Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Lukas Guzy

Entwicklung eines Gateways zur Umsetzung von Busprotokollen zwischen den Bussystemen PROFIBUS und SERVOLINK 4

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Lukas Guzy

Entwicklung eines Gateways zur Umsetzung von Busprotokollen zwischen den Bussystemen PROFIBUS und SERVOLINK 4

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im dualen Studiengang Informations- und Elektrotechnik
Studienrichtung Automatisierungstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Ulfert Meiners
Zweitgutachter: Prof. Henning Hasemann

Abgegeben am 12.April 2010

Vorwort

Die vorliegende Diplomarbeit habe ich im Rahmen des Diplomsemester in der Firma SIEB & MEYER AG angefertigt. Diese Arbeit befasst sich mit einem wichtigen Thema im Bereich der Automatisierungstechnik, der Entwicklung von selbstständig arbeitenden Mikroprozessorsystemen. In der heutigen Zeit sind Prozesse in der Industrie hoch komplex. Um diese Prozesse oder Teile solcher Abläufe kostengünstig, sicher und schnell durchführen zu können, ist ein einfacher und schneller Austausch von Prozessdaten notwendig. Durch die Implementierung von Mikroprozessorsystemen in FPGAs ist es möglich hochkomplexe Logik, die zur schnellen und sicheren Verarbeitung von Prozessdaten benötigt wird, kostengünstig zur Verfügung zu stellen. Zudem ist diese Logik platzsparend und durch SMD Technologie können heutzutage komplexe Mikroprozessorsysteme auf immer kleineren Leiterplatinen untergebracht werden.

Die Einsatzgebiete von FPGAs sind sehr vielfältig. Sie werden zur Echtzeit-Verarbeitung von einfachen bis komplexen Algorithmen z.B. Zustandsregler, zur digitalen Signalverarbeitung im Sinne von digitalen Filtern oder zur FastFourierTransformation und auch für Protokoll-Implementierungen verwendet. Die Industrie benötigt für solch anspruchsvolle Aufgaben hochqualifizierte Ingenieure und Informatiker. Diese sollten in ihrem Studium Erfahrungen im Bereich der Digitaltechnik und Automatisierungstechnik und den dazugehörigen Tools gesammelt haben.

Lukas Guzy

Thema der Diplomarbeit

Entwicklung eines Gateways zur Umsetzung von Busprotokollen zwischen den Bussystemen PROFIBUS und SERVOLINK 4

Stichworte

MicroBlaze, FPGA, Softcore, Busumsetzung, zyklischer Datenaustausch

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Entwicklung eines Gateways zum zyklischen Austausch von Prozessdaten zwischen dem Feldbussystem PROFIBUS und dem firmeneigenen Bussystem SERVOLINK 4. Zu diesem Zwecke wurde in einem FPGA ein Mikroprozessorsystem realisiert, in welchem eine interruptgesteuerte Routine abläuft. Das Gateway bewerkstelligt den Kommunikationsaufbau mit einer SPS über PROFIBUS, empfängt Daten von der SPS und sendet diese über Lichtwellenleiter an Antriebsregler, die dann Motoren antreiben. Der Datenaustausch wird ebenso von den Antriebsreglern zur SPS durchgeführt.Die Arbeit beinhaltet die Analyse der einzelnen Komponenten, die Planung der Systemstruktur und die Implementierung, welche den Entwurf eines Mikroprozessorsystems mit einem Softcore, die Anbindung an VHDL-Module, -Prozesse und -Komponenten und den Test an einem Prüfstand umfasst.

Lukas Guzy

Title of the paper

Development of a gateway for bus protocol conversion between the bus systems PROFIBUS and SERVOLINK 4

Keywords

MicroBlaze, FPGA, Softcore, bus conversion, cyclic data exchange

Abstract

This paper deals with the development of a gateway for cyclic exchange of process data between the fieldbus PROFIBUS and the company-owned bus system SERVOLINK 4. Therefor a microprocessor system has been implemented in an FPGA. In this microprocessor system an interrupt service routine is running. The gateway communicates to a PLC using PROFIBUS, achieves process data from the PLC and sends this data using SERVOLINK 4 to the connected drives. This data transfer also is executed from the drives to the PLC.This paper contains the analysis of the particular components, the design of the system structure and the implementation, which includes the generation of a softcore in a microprocessor system, the connection of VHDL modules, processes and components and the testing of the designed model on a test stand.

Diese Arbeit widme ich meinem Vater für seine Bemühungen mir die beste Ausbildung zu ermöglichen.

Danksagung

Ich danke meinen Eltern, meinen Geschwistern, Renate und Maya für alles, was sie mir auf meinen Lebensweg mitgegeben haben. Zudem möchte ich meinen Freunden danken, die mich immer unterstützt haben und mir ein ums andere Mal durch allerlei Unternehmungen Zerstreuung im Alltag gaben, ohne die mir mein Studium nicht so leicht gefallen wäre. Durch all diese Menschen bin ich zu dem Mensch geworden, der ich jetzt bin.

Inhaltsverzeichnis

Glossar

ALU	Die **A**rithmetic **L**ogic **U**nit (dt. arithmetisch-logische Einheit) ist ein elektronisches Rechenwerk.
Barrel Shifter	Ein Barrel Shifter ist eine Schiebelogik. Er kann Bitverschiebungen von mehreren Bits durchführen und vermeidet so zeitaufwendige Schleifen bei mehr bitigen Schiebeoperationen.
Burst	Burst beschreibt eine bestimmte Art des Lesens/Beschreibens eines Speichers.
Cache Line	Cache Line bezeichnet die kleinste Verwaltungseinheit innerhalb des Caches (Puffers) von Prozessoren. Es handelt sich dabei um eine Kopie eines Speicherbereichs mit mehreren aufeinander folgenden Adressen.
CacheLink	CacheLink sind spezielle Signale für das Arbeiten mit Fast Simplex Link (FSL).
CLB	**C**onfigurable **L**ogic **B**locks sind Blöcke in einer programmierbaren elektrischen Schaltung, deren interne Elemente, Look-Up Tables (LUT), AND-OR-Matrizen und Flipflops, je nach Erfordernis während der Programmierung in einen bestimmten Zustand gebracht werden und dann später während des Betriebs entsprechend agieren.
CoreConnect	CoreConnect ist eine von IBM entwickelte Bus-Architektur für Mikroprozessor-Busse.
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evice besteht im Wesentlichen aus programmierbaren AND/OR-Matrizen, programmierbaren Rückkopplungen, Eingabe- und Ausgabeblöcken.
CRC	**C**yclic **R**edundancy **C**heck ist ein Verfahren zur Bestimmung eines Prüfwerts für Daten, um Fehler bei Übertragung oder Speicherung erkennen zu können.

D-Sub	D-Sub bezeichnet eine Bauform eines Steckersystems für Datenverbindungen.
DCM	**D**igital **C**lock **M**anagement ist eine Manipulationsfunktion für Taktsignale.
DDR	**D**ouble **D**ata **R**ate bezeichnet in der Computertechnik ein Verfahren, mit dem Daten auf einem Datenbus mit doppelte Datenrate übertragen werden können.
DLMB	**D**ata **L**ocal **M**emory **B**us
DPRAM	Dual Port Random Access Memory ist ein Speicher auf den von zwei Seiten (Ports) aus zugegriffen werden kann.
DXCL	**D**ata side **X**ilinx **C**ache**L**ink interface bei Nutzung von FSL.
EAR	**E**xception **A**ddress **R**egister
ESR	**E**xception **S**tatus **R**egister
Exception	Eine Exception (dt. Ausnahmebehandlung) bezeichnet in der Computertechnik ein Verfahren, Informationen über bestimmte Programmzustände - meistens Fehlerzustände - an andere Programmebenen weiterzugeben.
FPGA	Ein **F**ield **P**rogrammable **G**ate **A**rray ist ein programmierbarer, integrierter Schaltkreis der Digitaltechnik und somit eine programmierbare logische Schaltung.
FPU	Die Floating Point Unit (dt. Gleitkommaeinheit) bezeichnet einen speziellen Prozessor, der mathematische Funktionen oder Gleitkommazahlen verarbeitet. Bei vielen modernen CPUs ist die FPU als Koprozessor realisiert.
FSL	Die **F**ast **S**implex **L**ink-Kanäle sind fest zugeordnete unidirektionale Punkt-zu-Punkt Datenfluss-Schnittstellen.
GCC	Die **G**NU **C**ompiler **C**ollection ist eine Sammlung von Compilern für die Programmiersprachen C, C++, Java, Objective-C, Fortran 95 und Ada.
HDL	**H**ardware **D**escription **L**anguage
ILMB	**I**nstruction **L**ocal **M**emory **B**us

IOB	Mit **I**nput **O**utput **B**locks können Verbindungen zwischen dem Innenleben eines FPGAs und der Außenwelt hergestellt werden.
IP	**I**ntellectual **P**roperty
ISE	**I**ntegrated **S**oftware **E**nvironment, siehe Kapitel 2 Seite 27.
ISO	**I**nternational **O**rganization for **S**tandardization
IXCL	**I**nstruction side **X**ilinx **C**ache**L**ink interface bei Nutzung von FSL.
JTAG	**J**oint **T**est **A**ction **G**roup bezeichnet den Institute of Electrical and Electronics Engineers-Standard 1149.1, der ein Verfahren zum Testen und Debuggen von elektronischer Hardware direkt in der Schaltung beschreibt.
LSI	**L**arge **S**cale **I**ntegration ist ein Integrationsgrad und bezeichnet die absolute Anzahl an Transistoren in einem integrierten Schaltkreis.
LUT	Ein **L**ook-**U**p**T**able ist eine Datenstruktur, die vorberechnete Daten einer aufwendigen Berechnung enthält und mit deren Hilfe es möglich ist komplexe Berechnungen auf die in der Regel erheblich schnellere Wertsuche eines Datenfeldes zu vereinfachen.
LWL	**L**icht**w**ellen-**L**eiter sind Leitungen, die mit Licht als Medium Daten übertragen.
MCH	**M**ulti **CH**annel
MHS	**M**icroprocessor **H**ardware **S**pecification beschreibt das Design der elektronischen Hardware mit seinen Komponenten, Ports und Parametern.
MMU	Die **M**emory **M**anagement **U**nit (dt. Speicherverwaltungseinheit) bezeichnet eine Funktionseinheit von Mikroprozessoren, die zum Zugriff auf den Arbeitsspeicher oder sonstige Hardware das Übersetzen von virtuellen Adressen in physikalische Adressen bewerkstelligt.
MPGA	**M**ask **P**rogrammable **G**ate **A**rray, siehe FPGA.
MSS	**M**icroprocessor **S**oftware **S**pecification beschreibt welche Treiberbibliotheken für welche Hardwarekomponente ausgewählt wurden.

Netzliste	Netzlisten sind im Bereich Elektronik-Entwurf eine textuelle Beschreibung der elektrischen Verbindungen (Schaltplan) zwischen Bauelementen.
OPB	**O**n-chip **P**eripheral **B**us
OSI	**O**pen **S**ystem **I**nterconnection
Phase-Lock-Loop	Eine Phasenregelschleife, auch als Phase-locked-Loop (PLL) bezeichnet, ist eine elektronische Schaltungsanordnung, die die Phasenlage und damit zusammenhängend die Frequenz eines veränderbaren Oszillators über einen geschlossenen Regelkreis so beeinflusst, dass eine möglichst kleine Phasenabweichung zwischen einem äußeren Referenzsignal und dem Oszillator oder einem daraus abgeleiteten Signal erzielt wird.
PLB	**P**rocessor **L**ocal **B**us
Program Counter	Der Program Counter (dt. Befehlszähler) ist ein spezielles Register innerhalb der CPU, dass je nach Systemarchitektur die Speicheradresse des derzeitigen oder des nächsten auszuführenden Befehls enthält.
PROM	**P**rogrammable **R**ead **O**nly **M**emory ist ein programmierbarer, nur lesbarer Speicherbaustein.
Prozessor Pipeline	Die Prozessor Pipeline bezeichnet bei Mikroprozessoren eine Art „Fließband“, mit dem die Abarbeitung der Maschinenbefehle in Teilaufgaben zerlegt wird, die für mehrere Befehle gleichzeitig ausgelöst werden.
PVR	**P**rocessor **V**ersion **R**egister
RAM	**R**andom **A**ccess **M**emory
RISC	**R**educed **I**nstruction **S**et **C**omputing ist eine bestimmte Designphilosophie für Prozessoren.
RS422	RS422 ist ein Schnittstellenstandard für eine leitungsgebundene differentielle serielle Datenübertragung.
SDK	**S**oftware **D**evelopment **K**it, siehe Kapitel 2 Seite 30.
Single-Beat	Single-Beat beschreibt eine bestimmte Art des Lesens/Beschreibens eines Speichers.
Slices	Ein Slice bezeichnet die kleinste logische Funktionseinheit in einem FPGA.

Soft-IP-Core	IP-Core ist ein in Software existierender, wiederverwendbarer Teil eines Chipdesigns.
Softcore	Ein Softcore ist ein IP-Core (IP-Kern) (intellectual property core) und existiert in Form von Quellcode in einer Hardwarebeschreibungssprache. Er kann auch als bereits vom Hersteller synthetisierte Netzliste oder als textuelle Beschreibung eines Schaltplans vorliegen.
SPS	Eine **S**peicher**p**rogrammierbare **S**teuerung ist eine Baugruppe, die zur Steuerung oder Regelung einer Maschine eingesetzt wird.
Testbench	Eine Testbench ist eine virtuelle Umgebung, die zur Verifizierung von Design oder Modellen genutzt wird.
Token	Ein Token ist ein Hilfsmittel zur Synchronisation paralleler Prozesse. Der Busteilnehmer, der den Token besitzt, darf auf die Ressource (einen Speicherbereich, eine Schnittstelle o.a.) zugreifen.
Toplevel	Der Toplevel ist in der ISE die hierachisch höchstgelegene Ebene oder vhd-Datei des Systementwurfs, in dem alle anderen vhd-Dateien als Instanzen eingebunden sind.
TOSLINK	TOSLINK ist ein standardisiertes Lichtwellenleiter-Verbindungssystem für optische Signalübertragungen.
UART	Ein **U**niversal **A**synchronous **R**eceiver **T**ransmitter ist ein elektronisches Bauelement, das der Realisierung von digitalen seriellen Schnittstellen dient.
VHDL	Kurzform von VHSIC
VHSIC	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit Hardware Description Language ist eine Hardwarebeschreibungsprache.
Wizard	Ein Wizard ist ein Benutzer Interface, das dem Benutzer beim Einrichten von z.B. Programmen hilft.
XCL	**X**ilinx **C**ache**L**ink ist eine High Performance Lösung von Xilinx für den Zugriff auf externen Speicher.
XMD	**X**ilinx **M**icroprocessor **D**ebug
XPS	**X**ilinx **P**latform **S**tudio, siehe Kapitel 2 Seite 28.

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Übersicht

Diese Arbeit beschäftigt sich mit der Konzeption und Implementierung eines Gateways zur Umsetzung von Busprotokollen. Das Gateway ist ein eigenständiges Hardwaremodul, das eine Protokollumsetzung zwischen einem Feldbus und dem firmeneigenen Bussystem SERVOLINK 4 der Firma SIEB & MEYER realisiert. Die Kommunikation des Gateways mit dem Feldbus und dem daran angeschlossenen Steuerungsrechner wird von einem Hardwaremodul übernommen. Das Übertragungsmedium des firmeneigenen Bussystems sind Lichtwellenleiter, über die die angeschlossenen Geräte (z.B. Antriebsregler) mit Prozessdaten versorgt werden. Auf der anderen Seite wird auch der Steuerungsrechner mit den aktuellen Istwerten der Geräte versorgt.

Das Gateway soll mit Hilfe eines FPGA realisiert werden. In diesem FPGA soll ein Softcore/Mikroprozessorsystem implementiert werden, in dem mittels einer Software-Applikation der Austausch von Prozessdaten zwischen den beiden Bussystemen stattfindet. Die vom Softcore benötigten internen Systemkomponenten in Form von logischer Hardware müssen ausgewählt und eingebunden werden. Die Anbindung an externe Hardware/Peripherie ist ebenfalls notwendig. Als interne Peripherie kommen sowohl fertige als auch selbst entwickelte VHDL-Komponenten zum Einsatz.

1.2 Projektumfeld

Diese Diplomarbeit wurde im Rahmen der Abschlussarbeit eines dualen Studiums in der SIEB & MEYER AG angefertigt. Die Firma wurde 1962 durch Reinhard Sieb und Johannes Meyer mit Sitz in Hamburg gegründet. Mittlerweile besitzt das Unternehmen neben dem Hauptsitz in Lüneburg (Deutschland) mit etwa 250 Mitarbeitern weitere Niederlassungen in Shenzhen (China) und Taiwan. Weltweit besitzt die Firma eine Spitzenstellung im Bereich Steuerungen für Leiterblattenbohr- und Leiterplattenfräsmaschinen.

1.3 Motivation und Ausgangssituation

Um eine einfache und schnelle Steuerung und Bedienung von Servoverstärkern und Frequenzumrichtern der Firma SIEB & MEYER für ihre Kunden zu ermöglichen, sollte im Rahmen dieser Diplomarbeit ein Adaptermodul/Gateway entwickelt werden, das einen unkomplizierten Anschluss dieser Geräte an einen Steuerungsrechner ermöglicht. An dieses Gateway können bis zu 10 Geräte an die SERVOLINK 4-Schnittstelle (Siehe 2.2 auf Seite 23) angeschlossen werden. Zudem ermöglicht das Gateway den Anschluss eines Feldbussystems, in diesem Fall PROFIBUS, über das mit dem Steuerungsrechner kommuniziert wird. Da dieses Projekt in sich geschlossen ist und keine Kooperation mit anderen Abteilungen der Firma nötig ist, eignet es sich als Thema einer Diplomarbeit innerhalb einer Firma, in der die Entwicklung eines Produktes im Allgemeinen sehr komplex ist.

1.4 Zielsetzung

Ziel dieser Arbeit ist es ein Gateway zur Umsetzung von Busprotokolelen zwischen PROFIBUS und SERVOLINK 4 zu entwickeln. Im Rahmen dieser Arbeit erfolgt dann die Einarbeitung in die Entwicklung eines Mikroprozessorsystems in einem FPGA mittels der Programmiersprache VHDL. Weiterhin wird in der Vorbereitungsphase die Einarbeitung in die Entwicklungsumgebung von *Xilinx* durchgeführt und die Programmierkenntnisse eines Mikroprozessorsystems mittels der Programmiersprache C werden vertieft, um einen effektiven und sinnvollen Aufbau des Systems gewährleisten zu können. Abschließend wird das entwickelte System in einer Adapterplatine eingesetzt und der zyklische Austausch von Daten zwischen zwei Bussystemen realisiert.

1.5 Gliederung

In Kapitel 2 wird eine Einführung in die Grundlagen der Programmierung eines FPGAs gegeben. Dies beinhaltet sowohl die Logik/Hardware, als auch die Software eines in einem FPGA enthaltenden Softcores. Es wird die Entwicklungsumgebung vorgestellt und nach der Erklärung der Begriffe Softcore (Microblaze), PROFIBUS und SERVOLINK 4 werden auch Anforderungen und die Vorteile dieser Technologien genauer beschrieben.

In Kapitel 3 werden die vorhandenen Systemkomponenten präsentiert und deren Nutzen bzw. ihre Funktionen im notwendigen Umfang erläutert, dazu wird kurz die Firma Hilscher GmbH vorgestellt.

