken, maar dan moet de SCLK van de BF561 een geheel veelvoud van de cameraklok zijn (hoofdstuk 3.1.3.2). Het zou dus ideaal zijn dat de processorkernen op 600 MHz en de andere componenten van de BF561 op 133 MHz geklokt zijn. Een PLL zorgt voor een frequentieverhoging uitgaande van een 30 MHz kristal (X-tal). In de BF561 zelf is het mogelijk deze klok terug te delen tot de CCLK (deling door 1, 2, 4 of 8) en de SCLK (deling door 1, 2, ..., 15) [6].

Bij de eerste poging werd de optimale prestatie nagestreefd, dus zowel de CCLK als de SCLK zijn maximaal (tabel 3.2).

Component	Afleiding	Waarde
X-tal		30 MHz
PLL	X-tal \times 40	1,2 GHz
CCLK	PLL / 2	600 MHz
SCLK	PLL / 9	133,33 MHz

Tabel 3.2: *De instellingen zijn gekozen om een maximale prestatie te halen. De maximum toegelaten uitgangsfrequentie van de PLL wordt echter overschreden waardoor de overklokbeveiliging ingrijpt.*

Bij deze instellingen grijpt de automatische beveiliging, tegen te hoge frequenties, in. De frequentie die de PLL genereert is begrensd op 600 MHz [7]. De beveiliging stelt de frequenties automatisch in op 600 MHz op de PLL-uitgang, 600 MHz voor de CCLK en 120 MHz voor de SCLK.

Bij een tweede poging werd er beslist om een deel van de CCLK-frequentie op te offeren, aangezien die voor deze toepassing niet zo veel uitmaakte, om zo de gewenste SCLK te verkrijgen (tabel 3.3).

Ook bij deze instellingen komt het automatische beveiligingssysteem in actie. De reden is niet volledig duidelijk. Waarschijnlijk is de 2 MHz meer dan 133 MHz bij de SCLK het probleem [7]. Het is ook mogelijk dat het systeem zodanig is gekalibreerd dat het altijd bij de maximale CCLK van 600 MHz functioneert. Hierover werd echter niets teruggevonden in de documentatie. De beveiliging stelt de frequenties bijgevolg automatisch in op 600 MHz na de PLL, 600 MHz CCLK en 120 MHz SCLK.

Component	Afleiding	Waarde
X-tal		30 MHz
PLL	X-tal \times 18	540 MHz
CCLK	PLL / 1	540 MHz
SCLK	PLL / 4	135 MHz

Tabel 3.3: Een deel van de snelheid van de processorkernen wordt opgeofferd om de juiste systeemklok te bekomen. De maximale systeemklok wordt echter overschreden waardoor de overklokbeveiliging ingrijpt.

Omwille van de te halen mijlpalen werd er geen extra tijd aan de tijdsreferentie besteed en zijn de instellingen in tabel 3.4 gebruikt (figuur 3.3).

Component	Afleiding	Waarde
X-tal		30 MHz
PLL	X-tal \times 20	600 MHz
CCLK	PLL / 1	600 MHz
SCLK	PLL / 5	120 MHz

Tabel 3.4: De processorkernen werken op de maximale frequentie van 600 MHz. De systeemklok bedraagt 120 MHz. Er wordt geen enkele limiet overschreden.

Timeremulatie De ADSP-BF561 EZ-KIT Lite[®] werkt bij de uitvoering van de instructiecode via een USB-emulatie. Het is cruciaal dat bij elke gebruikte *timer* de *EMU_RUN*-bit van het configuratieregister zo wordt ingesteld dat de *timer* tijdens emulatie blijft functioneren [6]. Standaard staat deze functie namelijk af. Op deze manier kan er veel tijd verloren gaan omdat het een detail is waar snel over wordt gekeken.

SPI emulatie met SPORT1 Het was vanaf het begin de bedoeling om de SPI met de SPORT na te bootsen. De gebruikte implementatie verschilt echter wel grondig van het oorspronkelijke plan. De manier waarop de klok en de data worden doorgegeven is onveranderd gebleven. Er is dan ook duidelijke een directe overeenkomst tussen respectievelijk *SCK* en *MOSI* van de SPI enerzijds en *TSCLK1* en *DT1PRI* van de SPORT1 anderzijds. Het is echter niet zo duidelijk welk alternatief de SPORT biedt voor de *slave-select* van de SPI. De problemen zijn dan ook rond dit signaal gesitueerd.

Volgens het initiële plan zou de tweede dataverbinding (*DT1SEC*), die synchroon met de eerste dataverbinding (*DT1PRI*) werkt, hiervoor gebruikt worden. Op het ogenblik dat er dan data wordt gestuurd over *DT1PRI*, zou er over *DT1SEC* ook een datastroom gestuurd worden waarin alle bits één zijn. De data om over de 2 datakanalen te sturen moet echter in hetzelfde register geschreven worden [6]. Het register was daardoor niet groot genoeg om het datapakket voor beide kanalen volledig te bevatten. Hierdoor gebeurde het regelmatig dat de data in het register

volledig verzonden was, maar dat de volgende data van het pakket nog niet klaar stond (*underflow*).

Er werd gekozen om het synchronisatiesignaal op een andere manier te produceren [15], namelijk met de framesynchronisatie (*TFS1*) van de SPORT1. Deze synchronisatie was bij de oorspronkelijke implementatie uitgeschakeld. De SPORT werd nu geconfigureerd in interne, hoog actieve, late modus (figuur 3.4). In deze modus is de BF561 *master* en heeft de synchronisatie dezelfde kenmerken als de *slave-select* van de SPI. Om dit te realiseren werden ook een paar modificaties aan de interface-PCB aangebracht (figuren 5.5 & 5.6 op pagina's 85 en 86).

PPI0-synch3/Field Voor de framesynchronisatie tussen de camera en de PPI0 zijn 2 synchronisatiesignalen vereist, de frame/beeld- en lijnsynchronisatie. De PPI0 beschikt over een derde synchronisatiesignaal (*field*), dat niet werd aangesloten. De PPI0 kan alleen ingesteld worden voor het gebruik van 0, 1 of 3 synchronisatiesignalen. Voor 2 synchronisatiesignalen [6] moet ook de optie voor 3 signalen gebruikt worden maar dan moet er een connectie gemaakt worden tussen de grondreferentie en de *field*-synchronisatie (figuur 3.5 op pagina 40).

Continu beelden verwerken Het systeem functioneerde correct voor één videoframe. Door eenvoudigweg de *interrupts* te activeren en een oneindige lus in het hoofdprogramma te plaatsen zou de continu verwerking van video mogelijk moeten zijn. Dit verliep echter niet zo eenvoudig als verwacht. Bij het eerste beeld ging alles zoals gepland, maar vanaf het tweede beeld begonnen er zichtbaar zaken fout te gaan. Daarom was er bij de verwerking van slechts één videoframe ook geen enkele fout opgemerkt. De oorzaak van het probleem zou 2-ledig blijken.

Het eerste probleem was dat de beelden die niet meer op de juiste plek in de circulaire buffer werden geschreven. Er ging iets fout met het doorgeven van de startadressen van de buffers. Globale variabelen (*static*) in de algoritmes vormden de basis van dit probleem. Het gebruik van globale variabelen wordt algemeen afgeraden. Omdat er niet meteen een andere methode gekend was om gegevens met de ISR uit te wisselen, werd deze methode toch gebruikt. Hierdoor werden echter de startadressen foutief doorgegeven aan de ISR. Er werden in omgekeerde zin ook geen gegevens van de ISR naar het hoofdprogramma doorgegeven. Door het toepassen van *printf*-functies werd duidelijk dat de veranderingen die werden aangebracht tijdens de ISR-uitvoering, meteen ongedaan werden gemaakt van zodra de ISR werd verlaten om het hoofdprogramma verder uit te voeren. Het probleem werd opgelost door alle globale variabelen te vervangen door 'extern' gedeclareerde variabelen.

Het tweede probleem was dat de beelden niet meer correct verwerkt werden. Het verminderen van het aantal grijswaarden, pixel per pixel en dat beeld per beeld, werd niet altijd even consequent uitgevoerd. Meestal werd de verschuiving maar gedeeltelijk of zelfs helemaal niet uitgevoerd. Bovendien zat er een volledig videoframe vertraging tussen het laatst ingelezen en het laatst bewerkte videoframe. Door op het begin en het einde van de verwerking van één videoframe de waarde van een *programmable flag* te flippen werd met een oscilloscoop de uitvoertijd nagemeten. De verwerking werd op zeer onregelmatige basis uitgevoerd. Bovendien kwam de verwerking regelmatig in conflict met de DMA omdat deze gelijktijdig tot hetzelfde

geheugenadres toegang had. In combinatie met het fout doorgeven van startadressen werden er uitvoertijden tot 120 ms opgemeten. Dit is uiteraard veel te lang om een *framerate* van 24 fps te halen (hoofdstuk 3.1.3.2). Het lezen van en het schrijven naar het externe SDRAM-geheugen werd pixel per pixel (16 bits) uitgevoerd. Door dit op een 32-bits basis (2 pixels per keer) te implementeren werd het aantal geheugentoeegangen gehalveerd. De geheugentoeegangen moeten verder minimaal gehouden worden door elke bit maar één keer uit het geheugen te lezen, dan eerst alle bewerkingen te doen om vervolgens de bit terug in het geheugen te schrijven. In combinatie met een correct gebruik van gedeelde variabelen werd de verwerkingstijd van één videoframe teruggebracht tot 16,8 ms.

