Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Studiengang Systemtechnik

Vertiefungsrichtung Infotronics

Diplom 2015

Nicolas Andres

Bildverarbeitungskarte CubeSat

Dozent/in
François Corthay

Experte/Expertin
Muriel Richard, EPFL Space Engineering Center (eSpace)

Datum der Abgabe des Schlussberichts
10.07.2015

HES-SO Valais-Wallis • rte du Rawyl 47 • C.P. • 1950 Sion 2
+41 27 606 85 23 • hei@hevs.ch • www.hevs.ch

Valais excellence

Données du travail de diplôme
Daten der Diplomarbeit

☒ FSI ☐ FTV	Année académique / *Studienjahr* **2014/15**	No TD / *Nr. DA* **it/2015/15**
Mandant / *Auftraggeber* ☐ HES—SO Valais ☐ Industrie ☒ Etablissement partenaire *Partnerinstitution* **EPFL Space Center**	Etudiant / *Student* **Nicolas Andres** Professeur / *Dozent* **François Corthay**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte (données complètes)* **Muriel Richard** Space Engineering Center EPFL (eSpace)	

Titre / *Titel*

Bildverarbeitungskarte für CubeSat

Description / *Beschreibung*

CubeSats sind kleine Satelliten mit einem Volumen von 1 Liter. Sie werden hauptsächlich gebraucht, um einfache Experimente durchzuführen oder um neue Technologien im Weltall zu testen. Verschiedene Anwendungen wenden eine Kamera an, und die erworbenen Bilder sollten verarbeitet werden, bevor sie zur Erde gesandt werden.

Ziel dieser Arbeit ist, eine Karte zu erstellen, welche Bilder von einer Kamera erhaltet, diese verarbeitet und die resultierende Daten dem Hauptprozessor sendet. Die Bildverarbeitung findet in einem Gumstix Overo Computer-On-Module (COM) board statt. Der Datenaustausch zwischen der Karte und dem Hauptprozessor wird mit Hilfe eines programmierbaren Bausteins von Typ FPGA durchgeführt.

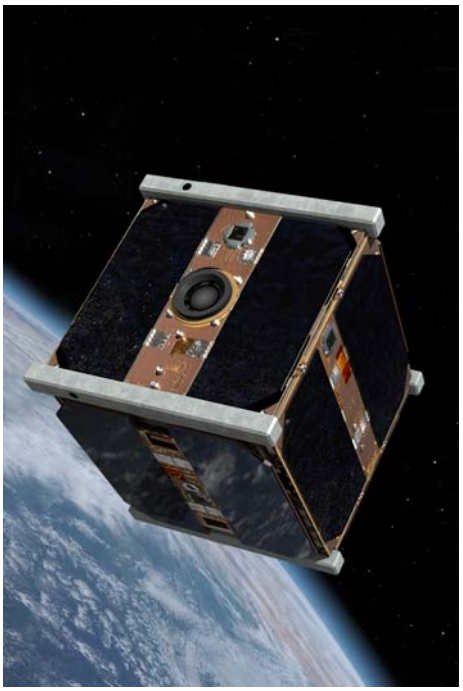
Objectifs / *Ziele*

— Adapter-Karten für den Anschluss an verschiedene Kameras sind zu erstellen
— Die Ersetzung des COM-Moduls mit einer Speicherkarten muss vorgesehen werden
— Ein PCB muss erstellt und getestet werden.

Signature ou visa / *Unterschrift oder Visum* Responsable de l'orientation *Leiter der Vertiefungsrichtung:*..... [1] Etudiant / Student :.....	Délais / *Termine* Attribution du thème / *Ausgabe des Auftrags:* 11.05.2015 Remise du rapport / *Abgabe des Schlussberichts:* 10.07.2015, 12:00 Expositions / *Ausstellungen der Diplomarbeiten:* 26 – 28.08.2015 Défense orale / *Mündliche Verfechtung:* Semaine | Woche 36

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....



Bildverarbeitungskarte CubeSat

Diplomand/in Nicolas Andres

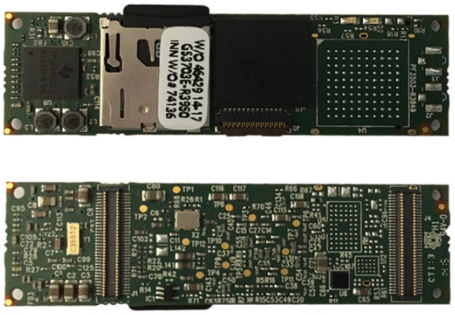
Ziel des Projekts

Das Ziel der Bachelorarbeit ist es, eine elektronische Karte zu entwickeln, welche Bilder einer Kamera aufnehmen kann, diese verarbeitet und die resultierenden Daten an die Hauptprozessorkarte des Satelliten weiterleitet.

Methoden | Experimente | Resultate

Für eine kommende Generation von CubeSat-Satelliten wurde ein erster Prototyp einer Bildverarbeitungskarte entwickelt. Auf der Karte wurde ein Computer-On-Modul (Gumstix Overo COM: WATERSTORM) untergebracht. Dieses Modul ist speziell für graphische Applikationen ausgelegt und es läuft ein Embedded-Linux Betriebssystem darauf. Auf dem Modul soll die Verarbeitung der Bilder durchgeführt werden. Mittels eines programmierbaren Bausteins von Typ FPGA wird die Kommunikation zwischen Kamera, Gumstix und der Hauptprozessor-karte geregelt.
Die Karte wurde so entwickelt, dass das COM-Modul durch ein alternatives Speichermodul ersetzt werden kann. Somit empfiehlt sich die Karte auch für eine allgemeinere Nutzung.
Ein 8-Kanal ADC-Konverter dient zur Messung der verschiedenen Spannungen auf dem Board. Drei nicht genutzte Eingänge wurden auf einem Konnektor herausgeführt und können für die Messung von externen Spannungen, wie beispielsweise einem analogen Sensor, eingesetzt werden.
Für mögliche Erweiterungen wurden zwei zusätzliche Anschlüsse auf die FPGA verbunden. Somit besteht die Möglichkeit, neben der Kamera noch zwei weitere Peripherien mit der Bildverarbeitungskarte zu verbinden.

Bildbearbeitungskarte V1.0
Ansicht des PCB von oben
Abmessungen: 95x95 mm



Gumstix Overo COM : WATERSTORM
Ansicht des Gumstix von oben / unten.
Abmessungen: 57x17 mm

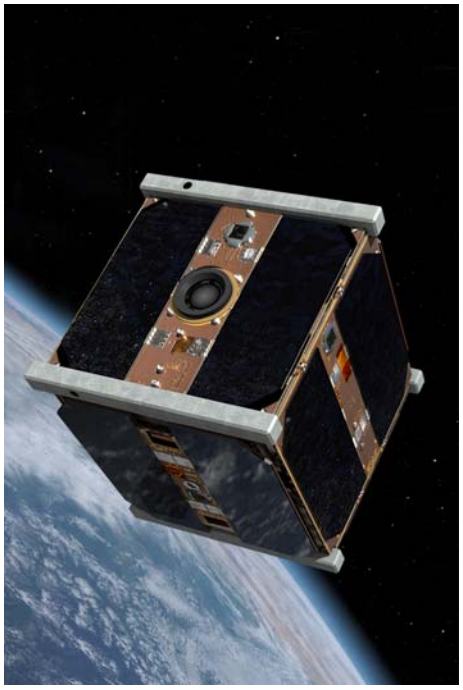
Diplomarbeit
| 2015 |

Studiengang
Systemtechnik

Anwendungsbereich
Infotronics

Verantwortliche/r Dozent/in
Francois Corthay
Francois.Corthay@hevs.ch

Partner
EPFL Space Engineering Center (eSpace)



Carte de traitement d'image pour CubeSat

Diplômant/e Nicolas Andres

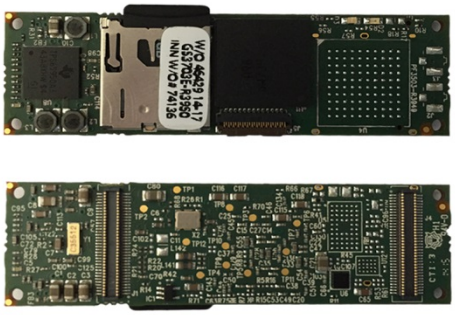
Objectif du projet

L'objectif du travail de bachelor est de développer une carte électronique qui peut prendre des photos d'une caméra, les traiter et transmettre les données résultantes à la carte processeur principale du satellite.

Méthodes | Expériences | Résultats

Un premier prototype de la carte de traitement d'image a été développé pour une prochaine génération de satellites CubeSat. Un Computer-On-Module (Gumstix Overo COM: WATERSTORM) a été intégré sur la carte. Ce module est conçu spécifiquement pour les applications graphiques et il fonctionne avec un système d'exploitation Linux embarqué. Le traitement des images est effectué sur le module. Un circuit programmable de type FPGA contrôle la communication entre la caméra, le Gumstix et la carte processeur principale.
Sur la carte développée, le module COM peut être remplacé par un module de mémoire. Donc la carte peut s'utiliser dans un cadre plus général.
Un convertisseur AD avec 8 canaux est utilisé pour mesurer les différentes tensions sur la carte. Trois entrées inutilisées sont branchées sur un connecteur et peuvent être utilisées pour mesurer des tensions externes, comme un capteur analogique.
Pour des extensions possibles, deux connecteurs supplémentaires sont raccordés à la FPGA. Ainsi, il est possible de connecter deux autres périphériques à la carte de traitement d'image.

Carte traitement d'image V1.0
Vue du PCB de dessus
Dimension : 95x95 mm



Gumstix Overo COM : WaterSTORM
Vue du Gumstix de dessus / de dessous
Dimension : 57x17 mm

Travail de diplôme
| édition 2015 |

Filière
Systèmes industriels

Domaine d'application
Infotronics

Professeur responsable
Francois Corthay
Francois.Corthay@hevs.ch

Partenaire
EPFL Space Engineering Center (eSpace)

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungserklärung

ADC	Analog-Digital-Wandler
CMOS	Metalloxid Halbleiter (Comlementary metal-oxid-semiconductor)
COM	Computer-On-Modul
DC/DC	Gleichspannungswandler
DSP	Digital Signal Processor
EPFL	École polytechnique fédérale de Lausanne
ETHZ	Eidgenössische Technische Hochschule Zürich
FPGA	Field Programmable Gate Array
FPS	Bilder pro Sekunde (Frame per Second)
GPIO	General Purpose Input/Output
GPMC	General Purpose Memory Controller
LiDAR	Licht Detektion
LVDS	Low Voltage Differential Signaling
OBC	On-Board-Computer
OpenGL	Open Graphic Library
PCB	Printed Circuit Board (elektronische Karte)
RAM	Random-Access Memory (Direktzugriffspeicher)
SCCB	Serial Camera Control Bus
SMD	Surface-Mounted Device
SPI	Serial Peripheral Interface
UART	Universal Asynchronus Receiver/Transmitter
VHDL	Hardware Description Language (Hardwarebeschreibungssprache)

1. Einleitung

CubeSats sind kleine Satelliten mit einem Volumen von einem Liter. Diese werden typischerweise von Universitäten genutzt, um einfache technische Experimente durchzuführen oder um neue Technologien im Weltall zu testen.

Verschiedene Peripherien wenden dabei eine Kamera an und die erworbenen Bilder sollen verarbeitet werden, bevor diese zur Erde gesandt werden.

Das Ziel der Diplomarbeit ist es, eine elektronische Karte zu entwickeln, welche Bilder von einer Kamera erhält, diese weiterverarbeiten kann und die resultierenden Daten dem Hauptprozessor übergibt, welcher diese zur Erde sendet. Die Bildverarbeitung findet auf einem Gumstix Overo COM Board statt. Der Datenaustausch zwischen Kamera, Gumstix und dem Hauptprozessor wird mit Hilfe eines programmierbaren Bausteins von Typ FPGA durchgeführt.

Als Vorbereitung für die Diplomarbeit wurde im 6. Semester des Studiums eine Semesterarbeit durchgeführt. Während dieser Arbeit wurden bereits einige Themen behandelt und die nötigen Abklärungen/Vorbereitungen durchgeführt, um die Diplomarbeit erfolgreich in Angriff nehmen zu können.

Diese Arbeit wird in Zusammenarbeit mit dem Space Engineering Center (eSpace) der EPFL Lausanne und der Firma ELSE SA realisiert.

Der folgende Bericht beschäftigt sich mit den während der Diplomarbeit durchgeführten Arbeiten.

2. Pflichtenheft

Dieses Kapitel zeigt einen Überblick über das genaue Pflichtenheft des Projekts.

2.1. Projektziel

Ziel der Diplomarbeit ist es, für eine kommende Generation von CubeSat-Satelliten eine neue Bildverarbeitungskarte zu entwickeln. Auf dieser Karte soll ein Mikroprozessor mit DSP-Coprozessor (Gumstix Overo COM)-Modul integriert werden. Dieses Modul wird für die Verarbeitung von Bildern gebraucht. Mittels einer FPGA wird die Kommunikation zwischen Kamera, Gumstix und Hauptprozessorkarte geregelt.

Die Bildverarbeitung auf dem Gumstix-Modul ist nicht Teil der Diplomarbeit. Dieser Teil wird an der Fachhochschule He-arc Neuchâtel als separater Teil umgesetzt.

Die Karte soll an den „ELSE xU Struktur“ -Standard (Elegant Systems Engineering, 2015) angepasst werden, welcher bereits bei den CubETH von der ETHZ und dem Swiss Space Center der EPFL genutzt wird. Durch Verwendung dieses Standards wird ein Satellit sehr modular und individuell anpassbar.

Die FPGA erhält von der Hauptprozessorkarte Befehle und steuert dementsprechend die Kamera. Die aufgenommenen Bilder werden anschliessend von der FPGA empfangen und an das Gumstix-Modul für die Bildverarbeitung weitergegeben. Auf der Bildverarbeitungskarte sollte die Speicherung einiger Bilder möglich sein.

Die verarbeiteten Bilder können anschliessend mittels der FPGA an den Hauptprozessor weitergeleitet werden. Dieser sendet die Bilder zur Erde.

Dieser Aufbau ist im nachfolgenden Blockschaltbild vereinfacht dargestellt (**Fig. 1** *Blockschaltbild*). Das komplette und detaillierte Hardwareschema wird im Kapitel 3: **Hardware-Blockschaltbild** behandelt.

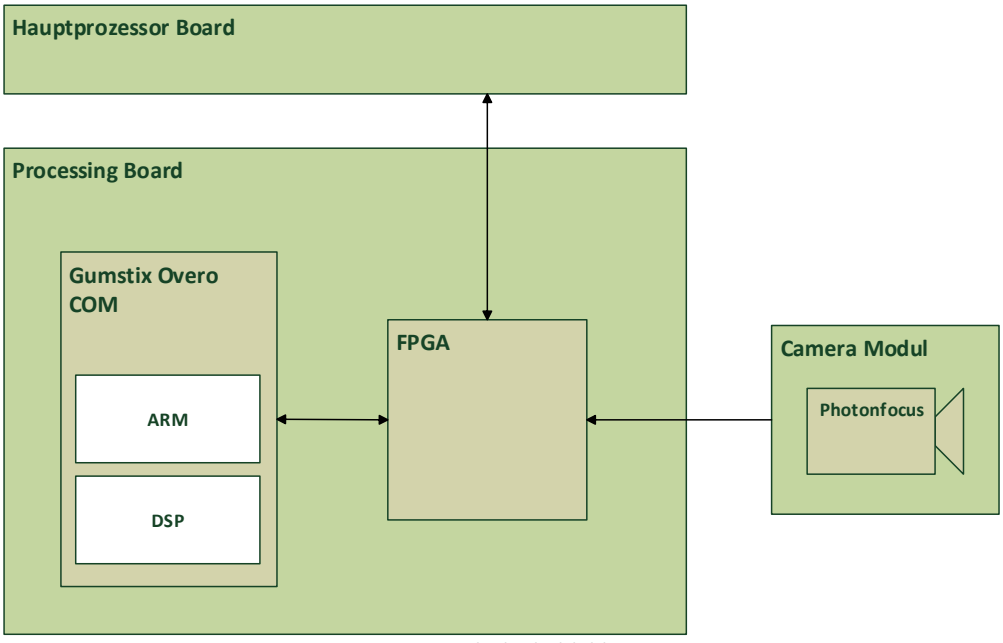


Fig. 1 Blockschaltbild

Folgende Ziele wurden für die Diplomarbeit definiert:

- **Adapter-Karten für verschiedene Kameramodule erstellen**
  Um ein Kameramodul an die Bildverarbeitungskarte anschliessen zu können, ist ein Adapterboard notwendig. Mittels eines geeigneten Interface können so verschiedene Kameramodule mit der Bildverarbeitungskarte verbunden werden.

- **Speichermodul als Alternative für Gumstix COM vorsehen**
  In einem simplen bzw. generelleren Anwendungsbereich der Bildverarbeitungskarte könnte der Gumstix überflüssig sein. Die Karte sollte deshalb so konzipiert werden, dass auf dem Gumstix Anschluss ein alternatives Modul angeschlossen werden kann, welches über einen/mehrere Speicher verfügt. Dazu müssen bei der Planung der Karte die nötigen Verbindungen vorgesehen werden.

- **PCB erstellen und testen**
  Es soll ein erster Prototyp der Bildverarbeitungskarte geplant und realisiert werden. Die Schaltung einer existierenden FPGA-Karte soll als Vorlage genommen werden und soweit erweitert/abgeändert werden, um den Anschluss des Gumstix und der Kamera sicherzustellen.

2.2. Material/Programme

Für die Umsetzung der Diplomarbeit wurden diverse Programme für die einzelnen Teilarbeiten benutzt:

Inkscape

Inkscape ist ein professionelles Programm zum Erstellen und Bearbeiten von Vektorgrafiken. Das Programm kann gratis von der Herstellerhomepage heruntergeladen werden.

Inkscape wurde verwendet, um eine Skizze vom PCB zu erstellen, um den Platzbedarf der Komponenten zu bestimmen.

Microsoft Visio

Visio ermöglicht die einfache Erstellung von professionellen Diagrammen. Es steht eine grosse Bibliothek mit Vorlagen zur Verfügung.
Mit Hilfe von Visio wurden Blockschaltbilder und Übersichten erstellt.

P-CAD 2006 (Schematic & PCB)

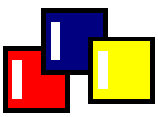
P-CAD ist eine Software für die Entwicklung von Leiterplatten in der Elektronik. Diese Software wird an der Hes-so Wallis in Sion ebenfalls verwendet.

„P-CAD Schematic“ diente zur Erfassung des Schaltplanes der Bildverarbeitungskarte und mit „P-CAD PCB“ wurde das Leiterplatten-Layout gezeichnet.

VMware

In einer virtuellen Umgebung kann parallel zum Betriebssystem des PC ein weiteres Betriebssystem genutzt werden.

Auf dem VMware Player wurde eine virtuelle Maschine mit einer Linux Distribution (Ubuntu 14.04) genutzt, um das Image für den Gumstix zu kompilieren.



HTerm

HTerm ist ein Terminal-Programm für die serielle Schnittstelle unter Windows. Das Programm kann gratis von der Herstellerhomepage heruntergeladen werden.

HTerm wurde für die Kommunikation zwischen der Bildverarbeitungskarte und dem PC genutzt. Das Programm bietet sehr viele Einstellungsmöglichkeiten und erlaubt den Export der empfangenen Daten.

PyCharme

PyCharme ist ein Programm für die Erstellung von Python-Projekten. Es handelt sich hierbei um eine Gratis-Software, welche auf der Herstellerseite zum Download verfügbar ist.

Vom der FPGA werden über eine UART-Schnittstelle (RS232) Daten seriell an den PC übermittelt. Um die Daten besser auswerten zu können, wurde eine kleine Applikation erstellt, welche die Daten graphisch anzeigt.

VHDL

HDL-Designer

HDL-Designer kombiniert tiefe Analysefähigkeiten, fortschrittliche kreative Editoren und ein komplettes Projektmanagement, um leistungsfähige HDL-Design zu erstellen.

Der HDL-Designer diente zur Erstellung des Hardware-Designs der FPGA.

Model SIM

Model SIM ist eine Simulations-Umgebung, um Hardwaredesigns welche mit VHDL erstellt wurden zu simulieren.

Model SIM wurde genutzt, um das erstellte Hardwaredesign zu testen, bevor dieses auf die FPGA programmiert wurde. Damit konnten allfällige Probleme im Voraus behoben werden.

ISE Project Navigator

Der ISE Project Navigator ist ein Programm von Xilinx, welches die verschiedenen Tools für das Synthetisieren und Programmieren ihrer FPGA-Bausteine bereitstellt.

Designs welche mit dem HDL-Designer erstellten wurden, können mit dem ISE Project Navigator für die FPGA von Xilinx bearbeitet und ins nötige Format umgewandelt werden, damit das Design auf die FPGA programmiert werden kann.

3. Hardware-Blockschaltbild

Dieses Kapitel befasst sich mit dem detaillierten Hardware-Blockschaltbild der Bildverarbeitungskarte.

3.1. Überblick Bildverarbeitungskarte

Im ***Anhang 1*** ist das detaillierte Hardware-Blockschaltbild ersichtlich.

Das „Processing Board“ stellt die Bildverarbeitungskarte dar, welche im Rahmen der Diplomarbeit entwickelt werden soll. Der Kern dieser Karte bildet die FPGA, welche für den Datenaustausch sowie die Steuerung der angeschlossenen Peripherien zuständig ist. Es ist eine grosse Anzahl von Ein/Ausgängen notwendig, um alle angeschlossenen Peripherien steuern zu können. Es wird deshalb eine FPGA von Xilinx, genauer gesagt das Model Spartan6, verwendet. Diese FPGA wird auch in anderen Projekten für Space-Anwendungen eingesetzt und hat sich dabei bewährt.

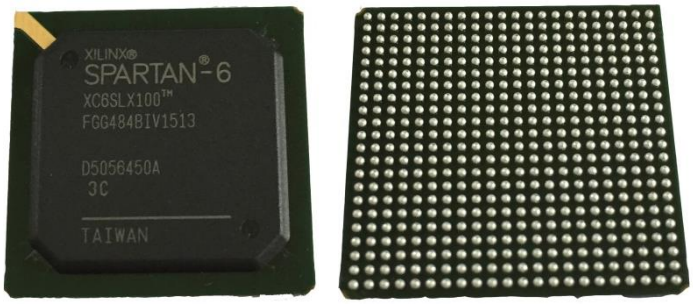


Fig. 2 Xilinx Spartan-6

Der Gumstix, welcher auf der Karte integriert wird, dient der Verarbeitung der erfassten Bilder. Das Model wurde bereits im Vorfeld vom Space Engineering Center (eSpace) der EPFL festgelegt. Anhand einer Studie, welche die Hes-so Wallis im Sommer '14 machte, wurden geeignete Lösungen gesucht, welche die nötigen Anforderungen erfüllen (LowPower, minimale Grösse). Es wird das Model „WaterSTORM“ verwendet, welches auf einem ARM Cortex A8-Processor (DM3730 von Texas Instruments) basiert. Zusätzlich ist die Library OpenGL und ein DSP (Fixed Point) integriert. Dieses Model ist speziell für grafische Applikationen ausgelegt. Auf dem Gumstix sind ein NAND-Flash-Speicher (512 MB), ein SDRAM-Speicher (512 MB) sowie ein microSD-Steckplatz als Speichererweiterung vorhanden.

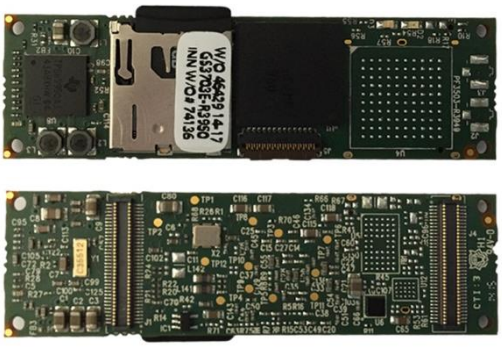


Fig. 3 Gumstix Overo COM WaterSTORM

Der Datenaustausch zwischen FPGA und Gumstix erfolgt über eine schnelle Schnittstelle, damit sichergestellt werden kann, dass auch alle nötigen Daten übertragen werden können. Es bieten sich hierbei 3 mögliche Schnittstellen an, welche auf den 2 Konnektoren des Gumstix herausgeführt sind: UART, SPI sowie der Extended Memory Bus.

Die 2 seriellen Schnittstellen (UART und SPI) sind für die Übertragung von Steuerbefehlen gedacht. Aufgrund des seriellen Aufbaus eignen sie sich nicht für die Übermittlung von Bilddateien, welche eine grössere Bandbreite benötigen, als diese Schnittstellen unterstützen.

Um die nötige Bandbreite für eine Bildübertragung zu erhalten, ist ein paralleles Interface nötig. Auf dem Gumstix kann hierfür der Extended Memory Bus genutzt werden, welche für den Anschluss eines zusätzlichen Speicherbausteins genutzt werden kann. In der FPGA kann ein Speicherbaustein implementiert werden, damit ein Datenaustausch durchgeführt werden kann.

Die Kommunikation zur Hauptprozessorplatine, dem OBC, erfolgt über die beiden Gosmoz-Stecker (Elegant Systems Engineering, 2015). Diese Stecker wurden im neuen CanSat-Standard definiert. Es werden dort die „Spring-Loaded-Connectors“ von ELSE verwendet. Die Konnektoren verfügen über Federkontakte, welche den Kontakt zu einer horizontalen Seitenplatine herstellen. Über diese Stecker wird die 5.0V Speisung der Bildverarbeitungskarte zugeführt.

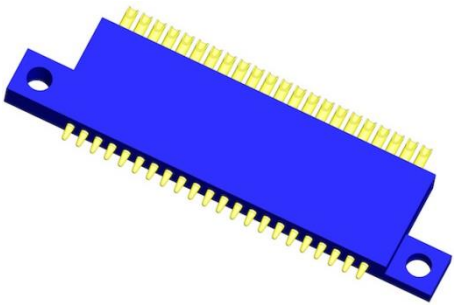


Fig. 4 Gosmoz Stecker (Spring-Loaded-Connector) (ELSE SA, 2015)

Da auf dem „Processing Board“ zusätzlich andere Spannungen benötigt werden, muss ein Power Management auf das Board integriert werden, um die nötigen Spannungen aus der 5.0V Speisung zu generieren.

Das „Processing Board“ verfügt über 4 Anschlüsse für externe Peripherien. Diese Anschlüsse sind direkt auf die FPGA verbunden. Für das Projekt wird nur einer dieser vier Anschlüsse fest verwendet. Es wird dort die Verbindung zur Kamera hergestellt. Für die Kommunikation dient eine serielle Schnittstelle und die Daten werden über ein simples, paralleles Interface übermittelt. Dies wird im Kapitel 4.2: **Kamera** genauer beschrieben.

Die 3 verbleibenden Anschlüsse sind für spätere Erweiterungen vorgesehen. Sie werden bereits auf die FPGA verbunden und können zu einem späteren Zeitpunkt genutzt werden.

Da die Pins der FPGA frei zuweisbar sind, kann später eine beliebige Peripherie angeschlossen werden. Es ist vorgesehen, eine Infrarotkamera und ein LiDAR als Ergänzung anzuschliessen. Auf dem letzten Konnektor sind analoge Eingänge zu finden, welche auf einen ADC verbunden sind. Man verfügt dadurch über die Möglichkeit, analoge Spannungen zu messen. Diese können für den Anschluss von Sensoren oder einer ähnlichen Anwendung genutzt werden.

Da die Datenmenge der Kamera viel höher ist als die von den 3 zusätzlichen Peripherien zusammen, sollte das System auch funktionieren, wenn diese zu einem späteren Zeitpunkt hinzugefügt werden. Für die Übertragungsrate der Kamera wird eine Marge von 100% gewünscht.

3.2. Überblick Kamera

Im Hardware-Blockschema sind 2 Kameras ersichtlich. Zum einen ist dort das „Photonfocus OEM-D1024E“-Kameramodul zu sehen, welches im fertigen Satelliten verbaut wird. Zusätzlich ist das „Omnivision OV7670“-Kameramodul ersichtlich, welches für die Entwicklung der Bildverarbeitungskarte verwendet wird.

Es wird ein alternatives Kameramodul für die Entwicklung verwendet, da das Photonfocus Modul nicht das Preisgünstigste ist. Deshalb kann nicht ein zweites Modul für Testzwecke angeschafft werden. Es wird ein preiswertes Modul verwendet, welches über ein vergleichbares Interface verfügt. Aufgrund dieses ähnlichen Interfaces ist jeweils nur ein kleines Adapterboard notwendig, um das jeweilige Kameramodul auf den vordefinierten Stecker der Bildverarbeitungskarte zu verbinden.

Das genaue Interface sowie alle weiteren Details zu den beiden Kameramodulen werden im Kapitel 4.2: **Kamera** behandelt.

4. Hardware

Dieses Kapitel befasst sich mit der spezifischen Hardware der Bildverarbeitungskarte. Es wird dabei das komplette Schema, die verwendeten Komponenten sowie ihre Spezifikationen behandelt.

4.1. Allgemein

Als Grundlage für die Entwicklung der Bildverarbeitungskarte wurde die Masterarbeit von Bastien Praplan verwendet. Diese Arbeit wurde im Frühling 2013 durchgeführt und es wurde dabei eine Prozessorkarte für Satelliten entwickelt. Es resultierte eine Prozessorkarte sowie eine Karte mit einer FPGA. Die FPGA-Karte ist so aufgebaut, dass sie sehr generell brauchbar ist und somit für verschiedene Projekte eingesetzt werden kann.

Das Schema dieser FPGA-Karte wurde als Grundstein genutzt und entsprechend den Anforderungen der Diplomarbeit abgeändert und erweitert. Das komplette Schema der Bildverarbeitungskarte ist im ***Anhang 2*** zu finden.

4.2. Kamera

Dieser Abschnitt befasst sich mit den Spezifikationen der Kameramodule, dem Interface für die Datenübertragung der Bilder.

4.2.1. Kameramodul Photonfocus

Im Satelliten wird ein Kameramodul (CMOS-Bildsensor) von Photonfocus verwendet. Es handelt sich hierbei um das *OEM-D1024E-160-LC* (Photonfocus, 2015). Dies ist eine Monochromkamera, welche Bilder mit einer Auflösung von 1024x1024 Pixeln aufnimmt (**Fig. 5** *Photonfocus OEM-D1024E*).

Für die Übertragung sollen maximal 20 Bilder/s mit maximaler Auflösung und 12bit/Pixel übertragen werden. Es soll eine Marge von 100% vorgesehen werden.

$$Bandwidth = 20\ images/s * 1024 * 1024\ Pixel * 12\ bit = 240\ Mb/s = 30\ MB/s$$

$$Bandwidth_{with\ Marge} = 2 * Bandwidth = 2 * 240\ Mb/s = 480\ Mb/s = \ 60\ MB/s$$

Es resultiert eine maximale Bandbreite von 60MB/s für das Kamera-Interface.

Dieses Modul ist speziell für Kunden geeignet, welche ihre eigene Hardwarelösung bevorzugen. Das Modul verfügt über einen 80 Pin Konnektor, welcher für den Anschluss der eigenen Hardware genutzt werden kann. Die genaue Pinbelegung ist im (Datenblatt OEM-D1024E, S. 48) zu finden. Das Modul ist mit einem parallelen Interface (12 Bit) für die Datenübertragung sowie einer RS232-Schnittstelle für die Steuerung der Kamera ausgestattet.

Fig. 5 Photonfocus OEM-D1024E (Photonfocus, 2015)

Da für die Entwicklung der Bildverarbeitungskarte dieses Modul nicht zur Verfügung steht, wurde ein alternatives Kameramodul gesucht, welches nicht so teuer in der Anschaffung ist und über ein ähnliches Interface verfügt. So kann die Karte mit diesem alternativen Kameramodul entwickelt werden. Später sind nur kleine Anpassungen notwendig, um das Photonfocus Kameramodul an die Bildverarbeitungskarte anzuschliessen.

4.2.2. Kameramodul Omnivision

Als Alternative zum Kameramodul von Photonfocus wird das OV7670 von Omnivision verwendet. Dieses Modul verfügt über eine VGA-Auflösung (640x480 Pixel @ 30 FPS) und einen stromsparenden CMOS-Bildsensor.

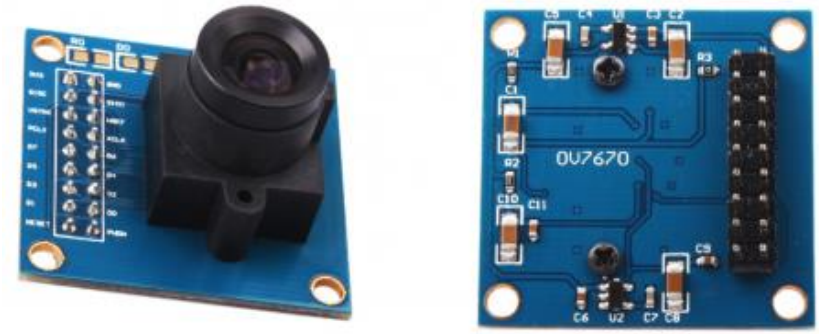


Fig. 6 Omnivision OV7670

Das Interface des Moduls ist sehr ähnlich und somit braucht es nur geringe Anpassungen, um das Kameramodul auszutauschen. Es verfügt ebenfalls über ein paralleles Interface (8 Bit) für die Datenübertragung. Für die Steuerung gibt es den SCCB-Bus von Omnivision, welcher wie I2C definiert ist. Das Kameramodul agiert als Slave.

Das Modul benötigt eine 3.3V Speisung. Die Logikpegel für die Datensignale sind ebenfalls 3.3V. Da das Modul über keinen Quarz verfügt, muss von Extern ein Clock-Signal für das Modul generiert werden. Es wird ein Clock zwischen 10 – 48 MHz benötigt (Datenblatt OV7670, S. 6). Für die meisten Anwendungen wird ein Clock um die 25 MHz verwendet. Dieser kann mit der FPGA erzeugt werden. Das Modul verfügt ausserdem über einen Reset (Activ Low) und einen PowerDown (Activ High).

Auf das genaue Interface der beiden Module wird im folgenden Abschnitt genauer eingegangen.

4.2.3. Kamerainterface

Für die Kommunikation zwischen der FPGA und der Kamera wird das parallele Interface genutzt, über welches beide Kameramodule verfügen. Es handelt sich hierbei um ein standardisiertes Kamerainterface (Camera Inetrface, 2015).

Für das Initialisieren und Steuern der beiden Module wird das jeweilige Zweidraht-Interface verwendet.

Das Kamera-Interface sieht im Allgemeinen wie folgt aus:

- Serielle Schnittstelle: Steuerung der Kamera
- 8-12 Parallel Data Bits: Datenaustausch
- Vertical Sync (VSYNC): Zeigt ein übermitteltes Frame
- Horizontal Sync (HSYNC): Zeigt eine übermittelte Linie
- Pixel Clock (PCLK): Zeigt die einzelnen Pixel

Das komplette Interface, welches für beide Kameras verwendet wird, ist im ***Anhang 3*** aufgeführt.