In Kapitel 4 wird die Funktionsweise des Gateways und das Konzept der Hardware und Software beschrieben.

In Kapitel 5 wird das im FPGA implementierte Mikroprozessorsystem, die MicroBlaze mit den funktional notwendigen Modulen, in seiner Grundstruktur erklärt. Die einzelnen Komponenten werden aufgelistet und es wird die genaue Untersuchung der benötigten Schnittstellen zum Verbinden der Komponenten miteinander beschrieben. Es folgt die Erläuterung zur Programmierung der MicroBlaze und es werden Hindernisse sowie Lösungen bei der Implementierung des Systems vorgestellt.

In Kapitel 6 wird die Einbindung des Gateways in ein reales Betriebsumfeld erläutert. Beschrieben werden die Simulation des Gateways bzw. von Hardwarekomponenten vor der Einbindung, sowie die Test- und Debugmöglichkeiten eines solches Systems.

In Kapitel 7 wird ein Fazit über das Projekt und die gesamte Projektarbeit innerhalb der Firma gezogen.

2 Grundlagen

2.1 Das Feldbussystem PROFIBUS

Die Entwicklung des Feldbussystems PROFIBUS (**Pro**cess **Fi**eld **Bus**) wurde 1987 in Deutschland ins Leben gerufen. Ein öffentlich gefördertes Verbundvorhaben von 21 Firmen und Instituten betrieb diese Entwicklung mit dem Ziel, die Realisierung und Verbreitung eines normierten Feldbussystems für die Fertigungs- und Prozessautomatisierung zu schaffen. Zu dieser Nutzerorganisation gehört auch das Unternehmen Siemens, das das Bussystem PROFIBUS in einer Vielzahl seiner Produkte einsetzt. Über PROFIBUS werden Daten (Produktdaten, Produktionsmitteldaten wie Information, Energie oder Material), die für einen Prozess relevant sind, zwischen den Busteilnehmern ausgetauscht [9, S. 13]. Zudem ist festgelegt, wie verteilte digitale Automatisierungsgeräte von der Feldebene bis zur Zellenebene miteinander vernetzt werden können. Da PROFIBUS ein Multi-Master-System ist, ermöglicht es den gemeinsamen Betrieb von mehreren Automatisierungs-, Engineering- oder Visualisierungssystemen mit den dezentralen Peripheriegeräten an einem Bus. PROFIBUS unterscheidet dabei unterschiedliche Gerätetypen [3, Seite 8].

- Master-Geräte:
  Sie bestimmen den Datenverkehr auf dem Bus. Ein Master darf Nachrichten ohne externe Aufforderung aussenden, wenn er im Besitz des Tokens ist. Master werden auch als aktive Busteilnehmer bezeichnet.
- Slave-Geräte:
  Sie sind Peripheriegeräte wie beispielsweise Ein/-Ausgabegeräte, Ventile, Antriebe und Messumformer. Sie erhalten keine Buszugriffsberechtigung, d.h. sie dürfen nur empfangene Nachrichten quittieren oder auf Anfrage eines Masters Nachrichten an diesen übermitteln. Slaves werden als passive Busteilnehmer bezeichnet. Sie verwenden im Gegensatz zu den Mastern nur einen geringen Anteil des Busprotokolls, dadurch wird die Implementierung vereinfacht.

Die Protokollarchitektur von PROFIBUS orientiert sich am ISO/OSI-Basisreferenzmodell entsprechend dem internationalen Standard ISO 7498. Dieses Referenzmodell gliedert sich in die folgenden sieben Schichten [2, Seite 251]. Siehe dazu auch Abbildung 1.

- (7) Anwendungsschicht - Application Layer:
  - zuständig für die Steuerung der untergeordneten Schichten
  - übernimmt die Anpassung an die jeweilige Anwendung
  - stellt dem Anwenderprogramm die Verbindung zur Außenwelt zur Verfügung
- (6) Darstellungsschicht - Presentation Layer:
  - zuständig für den gemeinsamen Zeichensatz und die gemeinsame Syntax
  - Umwandlung der lokalen Syntax in die für den Transport festgelegte Syntax und umgekehrt
- (5) Sitzungsschicht - Session Layer:
  - zuständig für den geordneten Ablauf des Dialogs zwischen den Endsystemen
  - Festlegen und Verwalten der Berechtigungsmarken für die Kommunikation
- (4) Transportschicht - Transport Layer:
  - zuständig für die Erweiterung von Verbindungen zwischen Endsystemen zu Teilnehmerverbindungen
  - bildet die Verbindungsschicht zu den anwendungsorientierten Schichten
- (3) Vermittlungsschicht - Network Layer:
  - zuständig für die Überbrückung geographischer Entfernungen zwischen den Endsystemen durch Einbeziehung von Vermittlungssystemen
  - steuert die zeitliche und logische getrennte Kommunikation zwischen verschiedenen Endsystemen
- (2) Sicherungsschicht - Data Link Layer:
  - definiert das Buszugriffsprotokoll
  - zuständig für den unverfälschten Datentransport über einen einzelnen Übermittlungsabschnitt
  - Flußsteuerung überwacht die vollständige und richtige Übertragung der Daten von den darüberliegenden Schichten
- (1) Übertragungsschicht - Physical Layer:
  - definiert und ist zuständig für den physikalischen Transport der digitalen Informationen

– überwacht die Funktion dieser Schicht durch zyklisches Prüfen von Steuerleitungen (getrennt von den Datenleitungen)

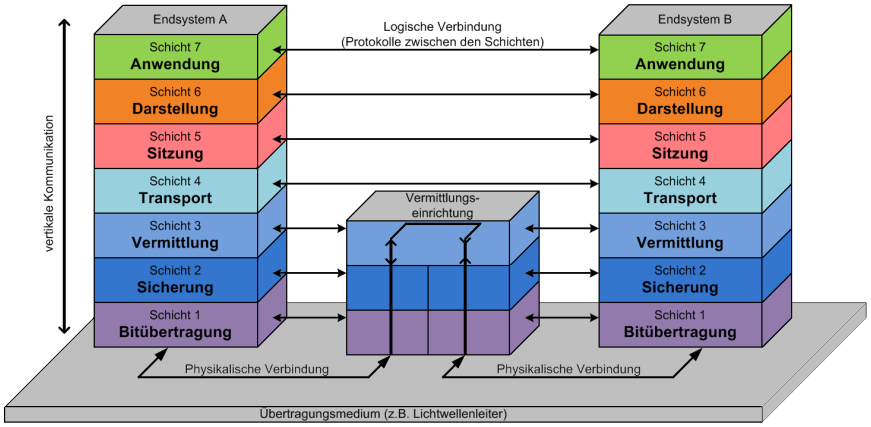


Abbildung 1: OSI-Referenzmodell

Die einzelnen Schichten nutzen Dienste der untergeordneten Schicht und stellen ihrerseits die Dienste für die übergeordnete Schicht bereit. Durch definierte Schnittstellen erfolgt die Schichteinteilung, wodurch einzelne Schichten ohne große Veränderung des gesamten Systems ausgetauscht werden können. Die Schichten 1 bis 4 sind die transportorientierten Schichten und die Schichten 5 bis 7 sind anwendungsorientierte Schichten. Das Übertragungsmedium ist im ISO/OSI-Modell nicht festgelegt.

Der PROFIBUS bietet unterschiedliche Dienstleistungen für die Automatisierungstechnik an: zyklischer Datenaustausch für Prozessdaten und azyklischer Datenaustausch für Parametrierdaten. Es sind Datenraten von bis zu 12 Mbit/s auf verdrillten Zweidrahtleitungen und/oder Lichtwellenleitern möglich.

2.2 Das Bussystem SERVOLINK 4

SERVOLINK 4 ist ein firmeneigenes Bussystem der Firma SIEB & MEYER. Es dient der Vernetzung von Antriebsverstärkern mit einem Steuerungsrechner. Als Medium stehen Lichtwellenleiter (TOSLINK) und RS422 zur Verfügung. Die Medien sind auch gemischt einsetzbar, da die Module über beide Schnittstellen verfügen. Dies ist in Abbildung 2 dargestellt. Zur

Ankopplung stehen für die Master- und die Slave-Seite FPGA-Lösungen zur Verfügung. Für die Masterseite existiert eine PCI-Einsteckkarte. Das SERVOLINK 4-Bussystem funktioniert in Echtzeit. Jeder Antrieb besitzt 16 Byte Solldaten und Istdaten, wobei die Inhalte von der Betriebsart abhängig sind. In Abbildung 3 ist der Aufbau eines SERVOLINK 4-Zyklusses zu sehen. Die gesamte Zykluszeit beträgt 500 μsec. In einem SERVOLINK 4-Ring können maximal 24 Antriebe in Reihe geschaltet sein, es ergibt sich die maximal zu übertragende Datenmenge von 384 Byte für einen SERVOLINK 4 Frame.

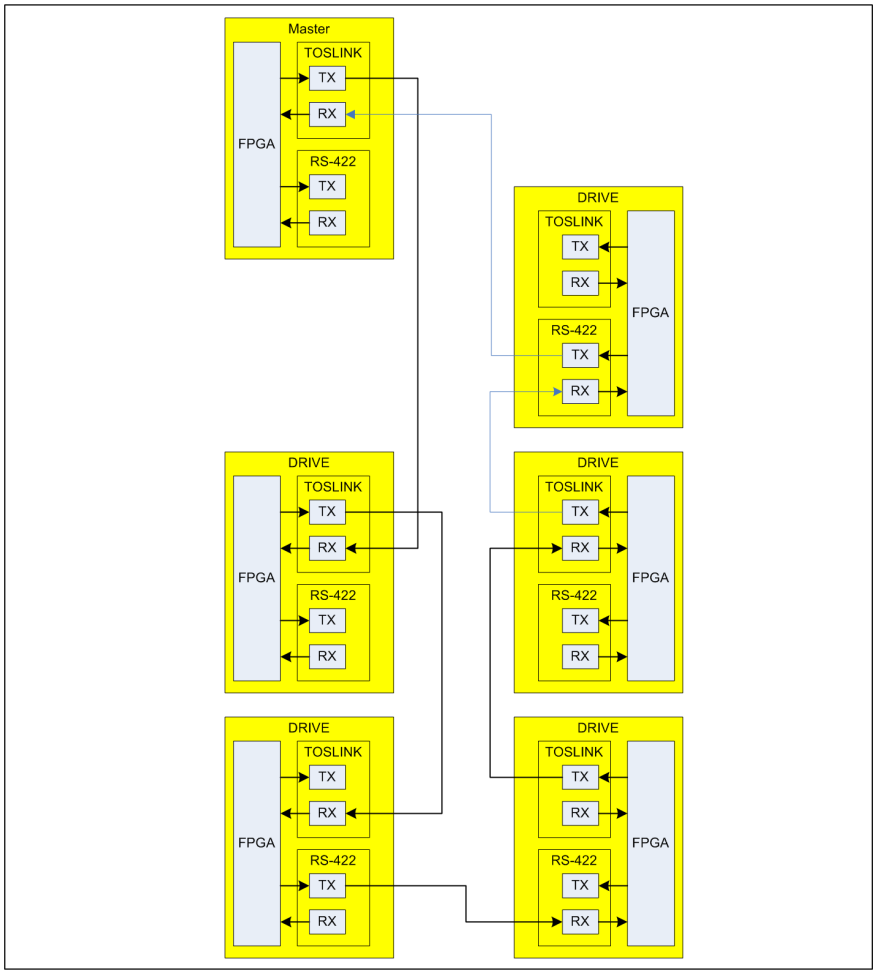


Abbildung 2: Blockdiagramm der Vernetzung von Antriebsverstärkern über SERVOLINK 4 [4]

$$t_{Frame} = t_{bit} * n_{Drives}(((n_{bytes} + n_{delay} + 1_{CRC}) * 10_{bit}) \tag{2.1}$$

Es folgt die Berechnung für die Dauer eines Frames mit 24 Antrieben aus Gleichung 2.2.

$$\begin{aligned} t_{Frame} &= 84nsec * 24 * ((16 + 2 + 1) * 10) = 380\mu sec \\ t_{Idle} &= t_{Cycle} - t_{Frame} = 500\mu sec - 380\mu sec = 120\mu sec \end{aligned}$$

Die Datenblöcke sind mit einer CRC-Prüfsumme versehen. Jeder Antrieb ersetzt die empfangenen Solldaten mit seinen Istdaten, dadurch ist gewährleistet, dass keine Verlängerung der Übertragungszeit stattfindet. Der CRC wird nach Gleichung 2.2 gebildet. Die Topologie des SERVOLINK 4-Systems ist ringförmig und der Prozess des Senden und Empfangen von Daten findet gleichzeitig statt.

$$CRC = 1 + x^1 + x^2 + x^8 \tag{2.2}$$

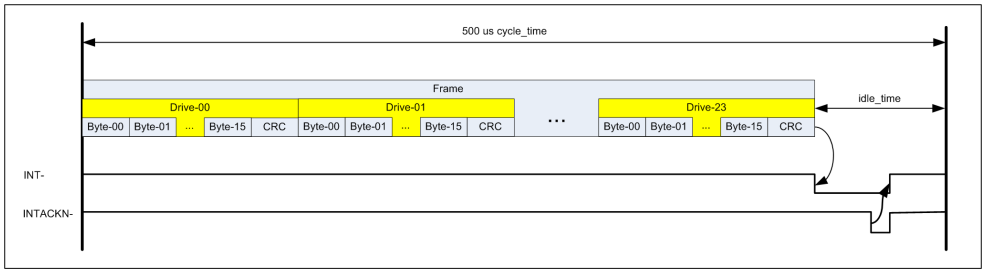


Abbildung 3: Aufbau eines SERVOLINK 4-Datenframes

2.3 Speicherprogramierbare Steuerung

"´Eine Speicherprogrammierbare Steuerung (SPS) ist eine Baugruppe, die zur Steuerung oder Regelung einer Anlage oder Maschine eingesetzt wird“ [12]. Die Sensoren und Aktoren sind mit dieser Baugruppe verbunden [12]. Das Betriebssystem stellt sicher, dass dem in der SPS ablaufenden Programm bei jedem Zyklus die aktuellen Zustände der Sensoren zur Verfügung stehen. Dazu wird bei jedem Zyklusbeginn ein Prozessdatenabbild gespeichert, und anhand dieser Informationen werden die Aktoren ein- oder ausgeschaltet, so wie es für die gewünschte Reaktion der Anlage nötig ist. Speicherprogrammierbare Steuerungen werden heutzutage annähernd wie Prozessleitsysteme genutzt und in Zusammenarbeit mit einer Visualisierungssoftware können dem Anwender immer aktuelle Daten über Fertigungsstände, Lagerbestände etc. zur Verfügung gestellt werden. Die SPS selber bewertet nur die Signale,

die ihr von Sensoren geliefert werden. In Abbildung 4 ist ein Beispiel für eine SPS abgebildet. In dieser Diplomarbeit stellt die SPS den Master und das Gateway bzw. das Hilscher COM-Modul den Slave für den PROFIBUS dar.

Abbildung 4: Siemens SPS

2.4 Die MicroBlaze

Die MicroBlaze bezeichnet in dieser Arbeit zweierlei Dinge. Zum Einen ist die MicroBlaze der Name der Komponente des Mikroprozessorsystems, zu dem das externe Speicher-Interface, die DPRAM-Blöcke etc. gehören, das in das VHDL-Projekt eingebunden wird, zum Anderen ist die MicroBlaze auch ein IP-Core, der den reinen Prozessor, mit ALU, Program Counter etc., darstellt, der in XPS in das Mikroprozessorsystem eingebunden wird. Allgemein ist eine MicroBlaze ein in einem FPGA der Firma Xilinx verwendbarer Mikroprozessor. Dieser Mikroprozessor existiert nicht als eigene Hardware in Form eines Bausteins, sondern ist nur als Softcore in Hardwarebeschreibungssprachen wie VHDL und Verilog verfügbar [11]. „Durch die spezielle Optimierung auf die Besonderheiten bestimmter FPGA-Bausteine ist der Logikbedarf dieses Mikroprozessors gering“ [11]. Weitere Details zu dem IP-Core MicroBlaze werden in Kapitel 5 aufgeführt. In Kapitel 3 wird erläutert, wie das Mikroprozessorsystem MicroBlaze als VHDL-Komponente aussieht.

2.5 Entwicklungsumgebung

2.5.1 Überlick

Die Entwicklungsumgebung besteht aus drei Anwendungen, der ISE-, der XPS- und der SDK-Umgebung. Jede Anwendung ist speziell für einen Teil der Entwicklung eines Projektes mit Hard- und Softwareteil zuständig.

2.5.2 Integrated Software Enviroment (ISE)

Die ISE enthält alles was zur Entwicklung von digitaler elektronischer Logik notwendig ist. Durch integrierte Tools und Wizards wird das Entwerfen von I/O-Verknüpfungen, Leistungsanalyse, Signaltimings und HDL-Simulationen den Anwender erleichtert. Abbildung 5 zeigt die ISE. In dem *Project Navigator Interface* hat der Anwender die Übersicht über sein gesamtes Projekt.

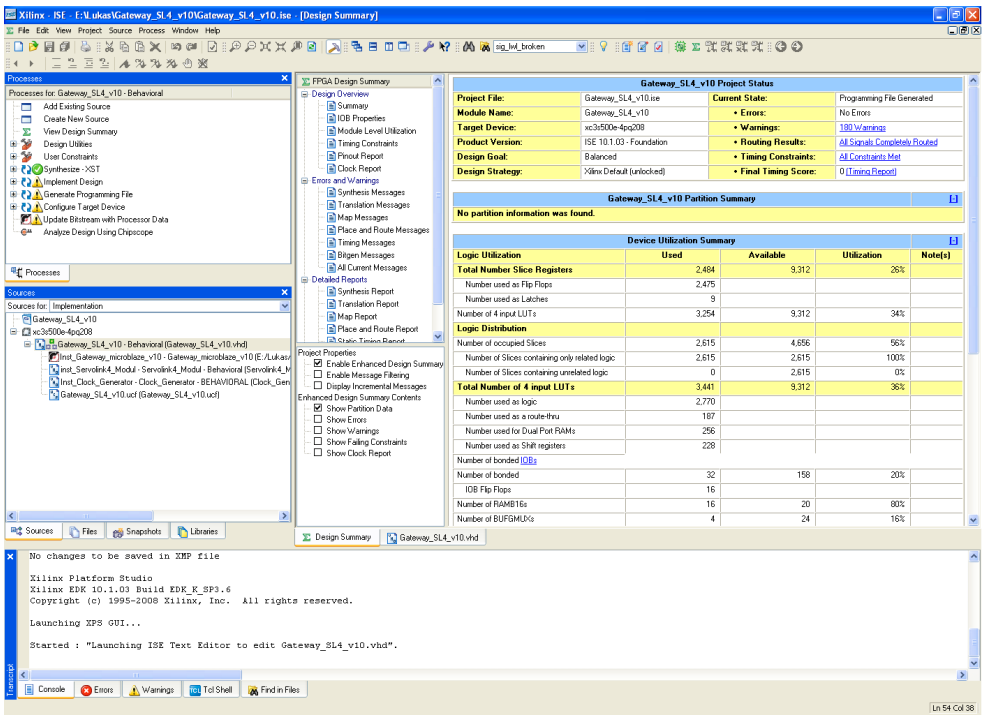


Abbildung 5: Integrated Software Environment

2.5.3 Xilinx Platform Studio (XPS)

Das Xilinx Platform Studio dient dem Anwender zum Entwerfen von Mikroprocessorsystemen. Ein System lässt sich dabei einfach mittels "Drag & Drop“ erstellen. Die Tools der Entwicklungsumgebung schreiben dann alle Informationen in Form von Quellcode für Hard- und Software in zwei Dateien. Die Hardware des Systems wird in die MHS-Datei und die Software des Systems in die MSS-Datei geschrieben. Diese beiden Dateien können auch von Hand bearbeitet werden. Aus den gespeicherten Informationen erstellt der Kompiler später die Netzlisten für das Hardwaredesign und Bibliotheken für die Softwareanwendungen.