Standaardinstellingen van de ADV7179 De standaardinstellingen van de ADV7179 (video-encoder) werkten niet voor onze toepassing. Daarom werd een IIC-emulatie gemaakt om de 30 interne registers van de ADV7179 te programmeren. Dit gebeurde via 2 *programmable flags*: *PF0* en *PF1* die respectievelijk de IIC-klok en de IIC-data voorstellen [4]. De parameters werden bestudeerd [16] en aan de hand van een voorbeeld werd een keuze gemaakt.

Dataconversie naar YCrCb 4:2:2 De data is opgeslagen in 16-bits registers, maar moet doorgegeven worden aan de ADV7179 in het YCrCb 4:2:2-formaat. Hiervoor was een omzetting nodig van zowel de grijsinten als de kleurinformatie. Deze omzetting nam teveel tijd in beslag zodat het niet meer mogelijk was om vloeiende beelden weer te geven. Om dit op te lossen werd de DMA ingesteld om eerst de beelden in blokken van het (trage) externe SDRAM-geheugen naar het (snellere) interne SRAM-geheugen te kopiëren, vervolgens de omzetting te doen en tenslotte de beelden terug te kopiëren naar het SDRAM-geheugen [23]. Of dit al voldoende snelheidswinst oplevert is op het moment van schrijven nog niet bekend. Als dit niet het geval is moet eventueel bekeken worden of beide processorkernen kunnen ingeschakeld worden om de omzetting in *pipelining* te doen.

hardwarematige SPI De ingebouwde SPI zou gebruikt worden om videoframes naar het Virtex-II Pro platform te sturen. Hoe de SPI ook geconfigureerd werd, het was niet mogelijk om een continue stroom van data te transfereren zonder dat er, ongeveer elke 8 bytes, ongewenste bits tussenkwamen. Met 25 MHz functioneert de SPI niet eens op zijn maximale frequentie van 30 MHz. De data wordt voldoende snel aangereikt door de *DMA-controller*. Die gebruikt een 16-bits parallel bus, die op een frequentie van 120 MHz (SCLK) functioneert. De SPI was de enige geactiveerde component op deze *DMA-controller* en had voor de zekerheid de hoogste prioriteit gekregen. Er werd geen underflow van de zendbuffer weergegeven in het statusregister. De instellingen van SPI en DMA stonden correct in de registers geschreven en ook de zendbuffer bevatte de juiste data. De exacte oorzaak van dit probleem is onbekend. Uit bronnen op het internet blijkt dat er bij de SPI van de BF561 regelmatig een hardware-anomalie voorkomt die tot dit soort problemen kan leiden. Of dit de oorzaak is of niet, het probleem werd omzeild door met de SPORT1 een SPI te emuleren [15] (figuur 3.14 op pagina 50). Ook op de interface-PCB werden de nodige modificaties aangebracht.

SPORT1 test De SPORT1 werd gebruikt om snel de instellingen voor SPORT0 te bepa-

len, alvorens de interface-PCB aan te passen. Er werd echter een vreemd fenomeen waargenomen. De frequentie waarop de interface functioneerde was steeds het dubbele van de daadwerkelijk ingestelde frequentie. De oorzaak voor dit fenomeen was de camera, die niet gebruikt werd voor deze test maar nog steeds aan de SPORT1 was aangekoppeld. Hierdoor waren er aan de SPORT nog een hele vertakking van *flatcables*, printsporen en het cameraregister verbonden, die tot het besproken probleem aanleiding gaven. Het afkoppelen van de camera en de interface-PCB volstond om dit probleem te verhelpen. Vervolgens werd de interface-PCB aangepast om SPORT0 te gebruiken voor het versturen van videoframes en kon de camera terug aangekoppeld worden.

Draaien van het beeld Voor het demomodel van MULTICARCOM zou de camera op zijn kant ingebouwd worden in de wagen. Om dit op te vangen eiste Recticel dat het beeld 90° gedraaid werd. Om dit te doen waren er 2 mogelijkheden: een algoritme of de DMA. Er werd voorlopig een functie geschreven die elke beeld pixel per pixel verwerkt en draait. In de toekomst zal dit via DMA gebeuren om de prestaties te verbeteren. Dit was niet echt een probleem maar is hier toch opgenomen omdat het een onvoorziene aanpassing was.

Hoofdstuk 4

Besluiten

4.1 Resultaten

4.1.1 Stageresultaten

De realisaties tijdens de stage zijn de eerste mijlpaal van deze masterproef. De stage duurde slechts 2 weken (van juli 2006), maar er werd hard gewerkt en de realisaties van deze periode liegen er niet om. Startend van enkele *datasheets* en een demo-opstelling van de camera, werden 2 belangrijke creaties voor de verdere masterproef gedaan: de interface-PCB en de beeldverwerkingsalgoritmes.

4.1.1.1 PCB

De interface-PCB was een cruciaal element voor het verdere verloop van de masterproef. Als de PCB niet tijdig af zou zijn, of er zouden later teveel aanpassingen nodig zijn, dan zou de masterproef onaanvaardbare vertragingen oplopen. Het duurt tenslotte ongeveer een maand (na het ontwerp) alvorens de PCB beschikbaar is. Stefan ontwierp de PCB volledig (hoofdstuk 3.1.1.4) tijdens 9 werkdagen in juli en met veel trots was de eerste print van eigen maak begin oktober beschikbaar.

De PCB is een ontwerp met twee lagen, dat de camera met het demobord verbindt. Om ervoor te zorgen dat de PCB ook in de toekomst compatibel met de algoritmes zou blijven werd elke keuze weldoordacht gemaakt. Het is namelijk niet efficiënt om bij elke nieuwe versie van de PCB ook de bestaande algoritmes volledig te moeten herschrijven. De PCB kan worden uitgebreid zonder dat de bestaande connecties moeten aangepast worden.

Meteen na de levering van de PCB kwamen er een aantal problemen aan het licht (zie hoofdstuk 3.3), maar zoals het de analytische geest van een (toekomstig) ingenieur betaamt, werd hiervoor telkens een efficiënte oplossing gevonden. Hierdoor werd het schema van de PCB meermaals herzien. Dit resulteerde in slechts 2 PCB-layouts; de andere wijzigingen gebeurden door het aanbrengen van patches op de bestaande layout om snel verder te kunnen testen. Voor iedere aanpassing werd wel een nieuw schematisch ontwerp gedaan (figuren 5.1 tot 5.10 op pagina's 81 tot 90).

4.1.1.2 Algoritmes

Dankzij de inspanningen van Dave zullen de klanten van Melexis niet langer zelf moeten instaan voor het manueel aanpassen van de cameraparameters bij wisselende lichtomstandigheden. De stage duurde slechts 9 werkdagen maar bij Melexis stonden ze versted van de kracht van de automatische beeldregulator (hoofdstuk 3.1.1.1). Deze functioneert *realtime* en presteert meer dan behoorlijk. Door de snelle resultaten werd het algoritme volledig gedocumenteerd en werd er zelfs een volledige *application note* bij geschreven. Omdat er nog tijd over was, werd ook het probleem van lensdistortie (hoofdstuk 3.1.1.2) onder handen genomen. Hiervoor werd er een correctie-algoritme geschreven. Dat is niet volledig afgewerkt maar het is toch al mogelijk om de kromming van de lens te corrigeren als de variabelen manueel ingesteld worden.

4.1.2 10 januari 2007

De tweede mijlpaal van deze masterproef was een demo, om te tonen hoever we stonden, op 10 januari 2007. Door een vertraging van bijna 2 maanden, ten gevolge van een probleem met de bestelling van de ADSP-BF561 EZ-KIT Lite[®], is er pas vanaf 17 november 2006 van start gegaan met het schrijven van instructiecode voor het camerasysteem. De tijd daarvoor werd opgevuld met het uitproberen van de code, op een simulator, en de assemblage van de PCB. In één maand tijd is het gelukt om van een aantal losse onderdelen tot een volledig camerasysteem te komen. Op de laatste lesdag van het eerste semester was het mogelijk om de camera aan te sturen en één camerabeeld via DMA naar het SDRAM-geheugen te schrijven (hoofdstuk 3.1.3). Het beeld kon dan met de *image viewer* van Visual DSP++ 4.5 worden weergegeven.

4.1.3 Huidige stand

Vanaf de start van het tweede semester werd er gewerkt aan een camerasysteem dat meerdere beelden kon inlezen (hoofdstuk 3.1.5) en deze vervolgens ook weergeven op een beeldscherm in *realtime*. Dit zou tot 2 afzonderlijke systemen leiden: de Composiet-link (hoofdstukken 3.1.7 & 3.1.8) en de SPI/MOST-link (hoofdstuk 3.1.9). De 2 systemen (hoofdstuk 2) zijn op het moment van schrijven nog niet voltooid.