Dort ist detailliert aufgelistet, wie die jeweiligen Kameras angeschlossen werden müssen, damit diese mit dem festgelegten Stecker kompatibel sind. Es ist dabei ersichtlich, dass die beiden Kameras bis auf die serielle Schnittstelle identisch sind.

Die serielle Schnittstelle, welche direkt auf die FPGA verbunden ist, ändert je nach Modul. Durch den Aufbau der FPGA kann je nach Bedarf die jeweilige Schnittstelle auf den 2 Pins implementiert werden. Somit sind die 2 Kameras kompatibel.

Die Grösse des Parallelbusses beträgt beim Modul von Omnivision nur 8 Bit, weil dort die Auflösung geringer ist. Die 4 nichtgebrauchten Bits werden nicht angeschlossen.

Auf dem „Processing Board“ wird für den Anfang ein gewöhnlicher 24 Pin Steckverbinder verwendet. Somit kann mit einem Flachbandkabel eine Verbindung zwischen der Bildverarbeitungskarte und dem Kamera Interfaceboard hergestellt werden. Später werden „High Reliability“ Stecker von Omnetics (Omnetics Connectors, 2015) verwendet, welche speziell für Anwendungen im Space ausgelegt sind.

4.2.4. Adapterboard OV7670

Während der Diplomarbeit wird ausschliesslich das OV7670 Kameramodul verwendet. Um dieses mit der Bildverarbeitungskarte verbinden zu können, wurde in Adapterboard (**Fig. 7** *Adapterboard OV7670*) erstellt, um das Modul über den definierten Stecker anschliessen zu können. Das Schema zum Adapterboard ist im ***Anhang 4*** zu finden.

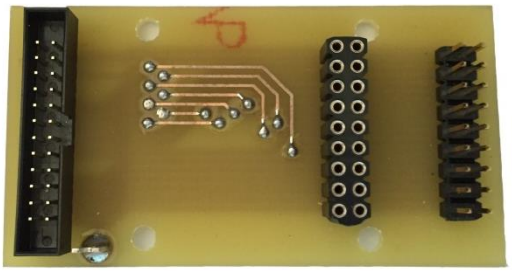


Fig. 7 Adapterboard OV7670

Auf der linken Seite ist der Konnektor ersichtlich, über welchen die Verbindung zur Bildverarbeitungskarte hergestellt werden kann. In der Mitte befindet sich eine Buchsenleiste, auf welche das Kameramodul gesteckt werden kann. Am rechten Rand des PCBs ist ein Pinheader zu finden, auf welchem alle Kamerasignale herausgeführt sind. Damit können diese Signale beim Debugging einfach gemessen werden. In **Fig. 8** *Adapterboard OV7670 mit Kameramodul* ist das Adapterboard mit dem Kameramodul ersichtlich.

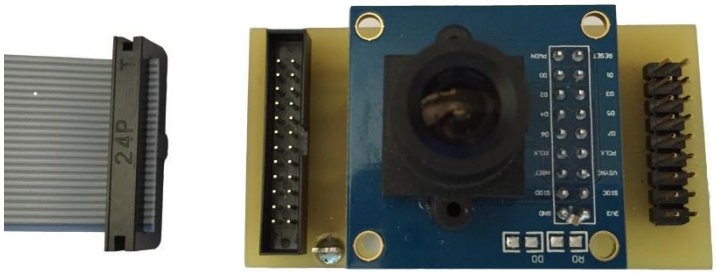


Fig. 8 Adapterboard OV7670 mit Kameramodul

4.3. Gumstix Overo COM

Der folgende Abschnitt befasst sich mit dem Gumstix Overo COM. Es wird das Hosten des Gumstix, die Verbindung zwischen Gumstix und FPGA, die Bildspeicherung und eine mögliche Alternative betrachtet.

4.3.1. Allgemein

Der Gumstix übernimmt auf der Karte die Verarbeitung der Bilder. Dies wird mit Hilfe des DSP und der Library OpenGL gemacht.

Auf dem Gumstix läuft ein Embedded Linux Betriebssystem namens Yocto (Yocto, 2013), welches direkt von der microSD-Karte gebootet wird.

Die Bildbearbeitung auf dem Gumstix wird wie bereits erwähnt von der He-arc Neuchâtel durchgeführt. In diesem Projekt soll primär das Hosten des Gumstix und das Testen der Schnittstellen für die Kommunikation mit der FPGA umgesetzt werden.

Der Gumstix verfügt über zwei 70 Pin Konnektoren, welche auf die Bildverarbeitungskarte verbunden werden. Mittels dieser sind die Übertragung der Bilder und die Kommunikation mit der FPGA möglich.

Der Anschluss des Gumstix ist im Schema der Bildverarbeitungskarte im ***Anhang 2, S. 7*** zu finden.

4.3.2. Yocto-Image

Das Yocto Projekt (Yocto, 2013) erlaubt die individuelle Erstellung einer Linux Distribution für Embedded Systeme.

Auf (Gumstix Yocto Manifest, 2015) findet man eine Anleitung, wie eine solche individuelle Distribution erstellt werden kann. Aufgrund einiger Probleme mit dem SPI Treiber, konnte ich das Image der He-arc Neuchâtel benutzen.

Für die Erstellung des Images ist ein Linux-Computer nötig. Deshalb wurde mit VMware eine virtuelle Maschine mit einem *Ubuntu 14.04* Betriebssystem erstellt. Anhand der Anleitung wurde anschliessend das Yocto-Image auf die microSD-Karte kopiert.

Schlussendlich muss die microSD-Karte in den Gumstix gesteckt und die Speisespannung eingeschaltet werden. Der Gumstix bootet automatisch das Yocto-Image von der microSD-Karte.

Das verwendete Yocto-Image ist im ***Anhang 28*** zu finden.

4.3.3. Hosten

Damit der Gumstix auf der Bildverarbeitungskarte auch funktioniert, müssen einige Grundelemente richtig angeschlossen werden. Für die Kommunikation/Interaktion mit anderen Peripherien, verfügt der Gumstix Overo COM über zwei 70 Pin Konnektoren (**Fig. 9** *Konnektoren Anordnung Gumstix*). Auf diesen Konnektoren sind die verschiedenen Schnittstellen angeschlossen.

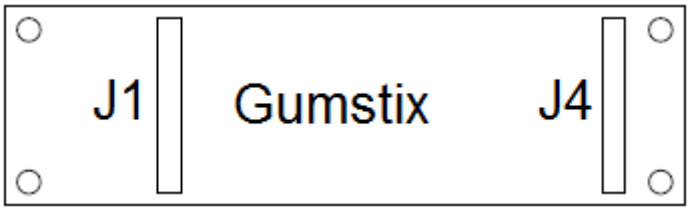


Fig. 9 Konnektoren Anordnung Gumstix

Die Speisung erfolgt über 4 Pins, jeweils 2 pro Konnektoren. Es ist eine Speisespannung von 3.3 - 4.2V nötig, damit der Spannungsregler des Gumstix funktioniert.

Es gilt zu beachten, dass die Logikpegel der I/Os des Gumstix bei 1.8V liegen. Das sollte beim Anschluss einer Peripherie nicht vergessen werden.

Konnektor J1

Auf diesem Konnektor sind ausschliesslich die Steuersignale angesiedelt.

Auf dem Pin 1 befindet sich der N_Manual_Reset, welcher als Hard-Reset genutzt werden kann. Auf diesem Pin werden ein Pull-Up Widerstand sowie ein Druckknopf gegen Masse verbunden. Durch drücken des Knopfes kann der Gumstix neu gestartet werden.

Der UART3 Anschluss des Gumstix ist auf den Pins 26 (TXD3) und 31 (RXD3) zu finden. Dieser Anschluss ist fix für die Verbindung einer seriellen Konsole gedacht. Der Konsolenanschluss kann sehr hilfreich sein fürs Debugging, um mit dem Betriebssystem zu interagieren. Mittels eines FTDI-Bausteins wird das UART-Signal in ein USB-Signal umgewandelt und kann somit über USB an den PC angeschlossen werden.

Auf dem Pin 59 (SYSEN) ist ein Output des Spannungsreglers herausgeführt, welcher mitteilt, ob die geregelte Speisespannung des Gumstix stabil ist oder nicht. Dieser könnte genutzt werden, um sicherzustellen, dass der Gumstix auch betriebsbereit ist.

Konnektor J4

Dieser Konnektor beherbergt die verschiedenen Schnittstellen des Gumstix.

Zum einen ist die SPI1 Schnittstelle herausgeführt. Diese kann für die Kommunikation zwischen Gumstix und FPGA benutzt werden. Die Schnittstelle kann im Master- und Slave-Mode agieren. Die maximale Clock-Frequenz beträgt 48 MHz (**Fig. 10** *Datenblatt DM3730, SPI Master Mode*).

NO.	PARAMETER		OPP100		OPP50		UNIT
			MIN	MAX	MIN	MAX	
SM0	$1/t_{c(CLK)}$	Frequency, mcspix_clk		48		24	MHz

Fig. 10 Datenblatt DM3730, SPI Master Mode (Datenblatt DM3730, S. 226)

Eine zweite UART-Schnittstelle (UART1) ist hier ebenfalls herausgeführt, welche frei genutzt werden kann. Diese eignet sich für die Kommunikation, jedoch nicht für die Datenübertragung.

Schlussendlich gibt es noch eine parallele Schnittstelle, den Extended Memory Bus (GPMC). Dieser ist dafür gedacht, um einen externen Speicher zu erreichen. Aufgrund des parallelen Aufbaus eignet er sich um grosse Datenmengen zu übermitteln. Laut **Fig. 11** *Datenblatt DM3730, GPMC* ist ein maximaler Clock von 100MHz möglich.

NO.	PARAMETER		OPP100		OPP50		UNIT
			MIN	MAX	MIN	MAX	
F0	$1 / t_{c(clk)}$	Frequency(15), output clock gpmc_clk		100		100	MHz

Fig. 11 Datenblatt DM3730, GPMC (Datenblatt DM3730, S. 160)

Mehr Informationen zum GPMC des Gumstix sind im (Datenblatt DM3730, S. 92-94, 160-182) zu finden.

Diese drei Schnittstellen wurden ausgewählt, um zwischen der FPGA und dem Gumstix zu kommunizieren.

Es sind noch weitere Signale/Schnittstellen auf den Konnektoren zu finden, welche hier nicht weiter betrachtet werden.

4.3.4. Interfaces FPGA-Gumstix

Für die Übertragung zwischen FPGA und Gumstix gibt es keine Vorgabe für die Wahl der Schnittstelle. Die Schnittstelle muss in der Lange sein, eine grosse Datenmenge zu übertragen, damit gewährleistet ist, dass die Daten der Kamera mittels der FPGA an den Gumstix übermittelt werden können.

Der Gumstix verfügt über diverse Schnittstellen, welche auf den zwei Konnektoren herausgeführt sind. Unter anderem gibt es dort die zuvor erwähnten Schnittstellen.

Die SPI-Schnittstelle kann mit maximal 48 MHz genutzt werden. Somit können bis zu 6 MB/s übermittelt werden. Diese Datenrate ist für die Bildübermittlung nicht ausreichend, jedoch für die Steuerung und den Austausch von kleinen Datenmengen.

Dasselbe gilt für die UART-Schnittstelle, welche ebenfalls nicht die gewünschten 60 MB/s erreicht.

Somit bleibt der Extended Memory Bus, welcher über einen 8 Bit Adressbus sowie einen 16 Bit Datenbus verfügt. Dieser ist die einzige parallele Schnittstelle, welche herausgeführt ist.

Durch die hohe Clockfrequenz sowie den parallelen Aufbau ist eine hohe Datenrate möglich. Es muss jedoch bedacht werden, dass ein Lese- bzw. Schreibevorgang mehrere Clock-Perioden dauert. Je nach Speicher werden mehr oder weniger Zyklen benötigt. Um genaue Kennwerte zu erhalten, müsste dieses Interface in Betrieb genommen und ausführlich getestet werden. Es müsste dafür ein spezieller Treiber geschrieben werden, was nicht sehr einfach ist.

Durch verwenden dieser drei Schnittstellen, welche für den Datenaustausch zwischen Gumstix und FPGA definiert wurden, stehen mehrere Kommunikationswege zur Verfügung.

Durch die UART- und SPI-Schnittstelle kann der Gumstix getestet werden, falls Probleme auftreten sollten. Über diese Schnittstellen können einfach Daten ausgetauscht werden.

Eine mögliche Umsetzung für die Bildübertragung wäre wie folgt:

In der FPGA wird ein Speicherbaustein implementiert, auf welchen mit dem Extended Memory Bus zugegriffen werden kann. Um die Datenübertragung zu koordinieren, kann ein serielle Schnittstelle (SPI/UART) genutzt werden. Dadurch kann der Ablauf und die Richtung der Übertragung geregelt werden.

4.3.5. Bildspeicherung

Anhand der Steuerbefehle der Hauptprozessorkarte werden Bilder aufgenommen. Diese können direkt oder mit einer Verzögerung an den Hauptprozessor zurückgegeben werden. Deshalb ist es notwendig, dass aufgenommene Bilder abgespeichert werden können.

Der Gumstix verfügt bereits über einen NAND-Flash (512 MB) und ein SDRAM (512MB). Somit sind insgesamt 1GB an Speicher verfügbar. Zusätzlich verfügt er über eine microSD-Karte, auf welcher das Betriebssystem läuft. Es wäre auch möglich, den freien Speicherplatz dieser microSD-Karte zu nutzen.

Es steht somit schon einiges an Speicher zur Verfügung, weshalb nicht noch zusätzlich ein weiterer Speicher auf der Bildverarbeitungskarte vorgesehen wurde. Diese 1 GB sollten ausreichend sein, um eine grosse Anzahl an Bildern zwischenspeichern zu können. Zusätzlich steht der freie Speicherplatz auf der microSD-Karte zur Verfügung.

4.3.6. Alternatives Speichermodul

Da eine FPGA für die Kommunikation mit den Peripherien wie der Kamera verwendet wird, könnte die Bildverarbeitungskarte allgemein dazu verwendet werden, um gegebenenfalls auch mit anderen Peripherien zu interagieren, wobei der Gumstix überflüssig ist.

Über die zwei 70 Pin Konnektoren, welche die Verbindung zum Gumstix herstellen, könnte ebenso eine andere Peripherie, anstelle des Gumstix, auf die Karte verbunden werden.

Ein Speicherbaustein wäre dafür sicher interessant, da ohne Gumstix kein echter Speicherbaustein auf der Bildverarbeitungskarte zu finden ist.

In diversen anderen FPGA-Boards sowie der Vorlage (Masterarbeit) wurde ein SDRAM von Micron verwendet. Dieses hat sich gut bewährt und wäre somit eine gute Wahl für dieses Speichermodul. Es wurde der Micron MT48LC16M16A2 verwendet, welcher über 4 Meg x 16 Bit = 64 Mbit verfügt. Es wurde beachtet, dass genügend Signale von der FPGA auf die zwei Konnektoren verbunden sind, damit ein Speichermodul mit diesem Baustein angeschlossen werden kann. Es sind 38 Signale nötig, um einen solchen Baustein anschliessen zu können. Über 50 Signale sind von der FPGA auf den Gumstix verbunden, wodurch der Anschluss eines solchen Bausteins möglich ist. Eventuell könnte parallel dazu ein weiterer Speicherbaustein mit einem seriellen Interface verbunden werden, um die verbleibenden Signale optimal auszunutzen.

4.4. OBC

Die Verbindung zum OBC, dem Hauptprozessorboard, erfolgt mittels der zwei Spring-Loaded-Connectors. Für die Entwicklung der Karte werden nicht diese Konnektoren verwendet, da eine Verbindung mit den Federkontakten nicht möglich ist. Deshalb wird unterhalb des Steckers ein Pinheader platziert, auf welchem alle Pins parallel zu diesem Stecker herausgeführt sind.

Für die Entwicklung kann ein Pinheader verwendet werden, wodurch alle Signale einfach erreichbar sind. Wird die Karte später in einem CubeSat getestet, kann das Pinheader entfernt und der Spring-Loaded-Connector verbaut werden. Die Logikpegel für diese Signale sind 3.3V.

Der passende Teil des Schemas ist im ***Anhang 2, S. 8*** zu finden.

4.4.1. Interfaces

Die genauen Interfaces, welche für die Kommunikation zwischen der Bildverarbeitungskarte und der Hauptprozessorkarte verwendet werden, sind noch nicht definitiv bekannt. Deshalb wurden für den Moment alle 44 Pins mit I/Os der FPGA verbunden. Sobald die Interfaces definiert sind, können diese in der FPGA hinterlegt werden.

4.4.2. LVDS

Es wurde gewünscht, dass einige der Signale als LVDS (2.5V) herausgeführt werden. Deshalb wurden 10 Pin-Paare auf der FPGA, welche als LVDS-Paare genutzt werden können, dafür vorgesehen.

Damit es mit dem Spacewire-Standard (von der ESA definierter Highspeed-Feldbus für Weltraum Anwendungen) (**Fig. 12** *LVDS Spacewire*) kompatibel ist, müssen noch Serie- und Parallelwiderstände hinzugefügt werden.

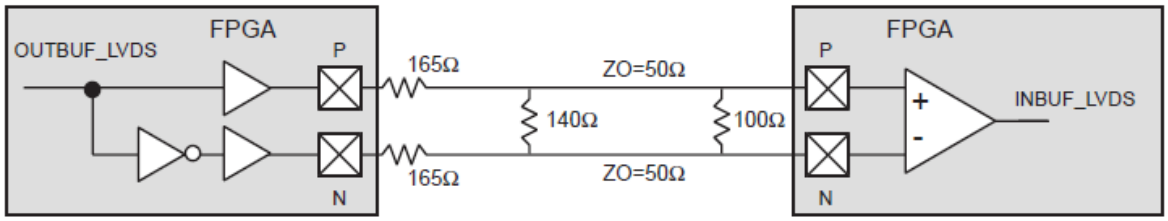


Fig. 12 LVDS Spacewire

Im Falle eines Outputs werden alle drei Widerstände anhand des Standards bestückt. Für einen Input werden die Serienwiderstände mit 0 Ohm und der Parallelwiderstand mit einem 100 Ohm Widerstand versehen.

Werden die Pins ohne LVDS verwendet, müssen die Serienwiderstände mit 0 Ohm Widerständen bestückt werden, während die Parallelwiderstände weggelassen werden.

Bei Verwendung der LVDS-Paare, betragen die Logikpegel 2.5V.

4.5. ADC

Da auf der Bildverarbeitungskarte verschiedene Spannungen zum Einsatz kommen, werden mit Hilfe eines Analog-Digital-Wandlers alle Spannungen gemessen. Damit kann sichergestellt werden, dass die Speisungen richtig funktionieren und keine Gefahr für die damit verbundenen Komponenten besteht.

Der ADC ist im Schema der Bildverarbeitungskarte im ***Anhang 2, S. 10*** zu finden.

Es wird der ADS8028 von Texas Instruments (Datenblatt ADS8028) verwendet. Dieser verfügt über 8 analoge Eingänge, sowie über ein SPI-Interface (Slave) zum Steuern und Auslesen der Daten. Die Speisungen werden fix auf den ADC verbunden und die restlichen analogen Eingänge werden auf einem Konnektor herausgeführt. Dadurch können zusätzliche analoge Spannungen gemessen werden, wie beispielsweise von einem analogen Sensor.

Der ADC (**Fig. 13** *ADS8028*) verfügt für das serielle Interface über eine 3.3V Speisung. Als analoge Referenzspannung dient die 5.0V Speisespannung.

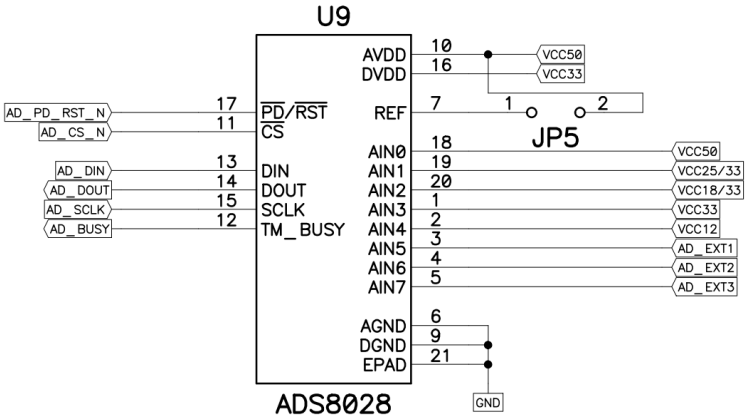


Fig. 13 ADS8028

Im Vorfeld wurde abgeklärt, ob der verwendete ADC mit der Gaisler GRLib kompatibel ist. Die GRLib (GRLIB IP Library User's Manual, 2015) ist eine freie Library, welche in den von der ESA unterstützten Prozessoren-Designs (Leon) verwendet wird. Dadurch wäre die Bildverarbeitungskarte mit diesem Standard kompatibel.

Mit der Gaisler GRLib (GRLIB IP Library User's Manual, 2015, S. 455) kompatible ADCs verfügen nur über einen analogen Eingang und über ein paralleles Interface. Somit ist der ADS8028 nicht kompatibel mit diesem Standard.

Man hielt am ADS8028 fest, da bei diesem über das serielle Interface direkt 8 Eingänge adressiert werden können und dafür nicht 8 einzelne ADCs notwendig sind. Dadurch wird Platz gespart und der Aufwand zum Lesen der Werte wird zusätzlich minimiert.

4.6. Spätere Erweiterungen

Auf den nichtbenutzen Pins der FPGA werden zusätzlich zwei Konnektoren für externe Peripherien angeschlossen. Dafür werden 24 Pin Konnektoren eingesetzt, wobei zwei Pins für die Speisung (3.3V) genutzt werden. Somit sind je 22 Pins für Signale vorhanden.

Zu einem späteren Zeitpunkt ist geplant, eine Infrarotkamera und ein LiDAR als Erweiterung auf die Bildverarbeitungskarte zu verbinden. Dies war eine Idee, welche zu Beginn der Planung gewünscht wurde. Was schlussendlich angeschlossen wird, steht noch offen. Durch die direkte Verbindung auf die FPGA können jedoch fast alle möglichen Peripherien angeschlossen werden.

Die genaue Beschaltung der beiden Konnektoren ist im Schema der Bildverarbeitungskarte im ***Anhang 2, S. 11*** zu finden.

4.7. Power

Um die diversen Peripherien auf der Bildverarbeitungskarte zu speisen, ist ein Power Management nötig. Als Spannungsinput dient eine einzelne 5.0V Speisung, welche für die Erzeugung aller anderen Spannungen verwendet wird. Für die Umwandlung der einzelnen Spannungen werden DC/DC-Wandler verwendet, welche effizienter sind als Spannungsregler.

Folgende Spannungen sind für die jeweiligen Komponenten notwendig:

Komponent	Spannung	Konverter/Regler	Grund
FPGA	3.3 V	DC/DC1	GPIO
	2.5 V/3.3V	DC/DC2	OBC (2.5V LVDS)
	1.8 V/3.3V	DC/DC3	Kommunikation Gumstix
	1.2 V	DC/DC4	Core
Gumstix	3.3 V	DC/DC1	Speisung
Kamera Modul "Photonfocus"	5.0 V 3.3 V 1.8 V	Separate Speisung direkt auf Kameramodul	
Kamera Modul "OV7670"	3.3 V	DCD/DC1	Speisung
ADC	5.0 V	Speisespannung	Analoge Referenzspannung
	3.3 V	DC/DC1	Digitales Interface
SPI Flash	3.3V	DC/DC1	Speisung
OBC	2.5V/3.3V	DC/DC2	Speisung
Peripherie 1	3.3V	DC/DC1	Speisung
Peripherie 2	3.3V	DC/DC1	Speisung

Legende:

DC/DC1: 3.3V **DC/DC3:** 1.8V/3.3V

DC/DC2: 2.5V/3.3V **DC/DC4:** 1.2V

Tab. 1 Übersicht Spannungen

In **Tab. 1** *Übersicht Spannungen* ist ersichtlich, dass 4 DC/DC-Wandler notwendig sind. Diese werden in den folgenden Abschnitten genauer betrachtet:

DC/DC1: 3.3V

Die 3.3V Speisung bildet die allgemeine Speisung für alle Komponenten, welche über eine fixe 3.3V Speisung verfügen. Dazu zählen der Gumstix, das Kameramodul OV7670, der ADC, der SPI-Flash sowie die externen Peripherien.

DC/DC2: 2.5V/3.3V

Diese Speisung wird einzig für die Kommunikation mit der Hauptprozessorplatine benötigt. Es werden die Pins gespiesen, welche zum OBC-Konnektor verbunden sind. Werden die LVDS-Pins eingesetzt, müssen 2.5V-Pegel verwendet werden. Sind die Pins als normale I/Os definiert, werden 3.3V-Pegel benötigt. Diese Speisung muss anhand der gegebenen Anwendung regelbar sein.

DC/DC3: 1.8V/3.3V

Die dritte Speisung ist für die Kommunikation zwischen FPGA und Gumstix gedacht. Wird der Gumstix auf die Bildverarbeitungskarte verbunden, müssen die Logikpegel 1.8V betragen. Dies ist beim Gumstix so definiert. Sollte das alternative Speichermodul verbunden werden, müssten die Logikpegel 3.3V betragen. Auch hier muss die Speisung anhand der Anwendung regelbar sein.

DC/DC4: 1.2V

Die letzte Speisung wird für den Core der FPGA benötigt. Dieser braucht eine fixe 1.2V Speisung.

Für die DC/DC1-3 wird der SC196 (Datenblatt SC196, 2007) verwendet, welcher ein regelbarer DC/DC-Wandler ist. Über einen Spannungsteiler kann die Ausgangsspannung angepasst werden. Für den DC/DC4 wird der SC189 (Datenblatt SC189, 2010) genutzt, der über eine fixe Ausgangspannung verfügt. Die beiden Wandler stammen aus derselben Serie, mit dem Unterschied, dass einer fix und der andere regelbar ist. Es gilt zu erwähnen, dass beide Wandler über einen integrierten Kurzschluss-Schutz verfügen, welcher als Schutz für die angeschlossenen Komponenten dient.

Am Ausgang jedes Wandlers befindet sich ein Jumper, um gegebenenfalls den Stromverbrauch eines Wandlers messen zu können.

Die genauen Berechnungen der Widerstände für die regelbaren DC/DC-Wandler sind im Schema aufgeführt. Man beachte, dass die Summe von $R_{Feedback1}$ und $R_{Feedback2}$ (**Fig. 14** *DC/DC-Wandler regelbar*, R5 & R6) zwischen 10k-1MOhm liegen sollte (Datenblatt SC196, 2007, S. 7). $R_{Feedback1}$ ist fix und $R_{Feedback2}$ kann anhand der gewünschten Spannung angepasst werden. Die Wahl der Widerstände ist im ***Anhang 5*** ersichtlich.

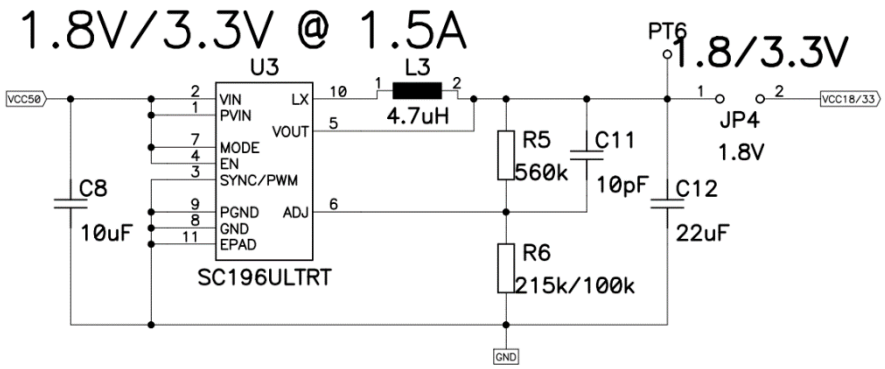


Fig. 14 DC/DC-Wandler regelbar

Die Beschaltung inklusive Berechnung der vier DC/DC-Wandler ist im Schema der Bildverarbeitungskarte im ***Anhang 2, S. 2*** zu finden.

Der ADC misst alle Spannungen der DC/DC-Wandler. Es bestände die Möglichkeit, anhand dieser Messungen die einzelnen Speisungen bei falscher Ausgangsspannung auszuschalten. Es wäre eine Steuerung nötig, welche auf der Karte nur die FPGA übernehmen kann. Dies ist nicht realisierbar, da die FPGA selber von diesen Speisungen versorgt wird.

Um diesen zusätzlichen Schutzmechanismus einzubauen, könnte ein kleiner Kontroller hinzugefügt werden, welcher als Sicherheitsbaustein fungiert und mit Hilfe des ADC die Speisungen notfalls ausschalten kann.

Auf dem ersten Prototyp sind die DC/DC-Wandler immer aktiv und es wurde auf diesen zusätzlichen Schutz verzichtet.

4.8. FPGA

Der folgende Abschnitt befasst sich mit den Spezifikationen der FPGA sowie der Beschaltung der jeweiligen Banken.

Die komplette Beschaltung der FPGA ist im Schema der Bildverarbeitungskarte im ***Anhang 2, S. 3-6*** zu finden.

4.8.1. Allgemein

Die FPGA bildet auf der Bildverarbeitungskarte die zentrale Steuereinheit. Gearbeitet wird mit einer FPGA von Xilinx aus der Spartan6-Familie. Zu dieser Familie gehören die Modelle LX45, LX100 & LX150, welche bis auf einige vereinzelte Pins pinkompatibel sind. Deshalb kann je nach Anforderungen der Karte eine passende Variante gewählt werden. Für den ersten Prototyp wird die LX100 verwendet, welche nur geringfügig schwächer als die LX150 ist. Die FPGA wird im Gehäuse FGG484 benutzt. Dieses verfügt über 484 Pins mit 326 brauchbaren User I/Os. Die FPGA ist in 4 Banken aufgeteilt, wobei jede Bank ihre eigene Spannungsversorgung hat. Somit kann mit Peripherien interagiert werden, welche über unterschiedliche Logikpegel verfügen.

4.8.2. Config

Ein Bitfile (.bit), welches mit dem ISE Project Navigator erstellt wurde, wird mittels JTAG auf die FPGA programmiert. Das Bitfile enthält die komplette Konfiguration der FPGA. Auf der Bildverarbeitungskarte ist ein FPGA_Done-LED zu finden, welches leuchtet, wenn die FPGA aktiv ist. Während der Programmierung oder wenn die Konfiguration aus dem Flash geladen wird, ist dieses LED inaktiv. Somit ist genau bekannt, in welchem Zustand sich die FPGA befindet.

Mit dem FPGA-Hardreset kann ein Neustart sowie das Laden der gespeicherten Konfiguration aus dem Flash erzwungen werden. Wird die Bildverarbeitungskarte unter Spannung gesetzt, lädt die FPGA ebenfalls die Konfiguration aus dem Flash.

Auf der Karte befindet sich kein Softreset, da bei einer späteren Anwendung im Satelliten niemand ein Software-Reset auslösen kann. Der Softreset würde alle Flipflops und Speicher zurücksetzen.

Als Clock-Input wird ein 100 MHz Quarz verwendet. Es wurde diese Frequenz gewählt, um sicherzustellen, dass die Clockfrequenz doppelt so hoch wie alle anderen Frequenzen ist. Dies ist notwendig, damit alle Signaländerungen erfasst werden können.

4.8.3. Flash (Konfiguration)

Die Konfiguration der FPGA ist nur solange vorhanden, wie die FPGA unter Spannung ist. Dies hat zur Folge, dass die FPGA theoretisch jedes Mal neu programmiert werden muss, nachdem die Speisespannung verbunden wird.
Als Gegenmassnahme wurde ein SPI-Flash auf der Karte untergebracht, um die Konfiguration zwischen zu speichern. Sobald die Speisespannung eingeschaltet ist, wird die Konfiguration automatisch aus dem SPI-Flash in die FPGA geladen.

Für den SPI-Flash muss ein spezielles File (.mcs) erstellt werden, welches die Konfiguration enthält. Dieses File wird mit Hilfe des ISE Project Navigator aus dem Bitfile erstellt.

Bei der Wahl des Flashs muss beachtet werden, dass die Grösse des Konfigurationsfiles je nach Typ der FPGA variiert. Für die grösste Ausführung, den LX150, welche auf der Bildverarbeitungskarte genutzt werden kann, sind somit 33‘909‘664 Bits notwendig.

Device	Total Number of Configuration Bits(1)
6SLX45	11,939,296
6SLX45T	11,939,296
6SLX75	19,719,712
6SLX75T	19,719,712
6SLX100	26,691,232
6SLX100T	26,691,232
6SLX150	33,909,664
6SLX150T	33,909,664

Fig. 15 Bitstream Länge (Spartan6 FPGA Configuration, 2014, S. 76)

Die Wahl eines 32Mbit Flash wäre riskant. Deshalb wurde ein 64Mbit Flash ausgewählt, um auf der sicheren Seite zu sein. Die Wahl fiel auf den M26P64, welcher sehr einfach aufgebaut und sehr preisgünstig ist.

In **Fig. 16** *SPI-Flash* ist die Beschaltung des SPI-Flashs zu sehen.

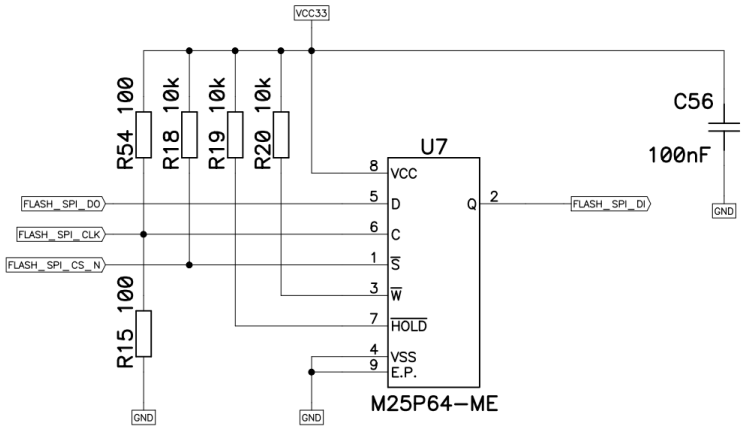


Fig. 16 SPI-Flash

4.8.4. Banken 0-3

Die drei Modelle der Spartan6-Familie, welche für die Karte vorgesehen sind, haben alle denselben Aufbau. Sie können grundsätzlich in 5 Teile aufgeteilt werden: Banken 0-3 und Core.

Die Banken 1-3 verfügen jeweils über rund 100 nutzbare I/Os. Einzig die Bank 0 verfügt über weniger I/Os, nämlich deren 56.

Für die Aufteilung der Peripherien auf die jeweiligen Banken mussten die verschiedenen Speisespannungen in Betracht gezogen und auf die Banken verteilt werden, damit alle Komponenten angeschlossen werden konnten.