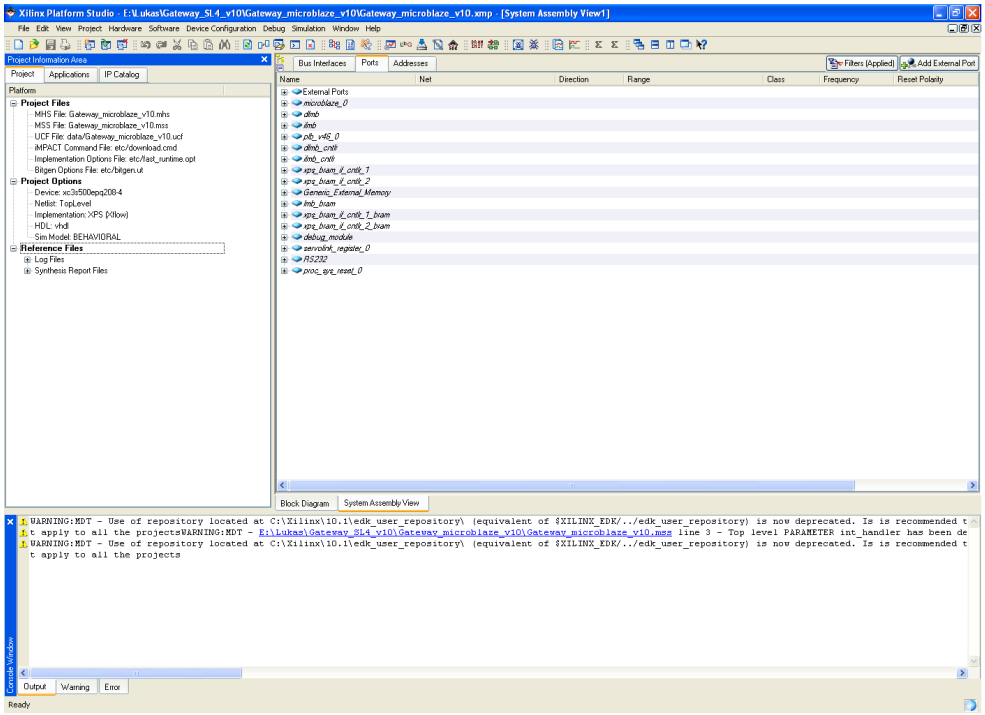


Abbildung 6: Xilinx Platform Studio Projekttab und Portliste

In Abbildung 7 ist der Application-Tab zu sehen. Dort kann der Anwender Softwareprojekte für die MicroBlaze erstellen und verwalten. Die Abbildung 8 zeigt den IP Catalog-Tab. Dort sind die Komponenten aufgelistet, die dem Anwender für den Aufbau seines Systems zur Verfügung stehen. In Abschnitt *Project Local pcores* werden die vom Anwender erstellten Komponenten

aufgelistet. Die benötigten Komponenten werden aus dem IP Catalog in das Fenster auf der rechten Seite des XPS gezogen. In diesem Fenster können alle Dokumente des Projektes angezeigt werden. Es sind die Komponenten mit ihren System-internen und System-externen Ports, ihren Busanbindungen und ihren Adressbereichen in einzelnen Tabs aufgelistet (siehe Abbildungen 6, 7 und 8). Im Bus-Interface-Tab muss der Anwender festlegen, an welche Busse eine Komponente angeschlossen wird. Zudem muss der Adressbereich jeder Komponenten festgelegt werden, diese Adressbereiche sind im Addresses-Tab zu sehen.

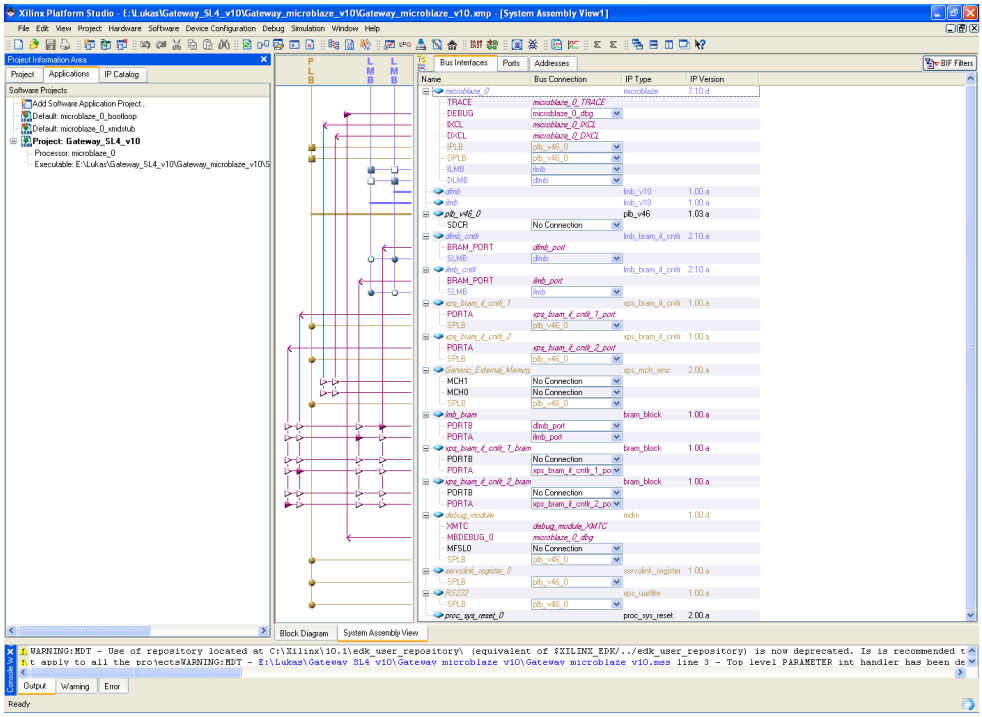


Abbildung 7: Xilinx Platform Studio Applications und Bus Interfaces

Ist die Vergabe der Adressen abgeschlossen, kann unter dem Ports Tab das Routen der Signale unter den einzelnen Komponenten stattfinden, oder es können auch Signale über die *External Ports* aus der MicroBlaze herausgeführt werden, so dass diese z.B. an andere Instanzen im Gesamtprojekt angebunden werden können.

Sind alle diese Einstellungen vom Anwender vorgenommen worden, kann er sein Design unter *Hardware* →*Generate Netlist* synthetisieren lassen, so dass die MicroBlaze in der ISE-

Umgebung als Instanz eingebunden werden kann. In der XPS-Umgebung kommt die GCC zur Anwendung. Diese wurde von Xilinx um eine grafische Oberfläche, welche als XPS bezeichnet wird, ergänzt.

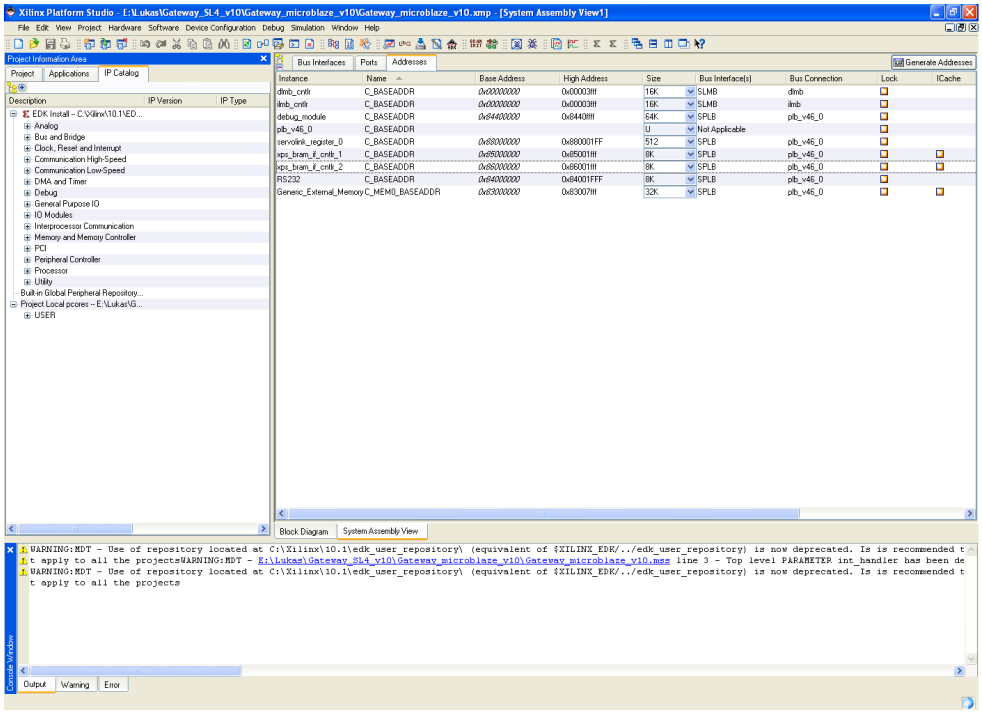


Abbildung 8: Xilinx Platform Studio IP Catalog und Adressvergabe

2.5.4 Software Development Kit (SDK)

Das Software Development Kit dient zur Programmierung des in XPS generierten Mikroprozessorsystems. In Abbildung 9 ist die SDK-Umgebung dargestellt. Ein SDK-Projekt kann in der XPS-Umgebung unter dem Applications-Tab gestartet werden oder aber auch im SDK selbst. Ein neues Softwareprojekt muss mit dem System der MicroBlaze verbunden werden, damit die richtigen Adressen in der Software angesprochen werden. Auf der linken Seite befindet sich ein Fenster, in dem das MicroBlaze-System mit seinen fertig generierten Bibliotheken, Makros etc. aufgelistet wird. Dort kann der Benutzer eigene Projekte einfügen und diese mit eigenen Dateien füllen.

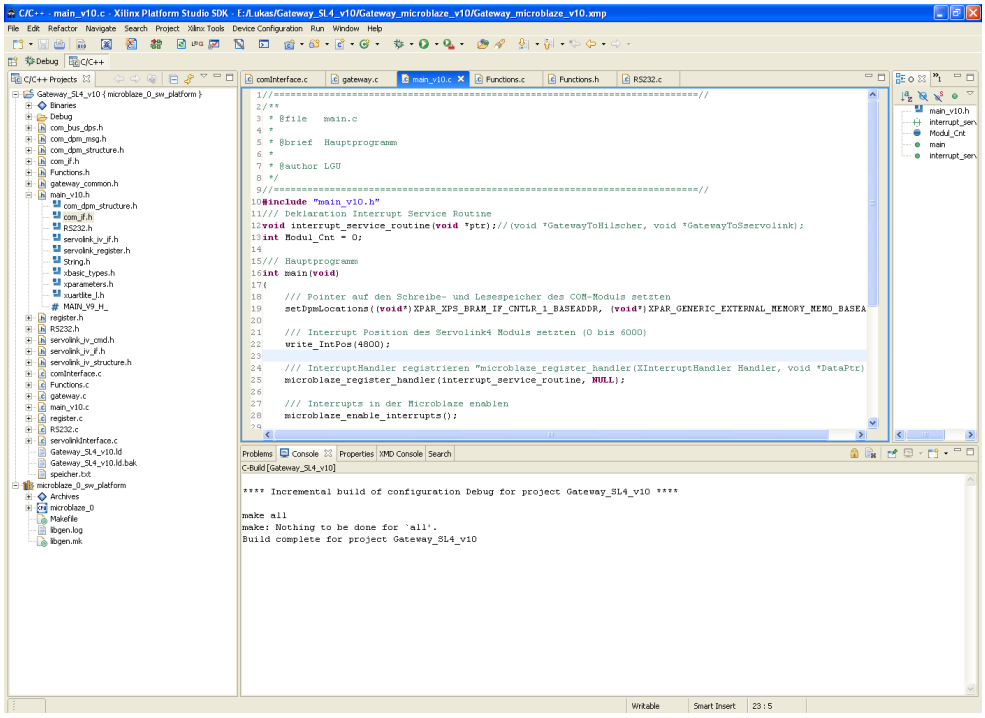


Abbildung 9: Software Development Kit-Oberfläche

2.5.5 Zusammenarbeit der drei Werkzeuge

In Abbildung 10 ist dargestellt, wie aus den einzelnen Quelldateien der ISE-, XPS- und SDK-Umgebung eine Hardware in einem FPGA entsteht. Der Entwicklungsablauf dieses Diplomprojektes wird in Kapitel 5 erläutert.

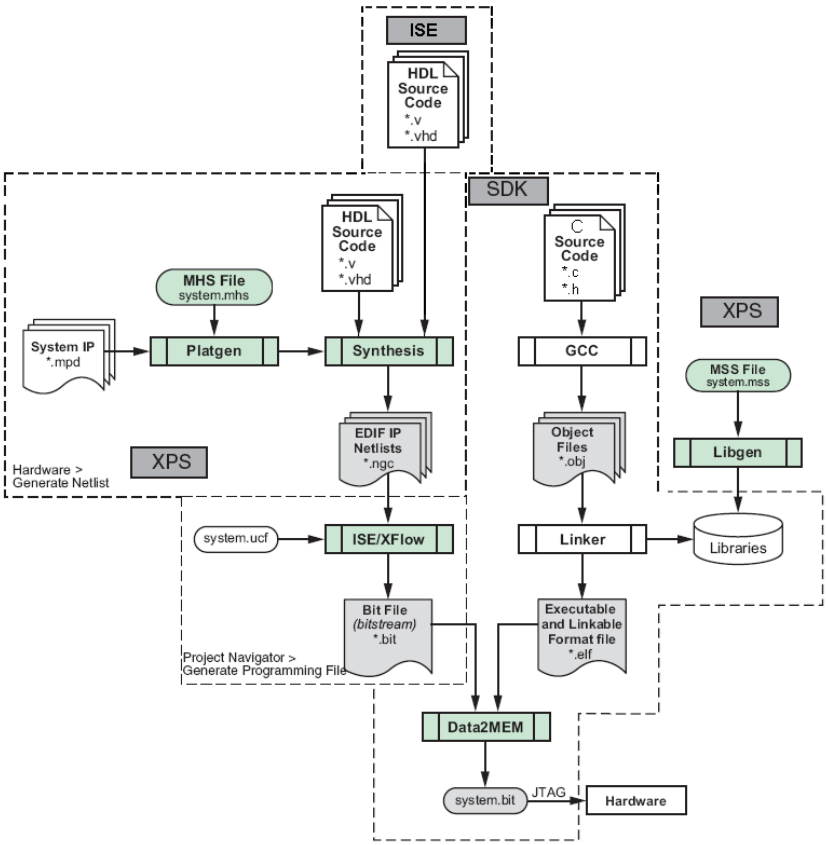


Abbildung 10: Elemente und Entwicklungsstufen von ISE, XPS und SDK zu einer FPGA-Konfiguration

2.6 Die Programmiersprache VHDL

VHDL steht kurz für VHSIC, auch VHSIC Hardware Description Language und ist mit einer Programmiersprache vergleichbar. Allerdings ist VHDL eine sehr umfangreiche Hardwarebeschreibungssprache, mit der digitale Systeme auf verschiedenen Abstraktionsebenen abgebildet und getestet werden können. Mit dieser Beschreibungssprache sollten Elemente bekannter Hochsprachen mit den nötigen Elementen zur effektiven Modellerstellung von parallelen Systemen auf verschiedenen Abstraktionsebenen kombiniert werden. VHDL wurde im Jahr 1987 vom IEEE standardisiert (IEEE 1076-1987). Vorwiegend werden mittels VHDL digitale Systeme beschrieben, allerdings gibt es sprachliche Erweiterungen in Form von VHDL-AMS, mit der auch analoge Systeme beschrieben werden können.

Mit VHDL können große und komplexe Schaltungen, die zeitlich wie ökonomisch eine hohe Effizienz erfordern, z.B. eines Mikroprozessors mit über 20 Mio. Transistoren, schnell entwickelt werden. Dabei wird in VHDL nicht das Verhalten einer Schaltung auf Ebene der elektronischen Bauteile, sondern auf einer höheren Abstraktionsebene beschrieben. So wird ein System simuliert, überprüft und anschließend wird eine Netzliste erstellt. Aus der Netzliste kann eine Maske für die Herstellung von MPGAs oder ähnlichen LSI-Chips produziert werden. Die Netzliste kann auch nach der Konvertierung in einen passenden Bitstream direkt in ein FPGA oder CPLD geladen werden.

VHDL ist sowohl vom Anwender als auch von Maschinen lesbar. Die Sprache eignet sich, um Schaltungsentwürfe zu beschreiben und auch mit Hilfe eines VHDL-Simulators zu simulieren. Zudem gewinnt VHDL als Schnittstellensprache zwischen Entwurfstools an Bedeutung. Mittels VHDL können komplexe Systeme bis hinunter zur Gatterebene modelliert und simuliert werden. Dabei können verschiedene Abstraktionsebenen in einem Modell auch gemischt werden. „Neben VHDL existieren Verilog und ABEL. Die weltweit meist genutzten Hardwarebeschreibungssprachen sind VHDL und Verilog. VHDL hat sich zum "Quasi-Standard“ in Europa entwickelt, in den USA ist dagegen Verilog die meist verwendete Sprache.“ [13]

Es ist nötig, zwischen synthesefähigem und funktionalem Code zu unterscheiden. Es gibt Entwürfe, die sich zwar simulieren lassen, aber nicht in reale Hardware übersetzt werden können. Das zur VHDL-Synthese gewählte Übersetzungsprogramm (Synthesetool) bestimmt primär, aus welchem funktionalem VHDL-Code tatsächlich synthesefähiger VHDL-Code wird. Funktionaler Code wird hauptsächlich zur Schaltungssimulation und zur Erstellung von Testbenches eingesetzt. Um synthesefähigen VHDL-Code zu erstellen, muss der Entwickler auf große Teile der Sprachmöglichkeiten von VHDL verzichten und die Zielhardware besser kennen, wodurch das Entwerfen von Systemen aufwendiger wird.

Ein VHDL-Modell lässt sich meist in drei Teilen beschreiben. Dazu gehören die Schnitt-

stellenbeschreibung, eine oder mehrere Architekturen und eine oder mehrere Konfigurationen. VHDL-Beschreibungen können hierarchisch aufeinander aufbauen.[10, Seite 8ff.]