4.1.3.1 Melexis: Composiet-link

In het SDRAM-geheugen wordt een testbeeld geschreven, dat voorzien is van horizontale en verticale synchronisatie. De video-encoder wordt correct ingesteld via IIC. De beelden worden door DMA vanuit het geheugen naar PPI1 gestuurd. PPI1 stuurt de beelden naar de video-encoder aan 50 fps. De Composiet-link werkt met zelfgemaakte testbeelden uit het geheugen. Als echter het algemene systeem, dat de camerabeelden inleest, wordt bijgevoegd dan lopen er zaken fout. De 2 afzonderlijke systemen functioneren correct, maar bij het samenvoegen worden er geen correcte beelden meer weergegeven. De oorzaak van het probleem is waarschijnlijk de *DMA1-controller* die overbelast wordt omdat deze door beide PPI's intensief wordt gebruikt.

4.1.3.2 MULTICARCOM/De Nayer: SPI/MOST-link

De SPI/MOST-link is in principe af, de instructiecode is van ongebruikte functies ontdaan en van commentaar voorzien. De SPI werd getest: zowel de hardware als de software werken correct. De beelden die in de SDRAM-geheugenbuffer zitten opgeslagen worden via DMA2 naar SPORT0 (SPI emulatie), en zo naar de eerste *MOST-transceiver* gestuurd. Deze zijn voorzien van synchronisatie, zoals in de specificaties (hoofdstuk 2) werd omschreven. Ook de LVDS-omzetting voor de SPI-connectie werd grondig getest. Er is echter een probleem in de communicatie met het MOST-systeem. Waarschijnlijk is er een synchronisatieprobleem met de eerste *MOST-transceiver* (Xilinx Virtex-II Pro). Dat is bij de tests ook een aantal keer voorgevallen. Het is ook mogelijk, dat de mediaspeler op de tweede *MOST-transceiver*, de ruwe beelddata niet kan verwerken. De mediaspeler

werkt normaal met MPEG-video. Tenslotte kan het probleem ook in het eigen systeem zitten maar volgens de tests functioneert dat correct. Voorlopig komt er nog geen beeld op het beeldscherm, dus er is nog geen demo-opstelling beschikbaar.

4.1.4 De Toekomst

4.1.4.1 Planning naar de masterproefverdediging toe

Nadat deze thesis is ingeleverd (2 mei 2007) is er nog één maand de tijd om extra mijlpalen te realiseren. Er is gepland om de beide systemen (zie hoofdstuk 2) nog voor de masterproefverdediging (29 mei 2007) af te werken.

Melexis: Composiet-video Door bepaalde algoritmes te optimaliseren, een efficiënter gebruik van de DMA en eventueel het inschakelen van de tweede processorkern zal er getracht worden om alsnog een correct functionerend demosysteem te realiseren (hoofdstuk 3.3).

MULTICARCOM/De Nayer: SPI/MOST-link De problemen met de SPI-synchronisatie zullen in samenwerking met de mensen die aan MOST werken moeten worden opgelost. Er zal getracht worden om de instructiecode in het flashgeheugen op te slaan. Het systeem kan dan vanaf het flashgeheugen opstarten en de code uitvoeren. Op deze manier kan het probleem sneller ontdekt worden, omdat de ontwerpers van het MOST-systeem er dan sneller mee kunnen testen.

Beeldverwerkingsalgoritmes Er was gepland om de beeldverwerkingsalgoritmes nog in de DSP te implementeren. Dit zal waarschijnlijk niet meer voor de masterproefverdediging kunnen gebeuren.

PCB ontwerp Om het systeem efficiënter te maken, zouden er nog een aantal aanpassingen aan de PCB kunnen worden aangebracht in de toekomst (figuren 5.7 tot 5.10 op pagina's 87 tot 90). Zo kan de parallelle bus, voor de camerabeelden, tot een busbreedte van 8 bits worden teruggebracht. Hierdoor is er ook minder verwerking door de software nodig en kan de DMA efficiënter gebruikt worden (hoofdstuk 3.2.2.6). Er kan ook een aparte connector voor SPORT0 op de interface-PCB gemaakt worden, dat is een voorziening voor de SPI/MOST-link (hoofdstuk 3.1.9). Omdat de hardwarematige SPI toch niet gebruikt wordt kunnen de *jumpers* gewoon worden weggelaten zodat enkel de SPORT1 nog wordt gebruikt (figuren 5.9 & 5.10 op pagina's 89 & 90). Ook deze aanpassingen zullen niet meer voor de masterproefverdediging gerealiseerd kunnen worden.

4.2 Besluit

Na maandenlang gewerkt te hebben met de camera kunnen we met trots zeggen dat we de camera onder de knie hebben. De werking van de DSP kennen we intussen ook door en door en met deze kennis is het zeker mogelijk om de tekortkomingen in deze masterproef aan te vullen en op te lossen. Realtime beelden inlezen lukt al, weergeven van enkelvoudige beelden lukt ook reeds via composiet. Bewegende beelden zichtbaar maken kon op het moment van schrijven nog niet, maar we zullen in de komende weken nog trachten dit aan de praat te krijgen. Een mogelijke oorzaak is dat de DSP er telang over doet om de beelden om te zetten van 9-bits data naar 8-bits YCrCb 4:2:2.

Lijst van figuren

1.1	<i>Een voorstelling van het MULTICARCOM-project, toegepast in de wagen</i>	18
1.2	<i>Een overzicht van het globale doel van deze masterproef</i>	19
2.1	<i>De specificaties van Melexis</i>	25
2.2	<i>De specificaties van MULTICARCOM/De Nayer</i>	26
3.1	<i>Voorbeeld van lensdistortie</i>	30
3.2	<i>De werking van de Second Slope</i>	32
3.3	<i>De tijdsreferentie van het systeem</i>	36
3.4	<i>De camera krijgt zijn instellingen van de DSP, via SPI</i>	38
3.5	<i>De beelden worden via een 9-bits parallele bus naar de DSP gekopieerd</i>	40
3.6	<i>DMA transfer voltooid: interrupt service routine</i>	41
3.7	<i>DMA transfer van PPI0 naar de circulaire buffer in het SDRAM-geheugen</i>	42
3.8	<i>De circulaire buffer in het SDRAM-geheugen met Visual DSP++ 4.5 bekeken</i>	44
3.9	<i>Connectoren op de BF EZ-kit LITE</i>	45
3.10	<i>Omzetten van pixels naar YCrCb</i>	46
3.11	<i>De YCrCb 4:2:2-standaard</i>	47
3.12	<i>Framing en synchronisatie van de CCIR656-standaard</i>	47
3.13	<i>Pixelvoorbeeld van de CCIR656-standaard</i>	48
3.14	<i>SPI-link voor de beeldtransfer van de DSP naar de MOST-transceiver</i>	50
3.15	<i>LVDS: de correcte testsequentie wordt ontvangen</i>	51
3.16	<i>Het volledige systeem van het evaluatiebord tot het beeldscherm</i>	52
5.1	<i>Versie 0.0 (blz. 1) van het schema van de Interface-PCB</i>	81
5.2	<i>Versie 0.0 (blz. 2) van het schema van de Interface-PCB</i>	82
5.3	<i>Versie 0.1 (blz. 1) van het schema van de Interface-PCB</i>	83
5.4	<i>Versie 0.1 (blz. 2) van het schema van de Interface-PCB</i>	84
5.5	<i>Versie 1.0 (blz. 1) van het schema van de Interface-PCB</i>	85
5.6	<i>Versie 1.0 (blz. 2) van het schema van de Interface-PCB</i>	86

5.7	<i>Versie 2.0 (blz. 1) van het schema van de Interface-PCB</i>	87
5.8	<i>Versie 2.0 (blz. 2) van het schema van de Interface-PCB</i>	88
5.9	<i>Versie 2.1 (blz. 1) van het schema van de Interface-PCB</i>	89
5.10	<i>Versie 2.1 (blz. 2) van het schema van de Interface-PCB</i>	90

Lijst van tabellen

3.1	EAV- en SAV-signalen	48
3.2	<i>DSP-timing probleem: CCLK = 600 MHz & SCLK = 133,33 MHz</i>	62
3.3	<i>DSP-timing probleem: CCLK = 540 MHz & SCLK = 135 MHz</i>	63
3.4	<i>DSP-timing correct: CCLK = 600 MHz & SCLK = 120 MHz</i>	63