Für das OBC wurden LVDS-Pin-Paare gewünscht. LVDS-Inputs können auf allen Banken implementiert werden, deren Outputs jedoch nur auf den Banken 0 & 2 (Spartan6 Select IO Resources, 2014, S. 41). Auf der Bank 0 gab es nicht ausreichend LVDS-Paare, welche sich auf den äusseren Pinreihen der FPGA befanden, weshalb für den Anschluss des OBC die Bank 2 gewählt wurde. Dort besteht die Möglichkeit, 10 LVDS-Paare herauszuführen. Die restlichen Signale des OBC werden auf I/Os der FPGA verbunden. Die Bank 2 wird mit 2.5V/3.3V gespiesen.

Die Bank 0 wurde für den Anschluss des Kameramoduls und des ADC verwendet. Diese beiden Peripherien verfügen über eine 3.3V Speisung, welche ebenfalls für die Bank genutzt wird.

Der Gumstix mit seinen drei Kommunikationsschnittstellen wurde auf die Bank 1 verbunden. Aufgrund der definierten Logikpegel des Gumstix wird diese Bank mit 1.8V/3.3V gespiesen.

Auf der verbleibenden Bank 3 werden die zwei zukünftigen Peripherien angeschlossen. Diese Bank verfügt über eine 3.3V Speisung.

Auf den Core der FPGA wird die 1.2V Speisung verbunden.

Der JTAG, der Hardreset, die FPGA_Done-LED sowie der SPI-Flash werden auf die speziell dafür vorgesehenen Pins auf der FPGA verbunden. Diese sind vom Hersteller fix vordefiniert.

Die Aufteilung der FPGA ist in **Fig. 17** *FPGA Aufbau* vereinfacht dargestellt.

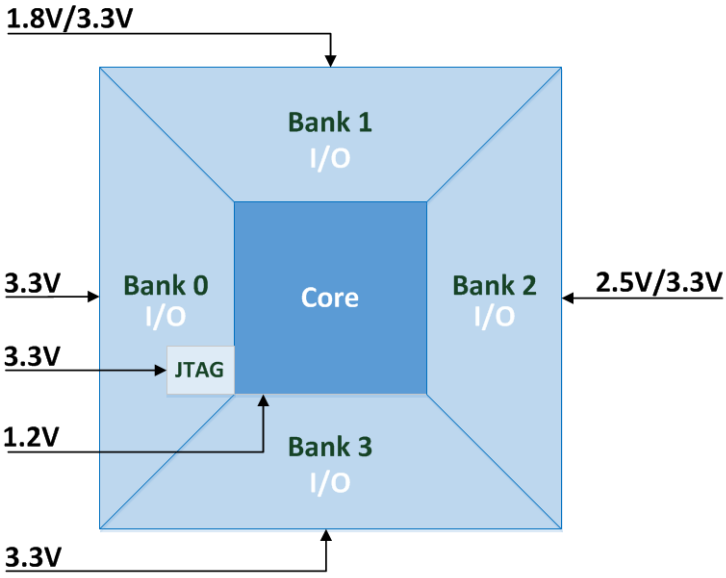


Fig. 17 FPGA Aufbau

Der Clock sowie alle anderen Clock-Signale (SPI-Clock, Pixelclock der Kamera etc.) werden auf die GCLK-Pins geführt. Auf jeder Bank sind einige dieser speziellen Pins vorhanden. Diese Pins sind extra für Clock-Signale gedacht, weil sie fest in der FPGA verbunden sind.

Für den SPI-Flash mussten noch 2 Pins (M[1:0]) wie folgt gesetzt werden:

Mode	M[1]	M[0]
SPI	0	1

Tab. 2 SPI Konfiguration

Es existieren noch andere Anschlussmöglichkeiten für einen Flash, welche hier nicht weiter behandelt werden.

Die komplette Aufteilung der Banken ist im Schema der Bildverarbeitungskarte im ***Anhang 2, S. 3-5*** zu finden.

5. PCB

Das folgende Kapitel befasst sich mit der Struktur des CubeSat Satelliten unter Berücksichtigung des neuen Standards von ELSE, den Abmessungen der Platinen und der Anordnung der Komponenten auf dem PCB.

5.1. xU-Struktur von ELSE

Die Firma ELSE SA hat die innovative xU-Struktur (**Fig. 18** *xU Struktur*) für CubeSats entwickelt. Ziel dieser Struktur ist es, den CubeSat modularer, anpassbarer, skalierbarer und zugänglicher zu machen.

Durch das Einfügen von einzelnen Lagen (Slices), welche über einen Eckpunkt herausgedreht werden können, ist jede Lage einfach erreichbar und auswechselbar. Ein PCB wird mit Hilfe einiger Schrauben auf einer solchen Halterung festgemacht.

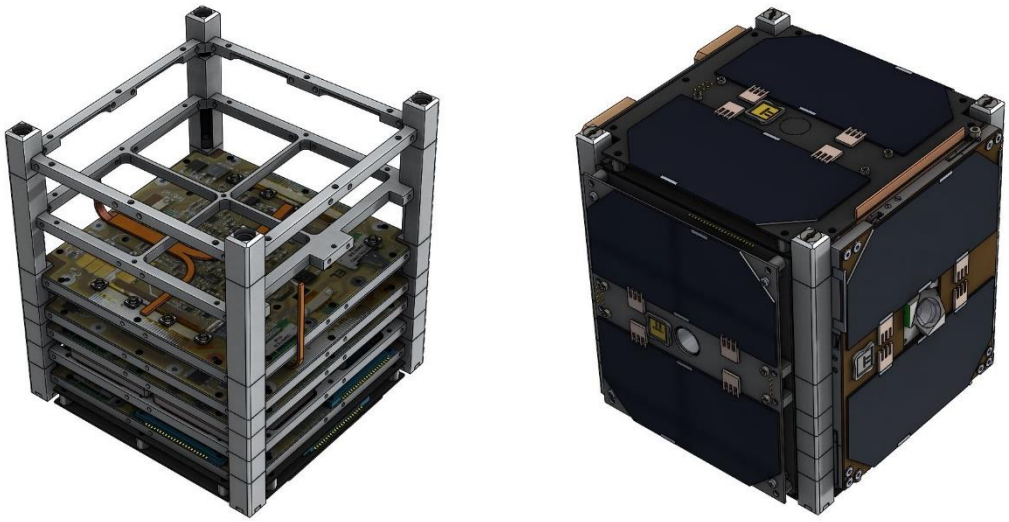


Fig. 18 xU Struktur (ELSE SA, 2015)

Mittels den „Spring-Loaded-Connectors“ wird die Verbindung der horizontalen PCB zu den vertikalen Verbindungs-PCBs hergestellt, welche auf den Aussenseiten des CubSats montiert sind.

Für den Anfang werden diese Konnektoren nicht genutzt, solange das PCB nicht in einem CubeSat eingebaut ist. Es wäre zu kompliziert auf die Konnektoren eine Verbindung herzustellen, da diese nicht einrasten, sondern über Druckkontakte verfügen.

Unter den Spring-Loaded-Connectors wird ein Pinheader platziert, auf welchem die gleichen Signale herausgeführt sind. Während der Entwicklung der Bildverarbeitungskarte kann das Pinheader genutzt werden, welches sehr einfach zugänglich für Messungen ist.

Betrachtet man eine einzelne Lage der Struktur (**Fig. 19** *Ein Slice der xU-Struktur*), ist ersichtlich, dass die Verbindungen zwischen den Eckpunkten individuell anpassbar sind. Die Verstrebungen können zwischen 4 - 12.5mm angeordnet werden. Ein PCB wird auf einer solchen Halterung montiert.

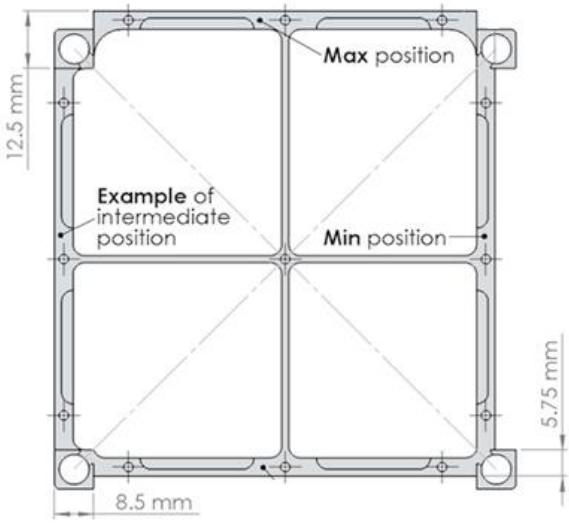


Fig. 19 Ein Slice der xU-Struktur (ELSE SA, 2015)

Für die Bildverarbeitungskarte werden die Dimensionen des PCB wie beim CubETH (95x95 mm) gewählt. Die genauen Abmessungen sind im ***Anhang 6*** zu finden.

5.2. PCB Planung

Im Gegensatz zu früheren Strukturen stehen in der neuen xU-Struktur nicht die kompletten 10x10 cm für das PCB zur Verfügung. Es entsteht ein Platzverlust, da jedes PCB auf eine Halterung montiert wird.

Um den Platzbedarf aller Komponenten der Bildverarbeitungskarte abzuschätzen, wurde mit Hilfe von Inkscape eine Skizze (**Fig. 20** *Skizze PCB*) erstellt, welche eine mögliche Aufteilung und Anordnung der wichtigsten Komponenten auf dem PCB veranschaulicht.

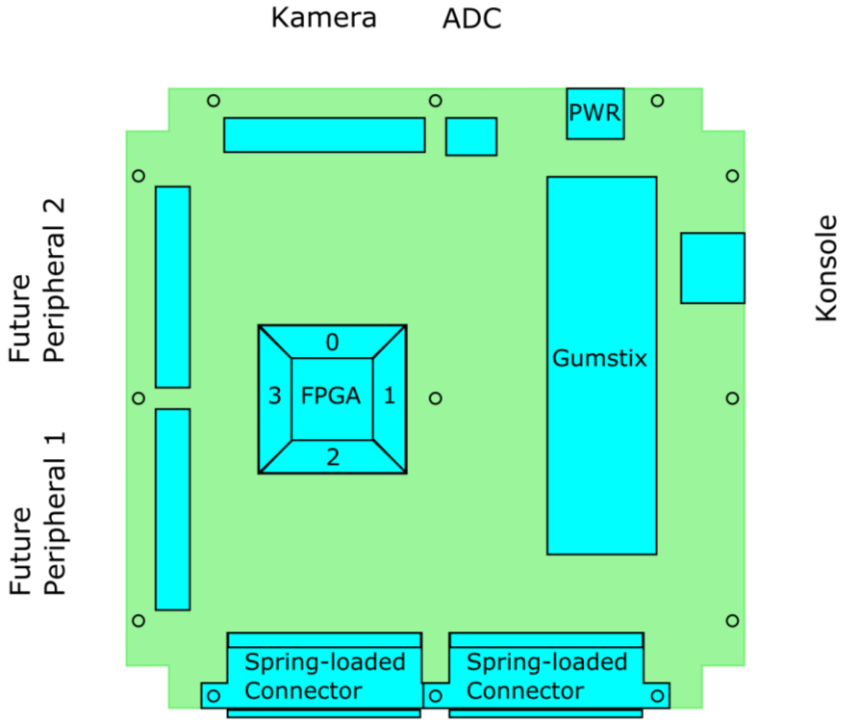


Fig. 20 Skizze PCB

Es wurde bei der Planung darauf geachtet, dass die jeweiligen Komponenten einfach auf die ihnen zugewiesene Bank der FPGA verbunden werden konnten.

Bank 0: Kamera & ADC
Bank 1: Gumstix
Bank 2: OBC (Spring-Loaded-Connectors)
Bank 3: Weitere Peripherien

Weitere Komponenten wie beispielsweise die Speisungen wurden auf dem freien Platz um die FPGA angeordnet. Auf der Unterseite des PCB befinden sich ausschliesslich kleine SMD-Komponenten wie Kondensatoren und Widerstände.

Für die Speisungen sowie den Clock wurden auf dem Board SMD-Testpunkte platziert, damit diese schnell und einfach gemessen werden können.

5.3. PCB Routing

Anhand des Schemas und der Planung für die Anordnung der Komponenten wurde des PCB von einem erfahrenen Elektroniker der Hes-so Wallis geroutet.

Aufgrund der Komplexität, welche die Bildverarbeitungskarte aufweist, entspricht dies einem grossen Zeitaufwand und das Routing beanspruchte 1,5 - 2 Wochen.

Das komplette Layout sowie die Pläne zum Bestücken der einzelnen Komponenten sind im ***Anhang 7*** zu finden.

Das PCB wurde von der Euro Circuit GmbH (Euro Circuit GmbH, 2015) angefertigt.

6. Komponenten

Um den Überblick über alle verwendeten Komponenten sowie ihre Kenndaten zu bewahren, wurde eine Excel-Tabelle erstellt, wo alle nützlichen Informationen aufgelistet sind.

Die komplette Komponentenliste mit Spezifikationen, Distributor, Preis usw. ist im ***Anhang 8*** zu finden.

Die Excel-Tabelle ist ebenfalls im digitalen Anhang vorhanden. Es ist dort eine Berechnung (**Fig. 21** *Berechnungstool*) eingebaut, mit welcher die Kosten für die Erstellung von einer/mehrerer Bildverarbeitungskarten berechnet werden kann.

Part	Quantity	Price/Piece	Price Tot.
Board Components	3	€ 270.04	€ 810.12
Gumstix	1	€ 128.98	€ 128.98
PCB Manufacturing (1 Piece) (Eurocircuit)	7	€ 20.00	€ 140.00
PCB Order Tax (Eurocircuit)			€ 200.00
		€	**1'279.10**

Fig. 21 Berechnungstool

7. Hardwarestruktur

In diesem Kapitel wird die Hardwarestruktur betrachtet, welche für die FPGA erstellt wurde. Sie diente dazu, sicherzustellen, dass die Bildverarbeitungskarte vollumfänglich funktioniert.

7.1. Struktur

Um alle Komponenten auf der Bildverarbeitungskarte testen zu können, wurde eine Hardwarestruktur erstellt, welche einen solchen Test ermöglicht.

Die Hardwarestruktur kann grundsätzlich in 3 Teile aufgeteilt werden: Kamera, ADC und Gumstix.

Kamera
Bei Startup wird das Kameramodul anhand eines Konfigurationsfile initialisiert. Anschliessend liefert das Modul in regelmässigen Abständen neue Bilder über das Kamerainterface. Diese werden in einem RAM zwischengespeichert. Das letzte Bild bleibt solange gespeichert, bis ein neues Bild empfangen wird.

ADC
Regelmässig wird der ADC mittels SPI abgefragt und die neuen Messwerte in einem RAM abgespeichert.

Gumstix
Die FPGA kann über SPI Nachrichten vom Gumstix erhalten. Da der Gumstix als Master agiert, muss die FPGA als Slave arbeiten.

Den Kern der Struktur bildet der Kontroller. Dieser kann die erhaltenen Nachrichten vom Gumstix per UART an den PC senden. Ausserdem kann der Kontroller vom PC Befehle erhalten und je nach Befehl das RAM der Kamera oder des ADCs auslesen.

Diese Struktur ist in **Fig. 22** *Blockschaltbild Hardwarestruktur* ersichtlich.

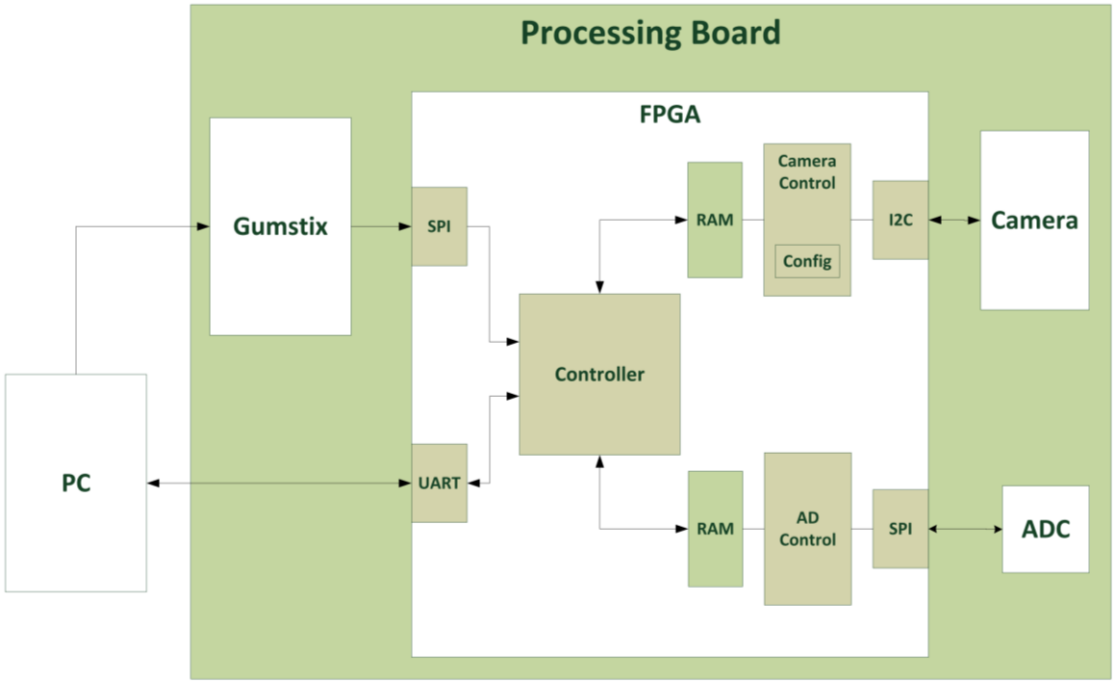


Fig. 22 Blockschaltbild Hardwarestruktur

7.2. Top-Level

Das Top-Level wird dafür genutzt alle Signale in die gewünschte Form zu bringen, um sie ohne Probleme im Design zu verwenden.

Die komplette Struktur des Top-Levels ist im ***Anhang 9*** zu finden.

Eingangssignale werden mit dem FPGA-Clock synchronisiert, um Metastabilitäten zu vermeiden. Es können jedoch nur Eingänge synchronisiert werden, bei welchen die maximale Änderungsfrequenz mindestens zweimal kleiner ist als diejenige der Clock-Frequenz. Idealerweise noch tiefer. Anderenfalls tritt das Problem auf, dass Signaländerungen auf den Inputs verpasst werden können.

Der Software-Reset wird ebenfalls synchronisiert. Da auf dem Board kein Software-Reset vorhanden ist, wurde ein Pseudo-Reset mit einem nicht angeschlossenen Pin erzeugt. Es wurde der Pin „D10“ der Bank 0 verwendet. Dies ist notwendig, da ansonsten nie ein Reset in irgendeinem Block ausgeführt wird. Somit bestände das Problem, dass nach dem Startup die FlipFlops nicht klar definiert sind.

Ein Tristate-Pin ist bidirektional, das heisst, es können mehrere Outputs auf dasselbe Signal verbunden sein. Deshalb wird ein solcher Pin für die tiefere Schicht der Struktur in einen Input und einen Output aufgeteilt.

Für das Kamerainterface ist dies auf der Datenleitung des I2C der Fall (**Fig. 23** *Tristate Pin*).

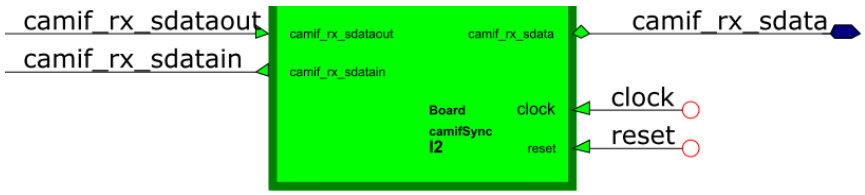


Fig. 23 Tristate Pin

Das Signal *camif_rx_sdata* wird hier in einen Input und einen Output aufgeteilt. Der Input entspricht immer gleich diesem Signal. Der Ausgang wird auf ‘0‘ gezogen, wenn der Output ‘0‘ ist, ansonsten ist er Hochohmig. Das setzt voraus, dass auf der Leitung ein Pull-Up Widerstand die Leitung standartmässig auf ‘1‘ zieht.

Im Top-Level werden ausserdem die globalen Deklarationen definiert.

In den Pre User Deklarationen (**Fig. 24** *Pre User Declarations*) werden die Geschwindigkeiten der verschiedenen Schnittstellen sowie die Grösse des Kamerabildes definiert.

Declarations

Pre User:

```
constant clockFrequency: positive := 100E6;
constant camClockFrequency: positive := 24E6;
constant baudrate: positive := 256000;
constant i2cClockFrequency: positive := 400E3;
constant spiClockFrequency: positive := 125E5;
constant pictureW: positive := 640;
constant pictureH: positive := 480;
```

Fig. 24 Pre User Declarations

Globale Deklarationen, welche schon im Top-Level verwendet werden (z.B. die Grösse eines Datenbusses), können nicht in den Pre User Deklarationen festgelegt werden, da diese erst nach den Entitys definiert werden. Deshalb befinden sich diese in den Generics, welche vor den Entitys definiert werden.

Die Signale des Hardwaredesigns müssen noch auf die Pins der FPGA zugewiesen werden. Diese Zuweisung erfolgt mit dem ucf-File.

Um diese Zuweisung einfach zu halten, wurden im Hardwaredesign die Top-Level-Signale gleich benannt wie im Schema (***Anhang 2, S. 4-5***).

Anhand des Schemas der Bildverarbeitungskarte konnte die Zuweisung recht einfach durchgeführt werden, da im Schema die Signale jeweils mit ihrem Ausgang der FPGA vorhanden sind.

Im ucf-File werden zusätzlich die speziellen Eigenschaften eines Pins deklariert. Beispielsweise für die I2C-Pins wurden hier die Pull-Up Widerstände hinzugefügt, aufgrund der Tristate Anwendung des Pins.

Für den Pseudo-Softreset wurde ebenfalls ein Pull-Up verwendet, da der Reset *activ Low* ist. Nach der erfolgreichen Programmierung der FPGA sind kurzzeitig alle Pins auf ‘0‘, wodurch der Reset aktiv wird und alle FlipFlops zurückgesetzt werden.

Für den Clock wird ausserdem eine TIMESPEC erstellt. Dieser wird beim Synthetisieren verwendet, um Durchlaufzeiten eines Signals zu berechnen.

Bei der Zuweisung von seriellen Schnittstellen, wie beispielsweise UART, muss darauf geachtet werden, dass nicht das TX-Signal der FPGA auf das TX der angeschlossenen Peripherie verbunden wird. Es muss eine Kreuzung, TX zu RX und RX zu TX definiert werden.

Das ucf-File für die Bildverarbeitungskarte ist im ***Anhang 10*** hinterlegt.

7.3. ProcessingBoard

Das ProcessingBoard wiederspiegelt den FPGA-Teil des Blockschaltbild der Hardwarestruktur (**Fig. 22** *Blockschaltbild Hardwarestruktur*).

Das grafische Design des ProcessingBoard ist im ***Anhang 11*** zu finden.

Die UART-Schnittstelle des PC wird auf die Signale *testRX* und *testTX* verbunden. Der serialPortFIFO <U_7> empfängt auf dem RxD die Befehle des PC. Die empfangenen Daten werden in einem FIFO zwischengespeichert. Solange *rxEmpty* = '1' ist, sind keine Daten empfangen worden (serialPortFIFO aus RS232-Library genutzt).

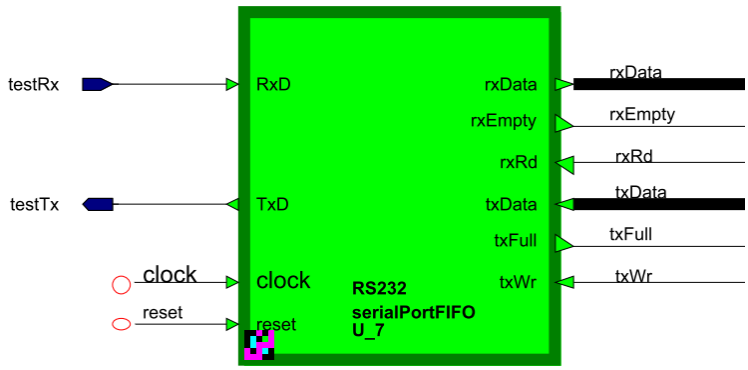


Fig. 25 serialPortFIFO

Nach dem gleichen Prinzip funktioniert der slaveSPI<U_11>. Als Input erhält er die SPI Signale des Gumstix (SPI1, CS0) und speichert empfangene Daten in einem internen FIFO.

Solange der interne FIFO leer ist, ist das Signal *sEmpty* = '1'. Sobald etwas empfangen und im FIFO abgespeichert wurde, wechselt das Signal auf '0'. Der FIFO könnte mit dem Signal *sRead* = '1' gelesen werden und dessen Inhalt würde auf den *sData*-Bus gelegt.

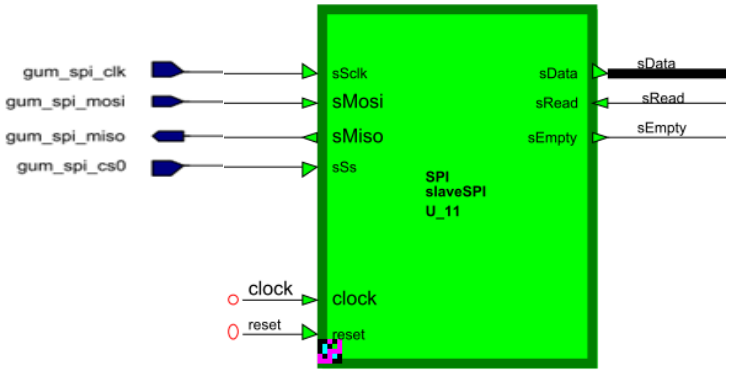


Fig. 26 slaveSPI

Der BoardController<U_2> befindet sich in einem Wartezustand, bis er auf einem dieser zwei Flags feststellt, dass etwas empfangen wurde. Er verlässt dann den Wartezustand, liest die Daten aus und verarbeitet sie.

Daten, welche vom Gumstix über SPI empfangen werden, liest er heraus und leitet sie direkt über UART an den PC weiter.

Wenn Daten über UART vom PC empfangen wurden, wird überprüft, ob sie mit den definierten Befehlen für die Kamera oder den ADC übereinstimmen. Trifft dies zu, werden die neusten Daten aus dem jeweiligen Block-RAM gelesen und über UART an den PC übermittelt.
Die genaue Funktionswiese des BoardController wird in einem späteren Teilkapitel:

Board Controller (Kontroller) detailliert betrachtet.

Parallel zum boardController laufen der camTest<U_9> und der ADS8028<U_3> Block.

Der camTest<U_9> steuert über I2C das Kameramodul und ist für die Aufnahme der Bilder über das parallele Interface zuständig. In regelmässigen Abständen wird ein neues Bild im Block-RAM<U_1> abgespeichert. Die komplette Funktionsweise wird im nächsten Teilkapitel: **camTest (Camera Control)** behandelt.

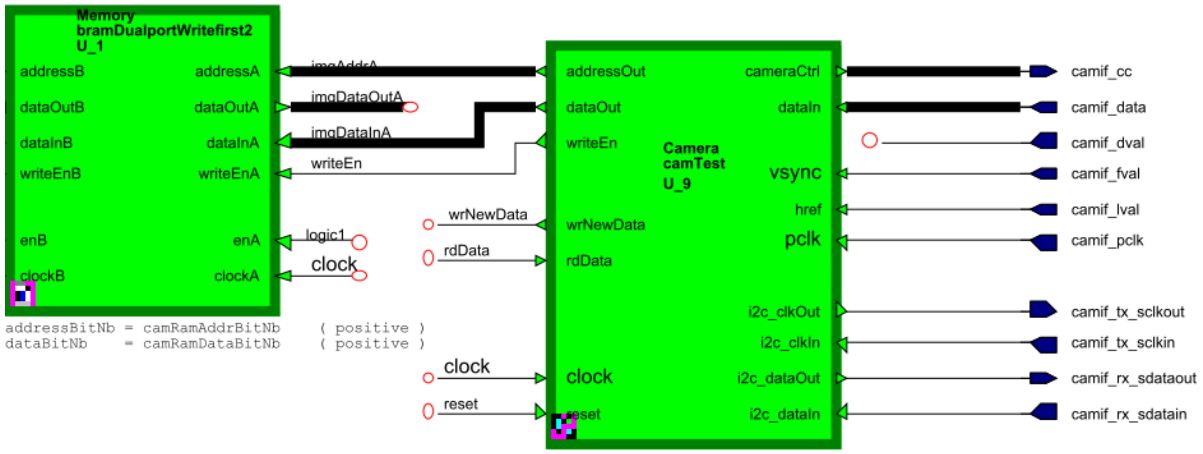


Fig. 27 Kamerainterface

Mit dem ADS8028<U_3> wird der AD-Wandler über SPI gesteuert. In einem definierten Zeitintervall liest er die Inputs des Wandlers und speichert die gelesenen Spannungswerte ins Block-RAM<U_4>. Der Aufbau wird im übernächsten Teilkapitel:

ADS8028 (ADC Control) genau betrachtet.

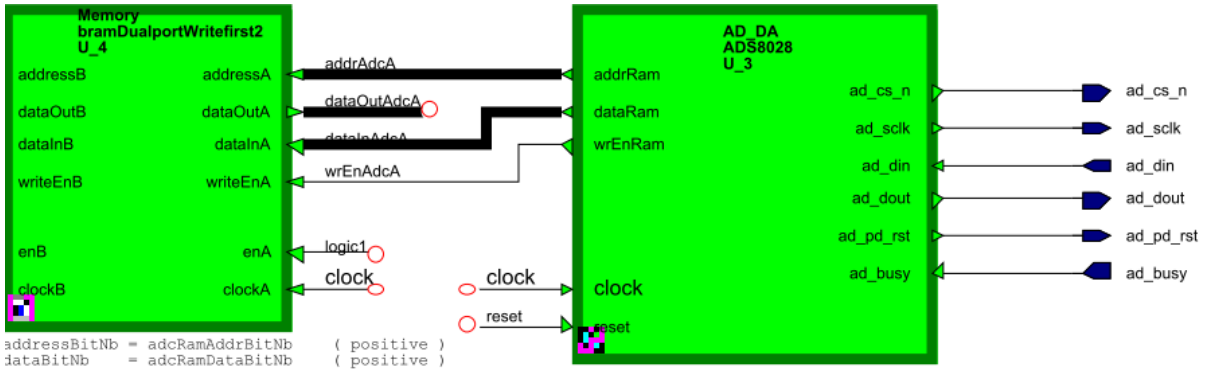


Fig. 28 ADC Dateninterface

7.4. camTest (Camera Control)

Der Block camTest ist für die Initialisierung des Kameramoduls und für das Empfangen der Bilddaten auf dem parallelen Kamerainterface zuständig. Zusätzlich muss das Clock-Signal für das Kameramodul erzeugt werden. Ohne diesen Clock, macht das Modul nichts.

Der interne Aufbau dieses Blockes ist im ***Anhang 12*** zu finden.

Allgemein

Mit Hilfe des cameraClk<U_0> wird der Clock für das Kameramodul erzeugt. Er erhält als Parameter die gewünschte Clockfrequenz und die Frequenz des FPGA-Clock. Anhand dieser erzeugt er den Kamera-Clock. Es gilt zu beachten, dass der Kamera-Clock maximal der halbe FPGA-Clock sein kann. Der Kamera-Clock beträgt 25MHz (1/4 des FPGA-Clock).

Die zwei Signale *camReset* und *camPWDN* werden vom Block camConfig gesteuert.

Konfiguration

Nach dem Startup der FPGA wird sofort die Konfiguration des Kameramoduls gestartet. Im Block-RAM<U_5> ist die Konfiguration gespeichert. Diese wird anhand eines Textfiles bei der Synthese des Designs in das RAM gespeichert.

Mittels dem camConfig<U_4> wird diese Konfiguration gelesen und über I2C (i2cFIFO<U_3>) an das Modul übermittelt (i2cFIFO aus I2C-Library genutzt).

Der interne Aufbau vom Block camConfig ist im ***Anhang 13*** zu finden.

Das Kameramodul wurde so konfiguriert, dass Bilder in VGA-Auflösung (640x480 Pixel) und mit 30 Bildern/s aufgenommen werden. Das Format des Bildes ist Raw Bayer RGB (Bayer Filter, 2015). Die nötigen Parameter für diese Konfiguration sind im (Datenblatt OV7670: Implementation Guide, 2005, S. 11) zu finden.

Die Kommunikation über I2C wird in einem Schreib-Lese-Vorgang durchgeführt. Das Kameramodul verfügt über zwei Adressen: 0x42 (Schreiben) und 0x43 (Lesen). Nach einem drei Zyklus Schreibbefehl wird immer direkt ein zwei Zyklus Lesebefahl ausgeführt. So

wird kontrolliert, ob das Schreiben der jeweiligen Register auch erfolgreich war. Ein solcher Vorgang ist in **Fig. 29** *I2C Write-Read* abgebildet.

Fig. 29 I2C Write-Read

Bildempfang

Für den Block processPictureData<U_7> dienen als Inputs die Signale des parallelen Kamerainterfaces. Der VHDL-Code befindet sich im ***Anhang 14***.

Der Start eines neuen Frames wird mit einer steigenden Flanke von *vsync* angezeigt. Anschliessend wird bei jedem Impuls des *pclk*, insofern *href* aktiv ist, ein Pixel à 8 Bit empfangen. Das empfangene Pixel wird umgehend im RAM abgespeichert. Dies wird so lange wiederholt, bis alle Pixel eines Frames übermittelt sind. Anschliessend wird auf das nächste Frame gewartet und der Vorgang beginnt von neuem.

Mit der momentanen Konfiguration werden 30 Bilder pro Sekunde empfangen. Es wird aber nicht jedes Bilder abgespeichert, sondern nur alle 2s ein Bild.

Um einen möglichen Konflikt zwischen den Auslesen eines Bildes vom Kontroller und dem Abspeichern eines neuen Bildes im RAM zu vermeiden, wurde eine gegenseitige Ausschliessung hinzugefügt.

Wenn der Kontroller ein Bild aus dem RAM lesen will, wird überprüft, ob momentan ein neues Bild abgespeichert wird (*wrNewData* = ‘1‘). Trifft dies zu, wird solange gewartet, bis der Speichervorgang abgeschlossen ist. Der Kontroller legt jetzt das *rdData*-Signal auf ‘1‘, solange er die Daten des Bildes aus dem Block-RAM liest. Sollte währenddessen ein neues Bild abgespeichert werden, sieht der processPictureData-Block den aktiven Lesevorgang anhand dieses Signales. Er überspringt diesen Speichervorgang und versucht es beim nächsten Bild wieder. Der Kontroller hat mit seinem Lesevorgang Priorität.

7.5. ADS8028 (ADC Control)

Im Block ADS8028 liest der ADS8028_Control<U_1> kontinuierlich die ADC-Werte und speichert sie in einem Block-RAM ab.

Das grafische Design des ADS8028 ist im ***Anhang 15*** zu finden.