- Funktionseinheit (Entity):
  In der Funktionseinheit werden die Ein- und Ausgänge (Schnittstellen) sowie Konstanten, Unterprogramme der zu modellierenden Komponente, des zu modellierenden Systems beschrieben sowie sonstige Vereinbarungen getroffen, die auch für alle zur Entity zugehörigen Architekturen gelten sollen.
- Architektur (Architecture):
  Eine Architektur enthält die Beschreibung der Funktionalität eines Modells. Hierfür gibt es verschiedene Möglichkeiten. Das Modell kann aus einer Verhaltensbeschreibung bestehen oder strukturalen Charakter (Netzliste) haben. Beide Möglichkeiten können miteinander kombiniert werden. Für eine Funktionseinheit können mehrere Architekturen definiert werden, d.h. es können für eine Komponentenschnittstelle mehrere Beschreibungen auf unterschiedlichen Abstraktionsebenen oder verschiedene Entwurfsalternativen existieren.
- Konfiguration (Configuration):
  In der Konfiguration wird festgelegt, welche der beschriebenen Architekturvarianten einer bestimmten Funktionseinheit zugeordnet ist und welche Zuordnungen für die möglicherweise verwendeten Submodule in der Architektur gelten. Hier können den untergeordneten Funktionseinheiten hierarchisch bestimmte Architekturen zugeordnet werden; außerdem ist es möglich, Parameterwerte an hierarisch tieferliegende Komponenten zu übergeben.

2.6.1 VHDL - Hardwarekomponenten

Es gibt schon viele Hardwarekomponenten, die von Xilinx für den Aufbau eines Systems zur Verfügung gestellt werden. Dazu gehören Komponenten wie z.B. Sigma-Delta-Wandler, IO-Module oder Peripheral Controller. Über den Menüpunkt *Hardware* $\rightarrow$ *Create or Import Peripheral...* in der XPS-Oberfläche können eigene Hardware-Komponenten, IPs genannt, wie z.B. Register erstellt und eingebunden werden. Die Schwierigkeit beim Erstellen und Einbinden einer eigenen Hardwarekomponente hängt ganz von der Komplexität der Komponente selbst ab. Soll diese Komponente an einen Bus angeschlossen werden, benötigt die Komponente ein entsprechendes Businterface, welches der Anwender entwerfen muss. In dieser Arbeit wurde ein einfaches Register mittels des Wizards erstellt und eingebunden. Ausführliche Informationen über das Erstellen eigener IP-Cores sind unter [17, Seite 37ff.], [8] zu finden.

2.7 Programmierung eines Softcores am Beispiel der MicroBlaze

2.7.1 Bibliotheken der MicroBlaze

Um in der SDK-Umgebung eine Referenz auf das MicroBlaze-System zu haben, werden alle nötigen Treiber, Makros etc. von der XPS-Umgebung automatisch aus den Systemdaten erstellt. Der Anwender kann bei der Erstellung einer neuen Softwareapplikation sein eigenes Softwareprojekt auf das MicroBlaze-System referenzieren und die benötigten Bibliotheken für die Hardware einbinden. In Abbildung 11 sind die automatisch erzeugten Header-Dateien des MicroBlaze-Systems dargestellt. Eine genaue Erklärung zum Ablauf ist unter [14, Seite 7ff.] zu finden.

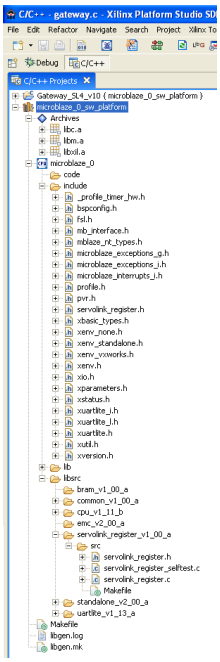


Abbildung 11: Die MicroBlaze-Bibliotheken im SDK

2.8 Softwarestand bei Beginn des Projektes

Zu Beginn der Entwicklung gab es schon einen Softwarestand. Dieser Softwarestand basierte auf einem Prototypen des Gateways. Bei diesem Projekt wurde das Hilscher-COM-Modul auf

einer PCI-Karte eingesetzt. Dabei übernahm der PC den Teil, den nun die MicroBlaze übernehmen sollte. Von der vorhandenen Software konnte ein großer Teil übernommen werden, allerdings mussten einige Anpassungen vorgenommen werden. Der übernommene Teil beinhaltet Zustandsmaschinen für das Hilscher-COM-Modul, das SERVOLINK 4-Modul und das gesamte Gateway. Die SERVOLINK 4-Zustandsmaschine wurde in diesem Diplomprojekt nicht benötigt und konnte daher weggelassen werden. Ungeändert übernommen wird die Hilscher COM-Modul-Zustandsmaschine und die Zustandsmaschine für das gesamte Gatewaysystem, welche im Zustand RUN um einen neuen Teil, der den Datenaustausch realisiert, erweitert wurde. Die Grundfunktionen der benutzten Zustandsmaschinen sind in den Abbildungen 12 und 13 dargestellt. Die Abbildungen sind einem firmeninternen Dokument, das einen Teil des Konzeptes für das Gateway auf Basis des Prototypen beinhaltet, entnommen.

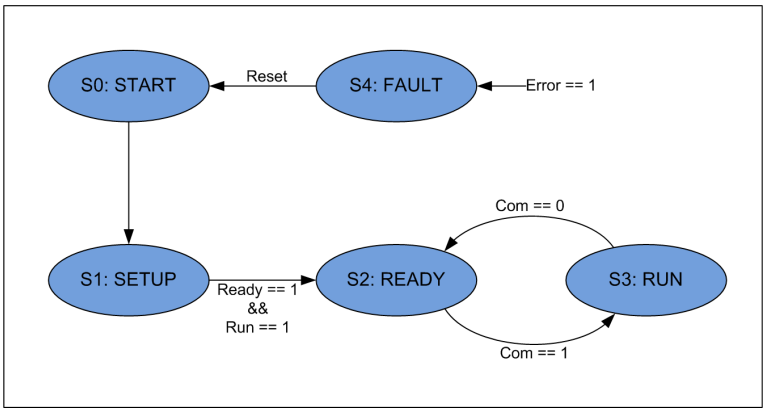


Abbildung 12: Zustandsmaschine des COM-Modules

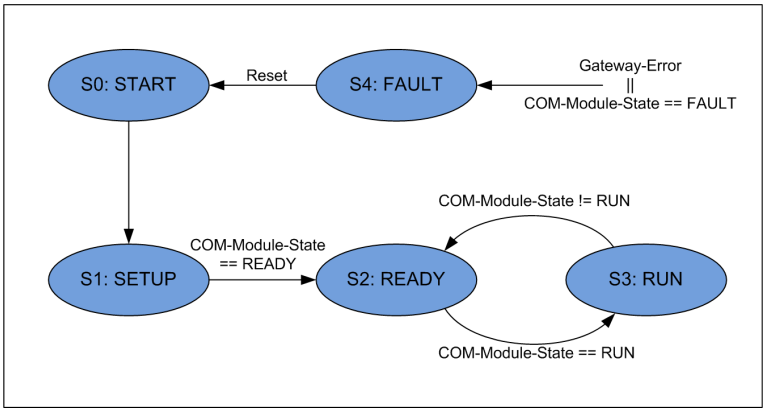


Abbildung 13: Zustandsmaschine des Gateways

3 Konzeption

3.1 Konzeption der Hardware

Das Konzept des Mikroprozessorsystems wurde anhand der vorliegenden Prototyp-Platine entwickelt. Dabei basiert die Entscheidung, als Softcore eine MicroBlaze zu implementieren, darauf, dass diese Art Softcore die Anforderungen, die an das Gateway gestellt sind, bewältigen kann. Das Gesamtkonzept des Systems ist in Abbildung 14 dargestellt.

Das Hilscher COM-Modul war als Kommunikationsschnittstelle mit PROFIBUS festgelegt. Daher wird ein externes Speicher Controller Interface (*External Memory Controller*) in das System implementiert, um auf den Dual Port RAM des Hilscher COM-Moduls zugreifen zu können. Das SERVOLINK 4-Modul generiert aus parallelen Daten die seriellen Daten, die über den Lichtwellenleiter gesendet werden. Genauso werden auch die seriellen Daten, die über den Lichtwellenleiter empfangen werden, wieder in parallele Daten umgesetzt. Daher wird jeweils ein DPRAM Block für die Sende- und Empfangsseite des SERVOLINK 4-Moduls verwendet. Jeder dieser beiden Speicherblöcke benötigt einen Dual Port RAM Controller, der von der MicroBlaze angesprochen wird und worauf hin die DPRAM Blöcke beschrieben bzw. ausgelesen werden. Bei einem einzelnen DPRAM Block müsste ein kompliziertes Timingkonzept für die Speicherzugriffe erstellt werden, da sonst Schreib- und Lesezugriffe auf den Speicher von Seiten der MicroBlaze und des SERVOLINK 4-Moduls gleichzeitig stattfinden könnten, wobei die Daten verfälscht würden bzw. verloren gingen. Zudem wäre die Zuordnung der zu jedem Antriebsregler zugehörigen Daten komplizierter. Dieser Sachverhalt wird im Kapitel 5 erläutert.

Der Softcore wird an einen 16 kByte großen Dual Port RAM angeschlossen, der als Daten- und Instruktionsspeicher dient. Es werden Register implementiert, die dazu dienen, Parameter und Anweisungen an die Hardware zu senden und Informationen und Zustände der Hardware zu lesen. Außerhalb der MicroBlaze, aber im FPGA, wird eine DCM eingebaut. Diese DCM erzeugt aus einem externen Quarztakt die für die MicroBlaze und die restliche Hardware benötigten Takte. Für Diagnosezwecke wird ein UART vom Typ RS232 eingebaut. Dieser UART wird in der fertigen Gatewayversion nicht mehr enthalten sein.

Da die Software Routine der MicroBlaze interruptgesteuert abläuft, wird aus Signalen des

SERVOLINK 4-Moduls mit Hilfe von parallelen Prozessen ein Interrupt erzeugt, der den Interrupt Eingang der MicroBlaze steuert.

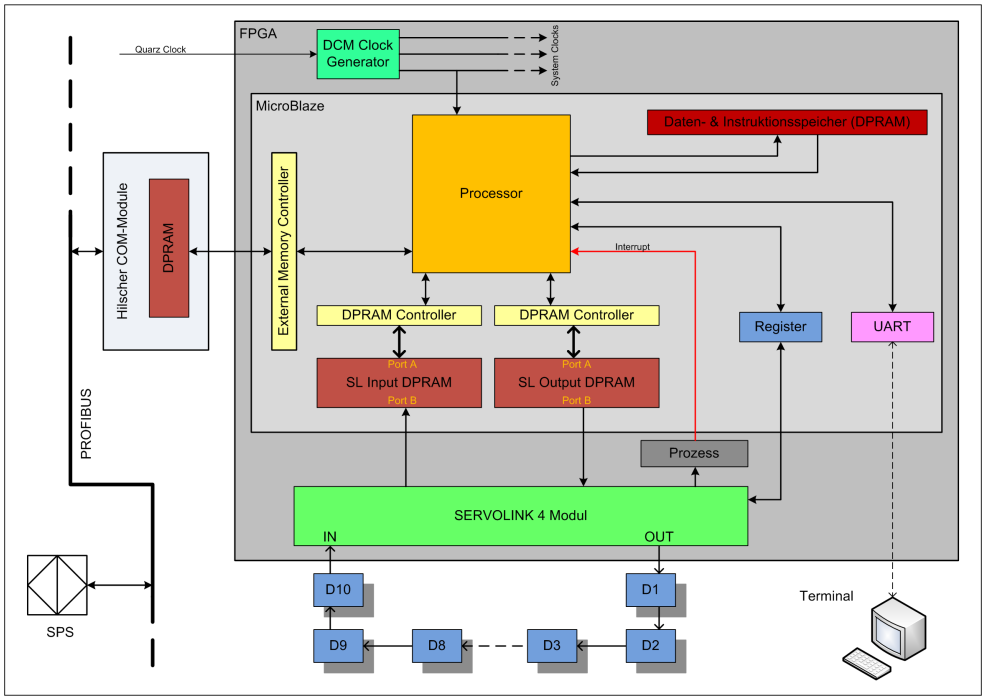


Abbildung 14: Konzept des MicroBlaze Systems

3.2 Interruptkonzept

Der Interrupt wird vom SERVOLINK 4-Modul am Ende eines Übertragungszyklus ausgelöst. Dann stehen maximal 120 μsec zur Verfügung, um die Daten zwischen dem Hilscher COM-Modul und SERVOLINK 4 auszutauschen. In Abbildung 15 ist zu sehen wie der Interrupt erzeugt wird. Das Signal „Daten kopieren“ nimmt High-Pegel an, wenn das SERVOLINK 4-Modul die Daten aller Antriebsregler empfangen hat. „Transmit in progress“ nimmt Low-Pegel an, wenn alle Daten übertragen wurden. Sobald diese Zustände mit der positiven Flanke des 96 MHz Taktes anliegen, wird der Interrupt erzeugt. Daraufhin werden die Daten kopiert und der Interrupt wird im ablaufenden Programm quittiert. Nun beginnt ein neuer SERVOLINK 4-Zyklus. Dies ist in Abbildung 16 dargestellt.

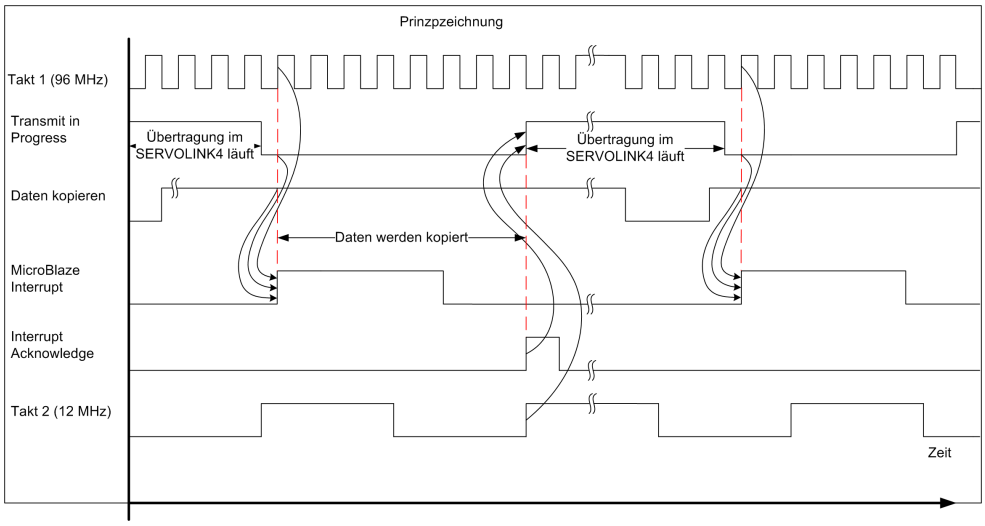


Abbildung 15: Der Hardware Interrupt und seine Quittierung

3.3 Konzeption der Software

In Abbildung 16 ist der prinzipielle Vorgang für die Verarbeitung und den Austausch der Prozessdaten zwischen PROFIBUS bzw. der Steuerungseinheit und SERVOLINK 4 bzw. den Antriebsreglern dargestellt. Dabei wird ein Interrupt vom SERVOLINK 4-Modul, wie in Abbildung 15 dargestellt, erzeugt. Beim Interrupt werden im Gateway die Daten aus dem Speicher des Hilscher COM-Moduls in den Ausgangsdatenspeicher für das SERVOLINK 4-Modul geschrieben. Ebenso werden die Daten aus dem Eingangsdatenspeicher des SERVOLINK 4-Moduls in den Speicher des Hilscher COM-Moduls geschrieben. Sind die Daten ausgetauscht, wird der Interrupt quittiert. Das SERVOLINK 4-Modul sendet daraufhin die Daten aus seinem Ausgangsspeicher an die Antriebe, und das Hilscher COM-Modul tauscht Daten mit der Steuerungseinheit aus. Da der SERVOLINK 4-Zyklus mit 500 μsec im schlechtesten Fall doppelt so schnell ist wie ein PROFIBUS-Zyklus, der mindestens 1 msec dauert, bekommen die Antriebe mehrmals die gleichen Prozessdaten bevor der PROFIBUS aktuelle Daten an das Hilscher-Modul sendet. Neben dem Datenaustausch muss das Hilscher COM-Modul noch initialisiert und parametrisiert werden, und es wird geprüft, in welchen Zustand sich das Hilscher COM-Modul und das Gesamtsystem befinden (siehe Kapitel Grundlagen).

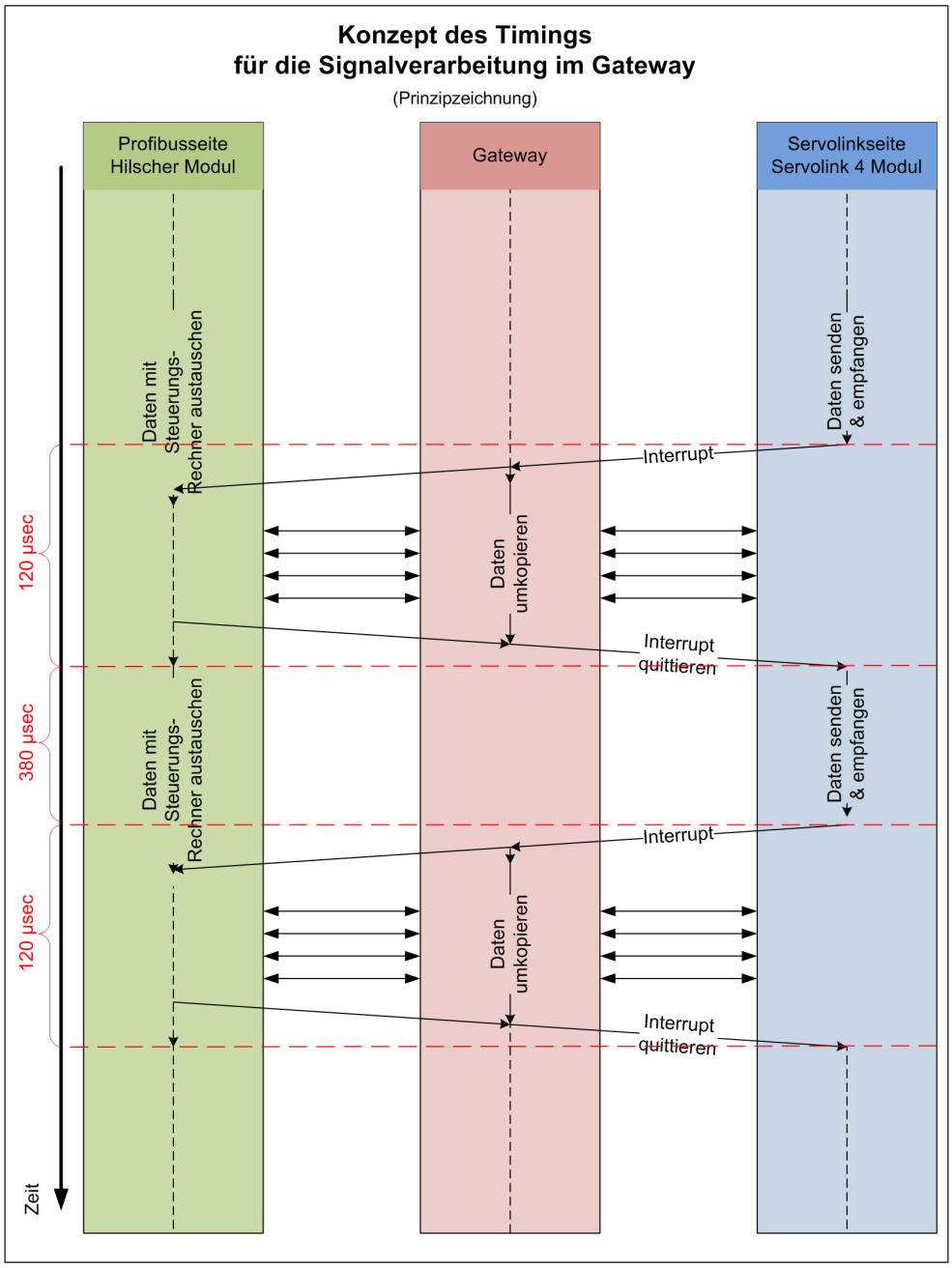


Abbildung 16: Timingkonzept

4 Verwendete Systemkomponenten

4.1 Das Gateway

Der Aufbau der Gateway-Platine besteht hauptsächlich aus den unten aufgelisteten Elementen. In Abbildung 17 ist die Platine abgebildet und die aufgezählten Elemente sind gekennzeichnet.