Bibliografie

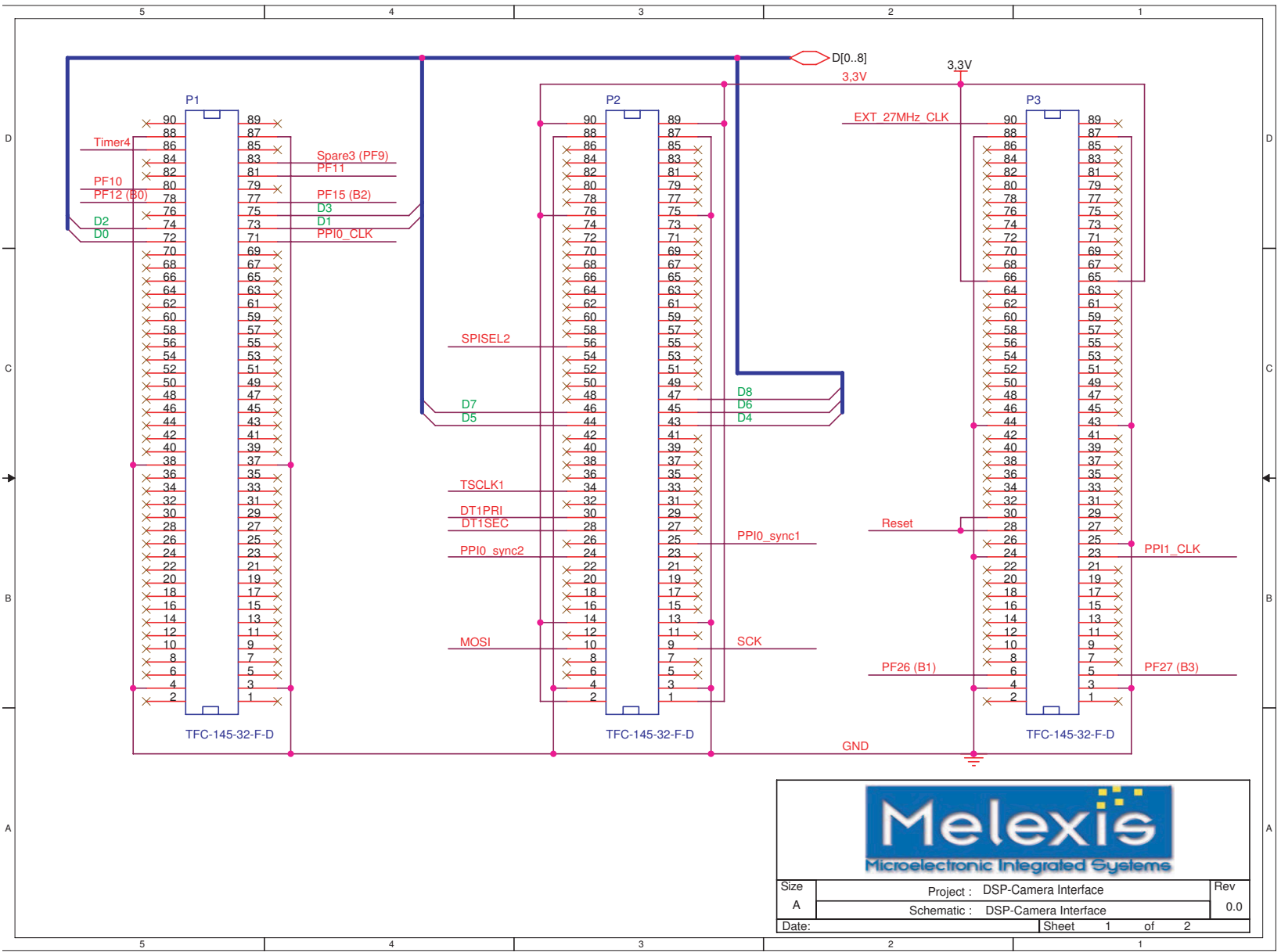
- [1] F. Drive, “Flanders’ drive nieuwsflash,” feb 2006.
- [2] Melexis, *MLX75007 Automotive CMOS PVGA camera*, 002 ed., feb 2006. (confidential).
- [3] Melexis, *MLX75006 & 75007 Automotive CMOS cameras – Family description*, 001 ed., oct 2006. (confidential).
- [4] Analog Devices, Inc., *ADSP-BF561 EZ-KIT Lite[®] Evaluation System Manual*, 3.0 ed., june 2006.
- [5] Newision, *Newision NW619VT*. <http://www.newision.com/NW619VT.htm>.
- [6] Analog Devices, Inc., *ADSP-BF561 Blackfin[®] Processor Hardware Reference*, 1.0 ed., july 2005.
- [7] Analog Devices, Inc., *Blackfin[®] Embedded Symmetric Multiprocessor*, a ed., 2006.
- [8] Melexis, *AN1: MLX75006 & 75007 Explanation of triple slope response*, 002 ed., aug 2005. (confidential).
- [9] Helmut Dersch, *Correcting Barrel Distortion*, july 1999.
- [10] P. Frantz, *Preparing for OrCAD Layout*. <http://cnx.org/content/m11677/latest/>.
- [11] Samtec, *Recommended PCB Layout for TFM(X) (Double)*, e ed.
- [12] Samtec, *Cost Saving Header TFC Series*.
- [13] Analog Devices, Inc., *Blackfin[®] A-V EZ-Extender[®] Manual*, 2.0 ed., april 2006.
- [14] Melexis, *AN8: MLX75006 & 75007 Frequently Asked Questions*, 001 ed., dec 2005. (confidential).
- [15] Analog Devices, Inc., *EE-304: Using the Blackfin[®] Processor SPORT to Emulate a SPI Interface*, 1 ed., nov 2006.
- [16] Analog Devices, Inc., *ADV7174/ADV7179 Chip Scale PAL/NTSC Video Encoder with Advanced Power Management*, a ed., 2004.
- [17] Rohde & Schwarz, broadcasting division, *The digital Video Standard according to ITU–R BT. 601/656*.

-
- [18] KHBO, *Digitale videoformaten en -coderingen*.
- [19] Tektronix, *A guide to standard and high-definition digital video measurements*, jan 2006.
- [20] Samtec, *Cost Saving Socket SFC Series*.
- [21] Samtec, *Micro Data Link TFMDL, FSIF Series*.
- [22] MathWorks, *MATLAB, The Language of Technical Computing, External Interfaces*, ch. MATLAB Interface to Generic DLLs. MathWorks, 7 ed., 2004.
- [23] Analog Devices, Inc., *EE-301: Video Templates for Developing Multimedia Applications on Blackfin[®] Processors*, 1 ed., sep 2006.
- [24] Analog Devices, Inc., *Visual DSP++ 4.5 C/C++ Compiler and Library Manual for Blackfin Processors*, 4.0 ed., april 2006.
- [25] Melexis, *AN3: MLX75006 & 75007 Digital simulation*, 001 ed., jan 2004. (confidential).
- [26] Melexis, *AN6: MLX75006 & 75007 Basic Application Circuits*, 001 ed., mar 2006. (confidential).
- [27] Melexis, *AN2: MLX75006 & 75007 Lexicon and technology*, 001 ed., may 2005. (confidential).
- [28] Analog Devices, Inc., *ADSP-BF561 Ez-Kit Booting / Flash Programmer Example*, 1.0 ed., mar 2004.
- [29] Analog Devices, Inc., *EE-203: Interfacing the ADSP-BF535/ADSP-BF533 Blackfin[®] Processor to NTSC/PAL video decoder over the asynchronous port*, sep 2003.
- [30] Analog Devices, Inc., *EE-276: Video Framework Considerations for Image Processing on Blackfin[®] Processors*, 1 ed., sep 2005.
- [31] Analog Devices, Inc., *EE-311: VisualDSP++[®] Flash Programmer API for Blackfin[®] Processors*, 1 ed., dec 2006.
- [32] Melexis, *AN9: MLX75006 & 75007 Safety Critical Applications*, 001 ed., nov 2004. (confidential).
- [33] Melexis, *AN10: MLX75006 & 75007 Layout guidelines*, 001 ed., sep 2005. (confidential).
- [34] F. Drive, "Flanders' drive nieuwsbrief 1."