Bei jedem *readEn*-Impuls, welcher jede Sekunde von counter <U_5> erzeugt wird, startet der ADS8028_Control einen neuen Lesevorgang. In einem ersten Befehl wird dem ADC über SPI übermittelt, welche Kanäle man lesen will und wie er konfiguriert werden soll. Der genaue Aufbau dieses Befehls ist im (Datenblatt ADS8028, 2012, S. 18) zu sehen. In unserem Fall sollen alle 8 Kanäle gelesen und die externe Referenzspannung verwendet werden.

In **Fig. 30** *ADC Lesevorgang* ist ein Beispiel eines Lesevorgangs zu sehen. Im WriteConfig wird dem ADC übermittelt, dass die Kanäle AN0-AN3 gelesen werden sollen. Anschliessend braucht der ADC einen Lesebefehl, um die Messungen bereitzustellen. In den 4 folgenden Lesebefehlen gibt er auf der MISO-Leitung nacheinander die Messwerte der 4 gewünschten Kanäle zurück.

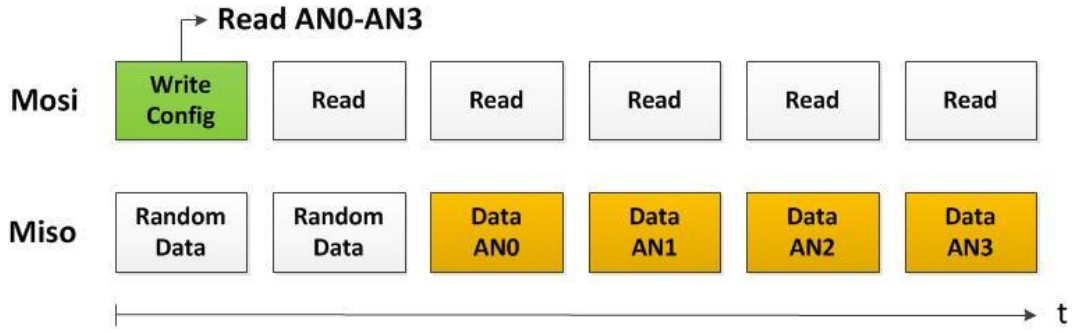


Fig. 30 ADC Lesevorgang

Der interne Aufbau des ADS8028_Control ist im ***Anhang 16*** zu finden.

Der spiFIFO<U_0> serialisiert die Befele und sendet diese über SPI. Die empfangenen Daten auf der MISO-Leitung werden in einem FIFO zwischengespeichert und können später vom Kontroller ausgelesen werden (spiFIFO aus SPI-Library genutzt).

7.6. slaveSPI (Gumstix)

Im Block slaveSPI empfängt der spiRead<U_0> einkommende Daten vom Gumstix über SPI. Die Daten werden deserialisiert und in einem FIFO zwischengespeichert. Die Struktur des slaveSPI ist im ***Anhang 17*** und der VHDL-Code des spiRead ist im ***Anhang 18*** zu finden.

7.7. Board Controller (Kontroller)

Der BoardController bildet die zentrale Steuereinheit.

Er überprüft, ob neue Daten von PC (UART) oder Gumstix (SPI) empfangen wurden. Empfangene Daten werden ausgewertet und dementsprechend verarbeitet.

Das Design des BoardController ist im ***Anhang 19*** zu finden.

Gumstix

Befindet sich der BoardController im Ruhezustand (IDLE), wartet er auf die Änderung des *sEmpty*-Signals des slaveSPI. Fällt dieses auf '0', wurde etwas empfangen und im FIFO zwischengespeichert. Der Kontroller kann diese Daten jetzt auslesen und direkt via UART an den PC weiterleiten. Danach kehrt er in den Ruhezustand zurück.

Dieser Ablauf ist in **Fig. 31** *BoardController: SPI Empfang* vereinfacht dargestellt.

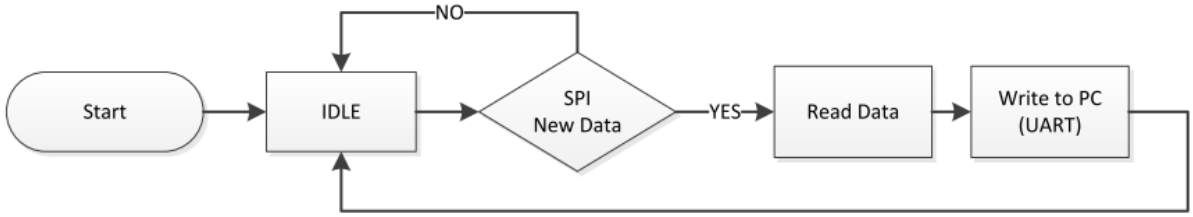


Fig. 31 BoardController: SPI Empfang

Camera / ADC

Wie der slaveSPI verfügt auch der serialPortFIFO über ein rxEmpty-Signal. Fällt dieses auf ‘0‘, wurde ein Befehl vom PC empfangen. Der Befehl wird aus dem FIFO ausgelesen und mit den bekannten Befehlen verglichen.

Folgende Befehle wurden definiert:

- Kamera Bild lesen: ‘I‘ oder ‘i‘, bzw. 0x49 / 0x69. (‘I‘ für Image)
- ADC Werte lesen: ‘A‘ oder ‘a‘, bzw. 0x41 / 0x61. (‘A‘ für ADC)

Wird ein Befehl empfangen, welcher nicht bekannt ist, wird er umgehend über UART zurück an den PC gesendet.

Ist der Befehl bekannt, wird das jeweilige Block-RAM mit den neusten Daten ausgelesen. Diese Daten werden über UART an den PC übermittelt. Ist die Übermittlung abgeschlossen, kehrt der Kontroller in den Ruhezustand zurück.

Der genaue Ablauf ist in **Fig. 32** *BoardController: UART Empfang* veranschaulicht.

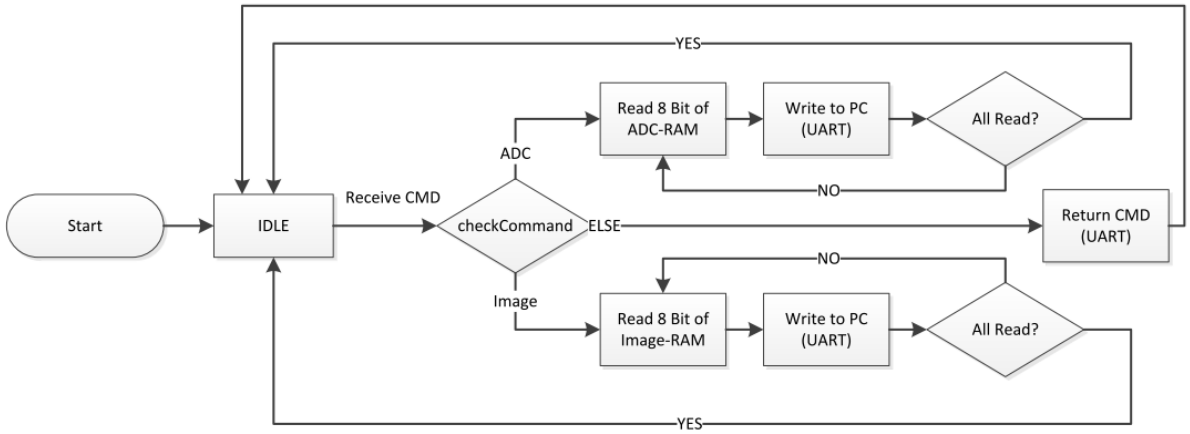


Fig. 32 BoardController: UART Empfang

8. Tests

Um sicherzustellen, dass auf der Bildverarbeitungskarte alles einwandfrei funktioniert, wurden diverse Tests durchgeführt.

8.1. Test der Hardware (PCB)

Nach dem Bestücken der ersten Karte wurden darauf verschiedene Test durchgeführt. Es wurde ein Testprotokoll erstellt, wo alle Tests aufgelistet sind.

Das Protokoll ist im ***Anhang 20*** zu finden. Alle durchgeführten Tests waren erfolgreich.

Die Stromaufnahme des kompletten Board betrug **360mA**, wenn das komplette Design auf die FPGA geladen wurde. In **Fig. 33** *Bildverarbeitungskarte V1.0* ist der erste Prototyp abgebildet.

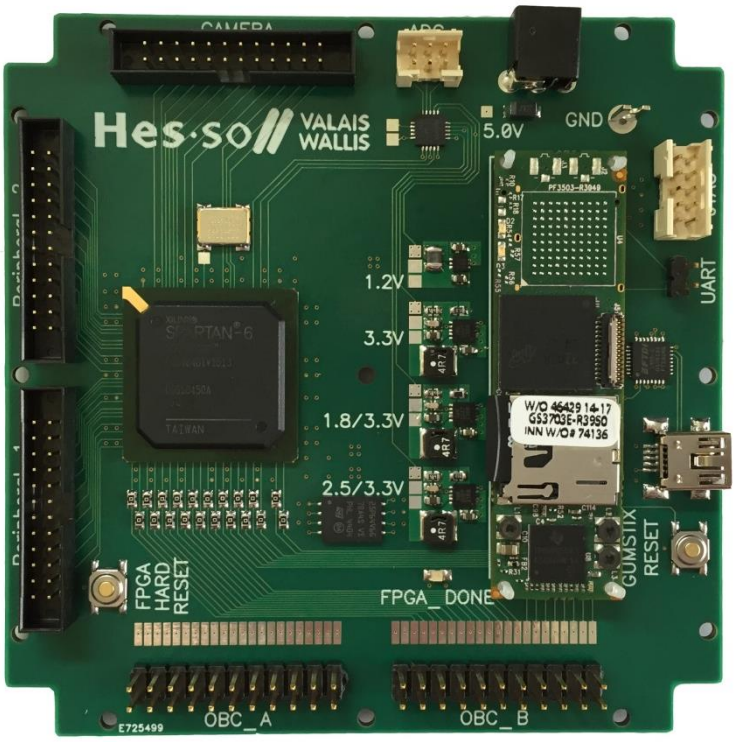


Fig. 33 Bildverarbeitungskarte V1.0

8.2. Simulation des Hardwaredesigns

Um das erstellte Hardwaredesign zu testen, wurden spezifische Testbanken für den Teil der Kamera und des ADC erstellt. Diese Testbanken werden in den folgenden Teilkapiteln genauer betrachtet.

8.2.1. Testbank Kamera

Die Konfiguration der Kamera kann mit einer Simulation nicht getestet werden. Dafür muss das Kameramodul an die Karte angeschlossen werden.

Was getestet werden kann, ist das Lesen der Daten auf dem parallelen Kamerainterface und die Speicherung dieser Daten im Block-RAM. Mit einer Testbank wurden die Kamerasignale simuliert. Um die Simulation zu vereinfachen, wurde die Grösse des Bildes auf 10x5 Pixel verkleinert. Der Aufbau der Simulation ist im ***Anhang 21*** ersichtlich.

In **Fig. 34** *Simuliertes Kamerasignal* ist das simulierte Kamerasignal zu sehen. Im roten Kreis ist der Start des Bildes ersichtlich. Im violetten Viereck ist der *href*-Impuls ersichtlich, welcher während 10 *pclk*-Impulsen anhält. Der Datenbus zählt in jeder Reihe von 0 auf 9.

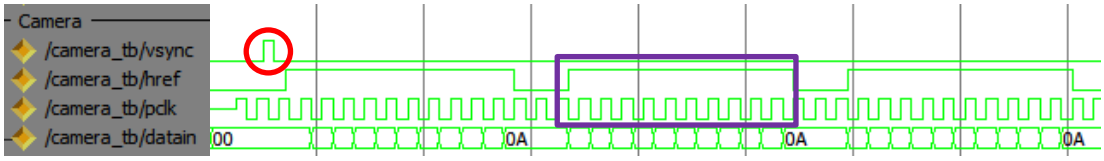


Fig. 34 Simuliertes Kamerasignal

Im ***Anhang 22*** ist das vollständige Ergebnis der Simulation zu sehen. Dort ist gut ersichtlich, dass nach der steigenden Flanke des *pclk* die Daten vom Parallelinterface gelesen werden. Kurz darauf kommt der Schreibimpuls (*imgWrEnA* = '1') für die Speicherung im Block-RAM.

Die Simulation zeigt, dass die Aufnahme und Speicherung der Pixeldaten funktioniert.

8.2.2. Testbank ADC

Um den ADC zu simulieren, wurde mit Hilfe einer Testbank die über SPI empfangenen Daten des ADC simuliert.
Der Aufbau der Simulation ist im ***Anhang 23*** ersichtlich.

Der ADS8028_Control-Block liest zu Beginn die 8 Eingänge des ADC. Nach der Übermittlung der 10 SPI Befehle werden die 10 empfangenen Pakete aus dem spiFIFO gelesen. Diese Werte werden mit der Testbank simuliert, da effektiv kein AD-Wandler angeschlossen ist.

Auf dem *sData*-Bus ist jeweils der Kanal(15:12) + Daten(11:0) ersichtlich. Nach dem *sRd*-Impuls wird der Addressbus (*addrADC)* angepasst und die Daten (*dataInADC)* auf den Bus gelegt, um diese ins RAM zu speichern. Beim nächsten Clock sind die Daten im RAM abgespeichert. In **Fig. 35** *Auswertung SPI Daten* ist die Verarbeitung von drei dieser Pakete sichtbar.

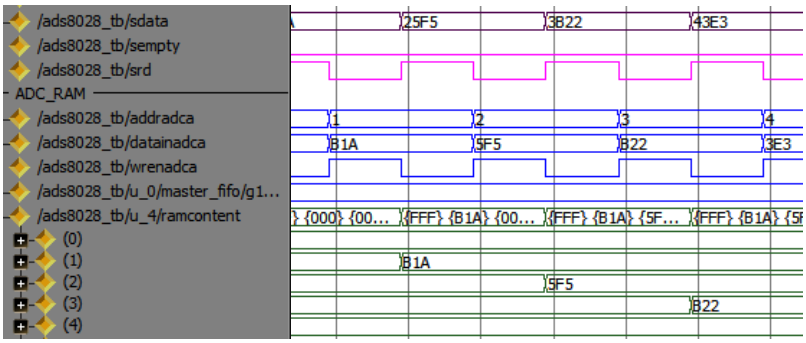


Fig. 35 Auswertung SPI Daten

Sobald der Lesevorgang abgeschlossen ist, wird eine UART-Abfrage des ADC simuliert. Diese liest alle Werte aus dem Block-RAM und übermittelt diese per UART.

Die kompletten Resultate der Simulation sind im ***Anhang 24*** zu finden.

8.3. Test des Hardwaredesigns (FPGA)

Schlussendlich wurde das Hardwaredesign der FPGA getestet. Es wurden die einzelnen Teile des Designs getestet.

Im ***Anhang 25*** ist eine Anleitung für das Anschliessen und die Inbetriebnahme der Bildverarbeitungskarte zu finden. Alle Tests wurden mit diesem Aufbau durchgeführt.

Auf dem PC wurde als serielles Terminal (HTerm, 2008) verwendet. In HTerm kann die Anzeige der Daten (ASCII, HEX, usw.) individuell eingestellt werden. Ausserdem bietet es die Möglichkeit, empfangene Daten zu exportieren, was für die Bilddaten der Kamera sehr nützlich war.

8.3.1. Test Kameramodul

In einem ersten Schritt wurde die Konfiguration des Kameramoduls über I2C kontrolliert. Dafür wurde mit einem *Logic Analyser* die Konfiguration auf dem Bus kontrolliert. In **Fig. 36** *Kamera: Konfiguration* ist der Schreib-Lesevorgang eines Registers ersichtlich. In das Register 0x70 wird der Wert 0x3A geschrieben. Im folgenden Lesevorgang wird derselbe Wert herausgelesen. Das Register wurde korrekt konfiguriert. Der komplette Initialisierungsvorgang des Moduls wurde so überprüft, wobei keine Fehler entdeckt werden konnten.

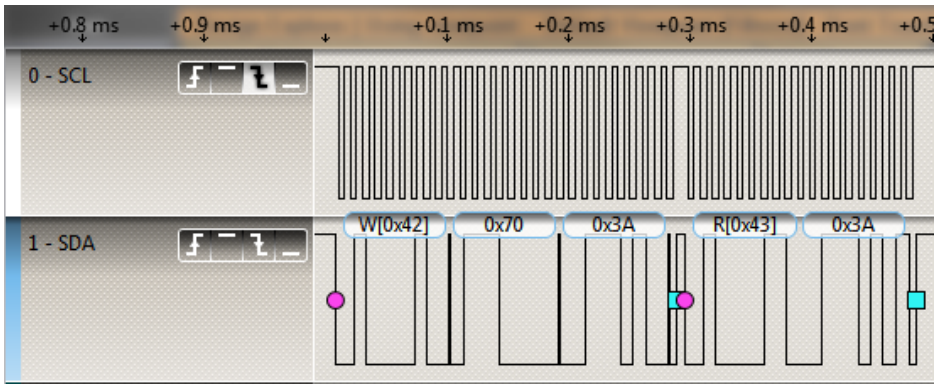


Fig. 36 Kamera: Konfiguration

Des Weiteren wurde die gegenseitige Blockade für den Zugriff auf die Kameradaten getestet. Das Signal *rdData* vom Kontroller und *wrNewData* vom camTest wurden auf 2 Pins herausgeführt.

In regelmässigen Abständen wird ein neues Bild abgespeichert (An den *wrNewData*-Impulsen ersichtlich in **Fig. 37** *Kamera: Gegenseitige Ausschliessung*). Wird jetzt über UART ein Bild gelesen, geht *rdData* auf '1'. Während dieser Zeit kommen keine *wrNewData*-Impulse, da keine neuen Bilder gespeichert werden. Ist der Lesevorgang abgeschlossen, können wieder neue Bilder abgespeichert werden.

Fig. 37 Kamera: Gegenseitige Ausschliessung

Um die Test für das Kameramodul abzuschliessen, wurde ein kompletter Ablauf für das Lesen des Bildspeichers durchlaufen.

Mit dem seriellen Terminal (*HTerm*) wird über UART ein Befehl für das Lesen des Bildspeichers ('I' oder 'i' für Image) an die Bildverarbeitungskarte übermittelt.

Anschliessend empfängt das Terminal die kompletten Daten eines Bildes (640x480 Pixel = 307200 Byte). Die empfangenen Daten können mit der Funktion *Save-Output* als RAW-Datei abgespeichert werden.

Da diese Datei nur die Bilddaten enthält, kann sich nicht angezeigt werden. Es fehlen die Informationen über die Auflösung des Bildes sowie die Farbtiefe. Um das Bild anzuzeigen, wurde die RAW-Datei mit Photoshop geöffnet.

Das Programm erkennt sofort, dass es sich um ein RAW-Bild handelt. Es wird ein *RAW Options*-Dialog Menu (**Fig. 38** *Photoshop RAW Options*) geöffnet, wo Parameter wie Grösse (640x480 Pixel) sowie Bildtiefe(8 Bit) eingestellt werden können.

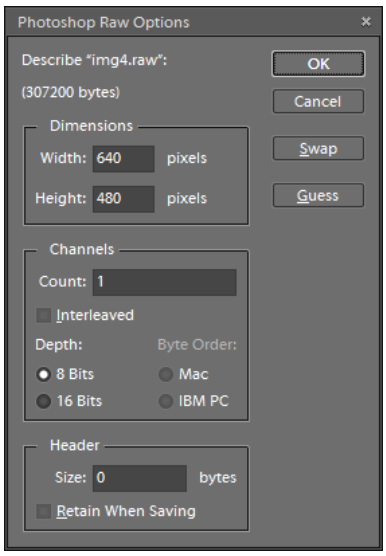


Fig. 38 Photoshop RAW Options

Das Bild wird jetzt in Graustufen angezeigt. Dieses kann als JPEG oder PNG abgespeichert werden. Es handelt sich hier jedoch nicht um das effektive Bild.

Um das Originalbild zu rekonstruieren, ist eine Umwandlung vom Bayer Formats (Bayer Filter, 2015) in ein RGB-Format notwendig. Matlab verfügt über eine Library namens Image Processing Toolbox. Diese Library liefert die Funktion *Demosaic* (Image Processing Toolbox: Demosaic, 2015), welche genau diese Umwandlung durchführt. Er wurde deshalb ein kleines Matlab Script erstellt, welches das abgespeicherte Rohbild in das Originalbild umwandelt. Das verwendete Matlab Script ist im ***Anhang 26*** zu finden.

In **Fig. 39** *Kamerabild: Vor und nach der Rekonstruktion* sind sich die beiden Bilder gegenübergestellt.

Fig. 39 Kamerabild: Vor und nach der Rekonstruktion

Es konnten somit erfolgreich Bilder mit dem Kameramodul aufgenommen und ausgelesen werden. Mit den nötigen Tools konnten diese auf dem PC angezeigt werden.

8.3.2. Test ADC

Auch beim ADC wurde getestet, ob das Konfigurieren und Lesen über SPI funktioniert. Die SPI-Siganle wurden auf einigen Testpins herausgeführt, um sie mit dem Logic Analyser messen zu können.

In **Fig. 40** *ADC: SPI Lesevorgang* ist eine solche Messung ersichtlich. Man erkennt gut die 10 Zyklen, welche für das Lesen von allen 8 ADC-Kanälen notwendig sind. Der erste Befehl auf der MOSI-Leitung teilt dem ADC die genaue Konfiguration mit.

Die ersten beiden Werte auf der MISO-Leitung sagen nichts aus. Die folgenden übermitteln jeweils die Daten der 8 Kanäle. Diese Werte sind wie folgt aufgebaut: 4 Bit für den Kanal und die restlichen 12 Bit für die Daten.

Man erkennt direkt, dass auf dem AN0 die 5.0V Speisung ist und AN5-7 auf Masse gelegt ist.

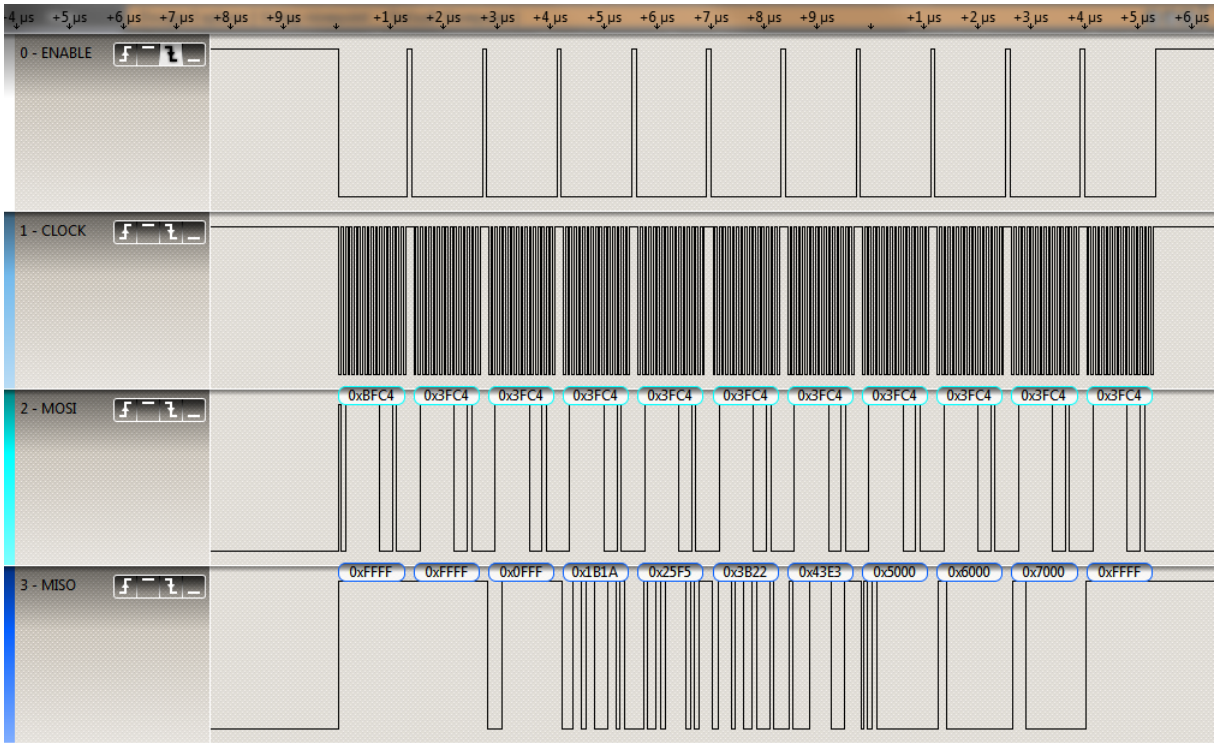


Fig. 40 ADC: SPI Lesevorgang

Um die Messwerte des ADC besser zu veranschaulichen, wurde mit Python eine kleine Applikation (**Fig. 41** *Python Applikation: ADC-Spannungen*) erstellt, welche die Spannungen grafisch darstellt. Über UART werden kontinuierlich die Daten des ADC abgefragt. Die erhaltenen Messwerte werden umgerechnet, um daraus einen Spannungsgraph zu zeichnen.

Die exakten Spannungswerte werden zusätzlich in der Konsole angezeigt.

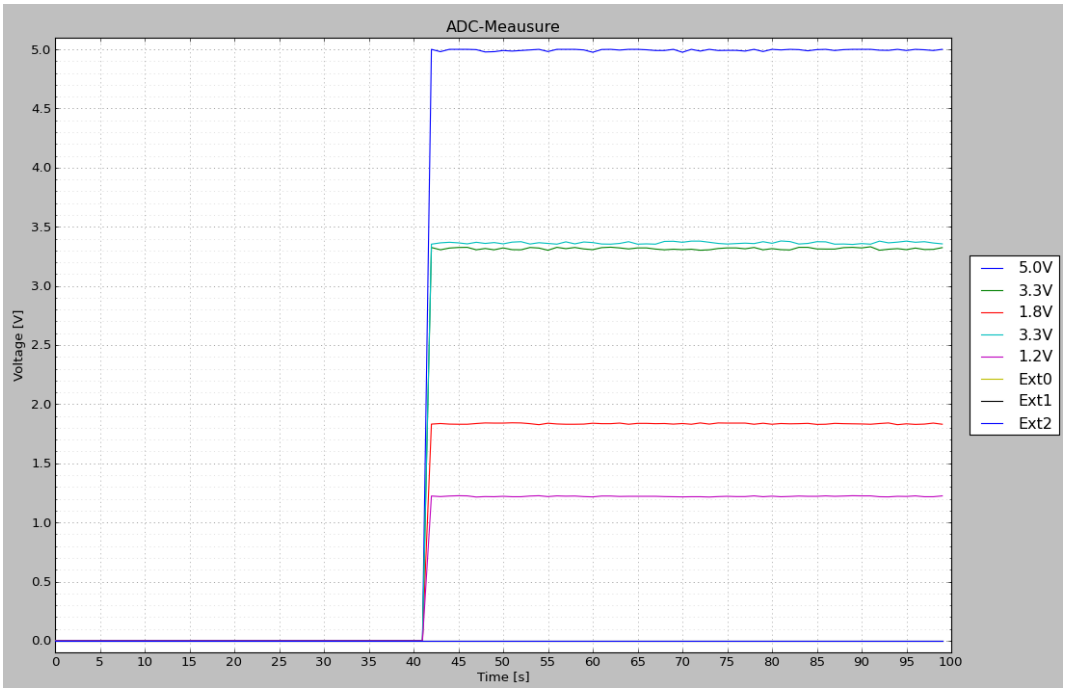


Fig. 41 Python Applikation: ADC-Spannungen

Der Code der Python Applikation ist im ***Anhang 27*** zu finden.

Im Programm muss der COM-Port manuell angepasst werden, bevor die Applikation gestartet werden kann. Es darf nicht bereits eine andere serielle Verbindung auf diesem Port offen sein.

8.3.3. Test Verbindung Gumstix-FPGA

Um zu testen, ob der Gumstix mit der FPGA kommunizieren kann (Logikpegel in Ordnung), wurde ein Testszenario festgelegt. Dabei sendet der Gumstix die Eingabe der Konsole über SPI an die FPGA. Diese empfängt diese Eingabe und leitet sie umgehend via UART an den PC weiter. In diesem Testversuch kann somit von der Konsole des Gumstix eine kleine Nachricht an den PC übermittelt werden.

In **Fig. 42** *Konsole Gumstix* ist die Eingabe in der Konsole des Gumstix ersichtlich. Bevor die Kommunikation möglich ist, muss die Konsole mit dem SPI gekoppelt werden. Dies wird mit dem *cat*-Befehl gemacht.

Es kann jetzt eine kleine Nachricht verfasst werden und durch drücken von *Enter* wird diese versandt.

```
root@overo:~$  cat < /dev/ttyO2 > /dev/spidev1.0
Hello. This is a Connection Test!
```

Fig. 42 Konsole Gumstix

Eine gesendete Nachricht wird nach einer kurzen Verzögerung auf dem PC empfangen. Die erhaltene Nachricht ist in **Fig. 43** *UART: Empfang PC* ersichtlich.

```
1    5    10    15    20    25    30    35
Hello. This is a Connection Test!\n
```

Fig. 43 UART: Empfang PC

Die empfangene Nachricht stimmt mit der ursprünglichen Nachricht überein. Es kann somit gesagt werden, dass die Kommunikation zwischen Gumstix und FPGA funktioniert und dass die Logikpegel in Ordnung sind, da ansonsten die Kommunikation nicht funktionieren würde.

8.3.4. Generell

Im erstellten Hardwaredesign sind diese drei separaten Teile in einem Design zusammengefügt.

Das Design funktioniert einwandfrei. Es gilt zu erwähnen, dass jeweils nur einer dieser Teil zu einem gegebenen Zeitpunkt genutzt werden kann. Sie können nicht parallel genutzt werden.

9. Schlussfolgerung

Die Diplomarbeit konnte erfolgreich abgeschlossen werden und alle gesteckten Ziele wurden erreicht.

Es wurde ein funktionsfähiger erster Prototyp der Bildverarbeitungskarte erstellt. Die Karte wurde anhand des neuen CubeSat-Standards von ELSE geplant und entwickelt.

Auf der Karte wurde ein Gumstix Overo COM integriert, welcher für die Bildverarbeitung genutzt wird. Der Gumstix ist speziell für grafische Applikationen ausgelegt und es läuft ein Embedded-Linux Betriebssystem darauf. Mit einer FPGA wird die Kommunikation zwischen Kamera, Gumstix und der Hauptprozessorkarte geregelt. Über einen Konsolenanschluss kann mit dem Gumstix interagiert und auf sein Filesystem zugegriffen werden.

Die Karte wurde so entwickelt, dass der Gumstix gegebenenfalls durch ein Speichermodul ersetzt werden kann. Dadurch würde sich die Bildverarbeitungskarte für eine allgemeinere Nutzung empfehlen.

Für die Entwicklung wurde mit einem preisgünstigen Kameramodul gearbeitet, welches über ein ähnliches Interface wie die Photonfocus-Kamera im Satelliten verfügt. Es wurde ein Interfaceboard erstellt, um dieses Kameramodul an die Bildverarbeitungskarte anschliessen zu können.

Ein 8-Kanal ADC-Konverter dient zur Messung der verschiedenen Spannungen auf dem Board. Die nicht genutzten Eingänge wurden auf einem Konnektor herausgeführt und können für die Messung von externen Spannungen, wie beispielsweise einem analogen Sensor, genutzt werden. Mit Hilfe einer kleinen Applikation, welche mit Python erstellt wurde, können die Messwerte des AD-Wandlers auf dem PC grafisch angezeigt werden.

Für mögliche Erweiterungen wurden zwei zusätzliche Anschlüsse auf die FPGA verbunden. Somit besteht später die Möglichkeit, neben der Kamera noch zwei weitere Peripherien mit der Bildverarbeitungskarte zu verbinden.

Für die Übermittlung von grossen Datenmengen zwischen FPGA und Gumstix müsste für den Extended Memory Bus des Gumstix ein Treiber geschrieben werden, um damit die FPGA adressieren zu können. Dafür fehlten die Zeit sowie die Grundkenntnisse für die Realisierung eines Treibers.

In einem nächsten Schritt könnte der Anschluss des Photonfocus-Kameramoduls realisiert sowie die Kommunikation mit der Hauptprozessorkarte in Angriff genommen werden.

Im Gegensatz zur FPGA-Karte, welche als Vorlage für diese Diplomarbeit diente, ist die Bildverarbeitungskarte speziell dafür entwickelt worden, um Bilder einer Kamera aufzunehmen und diese auf einem COM-Modul zu verarbeiten.

Die Bildverarbeitungskarte kann auch für eine sehr allgemeine Anwendung genutzt werden, indem das COM-Modul durch ein Speichermodul ersetzt wird.

10. Datum & Unterschrift

Dieser Bericht und das Projekt wurden realisiert von:

Sitten, den 10.Juli 2015 Andres Nicolas

11. Literaturverzeichnis

Cobham Haisler. (Mai 2015). GRLIB IP Library User's Manual. Abgerufen am 07. 2015 von http://www.gaisler.com/products/grlib/grlib.pdf

ELSE SA. (07. 2015). *Elegant Systems Engineering*. Von http://www.else.io/ abgerufen

Euro Circuit GmbH. (Juli 2015). Euro Circuit. Abgerufen am 07. 2015 von http://www.eurocircuits.de/

Github. (2015). *Gumstix Yocto Manifest*. Abgerufen am 07. 2015 von https://github.com/gumstix/yocto-manifest/blob/master/README.md

Hammer, T. (2008). *HTerm*. Abgerufen am 07. 2015 von http://www.der-hammer.info/terminal/

MathWorks. (2015). *Image Processing Toolbox: Demosaic*. Abgerufen am 07. 2015 von http://ch.mathworks.com/help/images/ref/demosaic.html

Omnetics Connector Group. (2015). *Omnetics Connectors*. Abgerufen am 07. 2015 von http://www.omnetics.com/

Omnivision. (September 2005). Datenblatt OV7670: Implementation Guide.

Omnivision. (August 2006). Datenblatt OV7670.