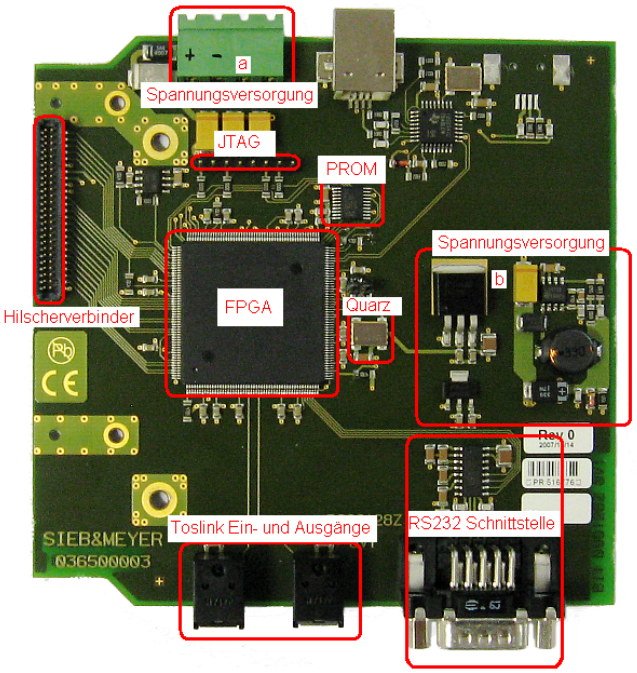


Abbildung 17: Die Gateway-Platine

- Xilinx Spartan 3E FPGA
- Xilinx PROM zum Laden des FPGAs
- Spannungversorgung

- Quarz aus dem alle wichtigen Takte gewonnen werden
- TOSLINK Ein- und Ausgangsbuchse für den LWL-Anschluss
- RS232 Pegelwandler mit Anschlussbuchse
- Verbindungsstecker zum Hilscher Modul
- JTAG Programmierschnittstelle

4.2 Xilinx Komponenten

4.2.1 Die Firma Xilinx

Das Gateway wird mit Hilfe eines FPGA der Firma Xilinx. Xilinx wurde 1984 gegründet und ist auf dem Gebiet der programmierbaren Logik weltweit führend. Im Jahr 2007 hatte das Unternehmen einen Marktanteil von über 51%. Das Unternehmen beschäftigt 3500 Mitarbeiter weltweit und hat über 20 000 Kunden. Die Hauptsitze des Unternehmens befinden sich in San Jose (USA), Dublin (Irland) und Singapur, daneben existieren unzählige Tochterunternehmen. Die Produktpalette umfasst Lösungen für etliche Problemstellungen der programmierbaren Logik und wächst stetig mit den Anforderungen der Kunden. Die Hauptmärkte, die Xilinx bedient, sind Luftfahrt, Verteidigung, Fahrzeugelektronik, Kommunikations- und Funktechnik, Industrie-, Forschungs- und Medizintechnik sowie Technik für den Verbraucher [15].

4.2.2 Spartan-3E XC3S500E FPGA

Die Spartan-3E Familie der Field-Programmable Gate Arrays (FPGAs) wurde speziell entwickelt, um die Bedürfnisse von kostensensiblen Großabnehmern für elektronische Anwendungen erfüllen zu können. Die Spartan-3E Familie besteht aus 5 FPGAs deren Anzahl an Systemgattern von 100 000 bis 1,6 Millionen reicht. In Tabelle 2 [20, Seite 3] sind die fünf FPGAs der Spartan-3E Familie mit ihren unterschiedlichen Attributen aufgelistet. In dieser Arbeit wird der Baustein XC3S500E verwendet.

Tabelle 2: Zusammenfassung der Spartan-3E FPGA Attribute

Device		XC3S100E	XC3S250E	XC3S500E	XC3S1200E	XC3S1600E
System Gates		100 K	250 K	500 K	1200 K	1600 K
Equivalent Logic Cells		2160	5508	10476	19512	33192
CLB Array	Rows	22	34	46	60	76
(1 CLB	Columns	16	26	34	46	58
=	Total CLBs	240	612	1164	2168	3688
4 Slices)	Total Slices	960	2448	4656	8672	14752
Distributed RAM bits		15 K	38 K	73 K	136 K	231 K
Block RAM bits		72 K	216 K	360 K	504 K	648 K
Dedicated Multiplies		4	12	20	28	36
DCMs		2	4	4	8	8
Maximum User I/O		108	172	232	304	376
Maximum Differential I/O Pairs		40	68	92	124	156

Die FPGA-Familie unterstützt eine große Liste an Features, die in der Dokumentation *„Spartan-3E FPGA Family: Complete Data Sheet“* auf der Homepage von Xilinx nachgelesen werden können. [20]
Viel wichtiger als die Funktionsumfang des FPGA ist die Architektur der Spartan-3E Familie. Diese besteht aus fünf grundlegenden funktionalen Elementen:

- Konfigurierbare Logik Blöcke (Configurable Logic Blocks (CLBs)): Die CLBs beinhalten flexible Look-Up Tables (LUTs), welche Logik- zuzüglich Speicherelementen aus Flip-Flops oder Latches umsetzen. Der Einsatz der CLBs ist sehr vielfältig. Sie können sowohl für logische Funktionen als auch für die Speicherung von Daten eingesetzt werden.
- Eingangs- und Ausgangsblöcke (Input/Output Blocks (IOBs)): Die IOBs kontrollieren den Datenfluss zwischen den I/O Pins und der internen Logik des Bausteins. Jeder IOB unterstützt bidirektionalen Datenfluss sowie Tristate. Es wird eine große Bandbreite an Signalstandards unterstützt inklusive vier high-performance differential Standards. Auch Double-Data Rate (DDR) Register werden unterstützt.

- die Block RAMs unterstützen eine Speicherung von Daten in Form von 18-KBit dual-port Blöcken.
- die Multiplizierblöcke können binäre Zahlen bis 18-Bit Datenlänge multiplizieren
- Digital Clock Manager (DCM) unterstützen Selbstkalibrierung, komplett digitale Lösung von Auflösen, Verzögern, Multiplizieren, Dividieren und Phasenverschieben von Takt Signalen.

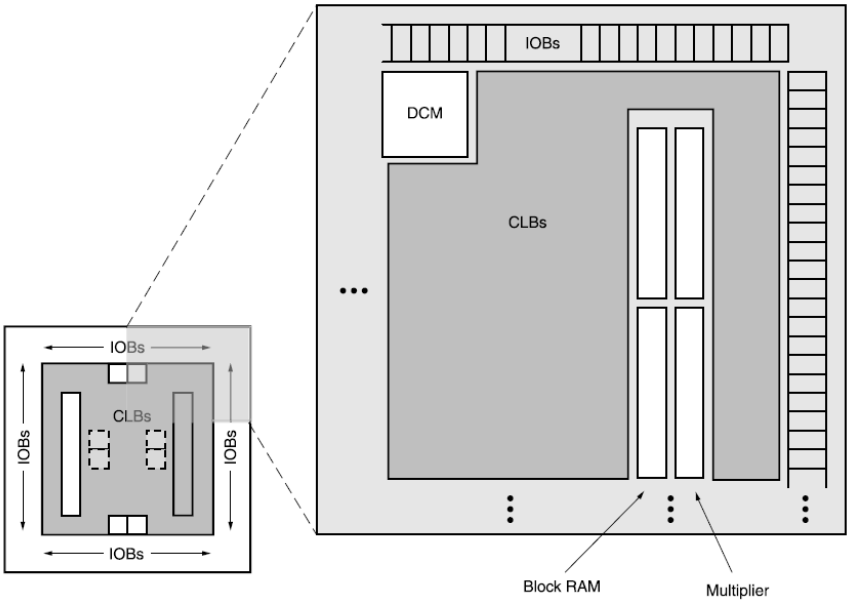


Abbildung 18: Architektur der Spartan-3E Familie [20, Seite 4]

In Abbildung 18 ist die Organisation dieser Elemente dargestellt. Die IOBs umranden ein geordnetes Feld von CLBs. Jeder Baustein besitzt zwei Block RAM Reihen, nur der XC3S100E besitzt eine Block RAM Reihe. Jeder RAM besteht aus mehreren 18-KBit großen RAM Blöcken und ist verbunden mit einem zugehörigem Multiplizierer. Es sind jeweils zwei DCMs in der Mitte des Bausteins oben und unten positioniert. Der XC3S100E besitzt jeweils nur eine DCM und der XC3S1200E und der XC3S1600E besitzen zusätzlich noch zwei DCMs, jeweils rechts und links. Die Spartan-3E Familie besitzt ein großes Netzwerk, welches alle fünf Elemente untereinander verbindet und in welchem Signale ausgetauscht werden.

4.2.3 Xilinx XCF04S PROM

Die XCFxxS-Serie von Xilinx ist eine Reihe von Platform Flash PROMs. Sie heißen PROMs, sind aber in Wirklichkeit löschbar und wieder beschreibbar. Diese PROMs sind in ihrer Funktion sehr ähnlich. Der wichtigste Unterschied zwischen den einzelnen Bausteinen ist ihre Speichergröße bzw. -dichte. Die Speichergröße des XCF04S PROMs beträgt 4 Mbit, und er kann mit mehreren anderen PROMS in Reihe geschaltet den zu ladenen FPGA-Code enthalten. In Abbildung 19 ist dargestellt wie zwei XCF04S Platform Flash PROMs zum Laden eines FPGAs genutzt werden. Sobald an den Bausteinen Betriebsspannung anliegt und ein Takt (CCLK) sowie das Startsignal (INIT) vom FPGA gesendet werden, beginnt das PROM die Daten (DATA) an das FPGA zu senden. Sobald das FPGA fertig geladen ist, meldet es dem PROM den Zustand Fertig (DONE) und der Datenstrom wird unterbrochen. Das FPGA beginnt nun den geladenen Programmcode auf der geladenen Hardware auszuführen [19].

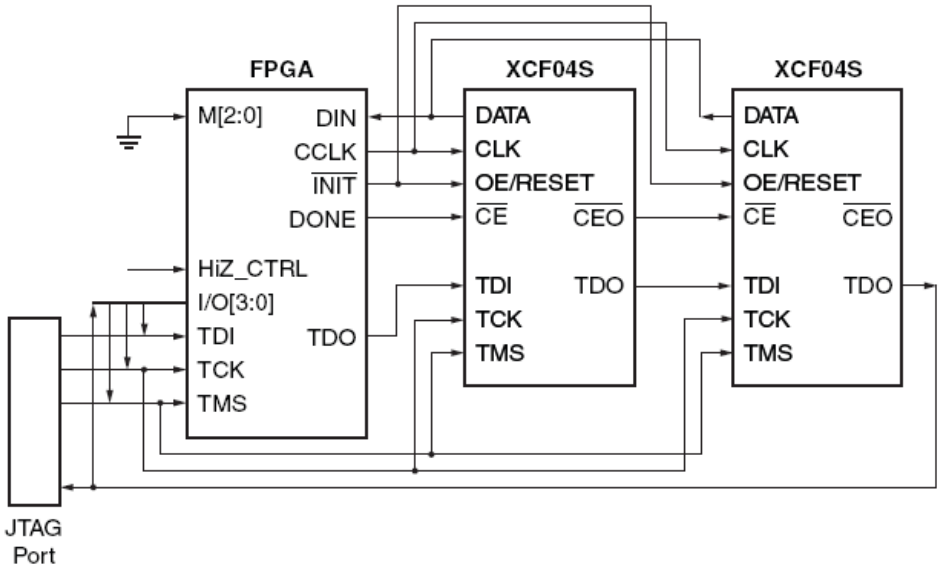


Abbildung 19: Ladevorgang eines FPGAs durch zwei XCF04S Platform Flash PROMs [21]

4.3 Spannungsversorgung

Die Betriebsspannungen für die Bauteile auf der Platine werden durch zwei zusammenhängende Funktionsteile geliefert. An der Spannungsversorgung 1 werden +24 Volt eingespeist. Durch

die Schaltung in Spannungsversorgung 2 werden daraus die Spannungen +3.3 Volt, +2.5 Volt und +1.2 Volt erzeugt (siehe Abbildung 17). Diese Spannungen sind die Betriebsspannungen für alle Bauteile auf der Platine.

4.4 Sonstige Elemente

Weitere nennenswerte Elemente auf der Platine sind eine RS232 Schnittstelle, die Verbindung zum Hilscher COM-Modul, eine JTAG-Verbindung zum Programmieren des FPGAs und des PROMs, die TOSLINK-Buchsen für die Lichtwellenleiter des SERVOLINK 4-Busses, eine USB-Schnittstelle und ein 48 MHz Quarz. Von diesen Elementen werden in der ersten Produktion die RS232- und USB-Schnittstelle nicht bestückt, da sie zunächst nicht verwendet werden. Weiteres dazu in Kapitel 6 auf Seite 100.

4.5 Hilscher Kommunikationsmodul

4.5.1 Die Firma Hilscher

Die Firma *Hilscher* ist ein unabhängiges, inhabergeführtes Familienunternehmen, das seit 1986 existiert und seit 2001 stetig gewachsen ist. Der Hauptsitz der Firma befindet sich in Hattersheim bei Frankfurt (Deutschland). Weitere Niederlassungen befinden sich in Vimodrone (Italien), Chicago (USA), Solothum (Schweiz), Bron (Frankreich), Berlin (Deutschland), Tokyo (Japan), Shanghai (China), Diepoldsau (Schweiz), Varna (Bulgarien) und Dehli (Indien). *Hilscher* bietet neben den Kommunikationsmodulen noch viele weitere Produkte im Bereich Kommunikationstechnik, wie PC-Karten, Analyzer und vieles Anderes, an.[6]

4.5.2 Das COM-Modul und seine Funktion

Für dieses Gateway wurde ein Hilscher COM-CA-DPS-Modul benutzt. Dieses Modul arbeitet als *PROFIBUS*-Slave und ist ein kleines, kompaktes Aufsteckboard, welches das gesamte PROFIBUS-Interface einschließlich der potentialfreien Treiberschnittstelle enthält. Der Datenaustausch mit dem Host-System erfolgt über eine Dual-Port Memory Schnittstelle. Hier werden die Prozessdaten in einem Abbildspeicher geführt, während Kommandos über eine Mailbox übertragen werden. Das COM-Modul übernimmt die Daten und führt die komplette Übertragung über das PROFIBUS Netzwerk selbständig ohne Belastung des Host-Systems aus.[5]

Das COM-CA-DPS-Modul besitzt einen 8 Kilobyte großen Dual-Port Speicher und einen Interrupt Ausgang. Es wird mit einer 50-poligen Pfostenbuchse im 1,27er Raster auf die

Gateway-Platine aufgesteckt. Die PROFIBUS-Schnittstelle EN 50 170 arbeitet mit einer Übertragungsrate von 9,6 kBaud bis 12 MBaud. Als Controller wird ein EC1 verwendet mit einer RS485-Schnittstelle über eine 9-polige D-Sub-Buchse. Das Modul zeigt seinen Status mittels zwei LEDs an, wobei zwischen den Status Ready, Run, Start und Error unterschieden wird. Zudem sind neben der D-Sub-Buchse zwei Drehschalter angebracht, mit denen die PROFIBUS-Adresse des Moduls eingestellt wird [5]. In Abbildung 20 ist das Modul abgebildet.

Die Aufteilung des Speichers des Hilscher-Moduls ist in Abbildung 21 dargestellt. Dabei wird die komplette Synchronisation der Daten und der allgemeine Zustand des Moduls in den beiden letzten Bytes des Speichers abgebildet.

Abbildung 20: Hilscher COM-CA-DPS Modul

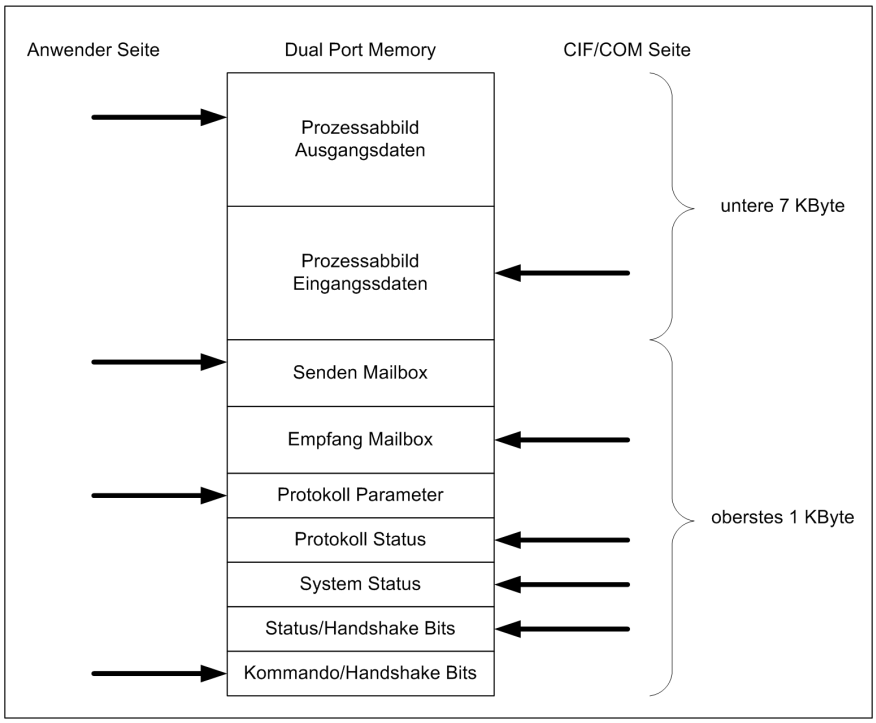


Abbildung 21: Speicherinterface des Hilscher Speichers

4.6 SERVOLINK 4-Modul

Das SERVOLINK 4-Modul ist eine Systemkomponente, die in digitaler Form vorliegt. Das Modul ist in VHDL geschrieben und wird auch in anderen Projekten eingesetzt. Die Aufgabe des Moduls ist es, die parallelen Ausgangsdaten aus dem zugehörigen DPRAM Block in 8-Bit-Blöcken zu lesen, diese 8-Bit-Blöcke in 10-Bit-Blöcke zu kodieren und diese Blöcke dann seriell mit 12 MHz an die über die Lichtwellenleiter angeschlossenen Geräte zu schicken. Auf der Eingangsseite wird der Lichtwellenleiter bzw. dessen digitales Abbild mit 96 MHz abgetastet. Aus den empfangenen Daten wird mittels Phase-Lock-Loop wieder ein 12 MHz Takt erzeugt, der synchron zu den empfangenden Daten ist. Sind 10 Bit bzw. 8 Bit empfangen, werden diese in das jeweils adressierte Byte eines 32 Bit breiten Datenblocks geschrieben. Das SERVOLINK 4-Modul gibt die Lese- und Schreibadressen der Speicherblöcke auf Grund der Drive-, Word- und Bitposition vor. Diese werden mittels Zählern sowohl für die Eingangsseite als auch für

die Ausgangsseite ermittelt. Sind alle Antriebe abgearbeitet, wird von dem Modul ein Interrupt erzeugt. Wird der Interrupt nicht quittiert, werden im nächsten Übertragungszyklus Nullen an die Antriebe gesendet. Bei einer Interruptquittierung werden die Prozessdaten gesendet.