Hoofdstuk 5

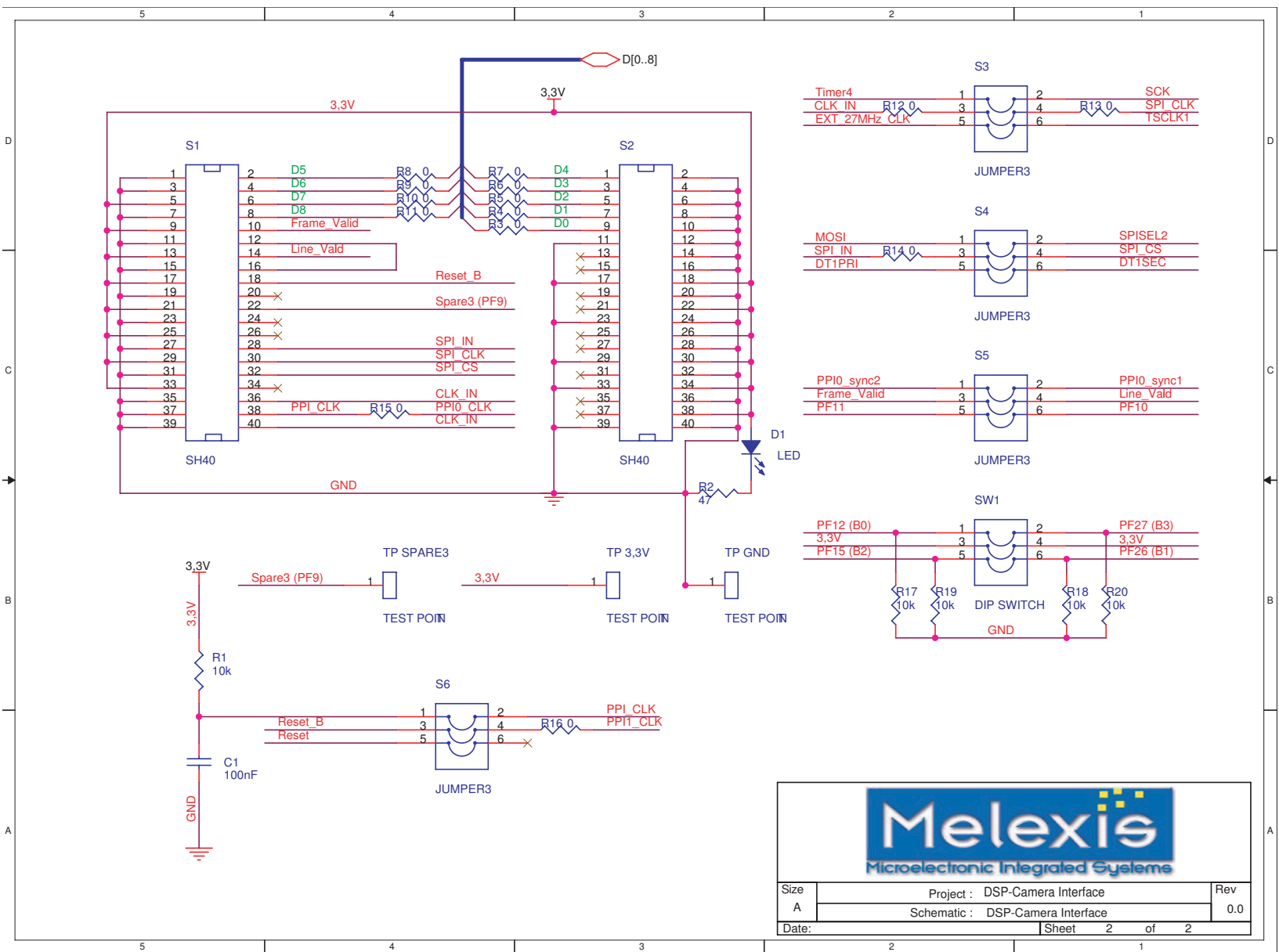
Bijlagen

Figuur 5.1: Versie 0.0 (blz. 1) van het schema van de Interface-PCB. Van dit schema werd ook een PCB-layout ontworpen. Daarmee werd de eerste interface-PCB gemaakt.



Size	Project : DSP-Camera Interface	Rev
A	Schematic : DSP-Camera Interface	0.0
Date:	Sheet 1 of 2	

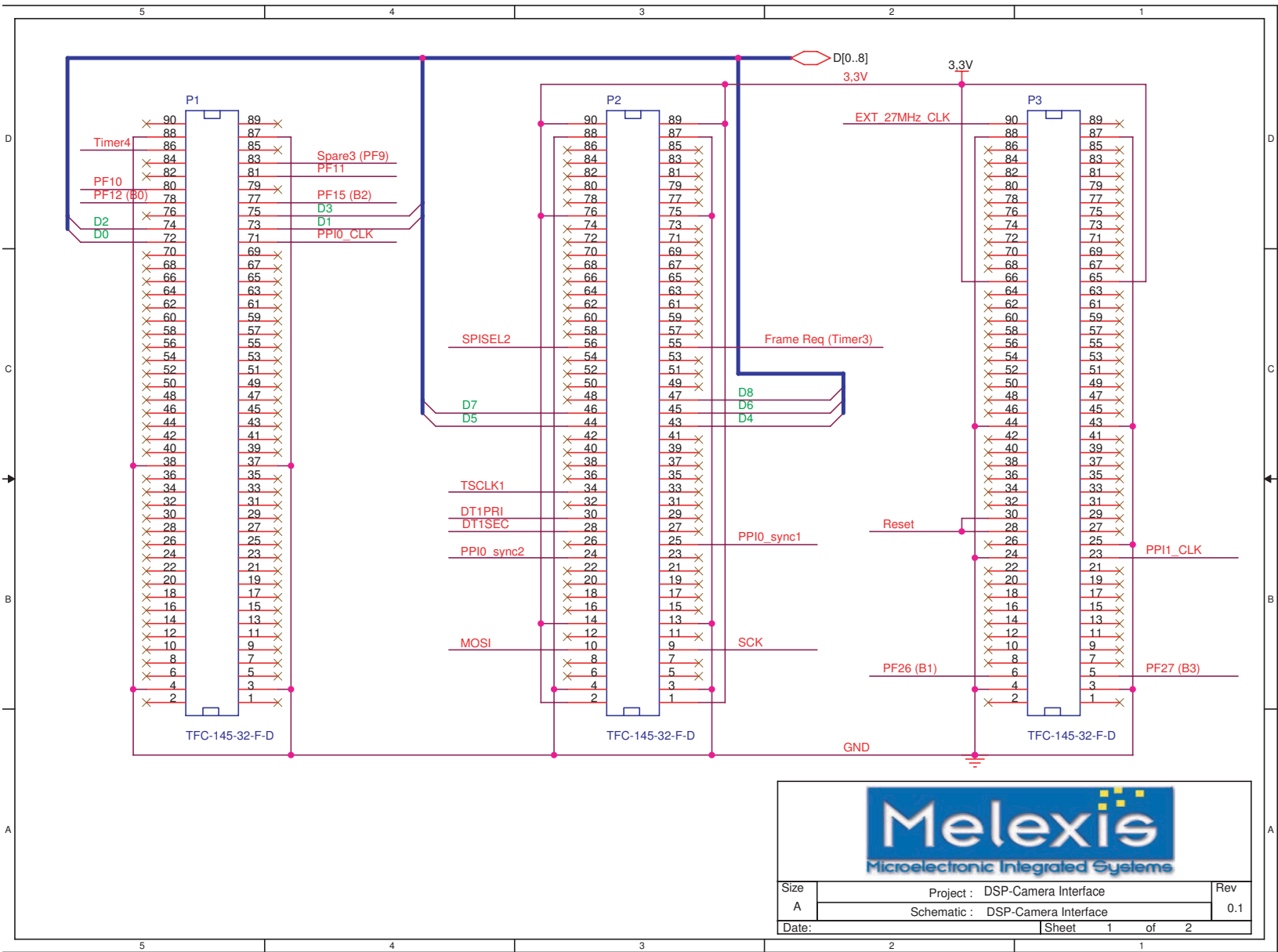
Figuur 5.2: Versie 0.0 (blz. 2) van het schema van de Interface-PCB. Van dit schema werd ook een PCB-layout ontworpen. Daarmee werd de eerste interface-PCB gemaakt.