Photonfocus. (März 2012). Datenblatt OEM-D1024E. Abgerufen am 07. 2015 von http://www.photonfocus.com/fileadmin/web/manuals/MAN032_e_V1_2_OEM_D1024E.pdf

Photonfocus. (2015). *Photonfocus*. Abgerufen am 07. 2015 von http://www.photonfocus.com/de/

Semtech. (Februar 2007). Datenblatt SC196. Abgerufen am 07. 2015 von http://www.semtech.com/images/datasheet/sc196.pdf

Semtech. (August 2010). Datenblatt SC189. Abgerufen am 07. 2015 von http://www.semtech.com/images/datasheet/sc189.pdf

Texas Instrument. (Juli 2011). Datenblatt DM3730. Abgerufen am 07. 2015 von http://www.ti.com/lit/ds/symlink/dm3725.pdf

Texas Instruments. (März 2012). Datenblatt ADS8028. Abgerufen am 07. 2015 von http://www.ti.com/lit/ds/symlink/ads8028.pdf

Wikipedia. (Juni 2015). *Bayer Filter*. Abgerufen am 07. 2015 von https://en.wikipedia.org/wiki/Bayer_filter

Wikipedia. (2015). *Camera Inetrface*. Abgerufen am 07. 2015 von https://en.wikipedia.org/wiki/Camera_interface

XILINX. (Oktober 2014). Spartan6 FPGA Configuration. *UG380*. Abgerufen am 07. 2015 von http://www.xilinx.com/support/documentation/user_guides/ug380.pdf

XILINX. (Februar 2014). Spartan6 Select IO Resources. *UG381*. Abgerufen am 07. 2015 von http://www.xilinx.com/support/documentation/user_guides/ug381.pdf

Yocto Project. (2013). *Yocto*. Abgerufen am 07. 2015 von https://www.yoctoproject.org/

12. Anhang

Anhang 1: Hardware Blockschema

Anhang 2: Bildverarbeitungskarte: Schema

Anhang 3: Kamera Interface

Anhang 4: Interfaceboard OV7670: Schema

Anhang 5: Berechnung Speisungen

Anhang 6: Abmessungen PCB

Anhang 7: Bildverarbeitungskarte: Layout

Anhang 8: Komponentenliste

Anhang 9: Hardwarestruktur: Top-Level (*digital*)

Anhang 10: FPGA Pinzuweisung: ucf-File

Anhang 11: Hardwarestruktur: ProcessingBoard (*digital*)

Anhang 12: Hardwarestruktur: camTest (*digital*)

Anhang 13: Hardwarestruktur: camConfig (*digital*)

Anhang 14: Hardwarestruktur: processPictureData

Anhang 15: Hardwarestruktur: ADS8028 (*digital*)

Anhang 16: Hardwarestruktur: ADS8028_Control (*digital*)

Anhang 17: Hardwarestruktur: slaveSPI (*digital*)

Anhang 18: Hardwarestruktur: spiRead

Anhang 19: Hardwarestruktur: boardController (*digital*)

Anhang 20: Hardwaretest PCB

Anhang 21: Testbank Kamera (*digital*)

Anhang 22: Simulation Kamera

Anhang 23: Testbank ADC (*digital*)

Anhang 24: Simulation ADC

Anhang 25: Anleitung: Bildverarbeitungskarte

Anhang 26: Matlab Scrip: Bayer Demosaic

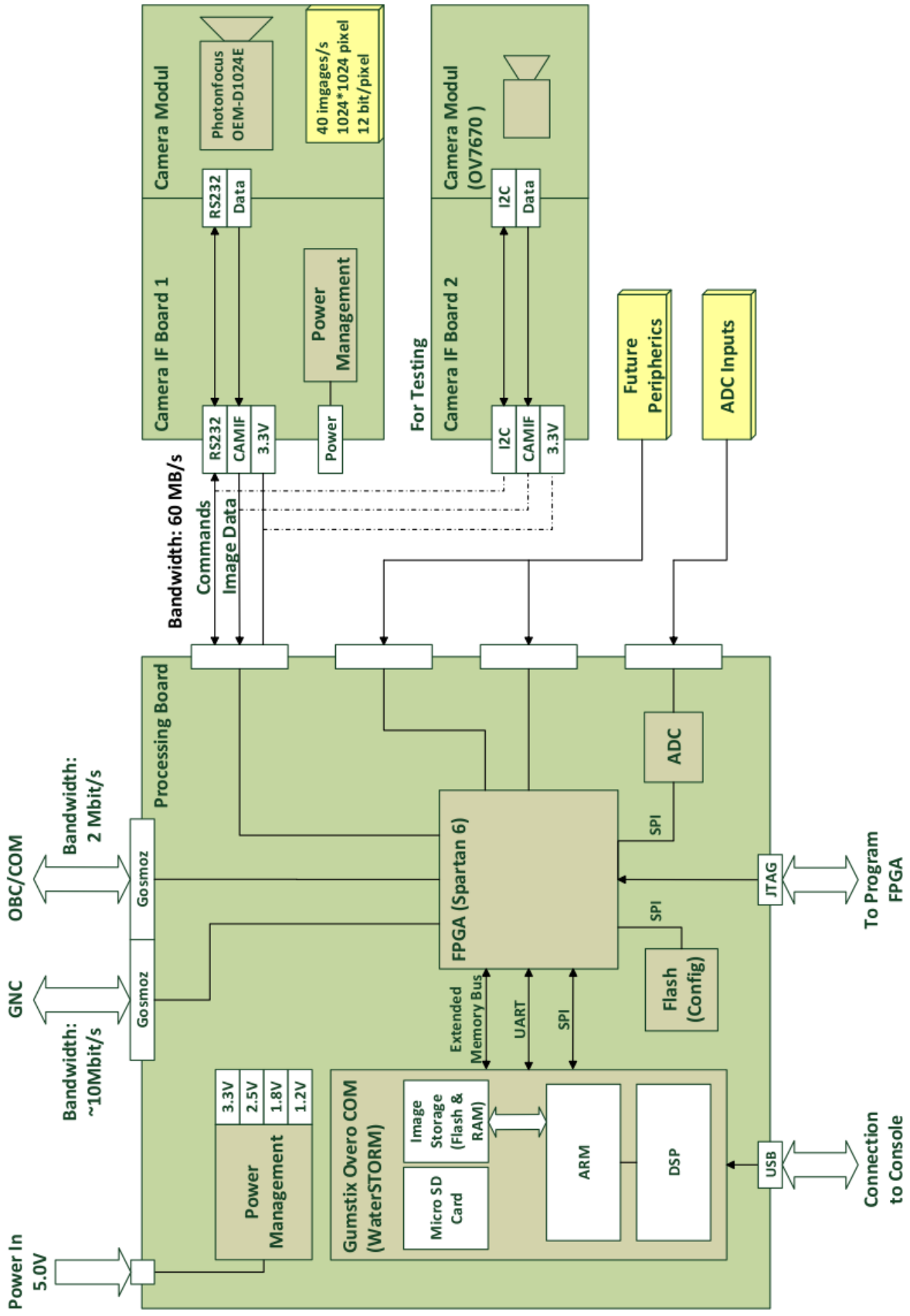
Anhang 27: Python Applikation: ADC

Anhang 28: Yocto-Image Gumstix (*digital*)

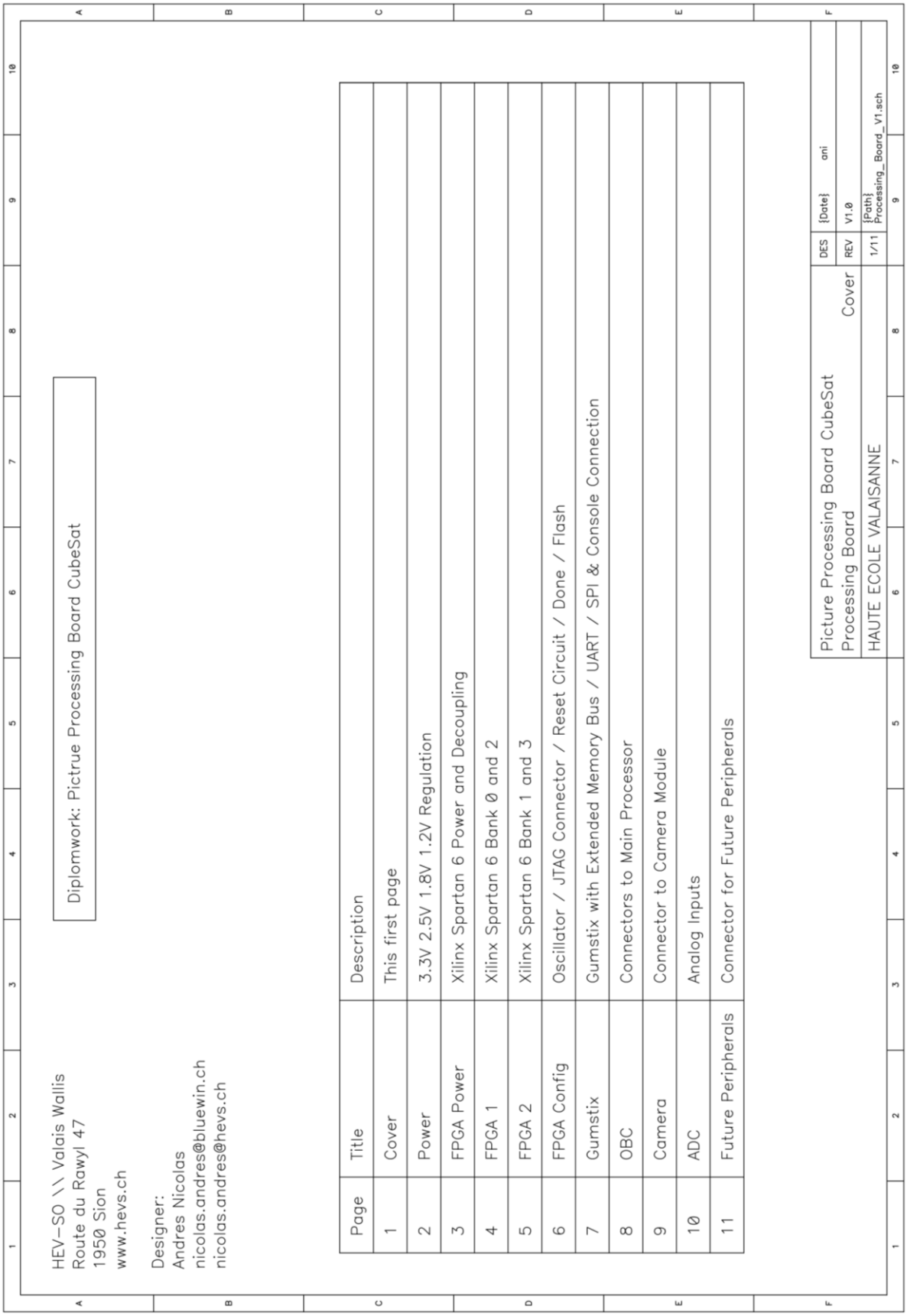
Alle Anhänge sind in digitaler Form auf der beigelegten CD vorhanden. Anhänge, welche den Vermerk *digital* aufweisen, sind nur auf der CD zu finden.

12. Anhang Verzeichnis

Anhang 1: Hardware Blockschema



Anhang 2: Bildverarbeitungskarte Schema



Diplomwork: Pictrue Processing Board CubeSat

HEV-SO \\ Valais Wallis
Route du Rawyl 47
1950 Sion
www.hevs.ch

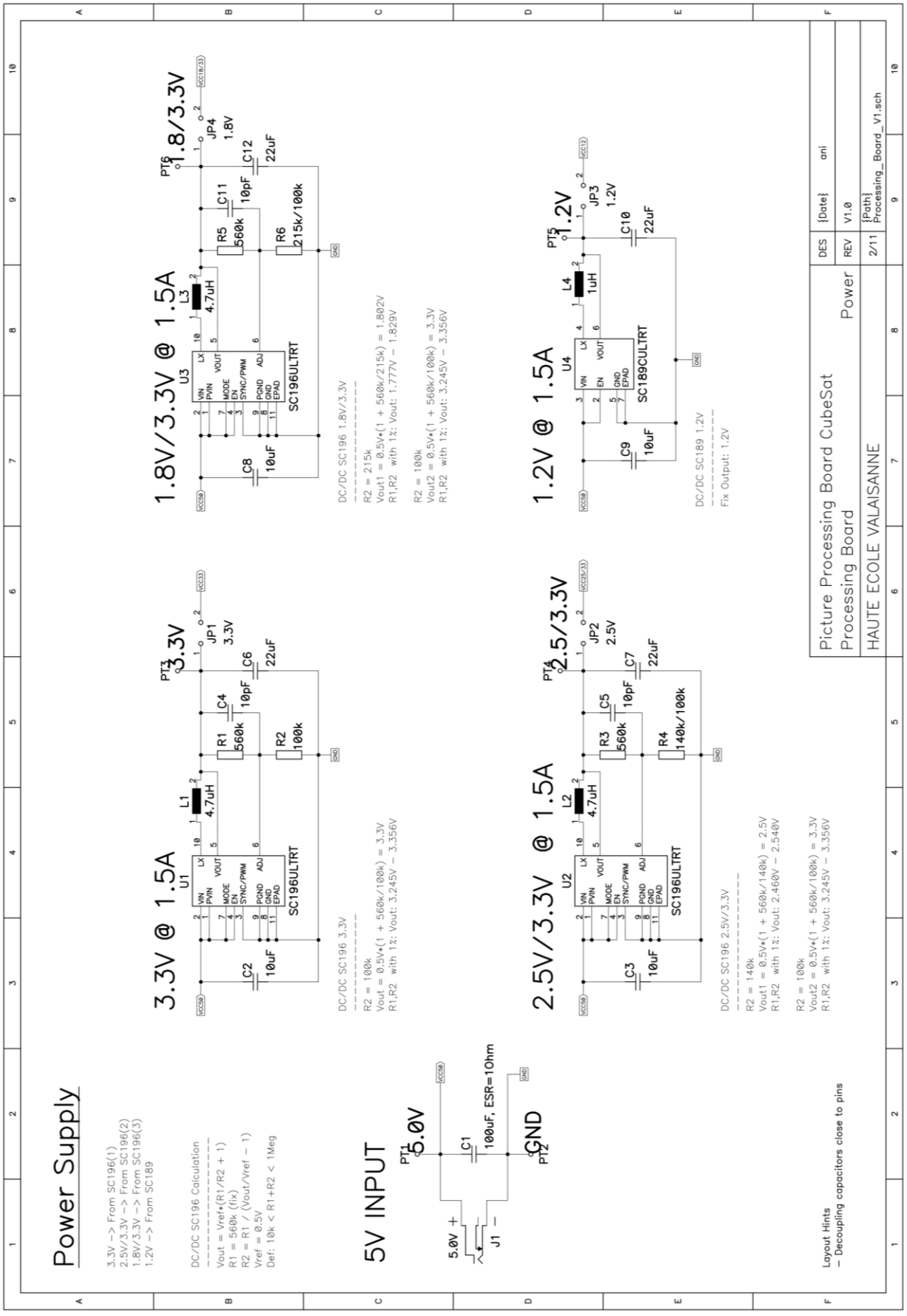
Designer:
Andres Nicolas
nicolas.andres@bluewin.ch
nicolas.andres@hevs.ch

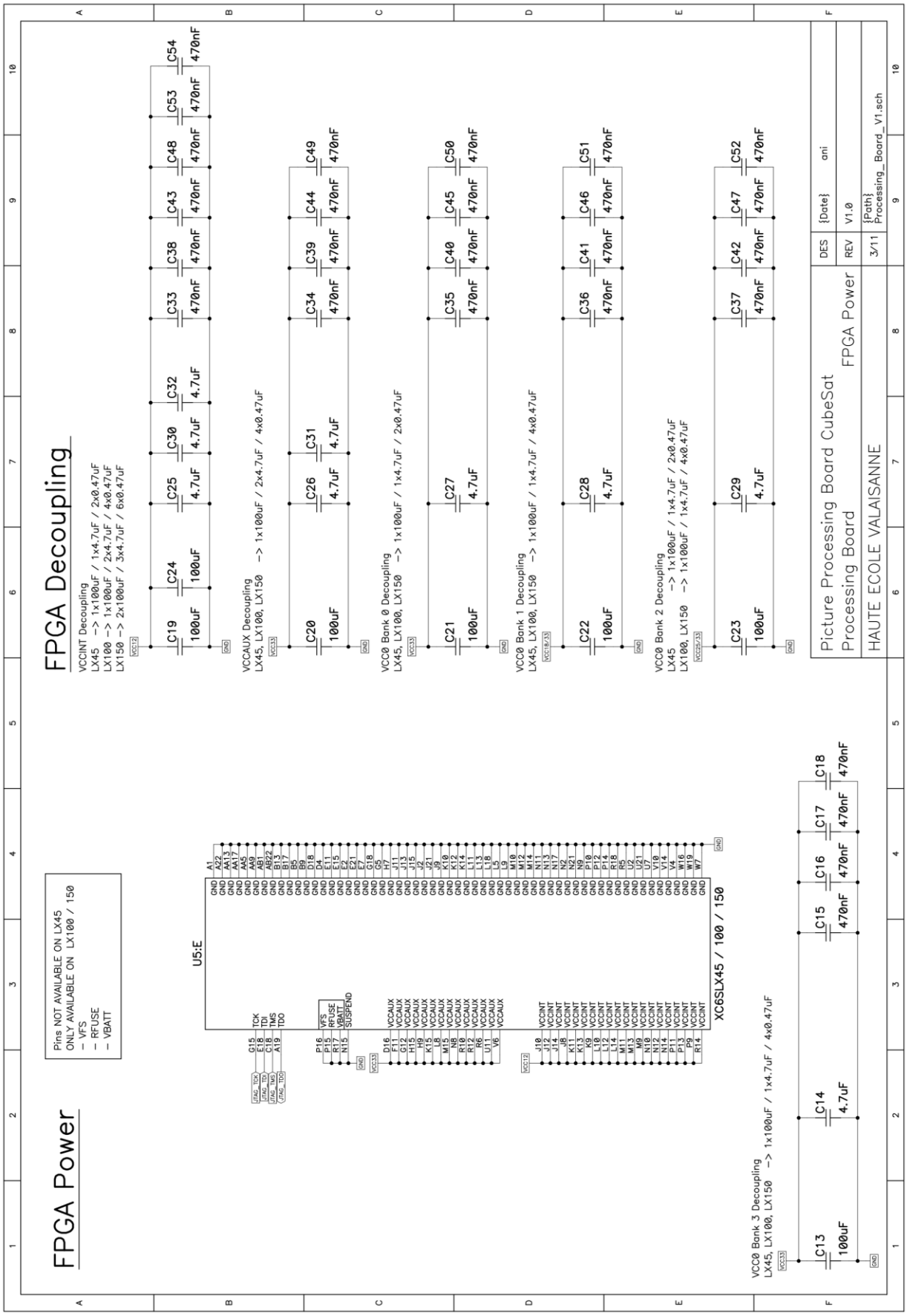
Page	Title	Description
1	Cover	This first page
2	Power	3.3V 2.5V 1.8V 1.2V Regulation
3	FPGA Power	Xilinx Spartan 6 Power and Decoupling
4	FPGA 1	Xilinx Spartan 6 Bank 0 and 2
5	FPGA 2	Xilinx Spartan 6 Bank 1 and 3
6	FPGA Config	Oscillator / JTAG Connector / Reset Circuit / Done / Flash
7	Gumstix	Gumstix with Extended Memory Bus / UART / SPI & Console Connection
8	OBC	Connectors to Main Processor
9	Camera	Connector to Camera Module
10	ADC	Analog Inputs
11	Future Peripherals	Connector for Future Peripherals

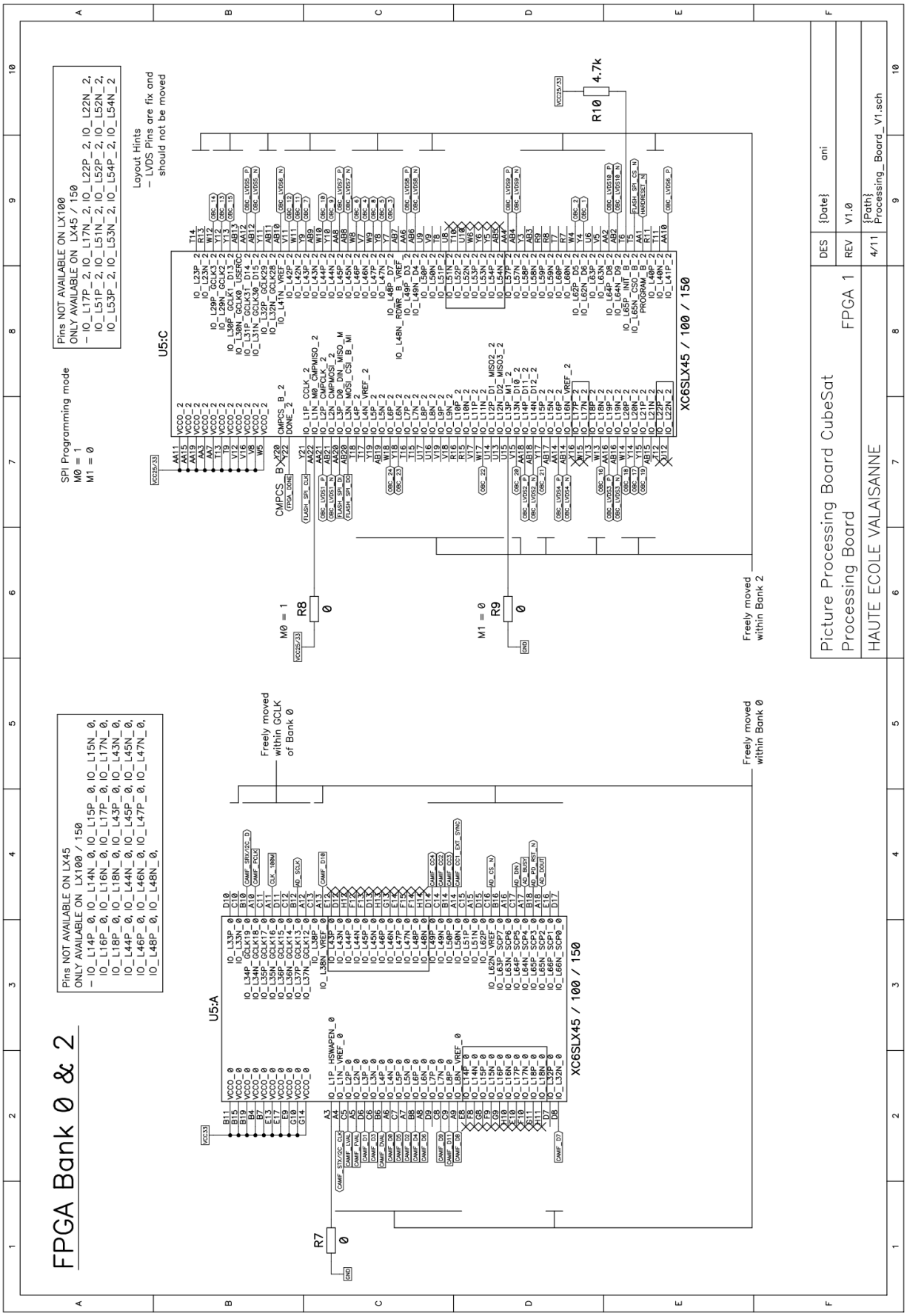
Picture Processing Board CubeSat
Processing Board
Cover

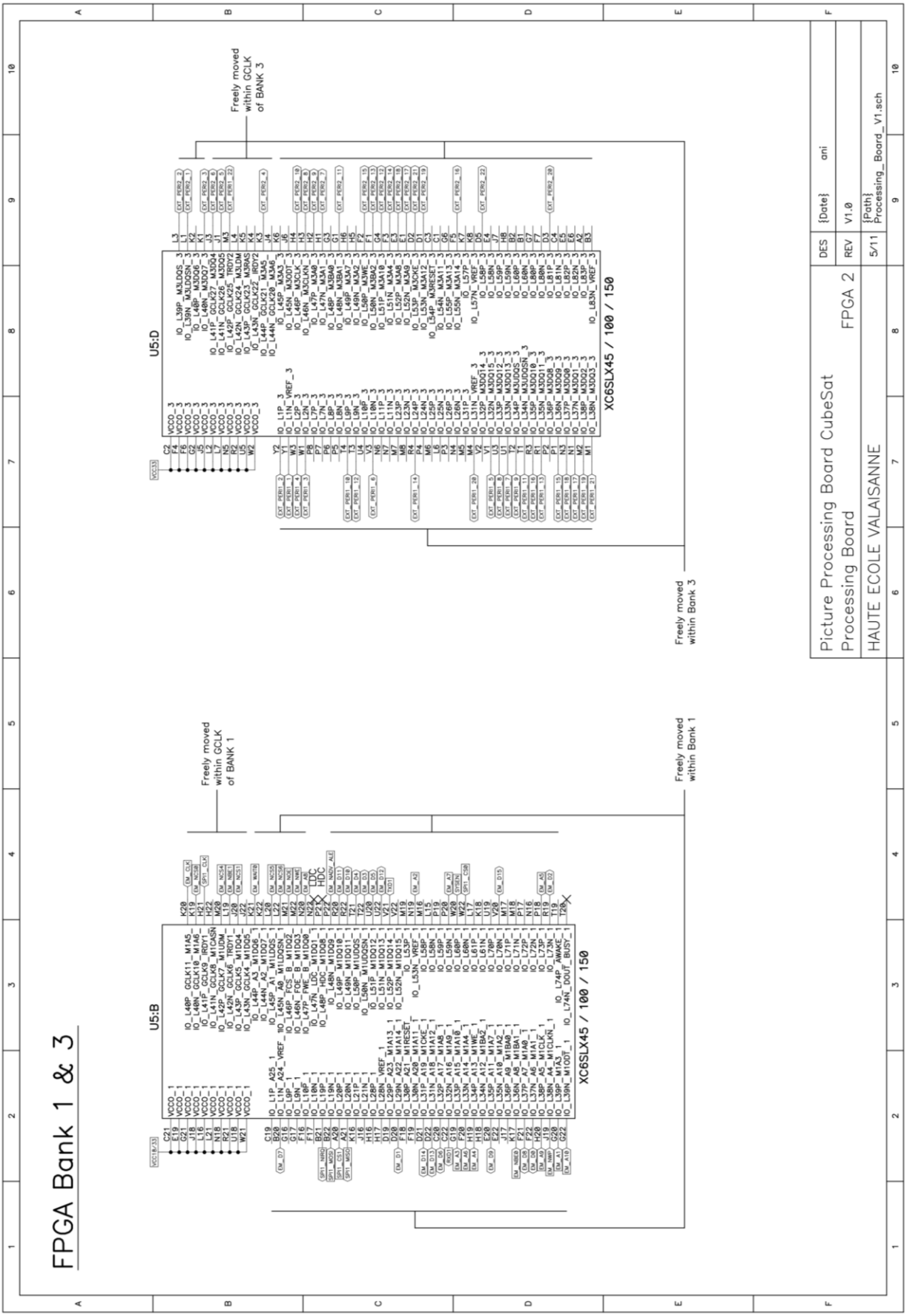
HAUTE ECOLE VALAISANNE

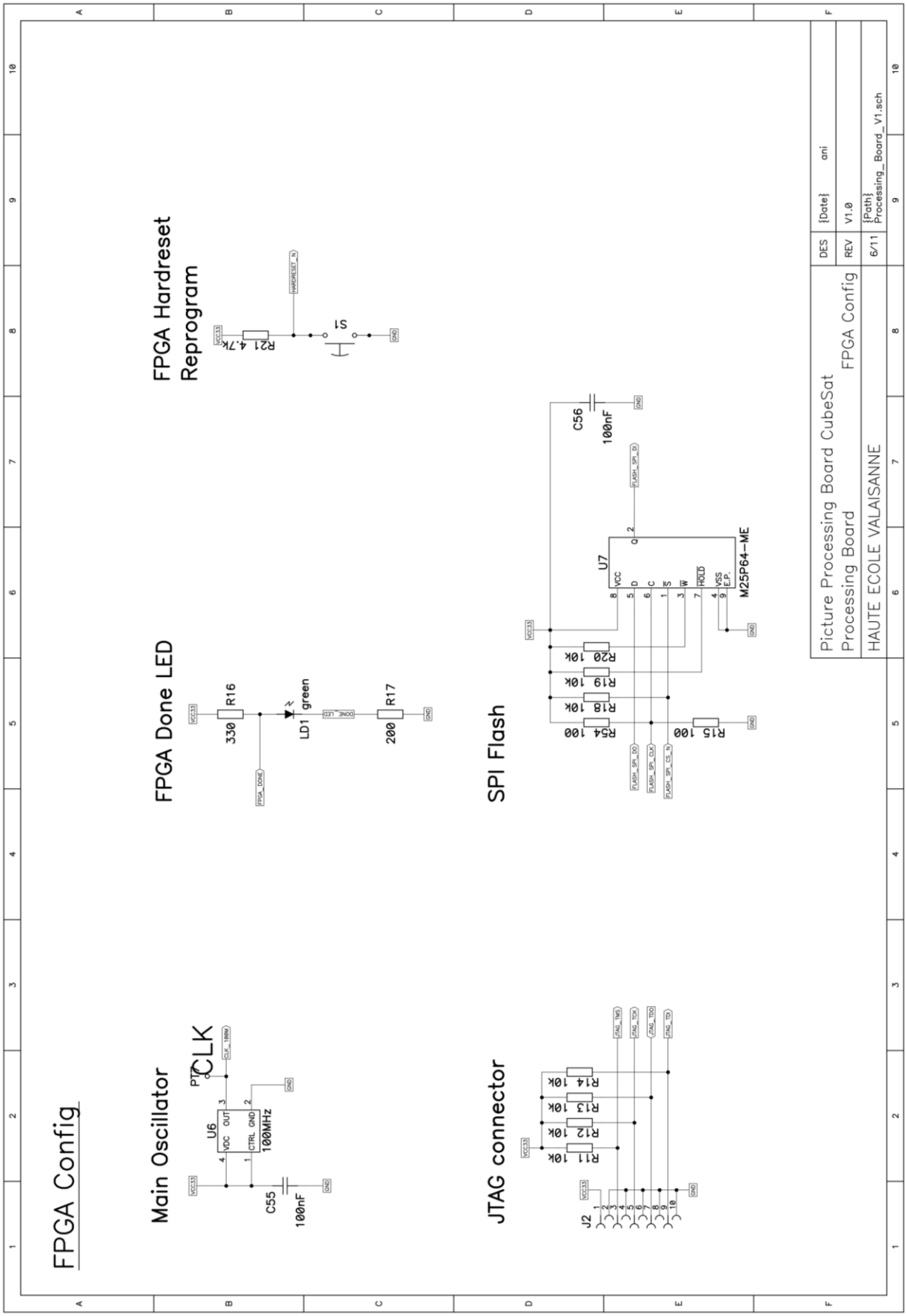
DES	{Date}	ani
REV	V1.0	
1/11	{Path}Processing_Board_V1.sch	

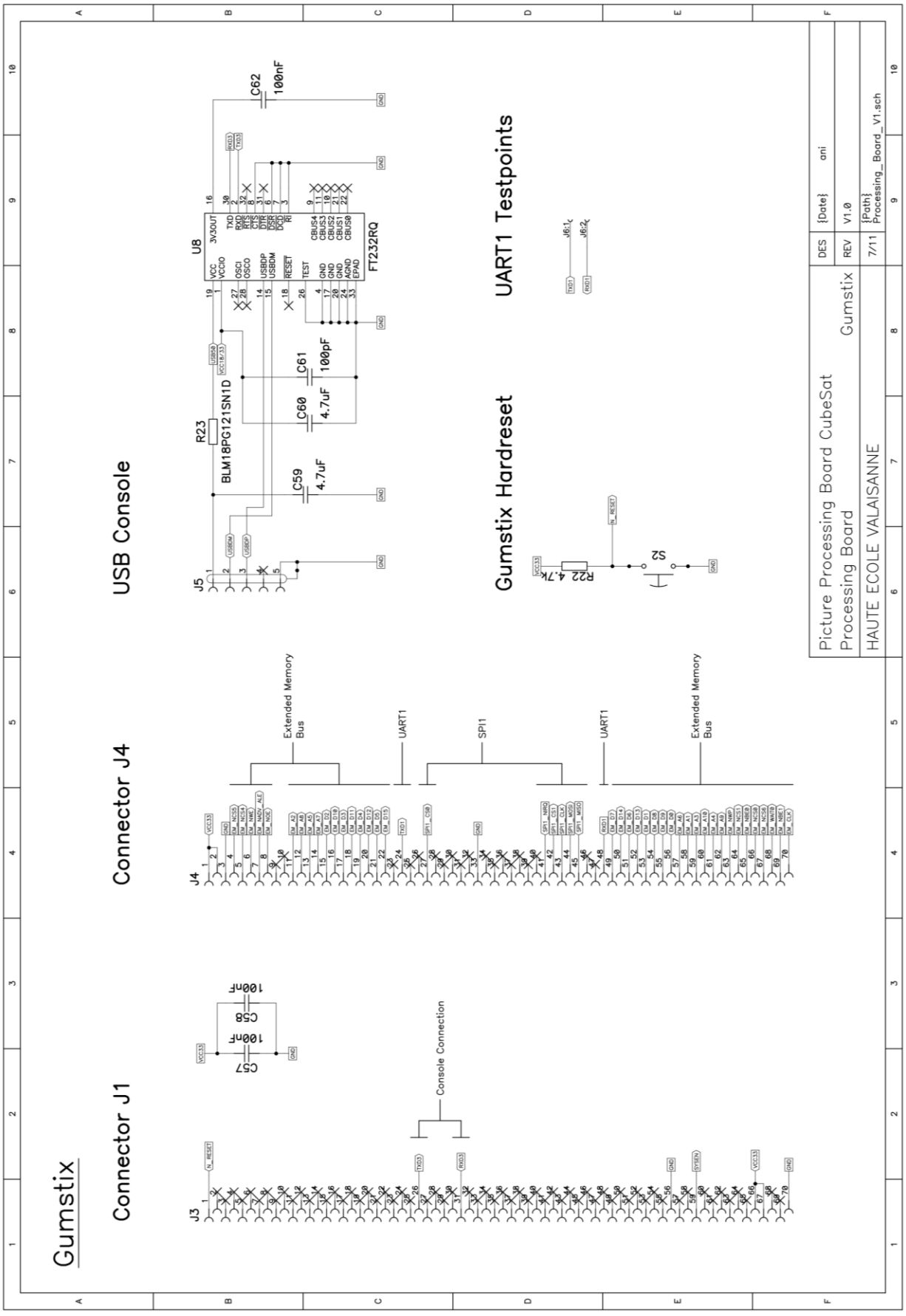

Power Supply
3.3V -> From SC196(1)
2.5V/3.3V -> From SC196(2)
1.8V/3.3V -> From SC196(3)
1.2V -> From SC189
DC/DC SC196 Calculation
Vout = Vref*(R1/R2 + 1)
R1 = 560k (fix)
R2 = R1 / (Vout/Vref - 1)
Vref = 0.5V
Def: 10k < R1+R2 < 1Meg
5V INPUT
5.0V
GND
100uF, ESR=10Ohm
3.3V @ 1.5A
2.5V/3.3V @ 1.5A
1.8V/3.3V @ 1.5A
1.2V @ 1.5A
SC196ULTRT
SC189CULTRT
DC/DC SC196 3.3V
R2 = 100k
Vout = 0.5V*(1 + 560k/100k) = 3.3V
R1,R2 with 1%: Vout: 3.245V - 3.356V
DC/DC SC196 2.5V/3.3V
R2 = 140k
Vout1 = 0.5V*(1 + 560k/140k) = 2.5V
R1,R2 with 1%: Vout: 2.460V - 2.540V
R2 = 100k
Vout2 = 0.5V*(1 + 560k/100k) = 3.3V
R1,R2 with 1%: Vout: 3.245V - 3.356V
DC/DC SC196 1.8V/3.3V
R2 = 215k
Vout1 = 0.5V*(1 + 560k/215k) = 1.802V
R1,R2 with 1%: Vout: 1.777V - 1.829V
R2 = 100k
Vout2 = 0.5V*(1 + 560k/100k) = 3.3V
R1,R2 with 1%: Vout: 3.245V - 3.356V
DC/DC SC189 1.2V
Fix Output: 1.2V
Layout Hints
- Decoupling capacitors close to pins
Picture Processing Board CubeSat
Processing Board
Power
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
2/11 {Path}Processing_Board_V1.sch