4.7 Antriebsregler

Als Antriebsregler wurden verschiedene Geräte aus der Palette der Antriebssysteme SD2 und SD2S benutzt. Diese sind in der Abbildung 22 abgebildet. All diese Antriebe können mit der Software *drivemaster2* der Firma SIEB & MEYER parametriert und in verschiedenen Betriebsarten betrieben werden, zudem dient die Software zur Diagnose. Sie enthält unter anderem einen Busmonitor, mit dem die Soll- und Istwerte, die über SERVOLINK 4 gesendet werden, beobachtet werden können. Teilweise wurden an die Antriebe Motoren angeschlossen, um zu sehen, ob die Datenübertragung funktioniert und die Motoren drehen.

Auf der linken Seite in Abbildung 22 ist ein SD2S-Kompaktgerät zu sehen. Daneben ist ein SD2-Kompaktgerät zu sehen. Dieses Gerät besitzt ein integriertes Netzteil und ist für Anwendungen mit wenigen Antrieben praktisch; es existieren aber auch Versionen ohne integriertes Netzteil. Die Geräte können Linearmotoren, rotative Motoren, hochpolige Torquemotoren oder synchrone/asynchrone Werkzeugspindeln mit oder ohne Sensor antreiben. Die Spezialität der Geräte ist das Antreiben von Synchron- und Asynchronmotoren mit Drehzahlen von bis zu 480.000 1/min. Zur Anbindung an die übergeordnete Steuerung werden analoge Sollwertsignale (+/-10 V) oder Puls-Richtung-Signale verarbeitet. Alternativ ist eine Ankopplung an eine CNC-Steuerung über das Bussystem SERVOLINK 4 möglich. Für eine PC-basierte Steuerung existiert eine SERVOLINK 4 PCI-Einsteckkarte. Die Verbindung zwischen der CNC-Steuerung und den Antriebsverstärkern wird durch Lichtwellenleiter realisiert, die eine besonders störsichere Verbindung gewährleisten.

- **Universelles Motorgeber-Interface** - SD2 und SD2S bieten die Möglichkeit, alle am Markt gängigen Messsysteme für rotative und lineare Motoren auszuwerten. Dazu gehören unter anderem Absolutwertgeber mit EnDat, Hiperface oder SSI-Schnittstelle, Encoder, Feldplatte, Hallsensor, linearer Hallsensor, Linearmaßstab mit 1Vss oder TTL-Pegel und Resolver.
- **Integrierte Sicherheit** - Mit der integrierten Anlaufsperre wird die Sicherheitskategorie 4 gemäß EN 954-1 erreicht, dadurch können externe Schutzschaltungen reduziert werden. Es werden die Anforderungen gemäß SIL 3 nach EN 61508 erfüllt.
- **Flexible Kühlung** (gilt nur für das SD2 Gerät) - In der Standardausführung wird der SD2 mit integriertem Kühlkörper ausgeliefert. Optional kann er als Coolplate-Version

Abbildung 22: SD2- und SD2S-Antriebsregler

bezogen werden. Dies ermöglicht den Einsatz einer Flüssigkeitskühlung, die Montage von Durchsteckkühlkörpern oder den Anbau von luftgekühlten Lamellenkühlkörpern.

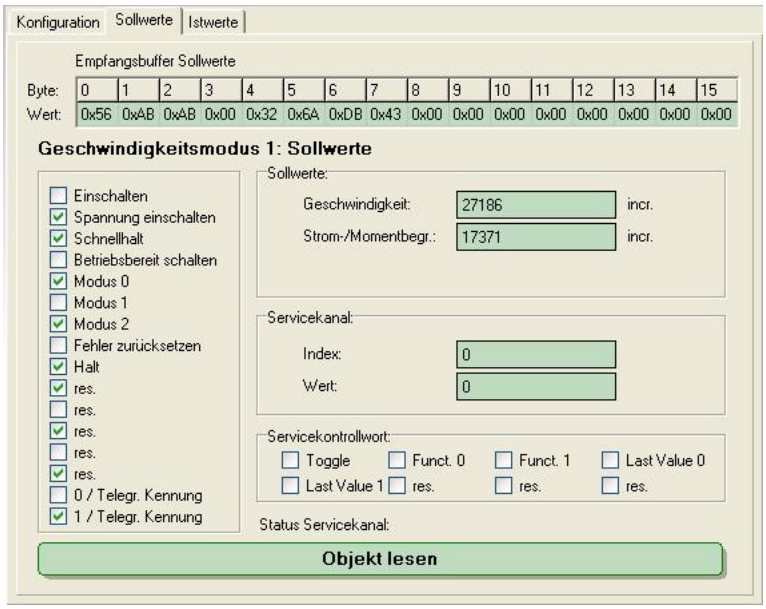


Abbildung 23: Sollwerte im SERVOLINK 4-Busmonitor

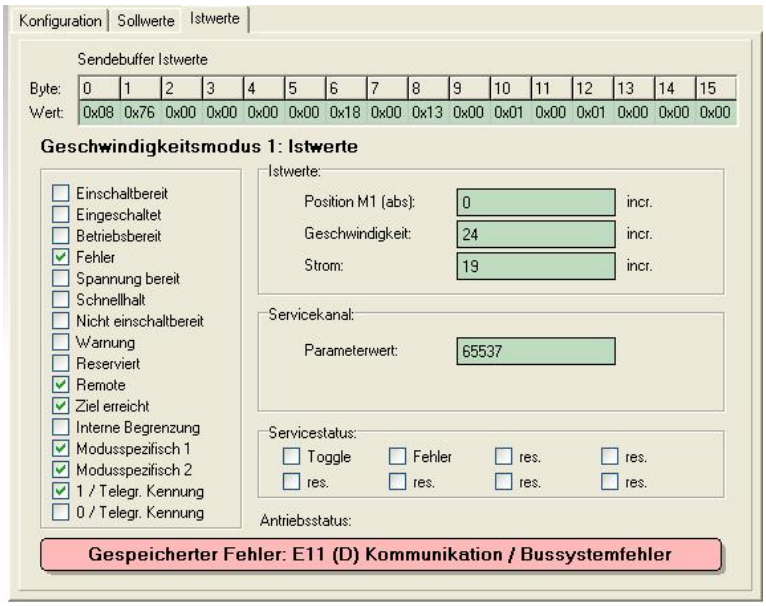


Abbildung 24: Istwerte im SERVOLINK 4-Busmonitor

4.8 PROFIBUS Master

Für die Entwicklung wird eine Siemens SPS als PROFIBUS Master eingesetzt. Die SPS ist vom Typ SIMATIC S7-300 und besitzt die unten aufgelisteten Baugruppen.

- PS307 2A:
  Die PS307 2A ist die Power-Baugruppe. Sie liefert eine Betriebsspannung von 24 Volt
- CPU315-2DP:
  Die CPU315-2DP ist die CPU Baugruppe. Dies ist die Recheneinheit der SPS.
- SM323:
  Die SM232 ist eine Ein- und Ausgabebaugruppe. Dort können jeweils acht digitale Ein- sowie Ausgänge verschaltet werden. Von dieser Baugruppe sind zwei Stück vorhanden.

Abbildung 25: Die verwendete SPS

5 Implementierung

5.1 Entwicklung der VHDL-Komponente MicroBlaze

Die VHDL-Komponente MicroBlaze besteht nicht nur aus dem reinen Prozessor, sondern beinhaltet zusätzliche IP-Cores. Die Abbildung 27 zeigt das Blockdiagramm der VHDL-Komponente MicroBlaze wie es von der Entwicklungsumgebung XPS aus den einzelnen IP-Cores gebildet wird. Abbildung 26 zeigt die zugehörige Legende. Es sind die einzelnen IP-Cores, d.h. der Prozessor (*microblaze*), die Register (*servolink_register*), der External Memory Controller (*xps_mch_ emc*), die DPRAM Blöcke mit ihren Controllern (*bram_block*) und (*xps_bram_if_cntlr*), der Daten- und Instruktionsspeicher mit seinen Controllern (*bram_block*) und (*lmb_bram_if_cntlr*) bzw. (*imb_bram_if_cntlr*) sowie der RS232 UART (*xps_uart*) und das Debug-Modul (*mdm*) abgebildet, und es sind die Verknüpfungen dieser IP-Cores über die einzelnen Systembusse dargestellt. Die unterschiedlichen Busse werden farblich gekennzeichnet, so dass klar zwischen dem PLB (*(PLB_v46_0)*) und dem DLMB bzw. ILMB unterschieden werden kann. Zudem ist gekennzeichnet, welcher IP-Core einen Master auf einem Bus darstellt und welcher einen Slave. Zusätzlich wird die Verbindung zwischen dem Debug-Modul und dem Prozessor (*microblaze_0_dbg*) dargestellt.

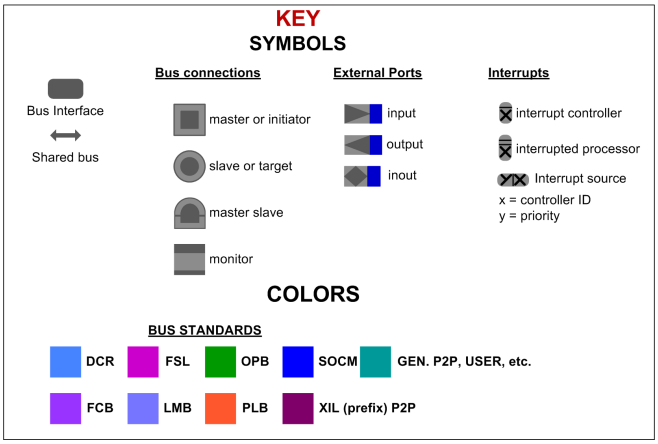


Abbildung 26: Legende zu Abbildung 27

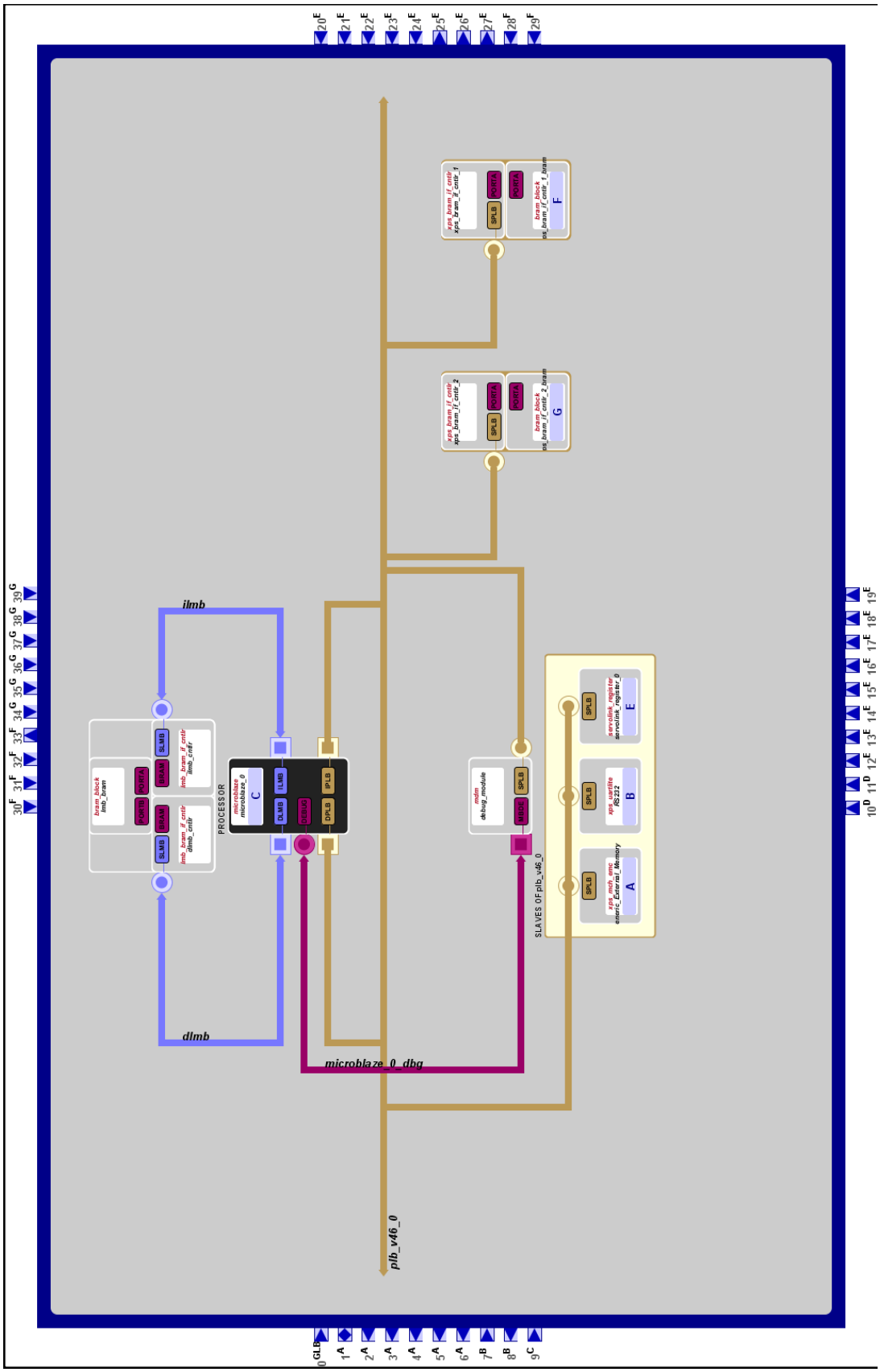


Abbildung 27: Blockdiagramm des MicroBlaze-Systems

5.1.1 Der MicroBlaze Rechenkern

Der reine Rechenkern des Mikroprozessorsystems, der IP-Core MicroBlaze, ist das Kernstück der gleichnamigen VHDL-Komponente. Er besitzt eine Reihe von konfigurierbaren und wenige feste Eigenschaften. Von diesen Eigenschaften wurden in dieser Arbeit nur einige genutzt. Die Architektur der MicroBlaze ist in Abbildung 28 zu sehen. Es handelt sich um einen 32 Bit RISC Mikroprozessor. „Dieser beinhaltet eine konfigurierbare Pipeline (3- bis 5-stufig), einen internen Cache, einen Interrupt, einen hardwarebasierten Multiplizierer und optional eine hardwarebasierte Divisionseinheit, eine Gleitkommaeinheit und spezielle Schieberegistereinheiten. Er nutzt mehrere unterschiedliche Busse, welche für den Anschluss von umfangreicher Peripherie und Speicher in einem FPGA vorgesehen sind. Der Prozessor ist primär an einem FPGA-internen CoreConnect-Bus (auch bekannt als PLB) angeschlossen. Zusammen mit optionalem externen Speicher und weiteren Peripherieeinheiten (IP-Cores) am PLB oder OPB entsteht ein System-on-a-Chip .“[1] Im Folgenden werden die Eigenschaften der MicroBlaze aufgelistet.[18, Seite 11]

Konfigurierbare Eigenschaften

- Anzahl der Stufen der Prozessor Pipeline
- OPB Interface für die Daten- und Instruktionsseite
- LMB Interface für die Daten- und Instruktionsseite
- Hardware Barrel Shifter
- Hardware Dividierer
- Hardware Debug Logik
- FSL Interface
- Setz- und Löschinstruktionen
- Instruktionspuffer über IOPB Interface
- Datenpuffer über IOPB Interface
- Instruktionspuffer über CacheLink (IXCL) Interface
- Datenpuffer über CacheLink (DXCL) Interface
- 4- oder 8-word Cache Line für XCL
- Unterstützung von Hardware Exceptions
- Bitmustervergleiche
- FPU

- Hardware Multiplizierer (auch für 64-Bit Ergebnisse)
- ESR und EAR bei Hardware Debug
- Versionsregister des Prozessors (PVR)
- Platz- oder Geschwindigkeitsoptimierung
- LUT Puffer Speicher
- PLB Interface für die Daten- und Instruktionsseite
- Fließpunkt Konvertierung und Wurzelrechnung
- MMU
- erweiterte FSL Instruktionen
- Benutzung von XCL für alle I-Cache Speicherzugriffe
- Benutzung von XCL für alle D-Cache Speicherzugriffe

Feste Eigenschaften

- 32 (Anzahl) 32-Bit Universalregister
- 32-Bit Instruktionen mit drei Operanden und zwei Adressierungsarten
- 32-Bit Adressbus
- Leitungen für einzelne Anforderungen - Single Issue Pipeline

Zu dieser mit einer CPU vergleichbaren Recheneinheit gehört ein Dual Port RAM Block, der als Daten- und Instruktionsspeicher dient. Dieser Speicherblock ist besitzt für jeden Port einen eigenen Speicher Interface Controller. Auf Port A sind die Instruktionen abgelegt und auf Port B die Daten. Dieser Speicher und dessen Anbindung an den Prozessor werden bei der Erzeugung der MicroBlaze mittels Wizard automatisch erzeugt. Dabei kann angegeben werden wie groß der Speicher sind soll. Das Auswählen der Eigenschaften wird beim Erstellen des Systems mit dem Assistenten sehr erleichtert. Bei der Verbindung der MicroBlaze zu anderen IP-Cores traten keine Probleme auf. Auch die XPS-Oberfläche erleichtert die Signalanbindung.

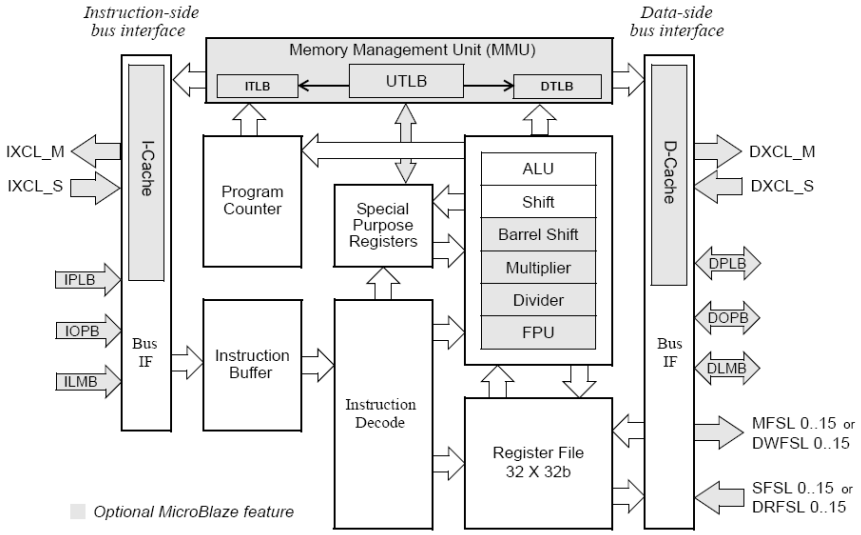


Abbildung 28: Blockdiagramm der Architektur der MicroBlaze [18, Seite 10]

5.1.2 Der XPS Multi-CHannel External Memory Controller

Der Xilinx Multi-CHannel External Memory Controller (XPS MCH EMC) stellt das Control Interface für externe synchrone, asynchrone SRAM- und FLASH-Speicher über den PLB dar. Das XPS MCH EMC ist ein Soft-IP-Core und ist für Xilinx FPGAs entworfen. Dieses Interface wird genutzt, um auf den Speicher des Hilscher COM-Moduls zugreifen zu können. Der XPS Multi-CHannel External Memory Controller besitzt folgende Eigenschaften:

- Anschluss als 32-bit Slave an einen PLB Bus mit 32, 64 oder 128 bit Busbreite
- Parametrierbares Channel Interface von bis zu 4 Kanälen
- Nutzung als reines PLB oder MCH Interface oder als Kombination aus beiden
- Anbindung von bis zu 4 Speicherbänken
- Single-Beat- and Burst-Übertragungen
- Geringe Latenzen bei PLB point-to-point Strukturen
- Synchrone, asynchrone SRAM- und FLASH-Speicher

- 8-bit, 16-bit und 32-bit Datenbreiten
- Abgleich von Datenbreiten wird unterstützt (die Datenbreite des Speichers muss dabei kleiner oder gleich der des PLB sein)
- Die Zykluszeiten für Lese- und Schreibzugriffe sind frei konfigurierbar

Ein Aufbau des XPS MCH EMC ist in Abbildung 29 dargestellt. Auf der linken Seite ist die Anbindung an den PLB bzw. an die MCHs zu sehen. Auf der rechten Seite sind die Signale zu sehen, die an den physikalischen externen Speicher angebunden werden müssen.