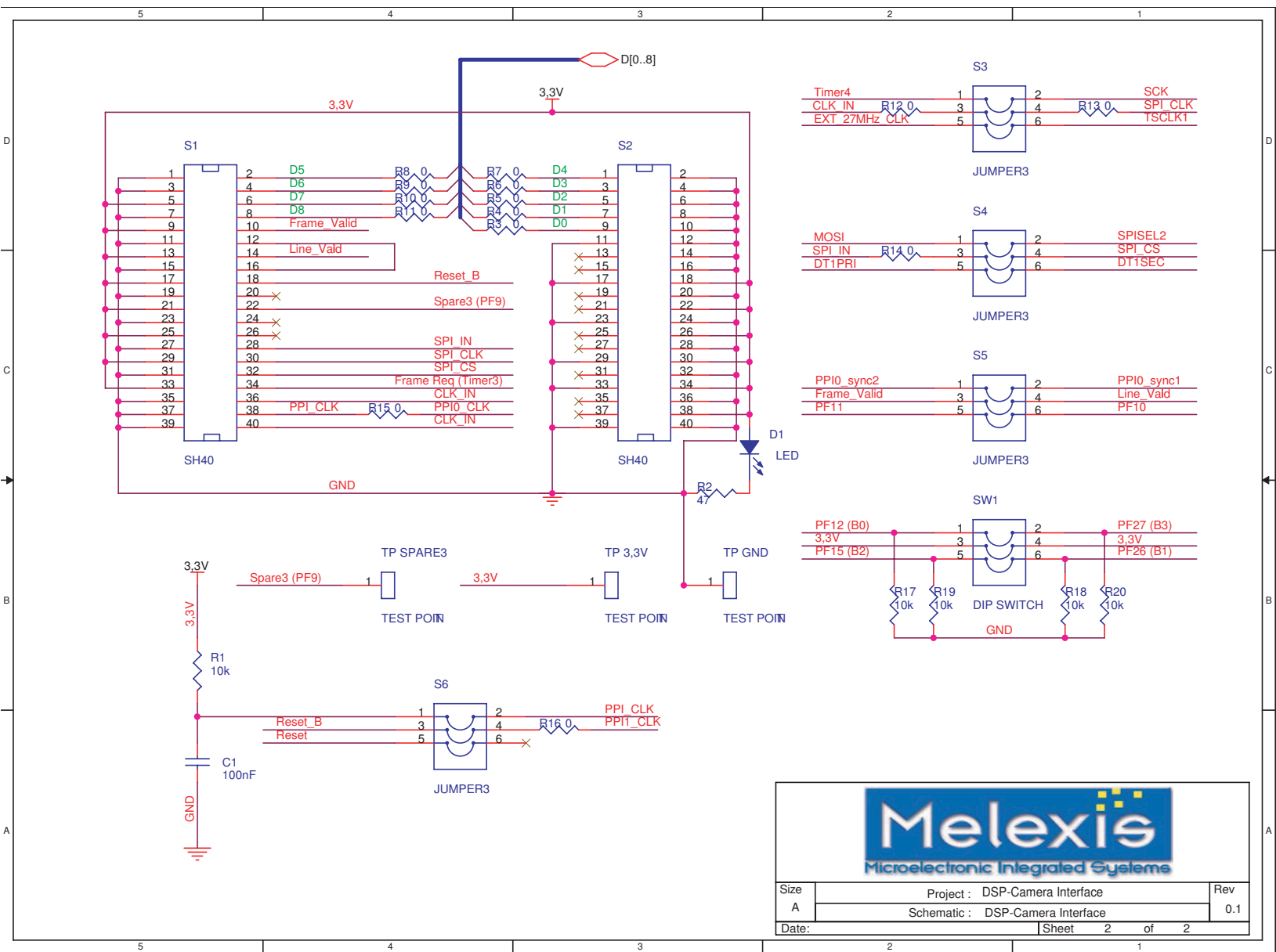


Melexis
Microelectronic Integrated Systems

Size	Project : DSP-Camera Interface	Rev
A	Schematic : DSP-Camera Interface	0.0
Date:	Sheet 2 of 2	

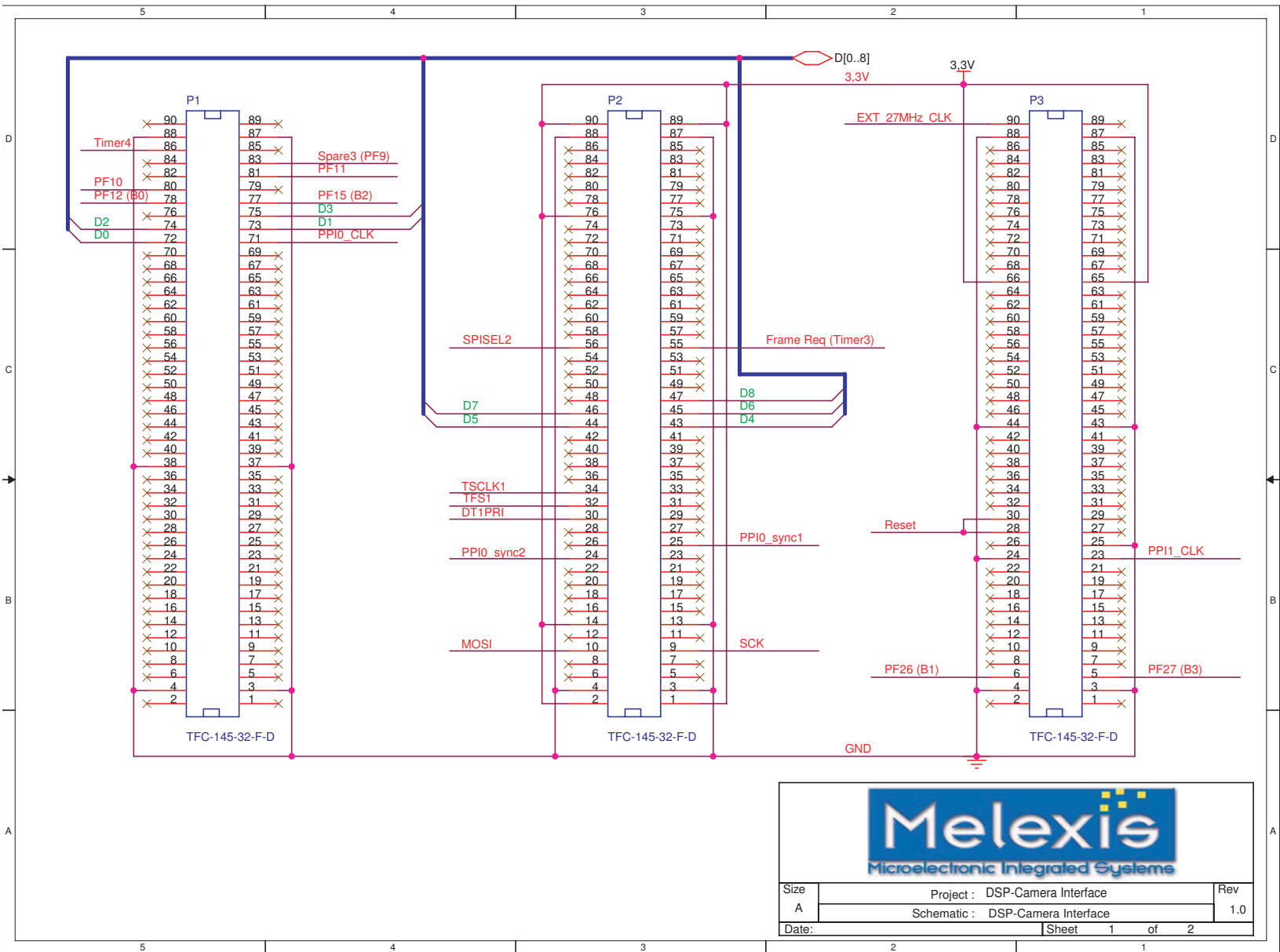
Figuur 5.3: Versie 0.1 (blz. 1) van het schema van de Interface-PCB. De ontbrekende frame-request connectie is toegevoegd. Van dit schema werd ook een PCB-layout ontworpen. Daarmee werd de tweede interface-PCB gemaakt.





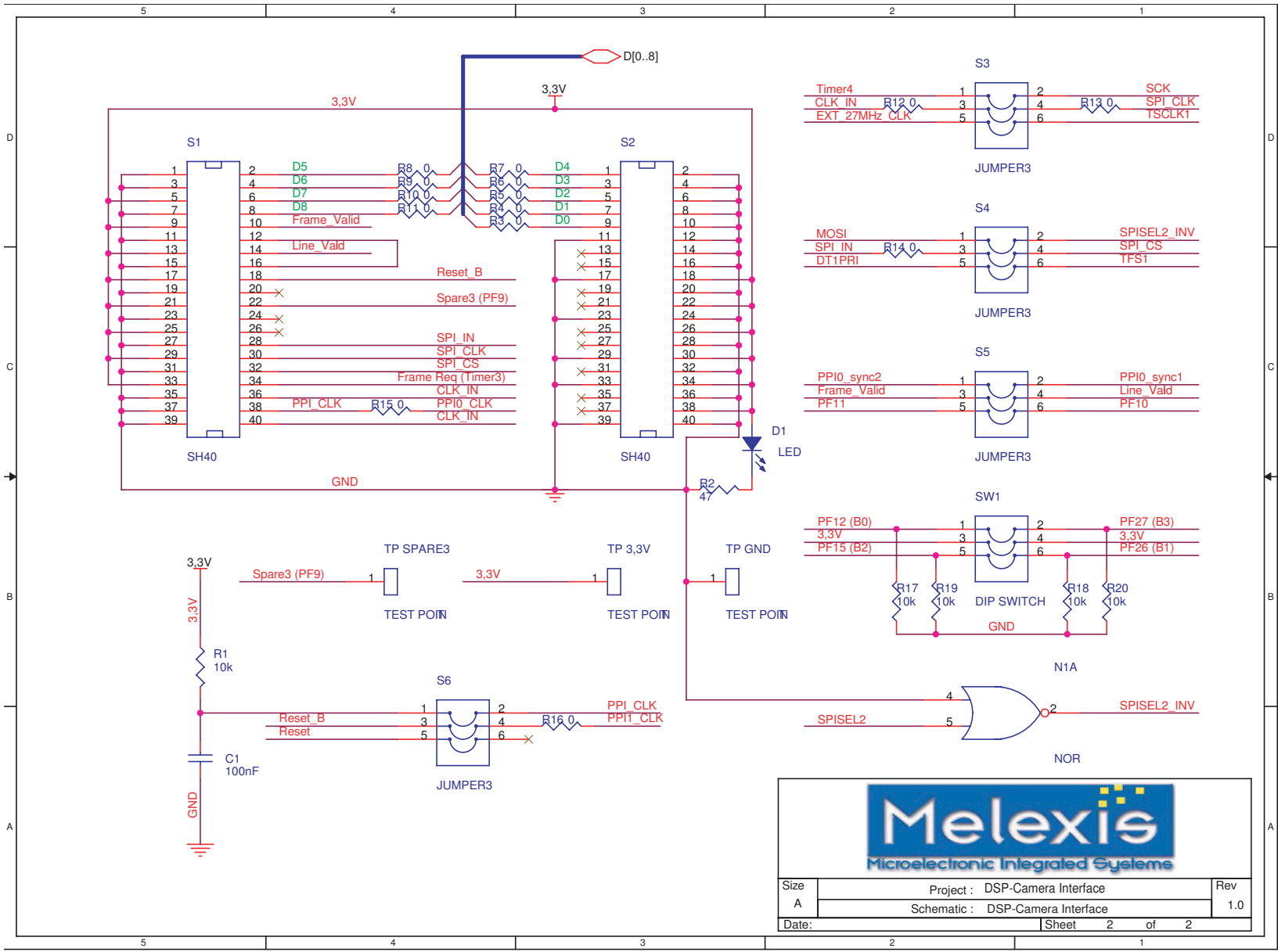
Figuur 5.4: Versie 0.1 (blz. 2) van het schema van de Interface-PCB. De ontbrekende frame-request connectie is toegevoegd. Van dit schema werd ook een PCB-layout ontworpen. Daarmee werd de tweede interface-PCB gemaakt.