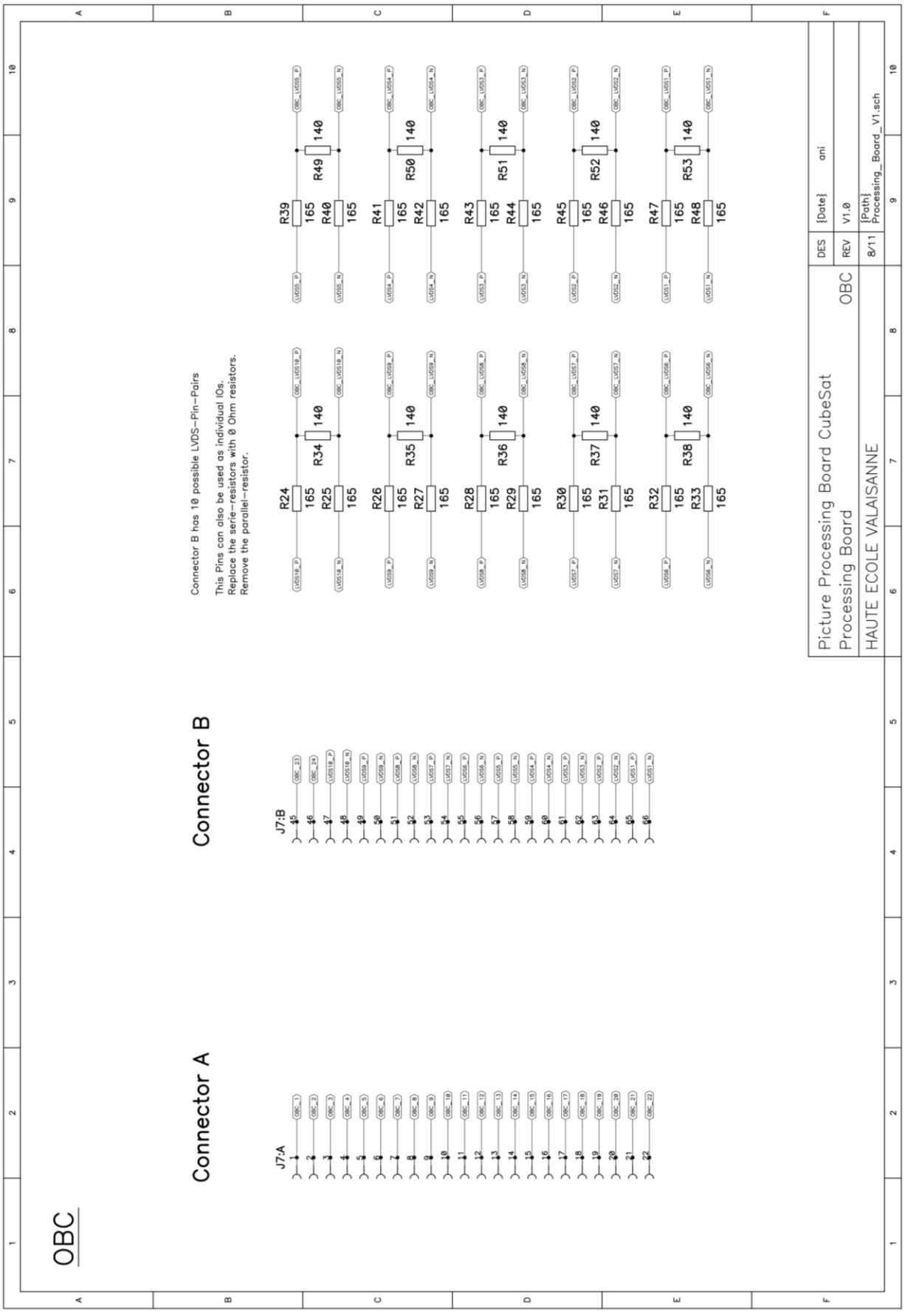

FPGA Decoupling
VCCINT Decoupling
LX45 -> 1x100uF / 1x4.7uF / 2x0.47uF
LX100 -> 1x100uF / 2x4.7uF / 4x0.47uF
LX150 -> 2x100uF / 3x4.7uF / 6x0.47uF
VCCAUX Decoupling
LX45, LX100, LX150 -> 1x100uF / 2x4.7uF / 4x0.47uF
VCC0 Bank 0 Decoupling
LX45, LX100, LX150 -> 1x100uF / 1x4.7uF / 2x0.47uF
VCC0 Bank 1 Decoupling
LX45, LX100, LX150 -> 1x100uF / 1x4.7uF / 4x0.47uF
VCC0 Bank 2 Decoupling
LX45 -> 1x100uF / 1x4.7uF / 2x0.47uF
LX100, LX150 -> 1x100uF / 1x4.7uF / 4x0.47uF
VCC0 Bank 3 Decoupling
LX45, LX100, LX150 -> 1x100uF / 1x4.7uF / 4x0.47uF
FPGA Power
Pins NOT AVAILABLE ON LX45
ONLY AVAILABLE ON LX100 / 150
- VFS
- RFUSE
- VBATT
U5:E
XC6SLX45 / 100 / 150
Picture Processing Board CubeSat
Processing Board
FPGA Power
HAUTE ECOLE VALAISANNE

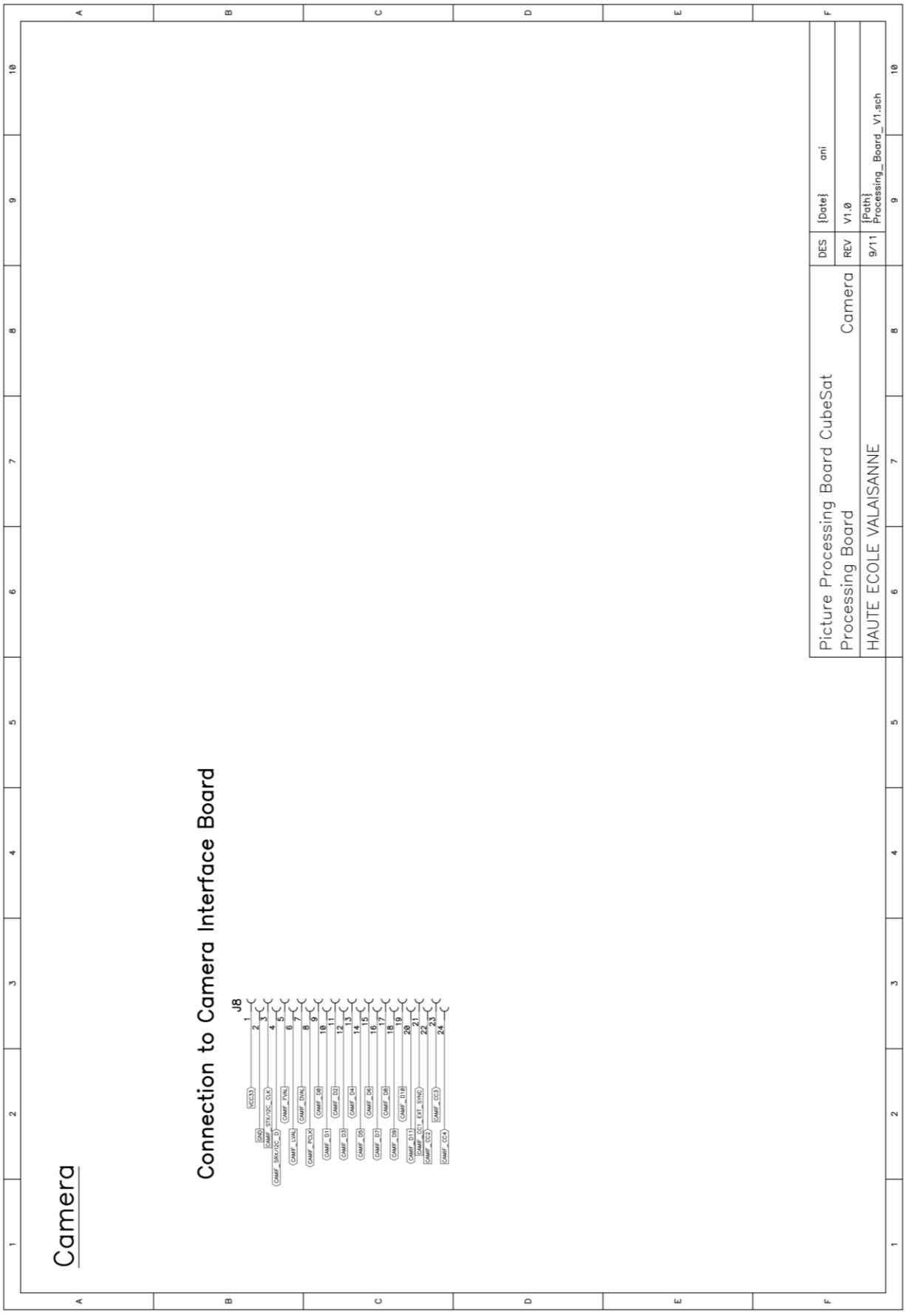

FPGA Bank 0 & 2
U5:A
U5:C
XC6SLX45 / 100 / 150
Pins NOT AVAILABLE ON LX45
ONLY AVAILABLE ON LX100 / 150
Pins NOT AVAILABLE ON LX100
ONLY AVAILABLE ON LX45 / 150
Layout Hints
– LVDS Pins are fix and should not be moved
SPI Programming mode
M0 = 1
M1 = 0
M0 = 1
M1 = 0
R7
R8
R9
R10
4.7k
Freely moved within GCLK of Bank 0
Freely moved within Bank 0
Freely moved within Bank 2
Picture Processing Board CubeSat
Processing Board
FPGA 1
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
4/11 {Path} Processing_Board_V1.sch

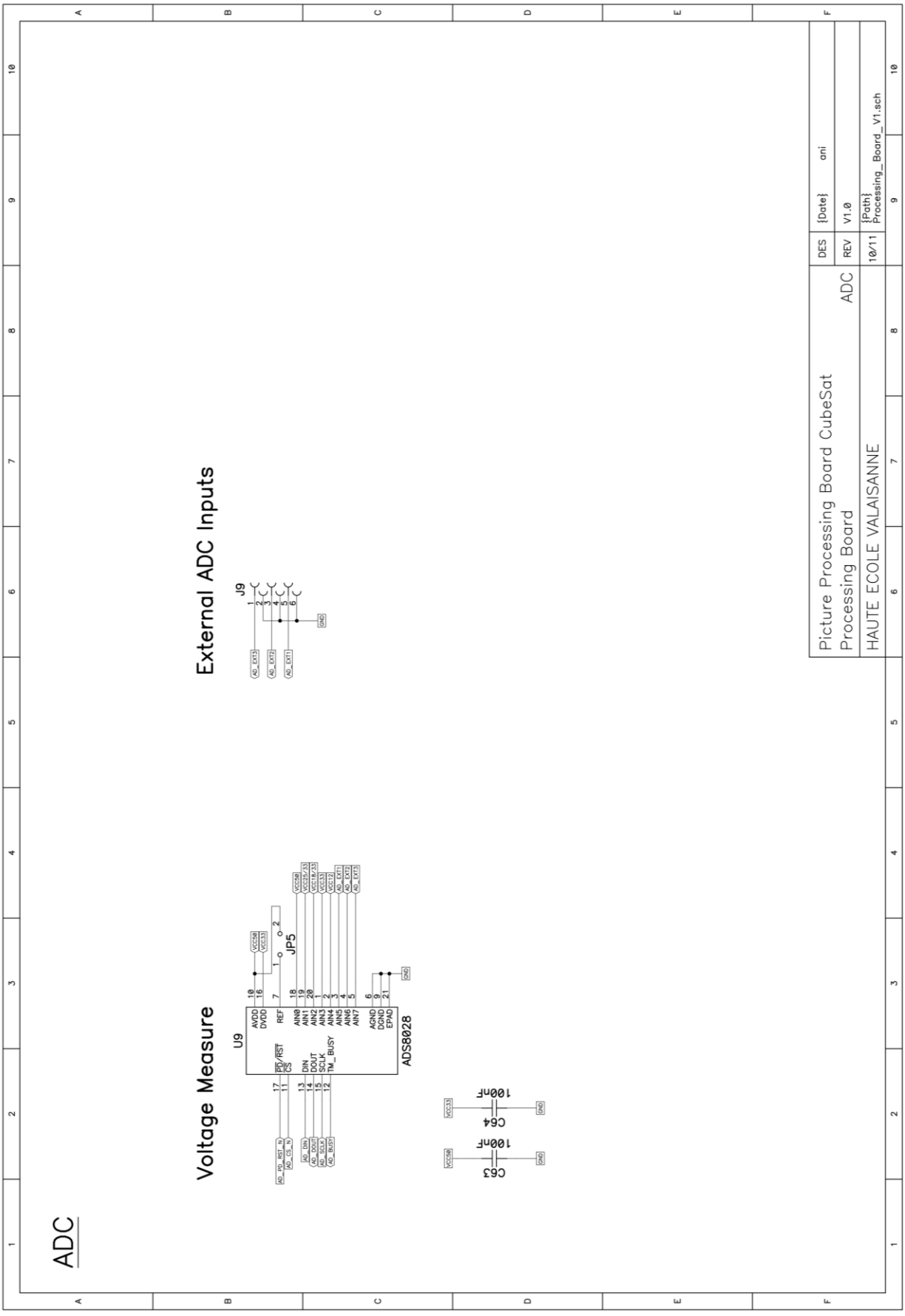

FPGA Bank 1 & 3
U5:B
XC6SLX45 / 100 / 150
U5:D
XC6SLX45 / 100 / 150
Freely moved within GCLK of BANK 1
Freely moved within Bank 1
Freely moved within GCLK of BANK 3
Freely moved within Bank 3
Picture Processing Board CubeSat Processing Board
FPGA 2
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
5/11 {Path} Processing_Board_V1.sch

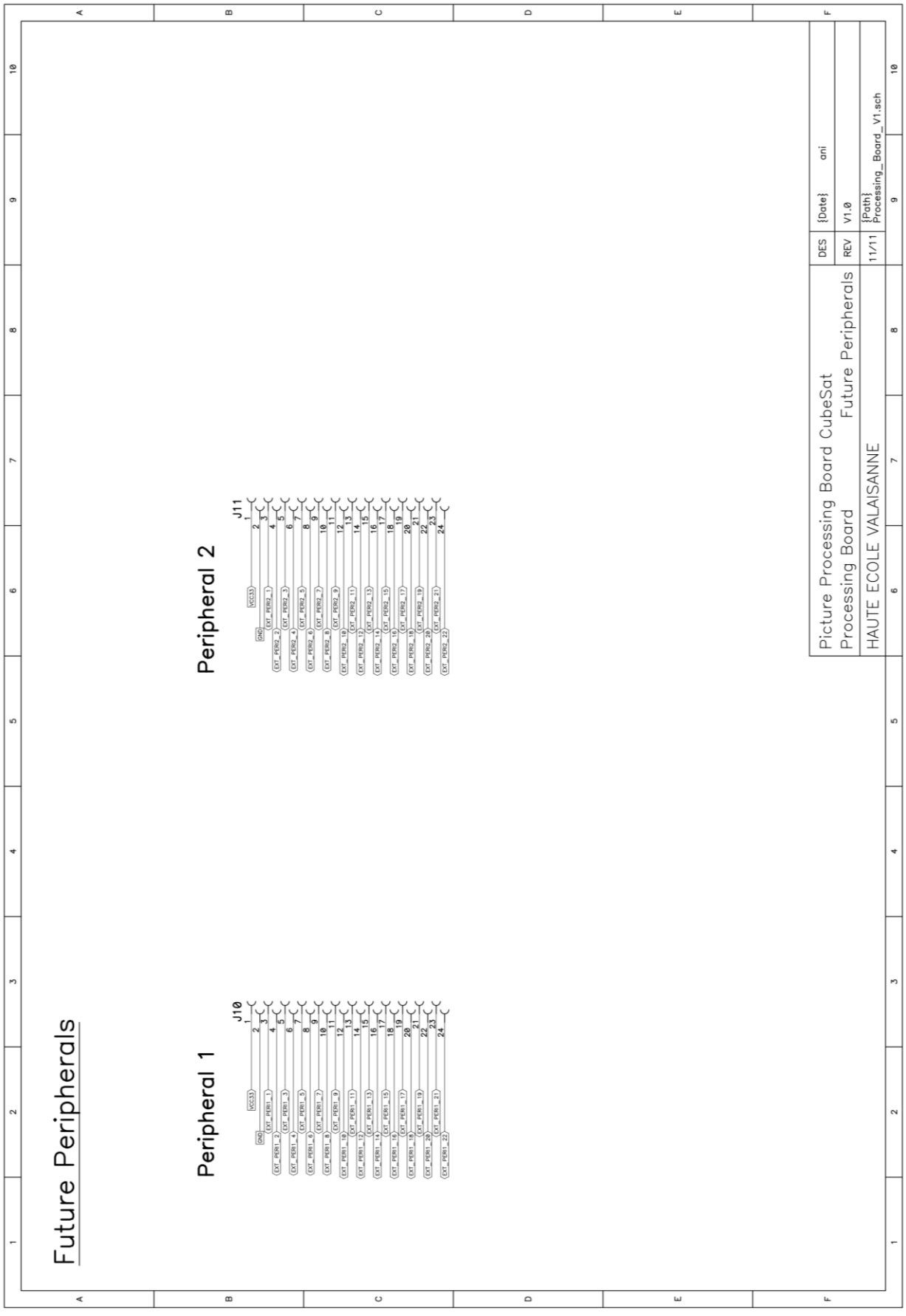

FPGA Config
Main Oscillator
U6
100MHz
C55
100nF
JTAG connector
J2
R11 10k
R12 10k
R13 10k
R14 10k
SPI Flash
U7
M25P64-ME
C56
100nF
R18 10k
R19 10k
R20 10k
R54 100
R15 100
FPGA Done LED
R16
330
LD1 green
R17
200
FPGA Hardreset
Reprogram
R21 4.7K
S1
Picture Processing Board CubeSat
Processing Board
FPGA Config
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
6/11 {Path} Processing_Board_V1.sch


Gumstix
Connector J1
Connector J4
USB Console
Gumstix Hardreset
UART1 Testpoints
Console Connection
Extended Memory Bus
UART1
SPI1
Extended Memory Bus
UART1
J3
J4
J5
J6
U8
FT232RQ
BLM18PG121SN1D
R23
C62 100nF
C61 100pF
C60 4.7uF
C59 4.7uF
C58 100nF
C57 100nF
R22 4.7K
S2
Picture Processing Board CubeSat
Processing Board
Gumstix
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
7/11 {Path} Processing_Board_V1.sch


OBC
Connector A
Connector B
J7:A
J7:B
Connector B has 10 possible LVDS-Pin-Pairs
This Pins can also be used as individual IOs.
Replace the serie-resistors with 0 Ohm resistors.
Remove the parallel-resistor.
R24 165 R25 165 R34 140
R26 165 R27 165 R35 140
R28 165 R29 165 R36 140
R30 165 R31 165 R37 140
R32 165 R33 165 R38 140
R39 165 R40 165 R49 140
R41 165 R42 165 R50 140
R43 165 R44 165 R51 140
R45 165 R46 165 R52 140
R47 165 R48 165 R53 140
Picture Processing Board CubeSat Processing Board
OBC
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
8/11 {Path} Processing_Board_V1.sch


Camera
Connection to Camera Interface Board
J8
VCC33
GND
CAMF_RTX/I2C_CLK
CAMF_SRX/I2C_D
CAMF_FVAL
CAMF_LVAL
CAMF_DVAL
CAMF_PCLK
CAMF_D8
CAMF_D1
CAMF_D2
CAMF_D3
CAMF_D4
CAMF_D5
CAMF_D6
CAMF_D7
CAMF_D9
CAMF_D10
CAMF_D11
CAMF_CC1_EXT_SYNC
CAMF_CC2
CAMF_CC3
CAMF_CC4
Picture Processing Board CubeSat
Processing Board
Camera
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
9/11 {Path}Processing_Board_V1.sch


ADC
Voltage Measure
External ADC Inputs
U9
ADS8028
JP5
J9
C64
100nF
C63
100nF
Picture Processing Board CubeSat
Processing Board
ADC
HAUTE ECOLE VALAISANNE
{Date} ani
V1.0
{Path}
Processing_Board_V1.sch
10/11

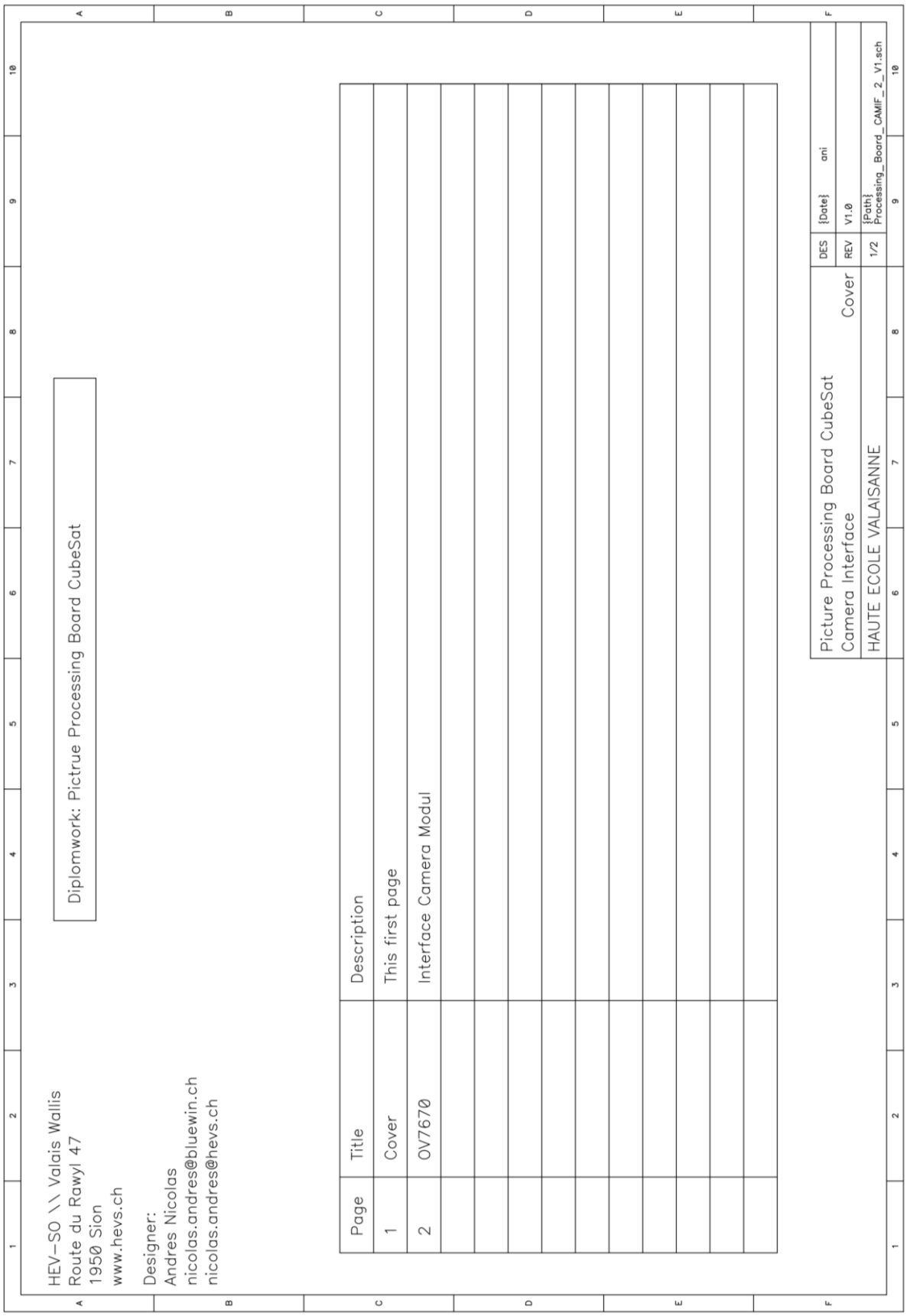

Future Peripherals
Peripheral 1
J10
Peripheral 2
J11
Picture Processing Board CubeSat
Processing Board
Future Peripherals
HAUTE ECOLE VALAISANNE
DES {Date} oni
REV V1.0
11/11 {Path}Processing_Board_V1.sch

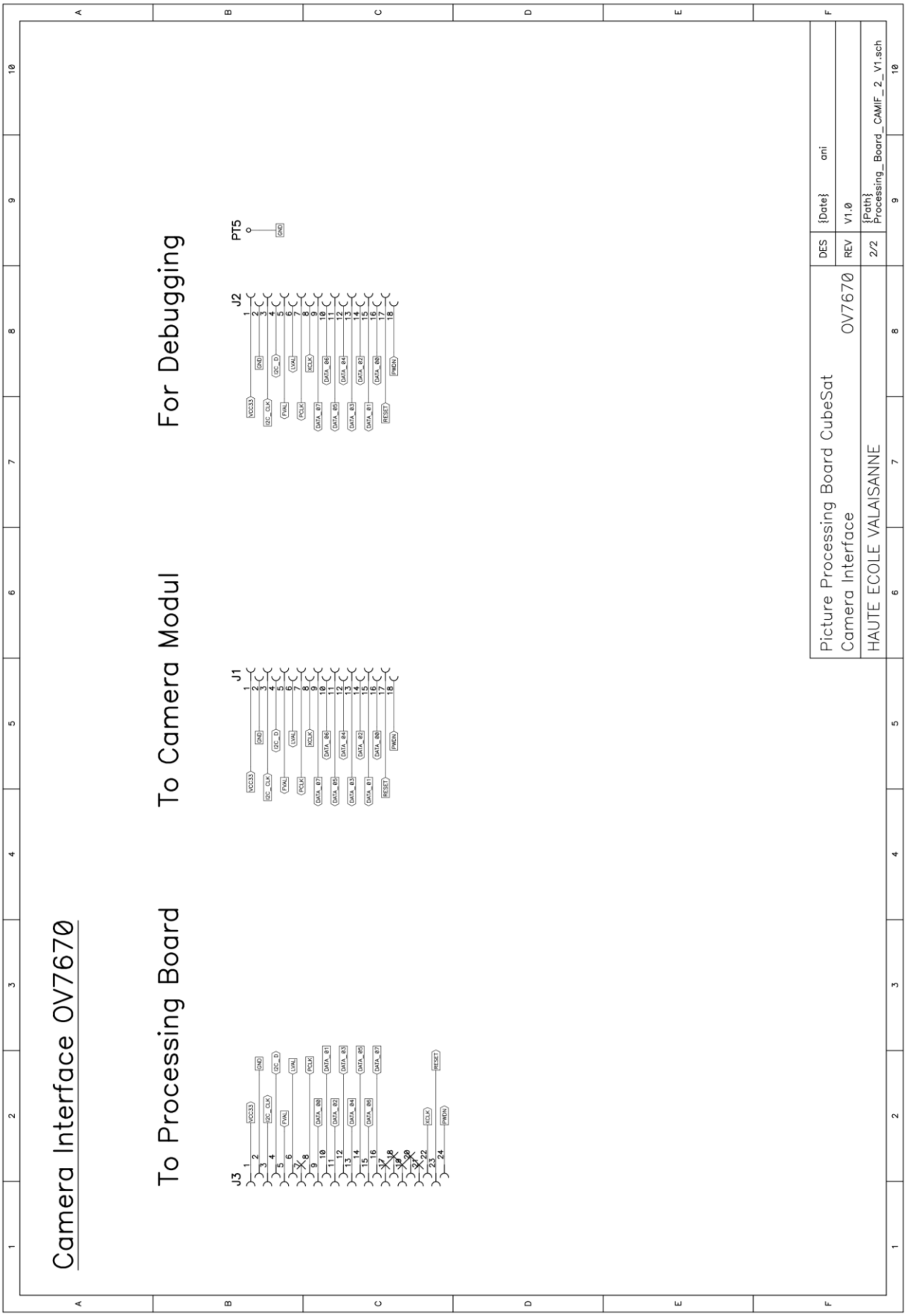
Anhang 3: Kamera Interface

Pin	Photonfocus	OV7670 -Modul	Funktion
1	- (Separat)	3.3V	Speisung
2	GND	GND	Masse
3	RS232 TX	I2C Clock	Ser. Schnittstelle
4	RS232 RX	I2C Data	Ser. Schnittstelle
5	Frame Valid	VSYNC	Zeigt neues Frame/Bild
6	Line Valid	HREF	Zeigt neue Linie
7	Data Valid	-	Daten zulässig oder nicht
8	PixelClock	Pixel Clock	Clock für Pixelübermittlung
9	Data[0]	Data[0]	Datenbits für Bildübermittlung
10	Data[1]	Data[1]	
11	Data[2]	Data[2]	
12	Data[3]	Data[3]	
13	Data[4]	Data[4]	
14	Data[5]	Data[5]	
15	Data[6]	Data[6]	
16	Data[7]	Data[7]	
17	Data[8]	-	
18	Data[9]	-	
19	Data[10]	-	
20	Data[11]	-	
21	CC1: External Sync	-	Externer Trigger für Kamera
22	-	XCLK (20MHz)	Clock für Kameramodul
23	-	Reset ()	Restart
24	-	PowerDown	Ein/Ausschalten

- not Used

Anhang 4: Interfaceboard Schema




Camera Interface OV7670
To Processing Board
To Camera Modul
For Debugging
J3
J1
J2
PT5
Picture Processing Board CubeSat
Camera Interface
OV7670
HAUTE ECOLE VALAISANNE
DES {Date} ani
REV V1.0
2/2 {Path} Processing_Board_CAMIF_2_V1.sch

Anhang 5: Berechnung Speisungen

Power Management: SC196

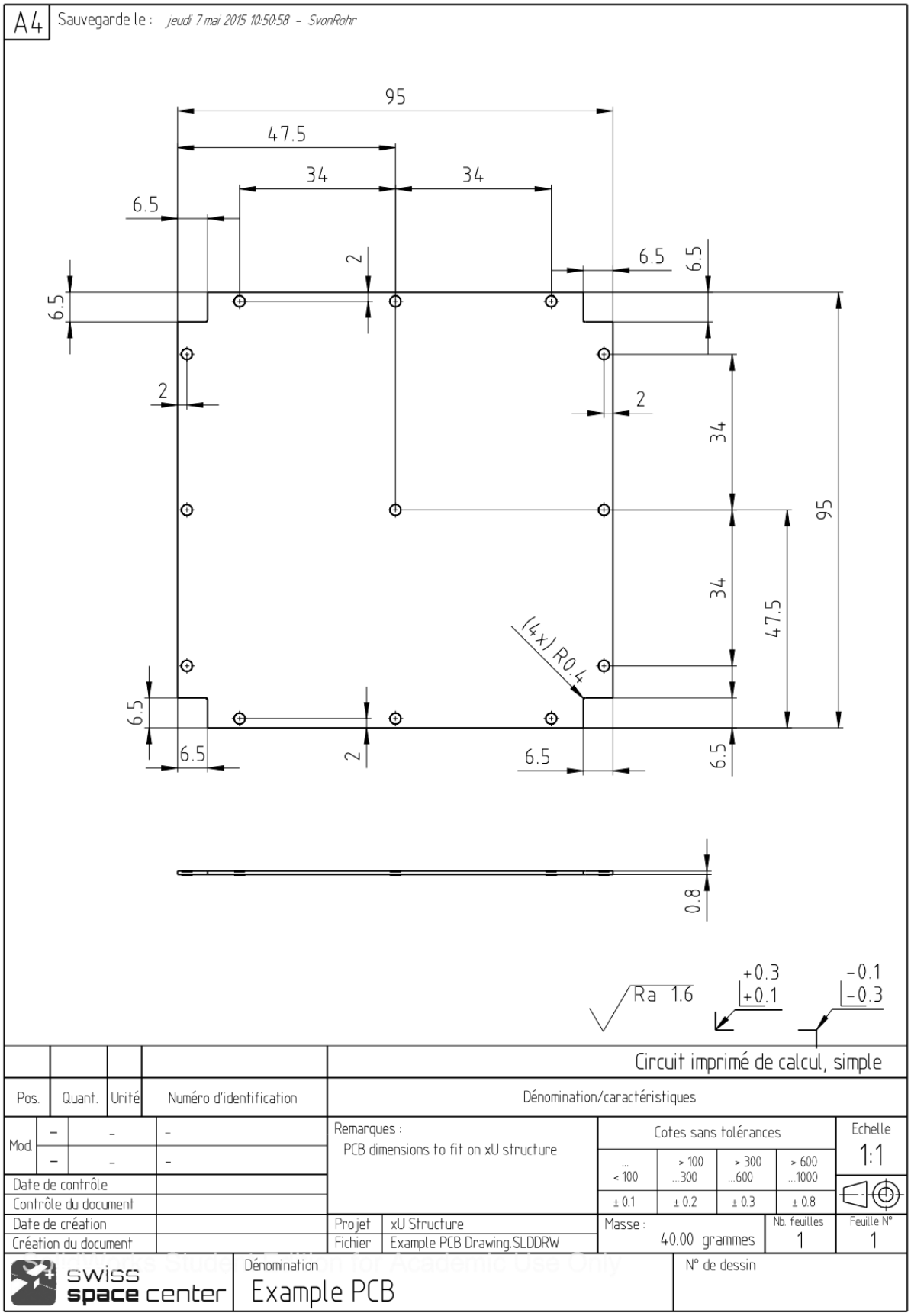
E48-Reihe 1%

3.3V		
VREF	0.5	[V]
VOUT	3.3	[V]
R1	560	[kΩ]
R2	100	[kΩ]
VOUT	3.3	[V]
R1	560	[kΩ]
R2	100	[kΩ]