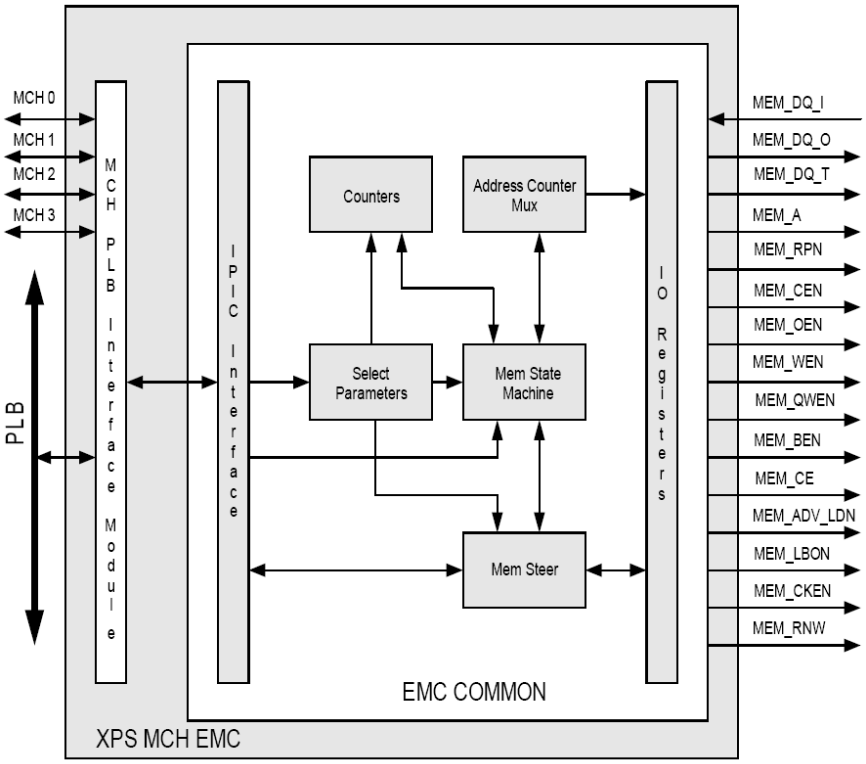


Abbildung 29: Blockdiagramm des External Memory Controller [25, Seite 2]

Die in dieser Arbeit genutzten Signale des XPS MCH EMC sind in Tabelle 9 im Anhang A auf Seite 107 dargestellt. Dies ist nur ein kleiner Teil aller Signale der Komponente. Die gesamte Liste der Signale kann in [25] auf den Seiten 5 bis 7 nachgelesen werden. Die Tabellen 11

und 10 im Anhang A auf den Seiten 109 und 108 enthalten die wichtigsten Parameter, die komplette Liste kann ebenfalls in [25] auf den Seiten 8 bis 11 eingesehen werden.
Die Parameter, die für das Timing bei den Schreib- und Lesezugriffen verantwortlich sind, sind in Tabelle 3 aufgelistet.

Tabelle 3: Verwendete Parameter

Parameter Name	**Wert**
C_TCEDV_PS_MEM_0	150000 psec
C_TAVDV_PS_MEM_0	150000 psec
C_THZCE_PS_MEM_0	70000 psec
C_THZOE_PS_MEM_0	70000 psec
C_TWC_PS_MEM_0	150000 psec
C_TWP_PS_MEM_0	120000 psec
C_TLZWE_PS_MEM_0	30000 psec

Diese Werte wurden aufgrund der Timing-Diagramme in den Abbildungen 40 und 41 auf Seite 73 für den Speicher des Hilscher COM-Moduls eingestellt.

5.1.3 Der XPS Block RAM Interface Controller

Die Implementierung von Speicherblöcken in einer MicroBlaze benötigt zum Einen den Speicherblock selbst, zum Anderen ein Interface zum PLB, über den die MicroBlaze Schreib- und Lesezugriff auf den Speicherblock hat.

Der XPS Block RAM Interface Controller ist ein Xilinx IP-Core, der ein PLB v4.6 Interface besitzt. Dieser Controller ist für byteweisen Zugriff entworfen und jede Zugriffsweite (in Byte) bis zur parametrierten Datenbreite des Speicherblocks ist erlaubt. Der XPS Block RAM Interface Controller besitzt folgende Eigenschaften:

- PLB v4.6 Interface mit Byte Enable Funktion
  - 32, 64 und 128 Bit breites PLB Interface
  - 32, 64 und 128 Bit Speicherbreite
  - 32 Bit Adressbreite

- Es werden drei verschiedene Übertragungsarten unterstützt
  - Single Data Beat
  - Cacheline 4 oder 8 Bit
  - feste Burstlängen von 2 bis 16 Data Beats

Ein Aufbau des XPS Block RAM Interface Controllers ist in Abbildung 30 dargestellt. Der Block RAM Controller wurde bei der Erstellung der MicroBlaze automatisch vom Assistenten an den MicroBlaze Prozessor sowie an den Block RAM angeschlossen. Eine Verbindung des Controllers mit dem Block RAM kann auch manuell erstellt werden, da die Signale/Ports des Speichers und des Controllers die gleichen Namen tragen. Die Signaltabelle für den Block RAM Interface Controller ähnelt der des Block RAM auf Seite 111. Einige der Parameter für den Controller sind in Tabelle 12 im Anhang A auf Seite 110 abgebildet. Die vollständigen Tabellen befinden sich in dem Dokument [22, Seite 3ff.].

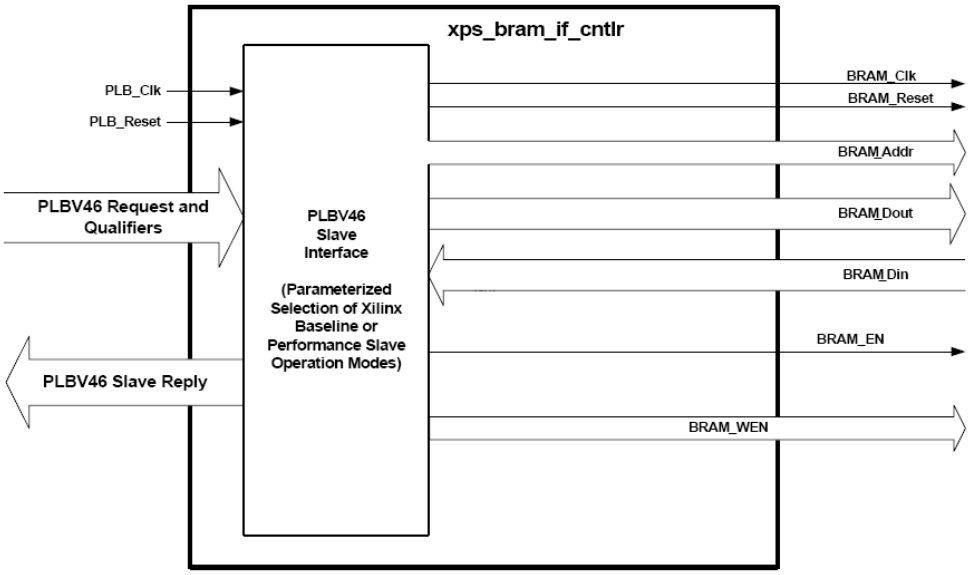


Abbildung 30: Blockdiagramm des Block RAM Interface Controllers [22, Seite 2]

5.1.4 Der XPS DPRAM Block

Für den Speicherblock, der an den XPS Block RAM Interface Controller angeschlossen wird, wird der BRAM Block verwendet. Dieser ist ebenfalls ein IP-Core, der sich an den XPS Block

RAM Interface Controller anschließen lässt. In dieser Arbeit wurden zwei DPRAM Blöcke eingesetzt, über die die Prozessdaten mit dem SERVOLINK 4-Modul ausgetauscht werden. Ein Block für die Sollwerte und ein Block für die Istwerte. Jeder dieser Blöcke ist 8 KByte groß. Da nur maximal 384 Byte in jede Richtung kopiert werden müssen, bleibt bei 8 KByte Speicherblöcken Speicher ungenutzt. Weiteres dazu in Kapitel 6 auf Seite 100. Ein DPRAM Block besitzt folgende Eigenschaften:

- Komplett automatische Erzeugung und Konfiguration der HDL mittels EDK Platgen/Simgen Tools
- Es können Datenbreite, Adressraum, die Anzahl an Byte-Enables und die Zielarchitektur eingestellt werden
- Er besitzt zwei Ports, Port A und Port B, die an voneinander unabhängige Block RAM Interface Controller angeschlossen werden können
- Es werden byteweise, halbwortweise, wortweise und doppelwortweise Übertragungen unterstützt, sofern die richtige Anzahl an Byte-Write-Enables konfiguriert ist

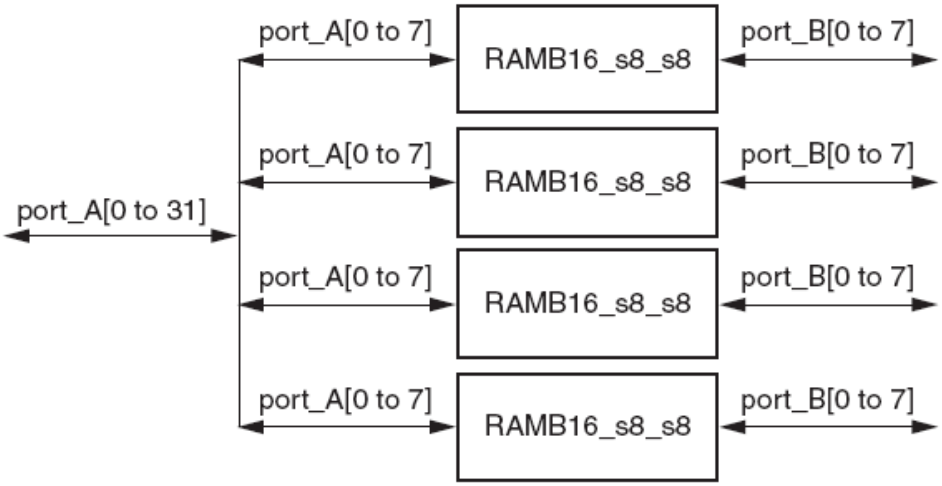


Abbildung 31: Beispiel einer Block RAM Implementierung mit vier 8-Byte RAM Blöcken

Abbildung 31 zeigt, wie der Zugriff auf DPRAM Blöcke funktioniert. Es sind vier 1-Byte große DPRAM Blöcke abgebildet, an deren Port A eine 32-Bit-Datenleitung angeschlossen ist. Aus Sicht auf Port A sind diese vier 1-Byte Speicher ein 4-Byte Speicher, wobei die letzten beiden Adressierungbits festlegen, auf welchen Speicherblock zugegriffen wird. Die B Ports sind alle einzeln verbunden, d.h. jeder Speicher wird unabhängig von den anderen drei Speichern adressiert, geschrieben und gelesen. Eine umfassende Dokumentation über den XPS DPRAM

Block ist unter [16] zu finden.

In dieser Arbeit wurde eine sehr ähnliche Anbindung an die DPRAM Blöcke realisiert. Die MicroBlaze besitzt eine 32-Bit- und das SERVOLINK 4-Modul eine 8-Bit-Schnittstelle. Das Modul bereitet die Daten auf und versendet sie seriell mit einem 12 MHz Takt. Mit jedem Takt wird ein Bit versendet. Auf der Empfangsseite wird das serielle Signal mit 96 MHz abgetastet, um bei 8-Bit-Daten wieder einen 12 MHz Takt zu erhalten, der synchron zu den empfangenen Daten ist. Der gewonnene 12 MHz Takt wird als Speichertakt für den Istwertspeicherblock genutzt, während der Sollwertspeicherblock den 12 MHz Takt aus dem DCM nutzt. Insgesamt werden 16 Byte pro Antriebsregler versendet und empfangen. Damit die Daten von der richtigen Adresse gelesen werden, müssen die Daten aus dem Sollwertspeicher mit einem Multiplexer an das SERVOLINK 4-Modul weitergegeben werden. Ähnlich müssen die Daten aus dem SERVOLINK 4-Modul mit einem Demultiplexer und einem „Byte-Write-Enable“-Signal an die richtige Adresse im Istwertspeicherblock geschrieben werden. Dieser Ablauf ist in den Abbildungen 32 und 33 auf Seite 63

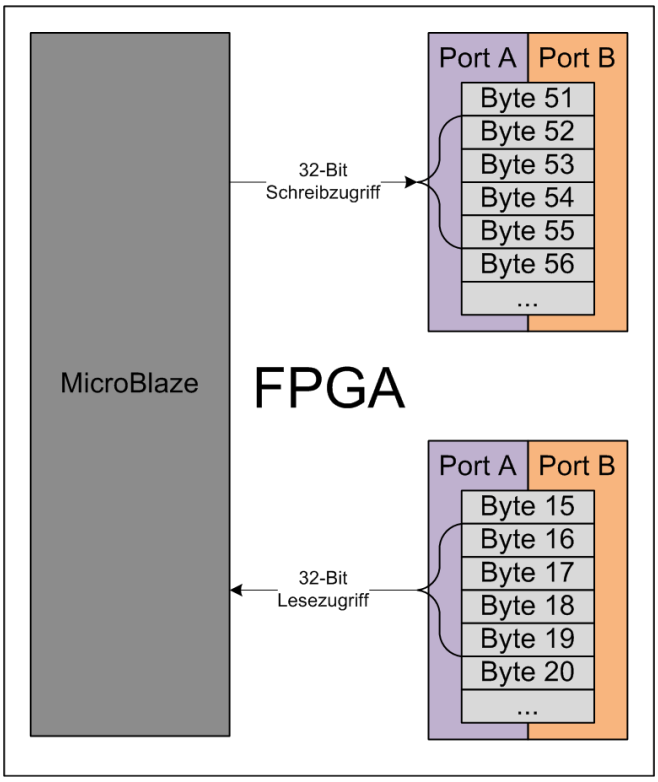


Abbildung 32: Prinzipzeichnung des 32-Bit-Lese-/Schreibzugriffs der MicroBlaze und die Anpassung an SERVOLINK 4 Teil 1

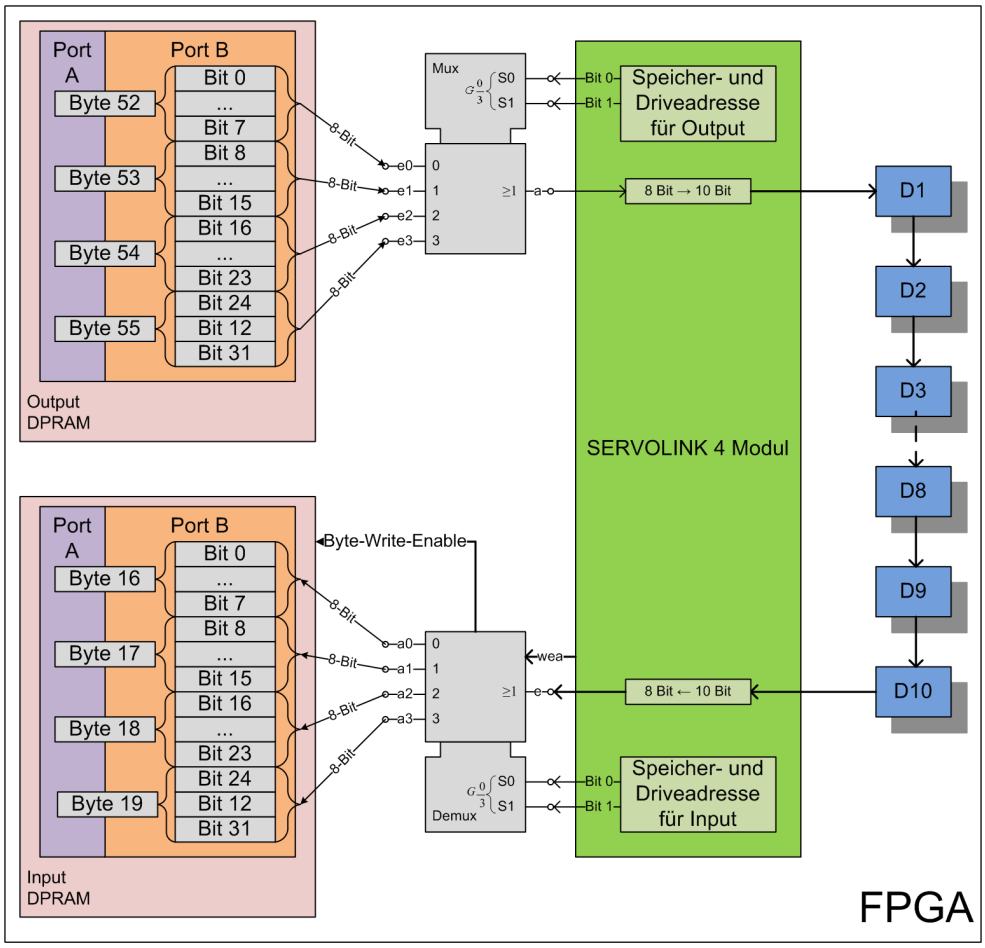


Abbildung 33: Prinzipzeichnung des 32-Bit-Lese-/Schreibzugriffs der MicroBlaze und die Anpassung an SERVOLINK 4 Teil 2

Im Anhang A in Tabelle 13 sind die Ein- und Ausgangssignale beschrieben, die in Abbildung 31 zu sehen sind. Es ist zu beachten, dass die In-/Out-Signale gegenteilig zu ihrer realen Richtung benannt sind. Die Signale sind der Betrachtung nach vom BRAM Controller aus benannt worden. „Daten In“ für den Controller ist auch „Daten In“ für den BRAM Block. Auf Seite 112 ist in Tabelle 14 abgebildet, welche Parameter die Block RAM Komponente besitzt und welche Werte diese haben können. Die Parameter werden automatisch an die Parameter des zum Block RAM zugehörigen Interface Controllers angepasst. Die vom System eingestellten Parameterwerte sind in Tabelle 4 dargestellt. Der Parameter C_MEMSIZE wird durch den dem Block RAM vergebenen Adressraum definiert.