Figuur 5.5: Versie 1.0 (blz. 1) van het schema van de Interface-PCB. De SPORT1 emuleert de slave-select van de SPI nu met TFS1 (synchronisatie) in plaats van met DT1SEC (secundair datatakmaal). De slave-select van de SPI zelf wordt nu geïnverteerd.



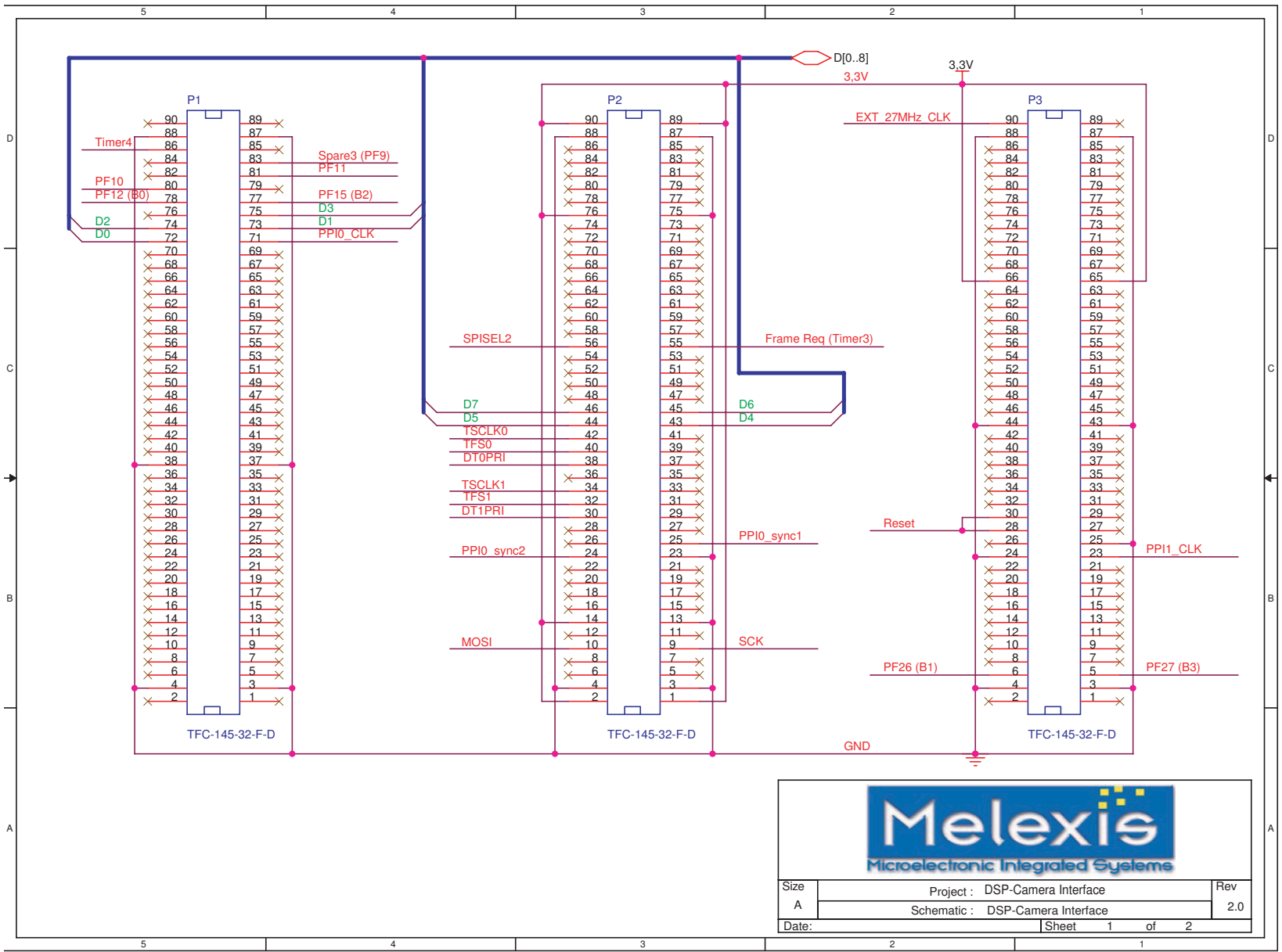
Melexis
Microelectronic Integrated Systems

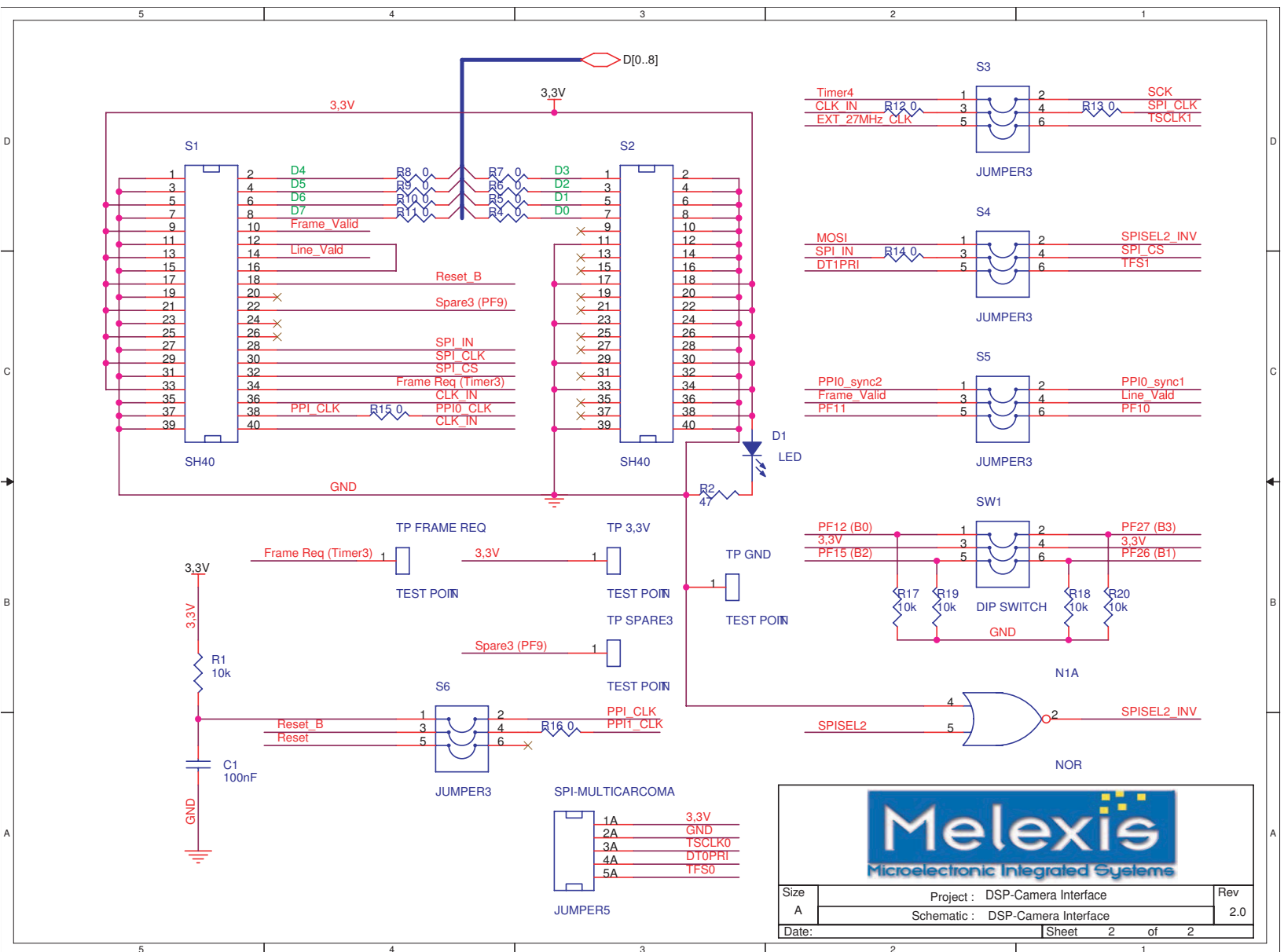
Size	Project : DSP-Camera Interface	Rev
A	Schematic : DSP-Camera Interface	1.0
Date:	Sheet 1 of 2	



Figuur 5.6: Versie 1.0 (blz. 2) van het schema van de Interface-PCB. De SPORT1 emuleert de slave-select van de SPI nu met TFS1 (synchronisatie) in plaats van met DT1SEC (secundair datakanaal). De slave-select van de SPI zelf wordt nu geïnverteerd.