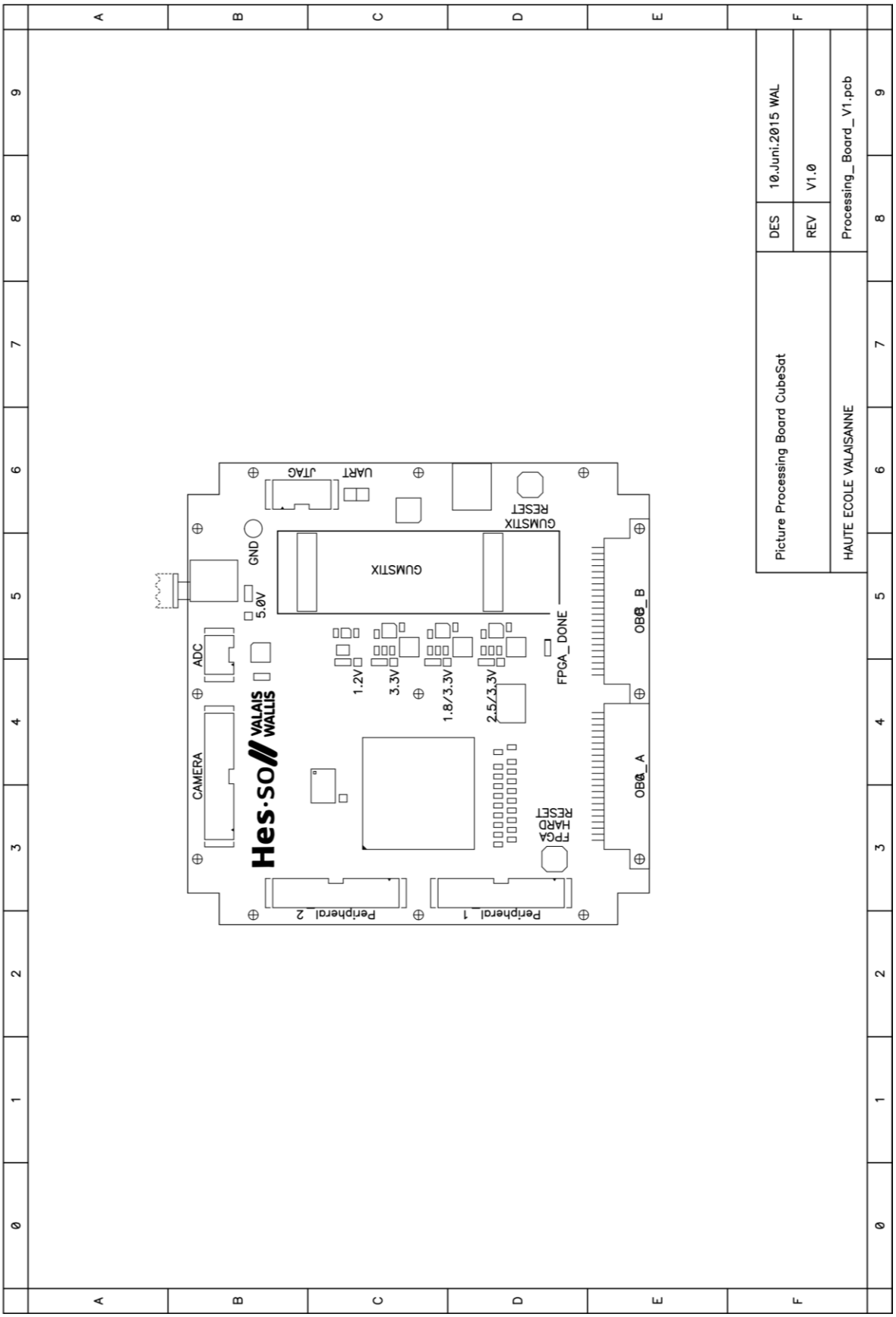
2.5V / 3.3V (Regelbar)		
VREF	0.5	[V]
VOUT1	2.5	[V]
VOUT2	3.3	[V]
R1	560	[kΩ]
R21	140	[kΩ]
R22	100	[kΩ]
VOUT1	2.5	[V]
VOUT2	3.3	[V]
R1	560	[kΩ]
R21	140	[kΩ]
R22	100	[kΩ]

1.8V / 3.3V (Regelbar)		
VREF	0.5	[V]
VOUT1	1.8	[V]
VOUT2	3.3	[V]
R1	560	[kΩ]
R21	215.38462	[kΩ]
R22	100	[kΩ]
VOUT1	1.8023256	[V]
VOUT2	3.3	[V]
R1	560	[kΩ]
R21	215	[kΩ]
R22	100	[kΩ]

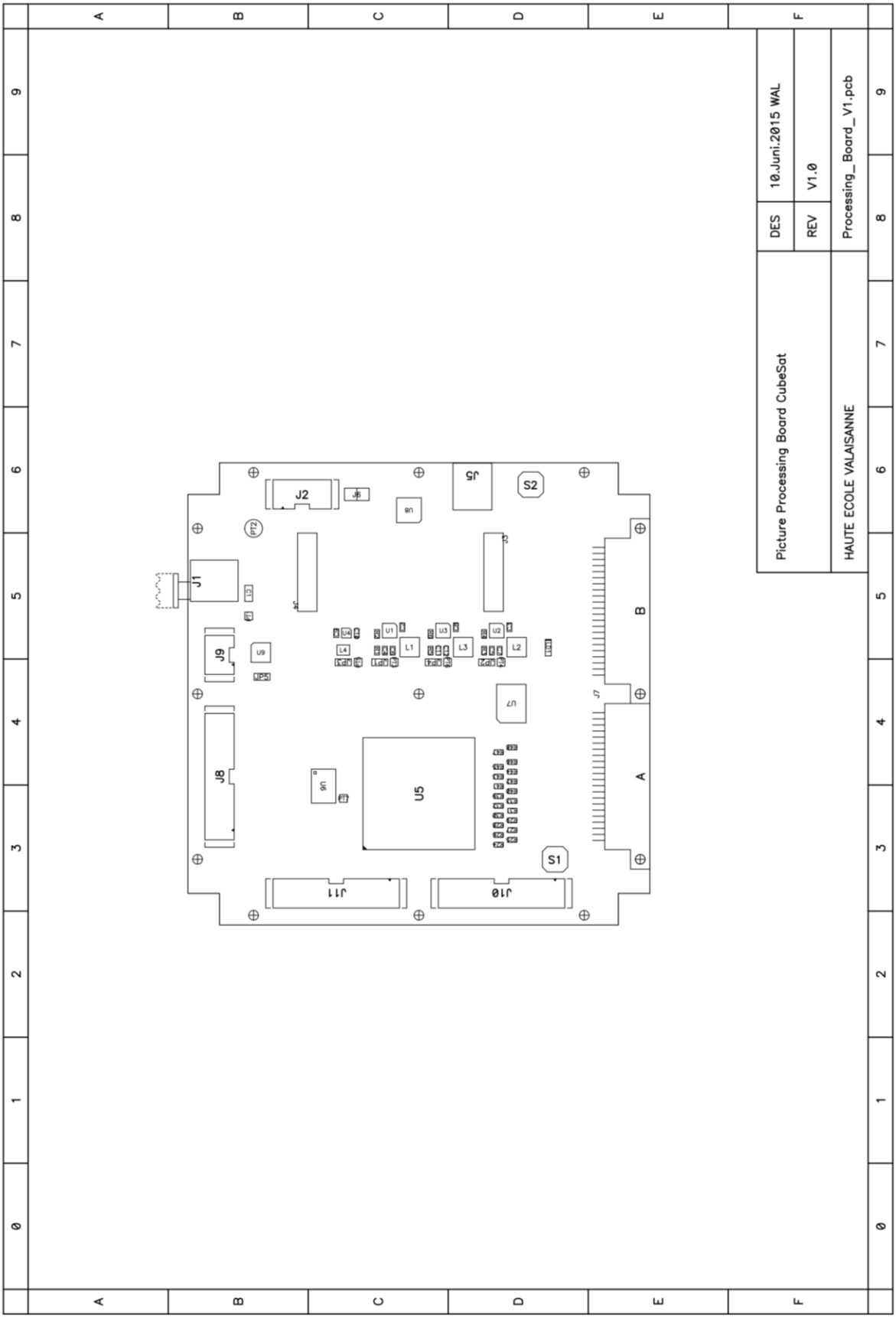
Anhang 6: Abmessungen PCB



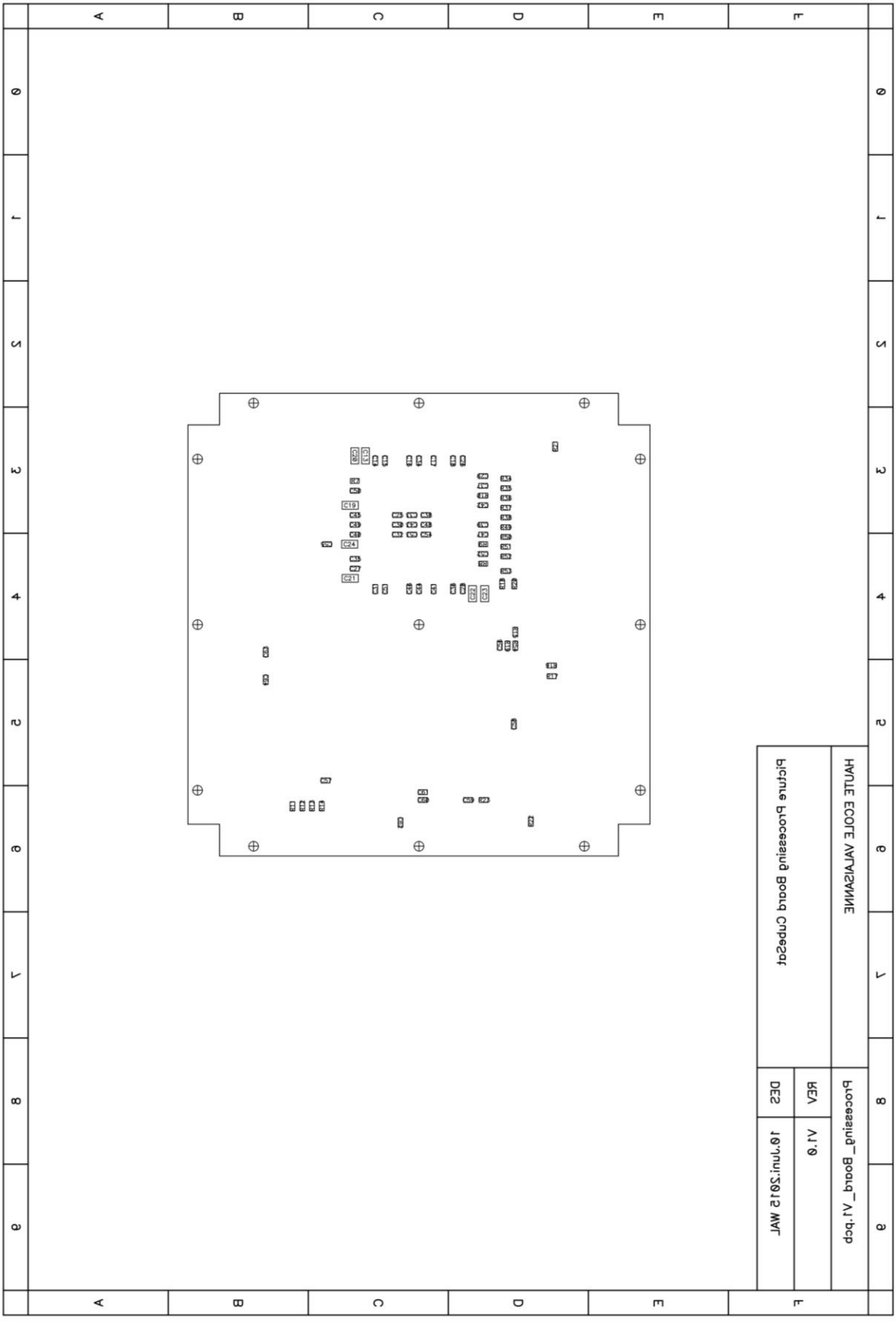
Anhang 7: Bildverarbeitungskarte Layout PCB



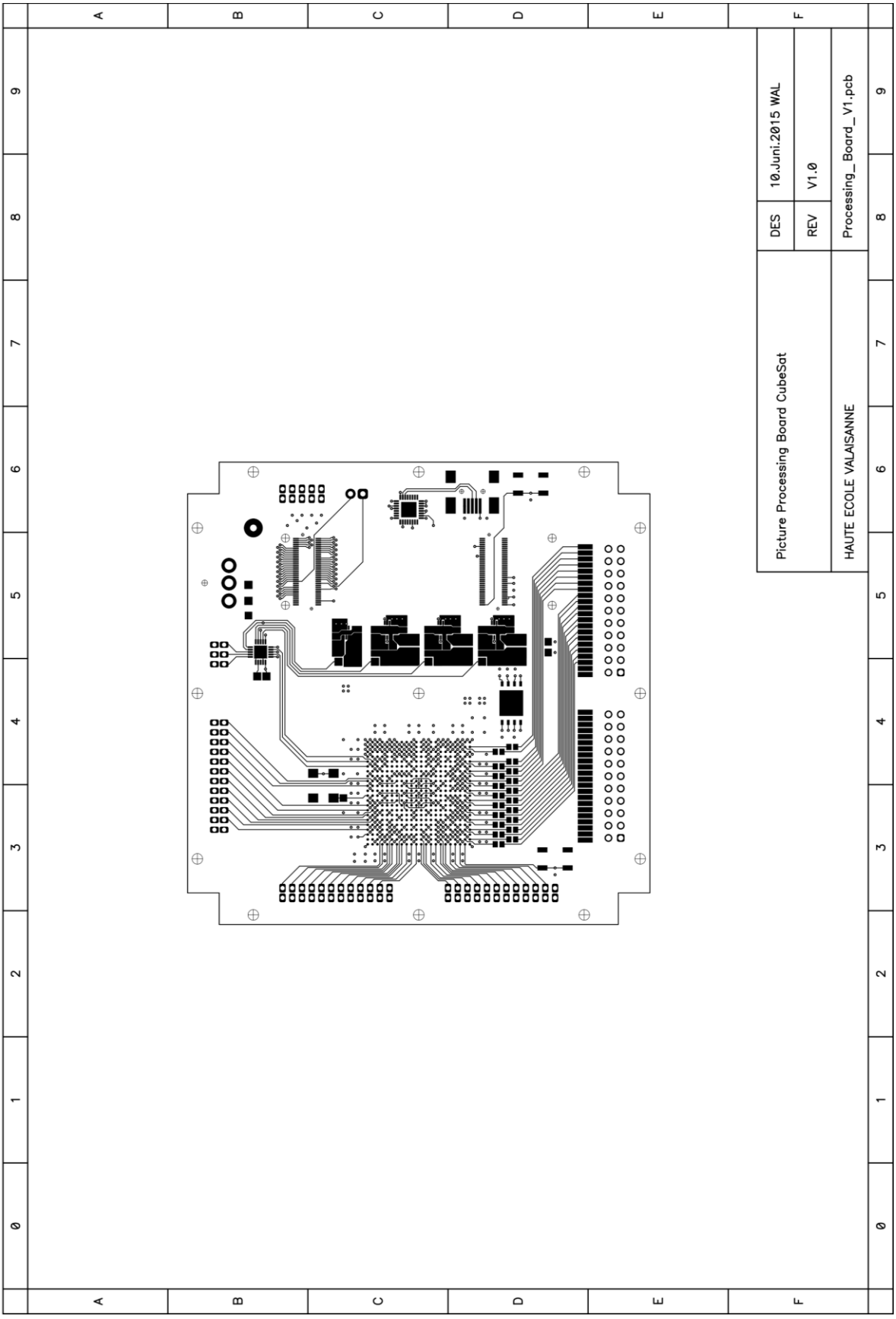
Komponenten Top



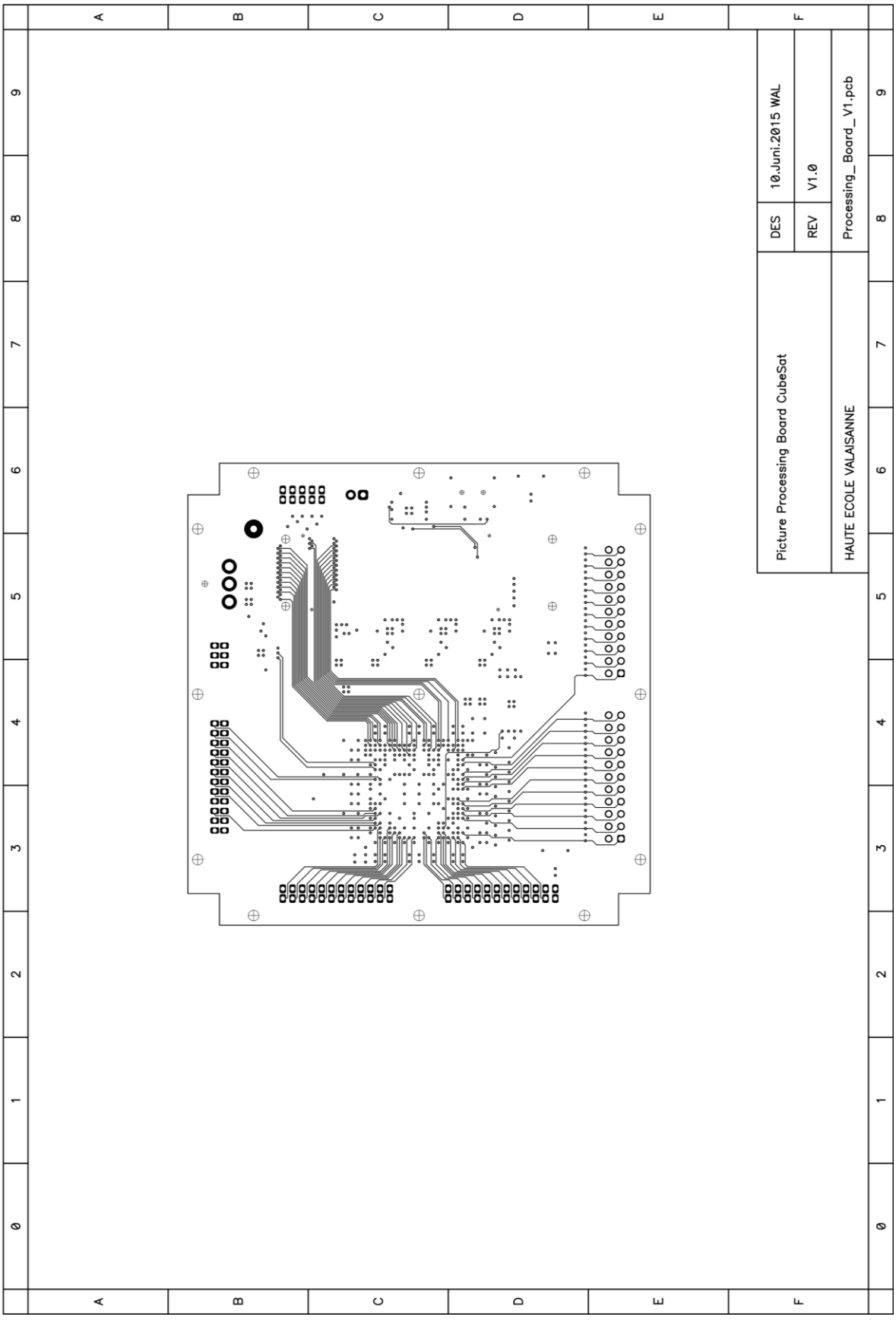
Komponenten Bottom



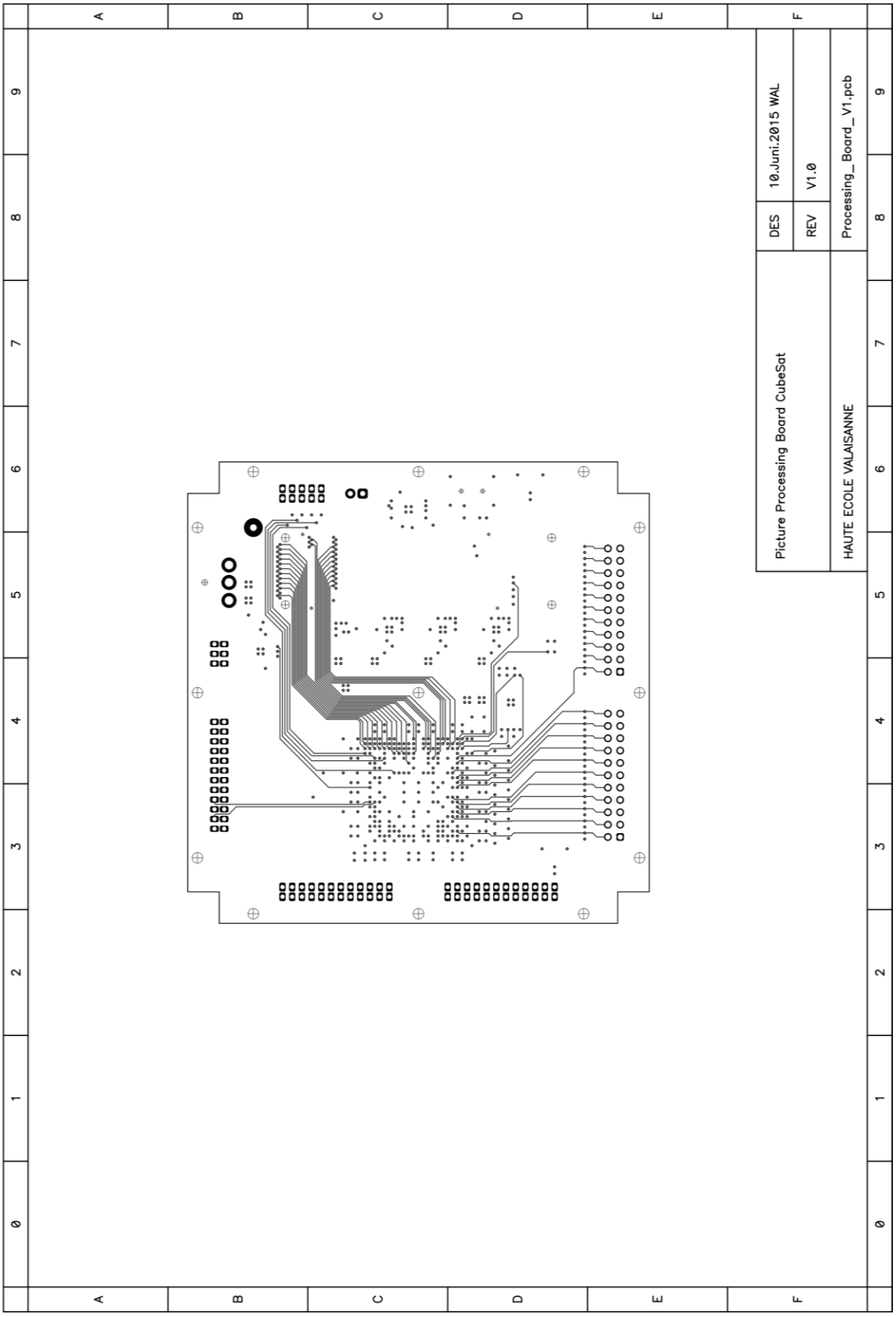
Layer: Top


Picture Processing Board CubeSat
HAUTE ECOLE VALAISANNE
Processing_Board_V1.pcb
DES
10.Juni.2015 WAL
REV
V1.0

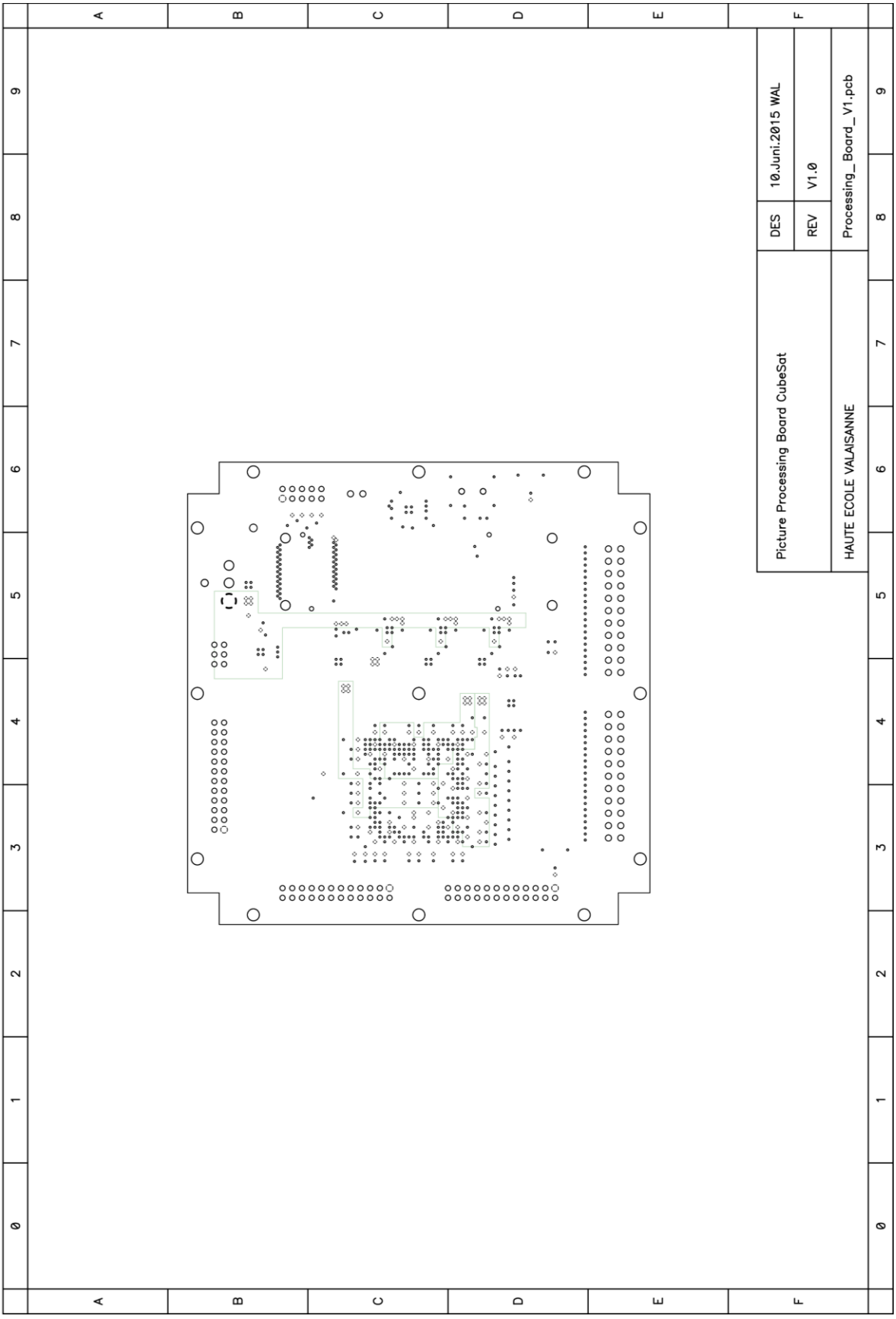
Layer: Sig1


Picture Processing Board CubeSat
HAUTE ECOLE VALAISANNE
Processing_Board_V1.pcb
DES
10.Juni.2015 WAL
REV
V1.0

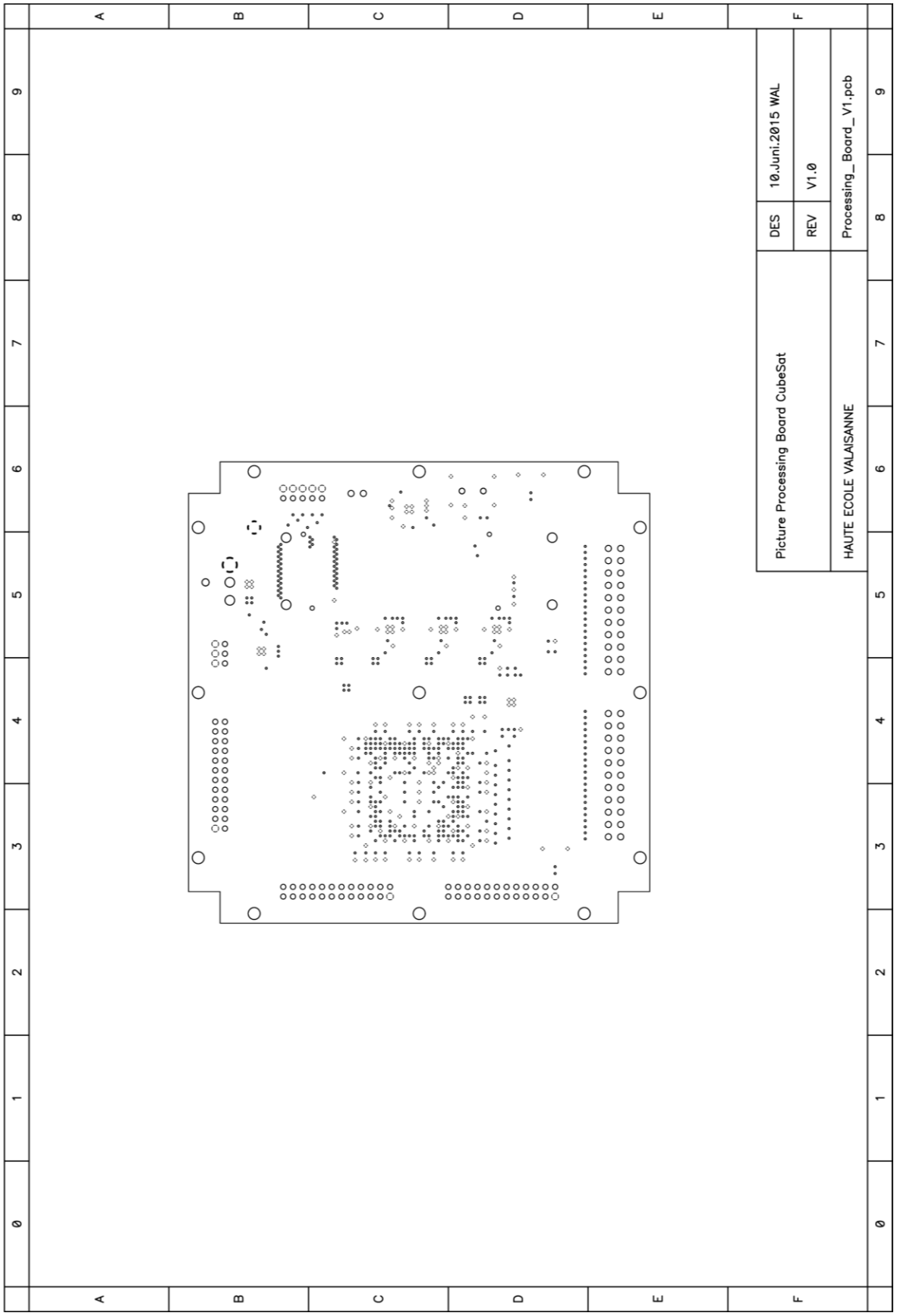
Layer: Sig2


Picture Processing Board CubeSat
HAUTE ECOLE VALAISANNE
DES
10.Juni.2015 WAL
REV
V1.0
Processing_Board_V1.pcb

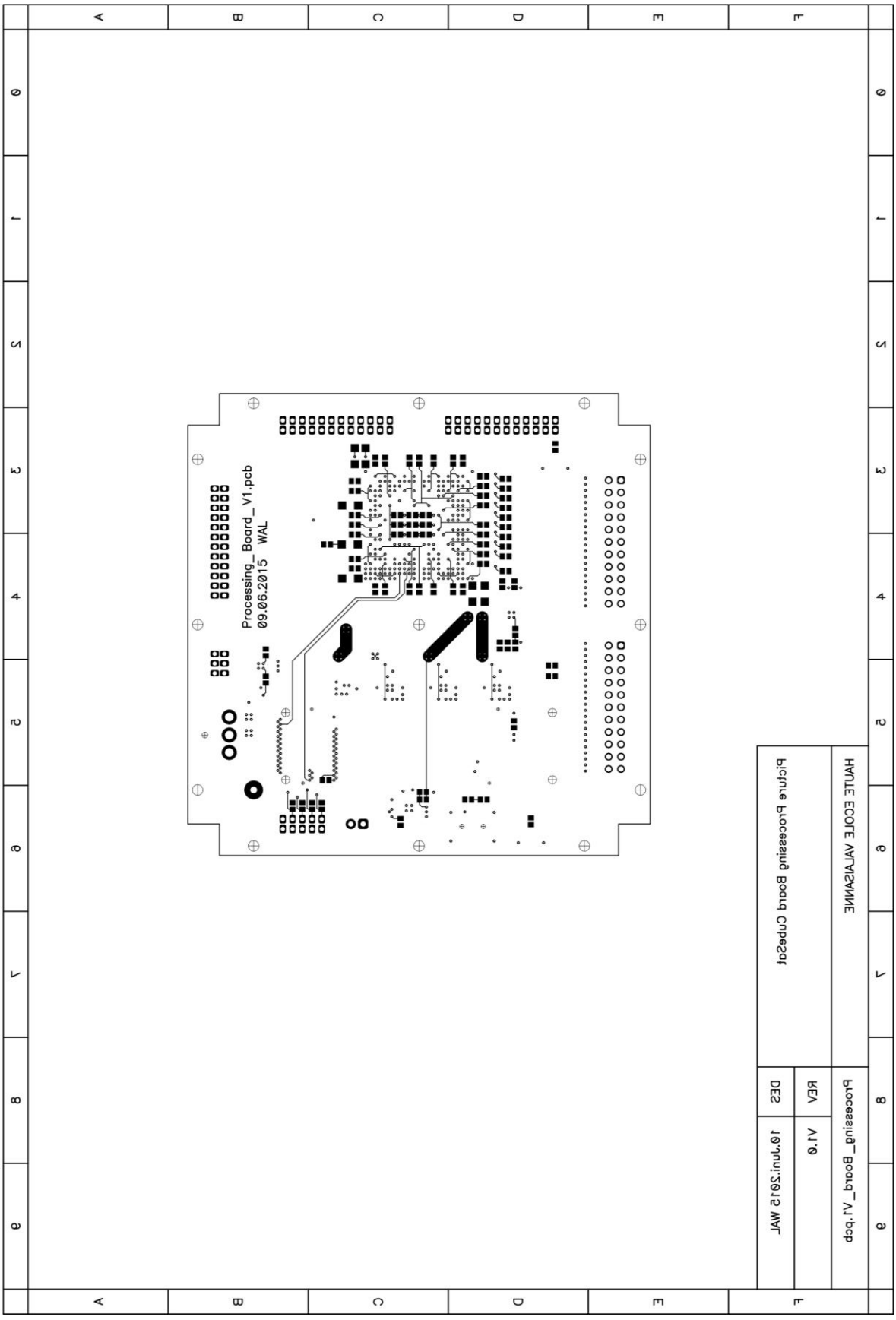
Layer: VCC


Picture Processing Board CubeSat
HAUTE ECOLE VALAISANNE
DES
10.Juni.2015 WAL
REV
V1.0
Processing_Board_V1.pcb

Layer: GND


Picture Processing Board CubeSat
HAUTE ECOLE VALAISANNE
DES 10.Juni.2015 WAL
REV V1.0
Processing_Board_V1.pcb

Layer: Bottom



Anhang 8: Komponentenliste

Picture Processing Board CubeSat	Price List	10.06.2015 V1.0

Part	Quantity	Price/Piece	Price Tot.
Board Components	3	€ 270.04	€ 810.12
Gumstix	1	€ 128.98	€ 128.98
PCB Manufacturing (1 Piece) (Eurocircuit)	7	€ 20.00	€ 140.00
PCB Order Tax (Eurocircuit)			€ 200.00
		€	1'279.10