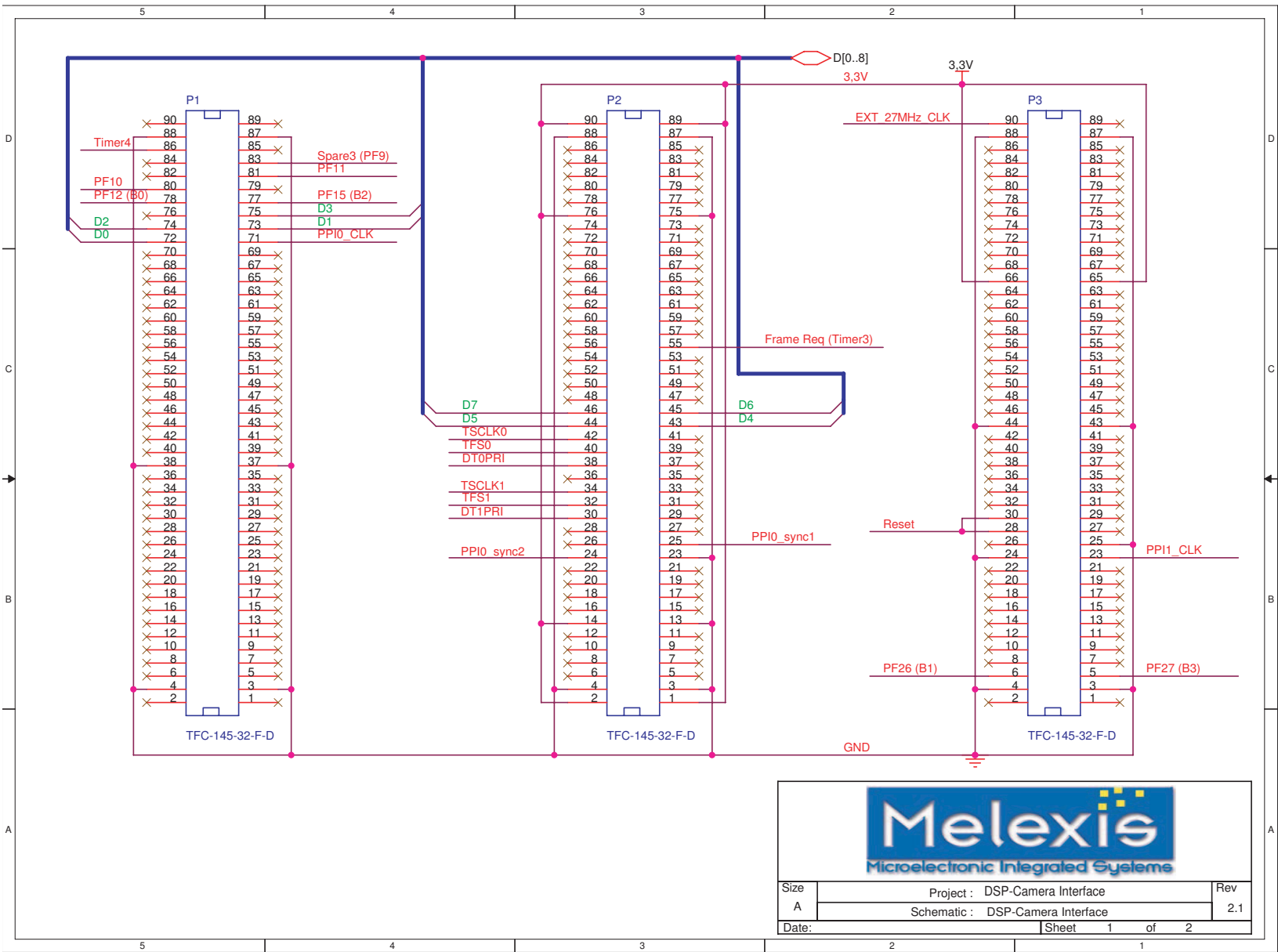
Figuur 5.7: Versie 2.0 (blz. 1) van het schema van de Interface-PCB. Dit is een ontwerp naar de toekomst toe. De busbreedte van de PPI is verminderd tot 8 bits en het derde synchroonistisch signaal van de PPI is aan de grond verbonden. De frame-request is van een testpunt voorzien. Tenslotte is er een connector voor de SPI/MOST-link op de PCB geplaatst.





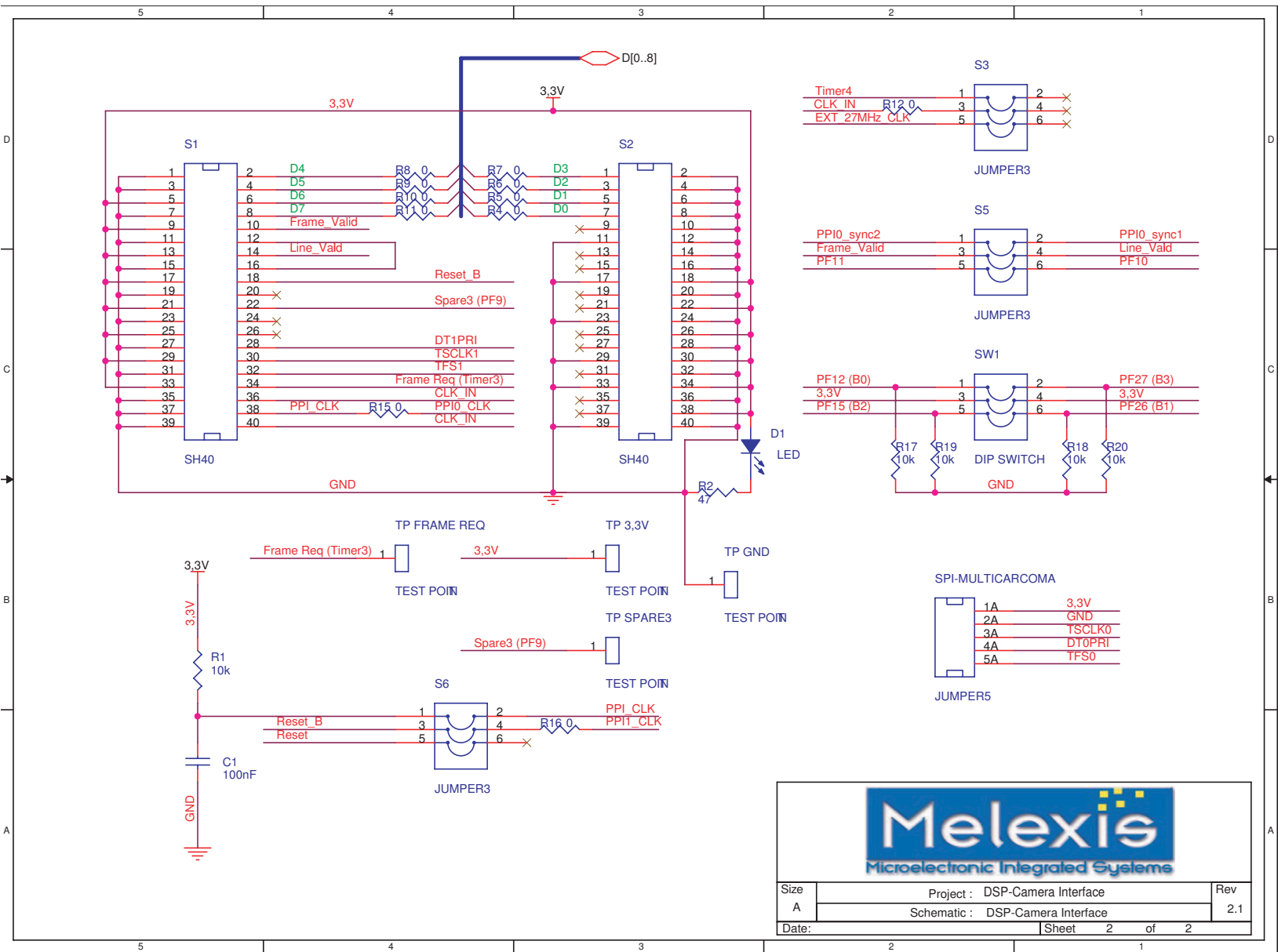
Figuur 5.8: Versie 2.0 (blz. 2) van het schema van de Interface-PCB. Dit is een ontwerp naar de toekomst toe. De busbreedte van de PPI is verminderd tot 8 bits en het derde synchronisatiesignaal van de PPI is aan de grond verbonden. De frame-request is van een testpunt voorzien. Tenslotte is er een connector voor de SPI/MOST-link op de PCB geplaatst.

Figuur 5.9: Versie 2.1 (blz. 1) van het schema van de Interface-PCB. Dit is een ontwerp naar de toekomst toe. De SPI wordt toch sowieso met de SPORT1 nagebootst, daarom zijn de jumpers voor SPI verwijderd.



Melexis
Microelectronic Integrated Systems

Size	Project : DSP-Camera Interface	Rev
A	Schematic : DSP-Camera Interface	2.1
Date:	Sheet 1 of 2	



Figuur 5.10: Versie 2.1 (blz. 2) van het schema van de Interface-PCB. Dit is een ontwerp naar de toekomst toe. De SPI wordt toch sowieso met de SPORT1 nagebootst, daarom zijn de jumpers voor SPI verwijderd.