Board Components

Component	Package	Quantity	Price/Piece	Price Tot.	Distributor	Art. Nr.
FPGA						
Xilinx XC6SLX100-3FGG484C, 326 I/O	FGG484	1	€ 134.96	€ 134.96	Digikey	122-1759-ND
M25P64, 64Mbit Flash NOR, 50MHz	8-VDFN	1	€ 2.87	€ 2.87	Digikey	M25P64-VME6TGCT-ND
Oszillator 100MHz	SMD	1	€ 2.80	€ 2.80	Digikey	CTX318LVCT-ND
Supply						
SC189CULTRT (DC/DC 1.2V)	6-UFDFN	1	€ 1.14	€ 1.14	Digikey	SC189CULCT-ND
SC196CULTRT (DC/DC, Adjustable 0.7-Vin)	10-UFDFN	3	€ 2.82	€ 8.46	Digikey	SC196ULCT-ND
Camera						
Connector Header Sockel, 24Pin, 2mm	2x12,2mm	6	€ 1.14	€ 6.84	Digikey	AE10564-ND
Connector Header Stecker, 24Pin, 2mm	2x12,2mm	6	€ 1.44	€ 8.64	Digikey	AE11130-ND
Flachbandkabel, 1m, AWG28, 24Pin, 1mm	-	2	€ 7.26	€ 14.52	Farnell	2064519
Peripherals						
ADS8028, 8AD Input, SPI	20-QFN	1	€ 8.34	€ 8.34	Digikey	296-30382-1-ND
FT232RQ, USB to Serial	28-SSOP	1	€ 4.45	€ 4.45	Digikey	768-1008-1-ND
General						
Power Connector DC8 (Socket)	-	1	€ 7.90	€ 7.90	Farnell	224947
Power Connector DCPP3 (Plug)	-	1	€ 12.05	€ 12.05	Farnell	224911
Pushbutton PTS525SM15SMTR2 LFS	SMD	2	€ 0.77	€ 1.54	Digikey	CKN9104CT-ND
Mini USB-B Connector	SMD	1	€ 1.70	€ 1.70	Farnell	1753807
Connector Pinheader, 22Pin, 2.54mm	2x11,2.54mm	2	€ -	€ -		
FCI 98414-G06-06ULF (Header, 6Pin) (ADC)	2x3,2mm	1	€ 1.21	€ 1.21	Digikey	609-3103-ND
FCI 98414-G06-10ULF (Header, 10Pin) (JTAG)	2x5,2mm	1	€ 1.11	€ 1.11	Digikey	609-3105-ND
Resistor 0Ω	0603 SMD	3	€ 0.10	€ 0.30	Digikey	P0.0GCT-ND
Resistor 100Ω 1% 0.1W	0603 SMD	2	€ 0.10	€ 0.20	Digikey	P100HCT-ND
Resistor 140Ω 1% 0.1W	0603 SMD	10	€ 0.10	€ 1.00	Digikey	P140HCT-ND
Resistor 165Ω 1% 0.1W	0603 SMD	20	€ 0.10	€ 2.00	Digikey	P165HCT-ND
Resistor 200Ω 1% 0.1W	0603 SMD	1	€ 0.10	€ 0.10	Digikey	P200HCT-ND
Resistor 330Ω 1% 0.1W	0603 SMD	1	€ 0.10	€ 0.10	Digikey	P330HCT-ND
Resistor 4.7kΩ 1% 0.1W	0603 SMD	3	€ 0.10	€ 0.30	Digikey	P4.7KCCT-ND
Resistor 10kΩ 1% 0.1W	0603 SMD	7	€ 0.10	€ 0.70	Digikey	P4.70KHCT-ND
Resistor 100kΩ 1% 0.1W	0603 SMD	3	€ 0.10	€ 0.30	Digikey	P100KHCT-ND
Resistor 140kΩ 1% 0.1W	0603 SMD	1	€ 0.10	€ 0.10	Digikey	P140KHCT-ND
Resistor 215kΩ 1% 0.1W	0603 SMD	1	€ 0.10	€ 0.10	Digikey	P215KHCT-ND
Resistor 560kΩ 1% 0.1W	0603 SMD	3	€ 0.10	€ 0.30	Digikey	P560KHCT-ND
Capacitor 10pF 50V 5% C0G	0603 SMD	3	€ 0.10	€ 0.30	Digikey	445-1269-1-ND
Capacitor 100pF 50V 5% C0G	0603 SMD	1	€ 0.10	€ 0.10	Digikey	445-1281-1-ND
Capacitor 100nF 25V 10% X7R	0603 SMD	7	€ 0.10	€ 0.70	Digikey	445-1316-1-ND
Capacitor 470nF 10V 10% X5R	0603 SMD	26	€ 0.10	€ 2.60	Digikey	445-1320-1-ND
Capacitor 4.7uF 6.3V 10% X5R	0603 SMD	11	€ 0.18	€ 1.98	Digikey	478-1416-1-ND
Capacitor 10uF 6.3V 10% X5R	0603 SMD	4	€ 0.24	€ 0.96	Digikey	478-1417-1-ND
Capacitor 22uF 6.3V 20% X7R	0603 SMD	4	€ 0.57	€ 2.28	Digikey	587-1334-1-ND
Capacitor 100uF 6.3V 20% X5R	1206 SMD	7	€ 1.38	€ 9.66	Digikey	490-4539-1-ND
Capacitor 100uF 6.3V ESR 1Ohm	1206 SMD	1	€ 1.46	€ 1.46	Farnell	1961933
BLM18PG121SN1D (Ferrite 120Ohm)	0603 SMD	1	€ 0.10	€ 0.10	Digikey	490-1037-1-ND
Inductivity 1uH, 55mOhm, 20%, LQM2HPN1R0MG0L	1008 SMD	1	€ 0.34	€ 0.34	Digikey	490-5112-1-ND
Inductivity 4.7uH, 98mOhm,30%, VLCF4020T-4R7N1R2	SMD	3	€ 0.76	€ 2.28	Digikey	445-3184-1-ND
LED Green, SMD	0805 SMD	1	€ -	€ -		
FTDI, USB-RS232-WE-1800-BT_3.3V, 1.8m	USB Wired	1	€ 23.25	€ 23.25	Digikey	768-1077-ND
			€	270.04		

Gumstix

1 US-Dollar= € 0.89

Component	Package	Quantity	Price/Piece	Price Tot.	Distributor	Art. Nr.
Gumstix Overo COM "WaterStorm"	-	1	€ 124.15	€ 124.15	Gumstix Store	GUM3703W
Hirose 70PIN AVX Connector (10 Piece)	-	0.2	€ 22.33	€ 4.47	Gumstix Store	KIT0023
Retaining Spacers 48/48 (White) (50 Piece)	-	0.08	€ 4.47	€ 0.36	Gumstix Store	KIT0027
			€	128.98		

Others

Omnivision OV7670 VGA Camera Modul	-	1	Fr. 29.70	Fr. 29.70	Play-Zone	P00000176

➔ Die Excel-Tabelle ist zusätzlich als digitaler Anhang vorhanden.

Anhang 10: Pinzuweisung FPGA

```
#-----------------------------------------------------------------------------
# Clock and reset
#
NET     "clock"                     LOC = "A11" ;           # System 100MHz
NET     "clock"  TNM_NET = "clock";
TIMESPEC "TS_clock" = PERIOD clock 10 ns HIGH 50.00%;

NET     "rst_n"                     LOC = "D10" | PULLUP ;  # SOFTRESET_N


#-----------------------------------------------------------------------------
# Test
#
# Out   Ext_Peri1
NET     "testOut<0>"                LOC = "Y1" ;
NET     "testOut<1>"                LOC = "Y2" ;
NET     "testOut<2>"                LOC = "W1" ;
NET     "testOut<3>"                LOC = "W3" ;
NET     "testOut<4>"                LOC = "V1" ;
NET     "testOut<5>"                LOC = "V3" ;
NET     "testOut<6>"                LOC = "U1" ;
NET     "testOut<7>"                LOC = "U3" ;
NET     "testOut<8>"                LOC = "T2" ;
NET     "testOut<9>"                LOC = "T4" ;
NET     "testOut<10>"               LOC = "T1" ;
NET     "testOut<11>"               LOC = "T3" ;
NET     "testOut<12>"               LOC = "R1" ;
NET     "testOut<13>"               LOC = "R4" ;
NET     "testOut<14>"               LOC = "P1" ;
NET     "testOut<15>"               LOC = "R3" ;
NET     "testOut<16>"               LOC = "N1" ;
NET     "testOut<17>"               LOC = "N3" ;
NET     "testOut<18>"               LOC = "M2" ;
NET     "testOut<19>"               LOC = "M4" ;
NET     "testOut<20>"               LOC = "M1" ;
NET     "testOut<21>"               LOC = "M3" ;


# RS232
NET     "testRx"                    LOC = "L1" ;    # Ext_Peri2.1
NET     "testTx"                    LOC = "L3" ;    # Ext_Peri2.2


# Camera Interface OV7670
NET     "camif_tx_sclk"             LOC = "A4" | PULLUP;    # I2C Clock
NET     "camif_rx_sdata"            LOC = "B10" | PULLUP;   # I2C Data

NET     "camif_fval"                LOC = "A5" ;
NET     "camif_lval"                LOC = "C5" ;
NET     "camif_pclk"                LOC = "A10" ;

NET     "camif_data<11>"            LOC = "C9" ;
NET     "camif_data<10>"            LOC = "A13" ;
NET     "camif_data<9>"             LOC = "C8" ;
NET     "camif_data<8>"             LOC = "A9" ;
NET     "camif_data<7>"             LOC = "D8" ;
NET     "camif_data<6>"             LOC = "A8" ;
```

```
58  NET     "camif_data<5>"             LOC = "C7" ;
59  NET     "camif_data<4>"             LOC = "B8" ;
60  NET     "camif_data<3>"             LOC = "C6" ;
61  NET     "camif_data<2>"             LOC = "A7" ;
62  NET     "camif_data<1>"             LOC = "D6" ;
63  NET     "camif_data<0>"             LOC = "A6" ;
64
65  NET     "camif_cc<0>"               LOC = "A14" ;
66  NET     "camif_cc<1>"               LOC = "C14" ;   # XLCK
67  NET     "camif_cc<2>"               LOC = "B14" ;   # Reset
68  NET     "camif_cc<3>"               LOC = "D14" ;   # PowerDown
69
70
71  # ADC
72  NET     "ad_cs_n"                   LOC = "C16" ;
73  NET     "ad_sclk"                   LOC = "B12" ;
74  NET     "ad_din"                    LOC = "A18" ;   # Dout of ADC
75  NET     "ad_dout"                   LOC = "C17" ;   # Din of ADC
76  NET     "ad_pd_rst_n"               LOC = "B18" ;
77  NET     "ad_busy"                   LOC = "A17" ;
78
79
80  # Gumstix: SPI
81  NET     "spi_clk"                   LOC = "H21" ;
82  NET     "spi_cs0"                   LOC = "W22" ;
83  NET     "spi_mosi"                  LOC = "B22" ;   # MOSI of Gumstix
84  NET     "spi_miso"                  LOC = "A21" ;   # MISO of Gumstix
```

Anhang 14: processPictureData_RTL

```
1  --
2  -- VHDL Architecture Camera.processPictureData.RTL
3  --
4  -- Created:
5  --          by - nicolas.andres.UNKNOWN (WE5403)
6  --          at - 10:08:08 19.06.2015
7  --
8  -- using Mentor Graphics HDL Designer(TM) 2009.2 (Build 10)
9  --
10 ARCHITECTURE RTL OF processPictureData IS
11
12 -- Edge detection of inputs
13   signal vsync_old : std_ulogic;
14   signal href_old : std_ulogic;
15   signal pclk_old : std_ulogic;
16   signal newFrame : std_ulogic;
17   signal newLine : std_ulogic;
18   signal newPixel : std_ulogic;
19
20   constant wBit : positive := requiredBitNb(pictureW);
21   constant hBit : positive := requiredBitNb(pictureH);
22   constant width : unsigned(wBit-1 downto 0) := to_unsigned(pictureW, wBit);
23   constant height : unsigned(hBit-1 downto 0) := to_unsigned(pictureH, hBit);
24   signal w : unsigned(wBit-1 downto 0);
25   signal h : unsigned(hBit-1 downto 0);
26
27   signal addr : unsigned(camRamAddrBitNb-1 downto 0);
28
29   -- For counting Frames, only save every 64. Frame = every 2s an Image
30   signal cnt : unsigned(5 downto 0);
31
32 BEGIN
33
34   -- get rising edge of these signals
35   newFrame <= vsync and not vsync_old;
36   newLine <= href and not href_old;
37   newPixel <= pclk and not pclk_old;
38
39   -- save old-state of signals, where edge will be detected
40   edgeDetection : process(clock,reset)
41   begin
42     if reset = '1' then
43       vsync_old <= '0';
44       href_old <= '0';
45       pclk_old <= '0';
46     elsif rising_edge(clock) then
47       vsync_old <= vsync;
48       href_old <= href;
49       pclk_old <= pclk;
50     end if;
51   end process edgeDetection;
52
53   -- Read image data from the parallel interface
54   readData : process(clock, reset)
55   begin
56     if reset = '1' then
57       addr <= (others => '0');
```

```
58          addrOut <= (others => '0');
59          dataOut <= (others => '0');
60          cnt <= (others => '1');
61        elsif rising_edge(clock) then
62          writeEn <= '0';
63          wrNewData <= '0';
64
65          -- Count Camera Images
66          if newFrame = '1' then
67            cnt <= cnt + 1;
68            addr <= (others => '0');  -- set address to 0 at beginning
69          end if;
70
71          if cnt = 0 then
72            -- Block write if read is in progress
73            if (enProcess = '1') and (rdData = '0') then
74              if addr < 307200 then   -- Block until all Pixels are written
75                wrNewData <= '1';
76              end if;
77
78              -- if new Pixel Data, save it to Block-RAM
79              if newPixel = '1'  AND href = '1' then
80                writeEn <= '1';
81                dataOut <= dataIn;
82                addrOut <= std_ulogic_vector(addr);
83                addr <= addr + 1;
84              end if;
85            end if;
86          end if;
87        end if;
88      end process readData;
89
90  END ARCHITECTURE RTL;
```

Anhang 18: spiRead_RTL

```
1   --
2   -- VHDL Architecture SPI.spiRead.RTL
3   --
4   -- Created:
5   --          by - nicolas.andres.UNKNOWN (WE5403)
6   --          at - 09:15:09 03.07.2015
7   --
8   -- using Mentor Graphics HDL Designer(TM) 2009.2 (Build 10)
9   --
10  ARCHITECTURE RTL OF spiRead IS
11
12    signal newBit : std_ulogic;
13    signal sSclk_old : std_ulogic;
14    signal cnt : unsigned(3 downto 0);
15    signal data : std_ulogic_vector(dataOut'range);
16    signal readFinish : std_ulogic;
17    signal readFinish1 : std_ulogic;  -- sync with clk spi
18    signal readFinish2 : std_ulogic;  -- sync with clock
19    signal readFinish2_old : std_ulogic;
20
21  BEGIN
22
23    -- detect edge of readFinish
24    readFinish <= readFinish2 and not readFinish2_old;
25
26    -- save old state of signal for edge detection
27    edgeDetection : process(clock, reset)
28    begin
29      if reset = '1' then
30          readFinish2 <= '0';
31          readFinish2_old <= '0';
32      elsif rising_edge(clock) then
33          readFinish2 <= readFinish1;
34          readFinish2_old <= readFinish2;
35      end if;
36    end process edgeDetection;
37
38    -- shift register
39    -- record of incoming spi message
40    readData : process(sSclk, reset)
41    begin
42      if reset = '1' then
43        cnt <= (others => '0');
44        readFinish1 <= '0';
45      elsif rising_edge(sSclk) then
46        readFinish1 <= '0';
47        if sSs = '1' then
48          cnt <= cnt + 1;
49          if cnt < 7 then
50            data <= data(dataBitNb-2 downto 0) & sMosi;
51          elsif cnt = 7 then
52            cnt <= (others => '0');
53            data <= data(dataBitNb-2 downto 0) & sMosi;
54            readFinish1 <= '1';
55          end if;
56        end if;
57      end if;
```

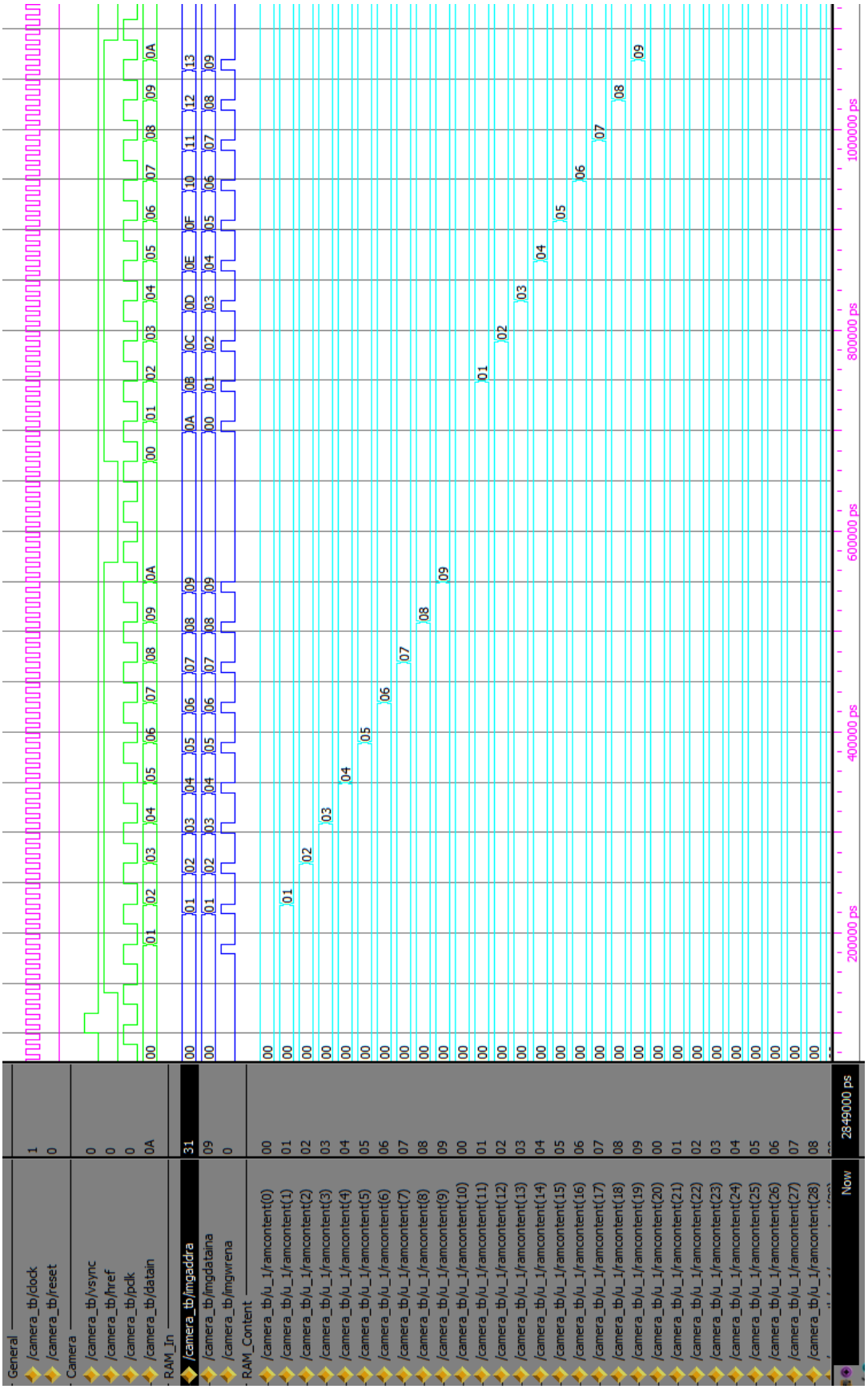
```
58    end process readData;
59
60    -- save a complete spi record in the FIFO
61    writeDataToFIFO : process(clock, reset)
62    begin
63      if reset = '1' then
64        dataOut <= (others => '0');
65        write <= '0';
66      elsif rising_edge(clock) then
67        dataOut <= (others => '0');
68        write <= '0';
69
70        if readFinish = '1' then
71          write <= '1';
72          dataOut <= data;
73        end if;
74      end if;
75    end process writeDataToFIFO;
76
77    sMiso <= '0';
78
79  END ARCHITECTURE RTL;
```

Anhang 20: Hardwaretest PCB

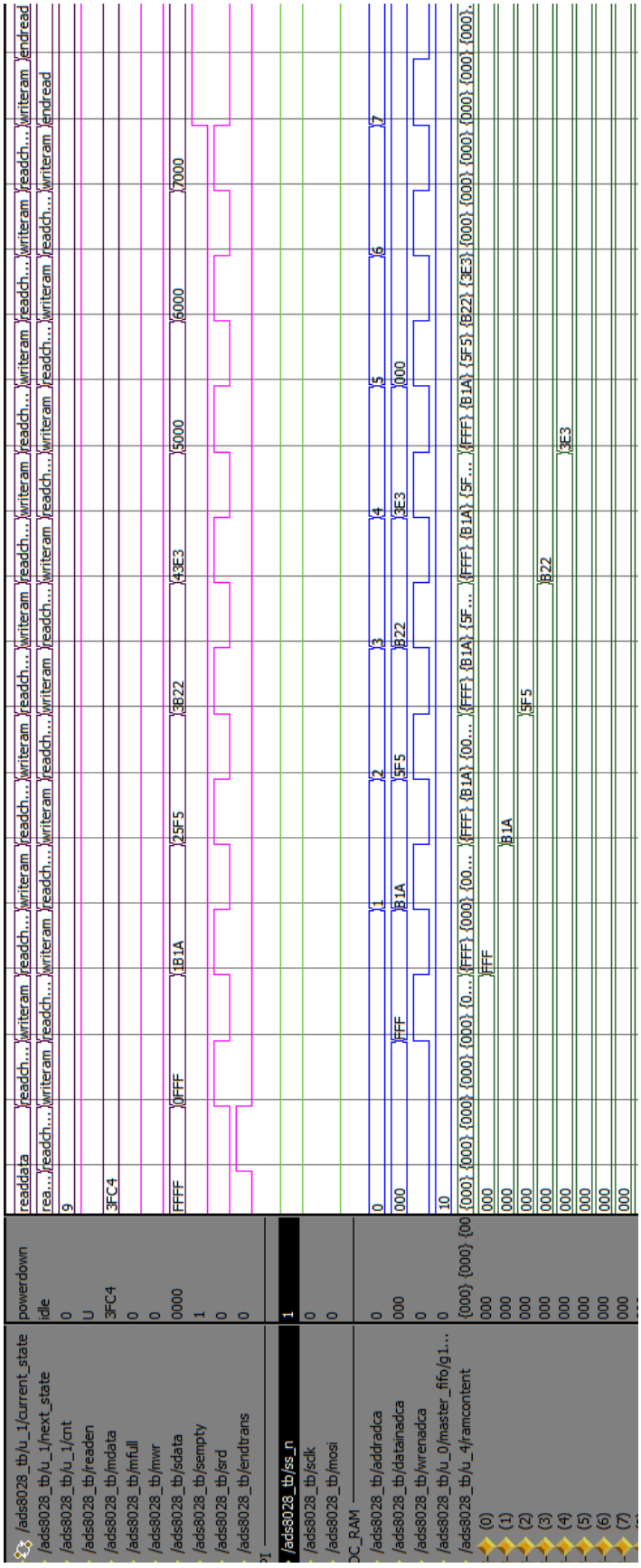
Hardwaretest: Bildverarbeitungskarte V1.0

Test / Erwatetes Resultat	Erhaltenes Resultat	Status	Bemerkungen
Allgemein			
Speisung der Karte mit 5.0V. Messung der verschiedenen Spannungen (5.0V, 3.3V, 3.3V, 1.8V, 1.2V), ohne Last	Speisung: 5.0V Teilspannungen: 3.33V, 3.29V, 1.82V,1.21V	OK	Jumper der Speisungen offen -> Alle Peripherien nicht angeschlossen **Ohne Last!**
Speisung der Karte mit 5.0V. Messung der verschiedenen Spannungen (5.0V, 3.3V, 3.3V, 1.8V, 1.2V), mit Last	Speisung: 5.0V Teilspannungen: 3.35V, 3.29V, 1.83V,1.21V	OK	Jumper der Speisungen geschlossen
Clockfrequenz (100MHz)	100MHz (Sinus)	OK	-
FPGA			
Input durch FPGA an Output führen (ohne Beeinflussung)	Input wurden übertragen	OK	-
1Hz Signal erzeugen (Test der Clockfrequenz)	1Hz Signal	OK	-
8-Bit Counter	Counter kann auf den 8 Ausgangspin gemessen werden	OK	-
SPI-Flash programmieren und bei Neustart Programm aus Flash laden	Test-Design wurde erfolgreich in Flash geschrieben und wird bei Reset der Karte neu geladen.	OK	-
Hardreset	Reset der FPGA und gespeicherte Konfiguration aus Flash wird geladen	OK	
Gumstix			
Verbindung auf PCB		OK	-
Verbindung auf Konsole des Gumstix	Serielle Verbindung konnte erfolgreich hergestellt werden und so auf das Filesystem des Gumstix zuzugreifen	OK	-
Reset Gumstix (Neustart) mittels des Reset-Button auf dem PCB	Gumstix wird neu gestartet	OK	-
ADC			
Verbindungsaufbau mit FPGA (SPI)	ADC-Werte können gelesen werden	OK	-
Stromaufnahme			
DC/DC-Wandler: 3.3V (ADC, Flash, Gumstix, Camera)	Stromaufnahme: 293 mA	OK	
DC/DC-Wandler: 3.3V (2.5V) (Kommunikation OBC)	Stromaufnahme: 1 mA	OK	Speisung 2.5/3.3V ist auf 3.3V, da LVDS nicht verwendet wird.
DC/DC-Wandler: 1.8V (3.3V) (Kommunikation Gumstix, FTDI)	Stromaufnahme: < 1 mA	OK	Speisung 1.8/3.3V ist auf 1.8V, da Gumstix verwendet wird.
DC/DC-Wandler: 1.2V (FPGA Core)	Stromaufnahme: 379 mA	OK	
Speisung: 5.0V	Stromaufnahme: 360 mA	OK	Gesamter Stromverbrauch

Anhang 22: Simulation Kamera



Anhang 24: Simulation ADC



Anhang 25: Anleitung Bildverarbeitungskarte

1. Aufbau & Anschluss

- Kameramodul *OV7670* auf den Konnektor *Camera* der Bildverarbeitungskarte verbinden
- miniUSB-Kabel zwischen PC und *Konsolen-Anschluss* des Gumstix verbinden
- FTDI auf *Peripheral_2* Konnektor verbinden
    a. Pin 1: -
    b. Pin 2: GND (Schwarz)
    c. Pin 3: TX (Orange)
    d. Pin 4: RX (Gelb)
- Adapterkabel auf ADC-Stecker verbinden
- Speisespannung (5.0V) auf *Power*-Konnektor verbinden

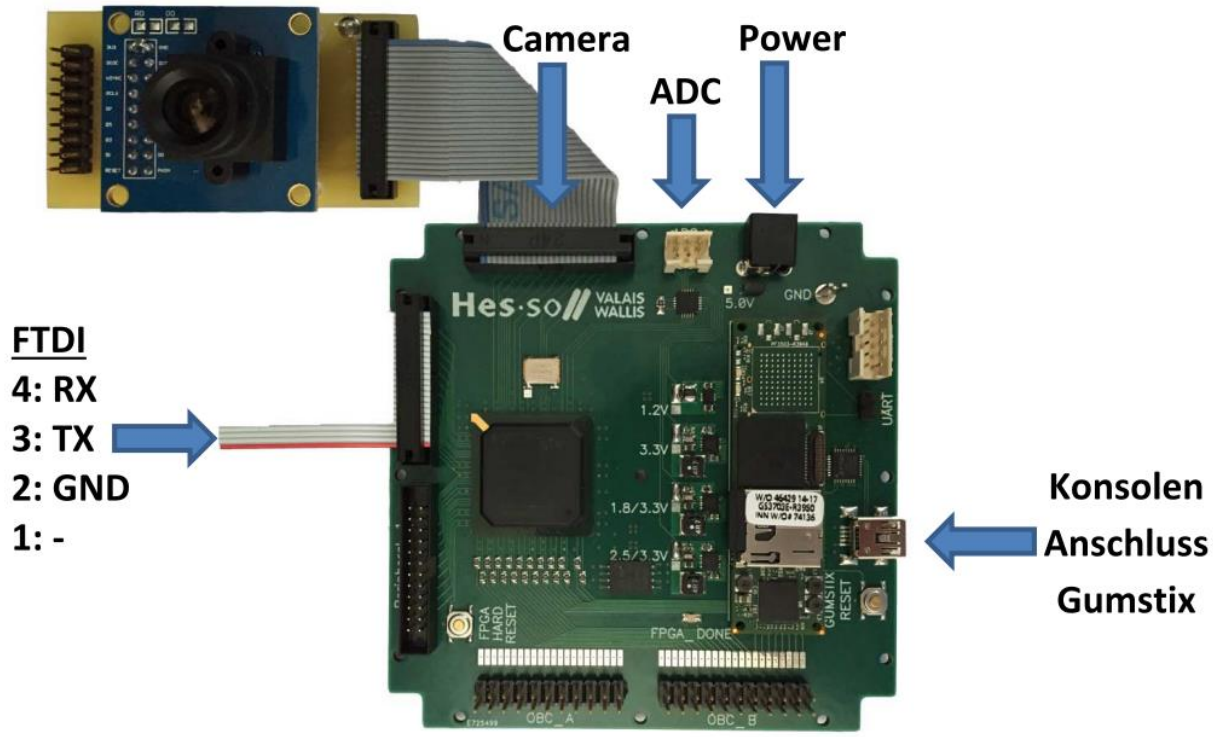


Fig. 1 Anschluss Bildverarbeitungskarte

Über das Adapterkabel des ADC können 3 externe analoge Spannungen angeschlossen werden (maximal 5.0V)

Fig. 2 ADC Adapterkabel

Wichtig: Sicherstellen, dass alles korrekt angeschlossen ist!

2. Inbetriebnahme

Das komplette Hardwaredesign wurde fest in den Flash der FPGA programmiert. Sobald die Spannungsversorgung eingeschaltet wird, lädt die FPGA das Design aus dem Flash. Wenn die FPGA_Done-LED aufleuchtet, ist die FPGA betriebsbereit.

Am PC muss eine Verbindung zu den 2 seriellen Schnittstellen hergestellt werden.
Im Geräte-Manager sollte überprüft werden, auf welchen COM-Ports die beiden Schnittstellen verbunden sind.
Anschliessend kann mit einem seriellen Terminal eine Verbindung zu den beiden Ports aufgenommen werden:

FTDI:	256‘000 Baud
Gumstix:	115‘200 Baud

Der Gumstix benötigt einige Sekunden für den Boot-Vorgang. Ist dieser abgeschlossen, werden die Login-Daten verlangt.

User: root (kein Passwort!)

Es fehlt noch die Kopplung zwischen der UART-Schnittstelle und dem SPI. Dafür muss folgender Befehl in der Konsole eingegeben werden:

```
cat < /dev/ttyO2 > /dev/spidev1.0
```

Es besteht jetzt eine Verbindung zwischen der Konsole und dem SPI.
Diese Verbindung kann bei Bedarf mit *CTRL + C* wieder beendet werden.

Die Inbetriebnahme ist soweit abgeschlossen, dass die Bildverarbeitungskarte getestet werden kann.

Anhang 26: Matlab Scrip: Bayer Demosaic

Dieses Skript nimmt als Input ein Bild (JPG oder PNG) und führt die 4 möglichen Transformationen des Bayern Pattern durch. Dadurch erhält man 4 Bilder, wobei nur eines brauchbar ist.

```
1
2   a1 = demosaic(a,'gbrg');
3   a2 = demosaic(a,'grbg');
4   a3 = demosaic(a,'bggr');
5   a4 = demosaic(a,'rggb');
6
7   scrsz = get(groot, 'ScreenSize');
8   figure('Position',[100 100 scrsz(3)-200 scrsz(4)-200])
9
10  subplot(2,2,1), subimage(a1)
11  subplot(2,2,2), subimage(a2)
12  subplot(2,2,3), subimage(a3)
13  subplot(2,2,4), subimage(a4)
```

Anhang 27: Python Applikation ADC

```
from matplotlib import pyplot as plt
from matplotlib import animation
import numpy as np
import serial


# General
channels = 8
width = 100
height = 5
serPort = 11
valCh = np.zeros((1, 8))
voltageCh = np.zeros((1, 8))
vRef = 5.0
rangeADC = 4095
i = 0


# Graph
fig = plt.figure(figsize = (15,10))
fig.canvas.set_window_title('Processing-Board: ADC')
x = np.arange(0, width, 1)

displayValues = np.zeros((width, channels)) # store Data-Line-Values
ax = plt.axes(xlim=(0, width), ylim=(0-0.1, height+0.1))
ax.set_xlabel('Time')
ax.set_ylabel('Voltage [V]')
ax.set_title('ADC-Meausure')


# Grid
ax.grid('on')
major_tick_x = width/20
major_tick_y = 0.5
minor_tick_y = 0.1

major_ticks_x = np.arange(0, width + major_tick_x, major_tick_x)
major_ticks_y = np.arange(0, height + major_tick_y, major_tick_y)
minor_ticks_y = np.arange(0, height + minor_tick_y, minor_tick_y)
ax.set_xticks(major_ticks_x)
ax.set_yticks(major_ticks_y)
ax.set_yticks(minor_ticks_y, minor=True)

ax.grid(which='minor', alpha=0.2)
ax.grid(which='major', alpha=0.5)


# Create Data-Lines
line0, = plt.plot([], label="5.0V")
line1, = plt.plot([], label="3.3V")
line2, = plt.plot([], label="1.8V")
line3, = plt.plot([], label="3.3V")
line4, = plt.plot([], label="1.2V")
line5, = plt.plot([], label="Ext0")
line6, = plt.plot([], label="Ext1")
line7, = plt.plot([], label="Ext2")
lines = [line0, line1, line2, line3, line4, line5, line6, line7]
```

```
58
59  # Legend of Graph
60  handles, labels = ax.get_legend_handles_labels()
61  plt.figlegend(handles, labels, loc=5)
62
63
64  # Serial Open
65  ser = serial.Serial(serPort - 1, 256000, timeout=0.1)
66  print ser.name
67
68
69
70  # Init-Function
71  def init():
72      for line in lines:
73          line.set_data([], [])
74      return lines
75
76
77  # Animate Function: Read new Data & update the Graph
78  def animate(i):
79      global displayValues
80      ser.write("a")
81      # Problems can occur with the Serial-Line, so put Try/Catch around it
82      try:
83          s = ser.readline()
84
85          for i in range(0, channels):
86              valCh[0, i] = int(s.encode("hex")[1 + 4 * i:4 + 4 * i], 16)
87              voltageCh[0, i] = valCh[0, i] * vRef / rangeADC
88
89          print(voltageCh)    # Display Current Values on Console
90
91          # Calculate Voltages
92          displayValues = np.vstack((displayValues, voltageCh))
93          displayValues = np.delete(displayValues, (0), axis=0)
94
95      except NotImplementedError:
96          print "ERROR: Serial Read"
97
98      for y, line in enumerate(lines):
99          line.set_data(x, displayValues[:, y])
100     return lines
101
102
103 # Calls in a given Frequency the animate Function to update the Graph
104 anim = animation.FuncAnimation(fig, animate, init_func=init, frames=100, interval=100
    , blit=True)
105
106
107 # Display Graph
108 plt.show()
109
110
111 # Close Serial Connection
112 ser.close()
```