Tabelle 4: Verwendete Parameter für das Block RAM

Parameter Name	Wert
C_PORT_AWIDTH	32 (Bit)
C_PORT_DWIDTH	32 (Bit)
C_NUM_WE	4
C_FAMILY	spartan3
C_MEMSIZE	8 KByte

5.1.5 Die Servolink-Register

Die Servolink-Register sind für den Austausch von Daten zwischen Software und Hardware notwendig. Diese Register werden über die Funktion „Create or Import Peripheral“ in der XPS-Umgebung erstellt. Sie sind eine eigene Konstruktion innerhalb des Gateway und gehören nicht zu dem SERVOLINK 4-Modul. Es wird dem Anwender mittels eines Assistenten die Möglichkeit gegeben, den Registern eine gewünschte Grundkonfiguration zu geben. Die Konfiguration wird dann in einem neuen Verzeichnis mit dem Namen der Komponente gespeichert. In diesem Verzeichnis befinden sich die Dateien *user_logic.vhd* und *servolink_register.vhd*, welche vom Anwender modifiziert werden müssen, um eine Anbindung an die übrige Hardware zu ermöglichen. Diese Modifikation beinhaltet die Festlegung, ob es sich bei dem einzelnen Register um ein Schreib- und/oder Leseregister handelt und wie der Name des Registers lautet. In Tabelle 15 im Anhang A sind die Ports der Servolink-Register aufgelistet. Diese Ports sind alle auch externe Ports. Eine Parameterliste zum Einstellen der Register gibt es nicht. Die Register werden automatisch an den Bus, an den sie angeschlossen sind, angepasst. Wie ein benutzerdefinierter IP-Core erzeugt und eingebunden wird ist unter [8, Seite 18 bis 24] nachzulesen.

5.1.6 Das Debug Modul

Das MicroBlaze Debug-Modul (MDM) wurde für JTAG basiertes Debuggen von einem oder mehreren MicroBlaze Prozessoren entworfen. Das MDM beinhaltet einen UART mit konfigurierbarem Slave Bus Interface für einen OPB oder PLBv46. Die UART TX- und RX-Signale werden über den JTAG-Port des FPGA vom und zum *Xilinx Microprocessor Debug Tool* übertragen. In Abbildung 34 ist das MDM dargestellt. Das Debug-Modul besitzt folgende Eigenschaften:

- Unterstützung von JTAG-basierenden Software Debug Tools

- Debuggen von bis zu acht MicroBlaze Prozessoren
- Synchrone Kontrolle von mehreren MicroBlaze Prozessoren
- Unterstützung von UARTs, die auf JTAG basieren mit einem konfigurierbaren OPD- oder PLBv46-Interface
- Chipscope ICON Kern Verbindung über BSCAN Signale
- einen Master Fast Simplex Link (FSL) für schnellen Download

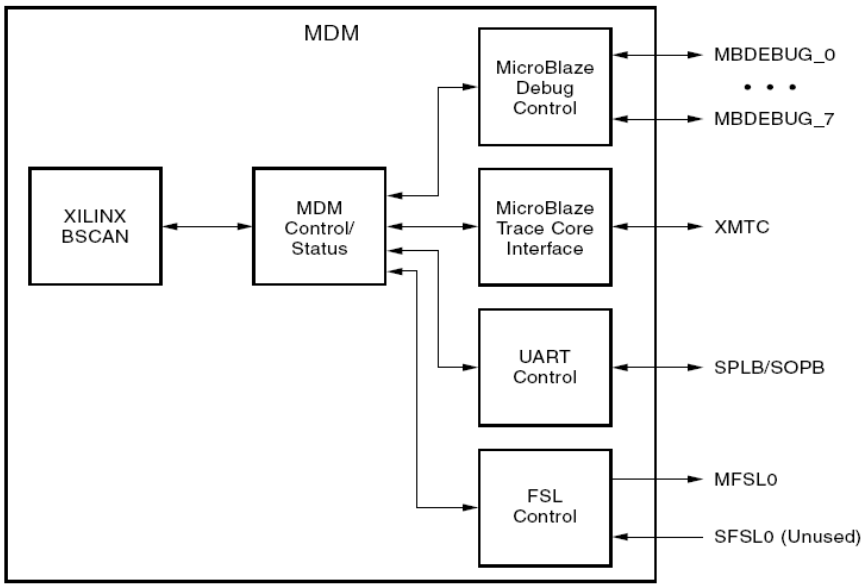


Abbildung 34: Blockdiagramm eines Mikroprozessor Debug Moduls [24, Seite 2]

In Tabelle 16 sind einige wichtige Signale aufgelistet. Diese Signale können vom Anwender genutzt werden. Alle anderen Signale werden automatisch durch die Entwicklungstools verbunden. Die vollständige Liste aller Signale und Parameter ist in [24] zu finden.

5.1.7 Schnittstellen der VHDL-Komponente MicroBlaze - External Ports

Um ein in XPS erstelltes System mit Komponenten außerhalb dieses Systems zu verbinden, werden externe Ports benötigt. Die systeminternen Ports der IP-Cores sind durch Signale mit diesen externen Ports verbunden. Um z.B. den Speicher des Hilscher COM-Moduls ansprechen zu können, müssen die benötigten Signale wie z.B. *Chip Select, Read, Write etc.* mit den

äquivalenten Signalen des externen Speicher-Interfaces der MicroBlaze verknüpft werden. Die externen Ports werden im Toplevel mit den entsprechenden Ports der Funktionseinheit verbunden, welche die Anbindung an die Pins des Hilscher COM-Moduls darstellen. Diese externen Ports können auch mit Signalen anderer Komponenten verbunden werden. Ein solches "´Signalrouting“ ist im Anhang B unter B.2 zu finden.

Um aus Ports eines IP-Cors externe Ports eines Mikroprozessorsystems zu machen, kann beim "´Signalrouting“ „Make External“ ausgewählt werden, ebenso kann dieses "´Signalrouting“ auch manuell in der MHS-Datei oder in der XPS-Oberfläche über „New Connection“ editiert werden. Die Tabellen 7 und 8 im Anhang A auf den Seiten 105 und 106 zeigen alle externen Ports der MicroBlaze.

5.2 Das Gateway Projekt in VHDL

Das Gateway wird als VHDL-Projekt in Xilinx ISE aufgebaut. Dazu gehören das Toplevel, die im Systemaufbau höchgelegene Instanz, und die in diesem eingebundenen Komponenten. Die Komponenten sind ein Clock Generator, das SERVOLINK 4-Modul und das Mikroprozessorsystem mit der MicroBlaze. Es muss zusätzlich eine Aufbereitung einiger Signale in Prozessen stattfinden, damit die Funktionalität des Gateways gegeben ist.

Zuerst wird die Funktionseinheit des Gateways aufgebaut, mit der die Anbindung an die Signale außerhalb des FPGAs realisiert wird, siehe Abbildung 35. Dabei wird in der Entity der Toplevel VHD-Datei lediglich der Name des Ports festgelegt. Damit der Compiler die Ports an die richtigen Pins verlinkt, muss der Anwender in der UCF-Datei diese Verbindung festlegen. Nun werden die anderen Komponenten als Instanzen eingebunden. Diese werden dann untereinander durch Signale verbunden. Wo das vollständige "´Signalrouting“ zwischen den einzelnen Komponenten und der FPGA-Pins stattfindet, ist im Anhang B unter B.3 dargestellt.

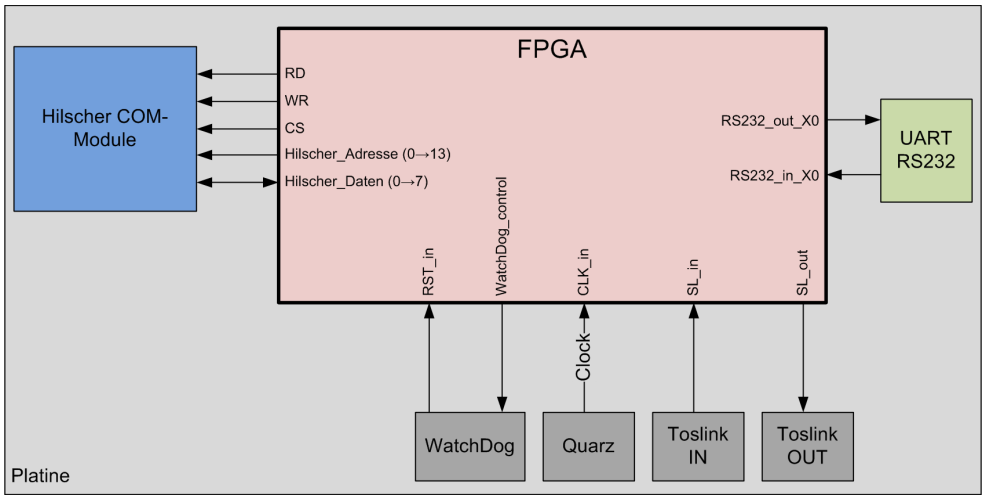


Abbildung 35: Prinzipieller Aufbau der Toplevel Entity

5.2.1 Die Komponente MicroBlaze

In Abbildung 36 ist dargestellt, wie die Komponenten-Ports der MicroBlaze heißen und ob sie Ein- oder Ausgänge sind. All diese Ports sind externe Ports und stimmen mit den External Ports unter 5.1.7 überein. Die Ports sind auch in Tabelle 7 und 8 auf Seite 105 und 106

abgebildet. Alle diese Ports sind als externe Ports in der XPS-Oberfläche für die MicroBlaze festgelegt und sind innerhalb der Komponente MicroBlaze über Signale an Komponenten gekoppelt. Im VDHL-Toplevel wird so z.B. der Interruptport der MicroBlaze mit dem des SERVOLINK 4-Moduls verbunden. Die vollständige Dokumenation der MicroBlaze ist unter [18] zu finden.

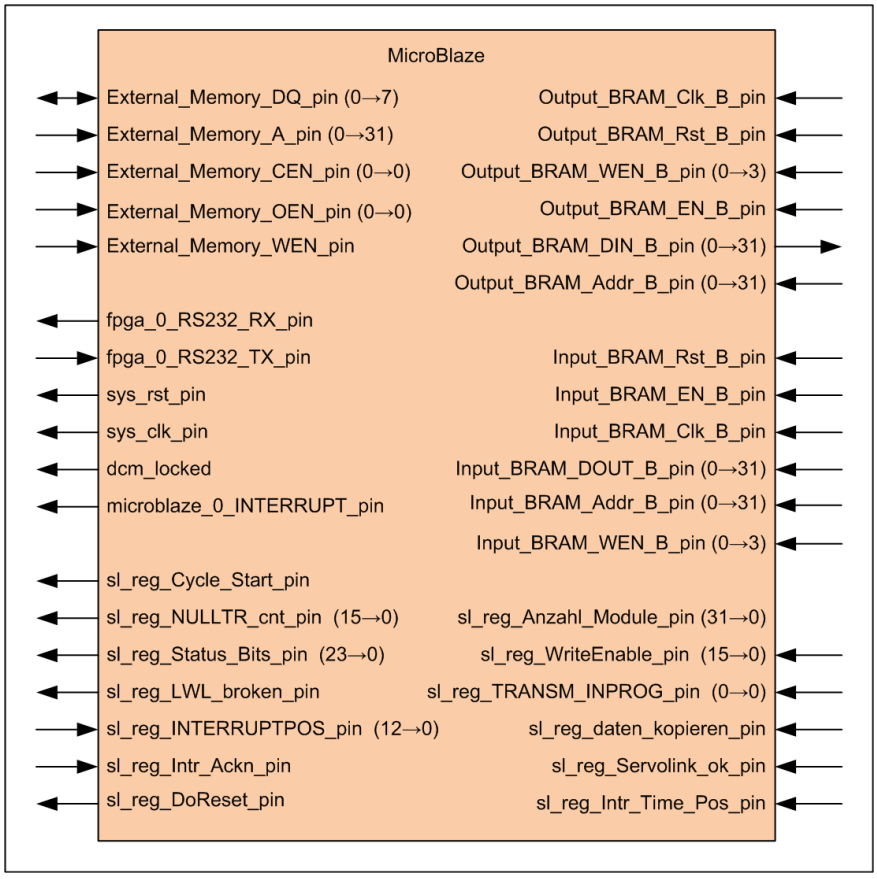


Abbildung 36: Die Komponente MicroBlaze

5.2.2 Die Komponente SERVOLINK 4-Modul und deren Schnittstellen

In Tabelle 21 im Anhang A ist dargestellt wie die Komponenten-Ports des SERVOLINK 4-Moduls heißen und ob sie Ein- oder Ausgänge sind. Die Ports des SERVOLINK 4-Moduls werden, wie die der anderen Komponenten, ebenfalls über Signale im Design verbunden.

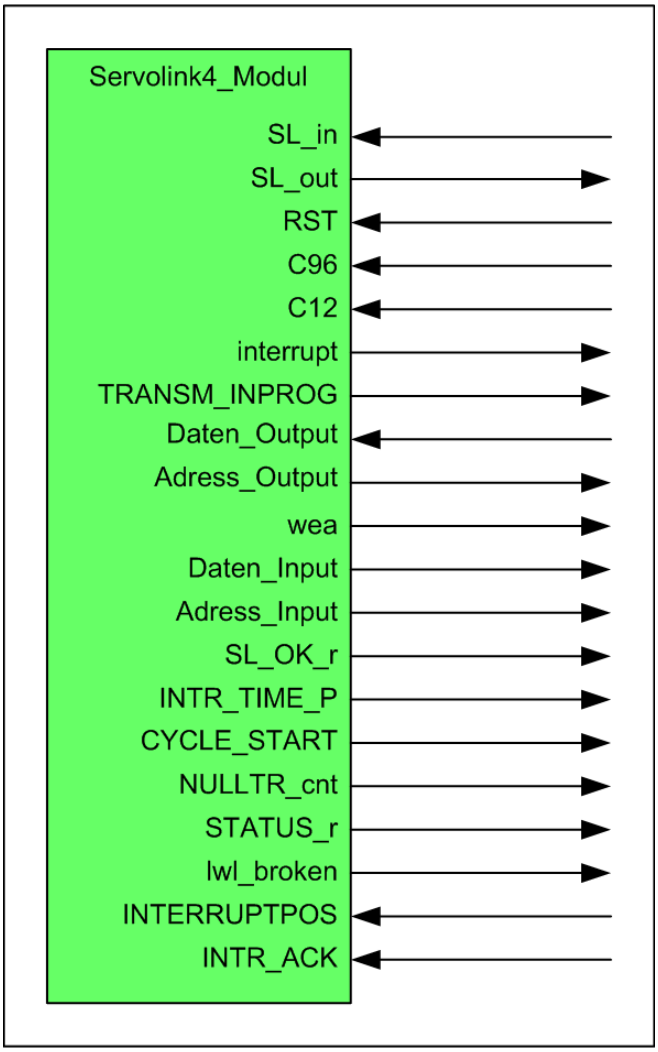


Abbildung 37: Die Komponente SERVOLINK 4-Modul

5.2.3 Die Komponente Clock Generator und ihre Schnittstellen

In Abbildung 38 ist dargestellt wie die Ports der Clock Generator-Komponente heißen und ob sie Ein- oder Ausgänge sind. Tabelle 18 im Anhang A ergänzt diese Abbildung. Der Clock Generator erhält seinen Eingangstakt direkt von dem 48 MHz Quarz außerhalb des FPGA und erzeugt daraus einen 12 MHz-, 48 MHz-, 96 MHz- und einen einstellbaren Systemtakt. Diese Takte werden für Prozesse oder zur Taktung von Speicherblöcken verwendet. In Tabelle 19 auf Seite 117 sind die wichtigsten Parameter zur Konfiguration der Komponente aufgelistet. Tabelle 5 zeigt die verwendeten Parameterwerte. Alle Informationen zum Clock Generator sind unter [23] zu finden.

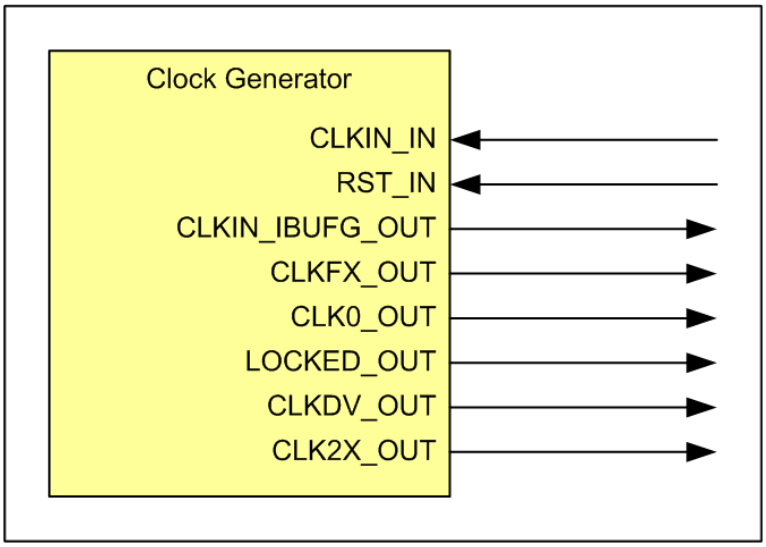


Abbildung 38: Die Komponente Clock Generator

Tabelle 5: Verwendete Parameter des Clock Generator

Parameter Name	Wert	Beschreibung
C_CLKDV_DIVIDE	4.0	Wert durch den der Eingangstakt für CLKDV_OUT geteilt wird
C_CLKIN_PERIOD	20.833	Periodendauer des Eingangstaktes in Pikosekunden
C_CLKFX_DIVIDE	2	Wert durch den der Eingangstakt für CLKFX_OUT geteilt wird
C_CLKFX_MULTIPLY	3	Wert mit dem der Eingangstakt für CLKFX_OUT multipliziert wird
C_CLKOUT_PHASE_SHIFT	TRUE	Gibt an, in welcher Form eine Phasenverschiebung gemacht werden soll
C_CLK_FEEDBACK	2x	Gibt an, welcher Takt als Feedback für die Synchronisation benutzt wird
C_DUTY_CYCLE_	TRUE	Gibt an, ob eine Korrektur des Tastverhältnisses stattfinden soll
C_PHASE_SHIFT	0	Wert für die Phasenverschiebung
C_STARTUP_WAIT	FALSE	Gibt an, ob es eine Start-verzögerung gibt

5.2.4 Die UCF Datei

In der UCF-Datei wird die Verbindung zwischen den zur Entity gehörenden Ports und den Pins des FPGAs hergestellt. Diese Verbindung kann direkt in der UCF-Datei editiert werden. Auch mit Hilfe des Floorplan IO Pre-Synthesis Tool können diese Verbindungen hergestellt werden. Dabei kann der Anwender den Port per „Drag & Drop“ an den gewünschten Pin ziehen. Der Floorplan ist in Abbildung 39 dargestellt.

Abbildung 39: Floorplan IO Pre-Synthesis

5.3 Hilscher COM-Modul: Schnittstelle und Anbindung

Das Hilscher COM-Modul [7] und [5] wird mittels einer 50-poligen Steckverbindung mit der Gateway-Platine verbunden. Zum Betrieb des COM-Moduls werden nicht alle Pins benötigt, deshalb werden in Tabelle 22 nur die verwendeten Pins aufgelistet. Damit auf den Speicher des Moduls zugegriffen werden kann, muss das Timing des External Memory Controllers auf das Speicher-Timing des COM-Moduls abgeglichen werden. Die Timing-Diagramme für die Lese- und Schreibzugriffe des Speichers sind in den Abbildungen 40 und 41 zu sehen. Die Abbildungen werden durch Tabelle 6 ergänzt.

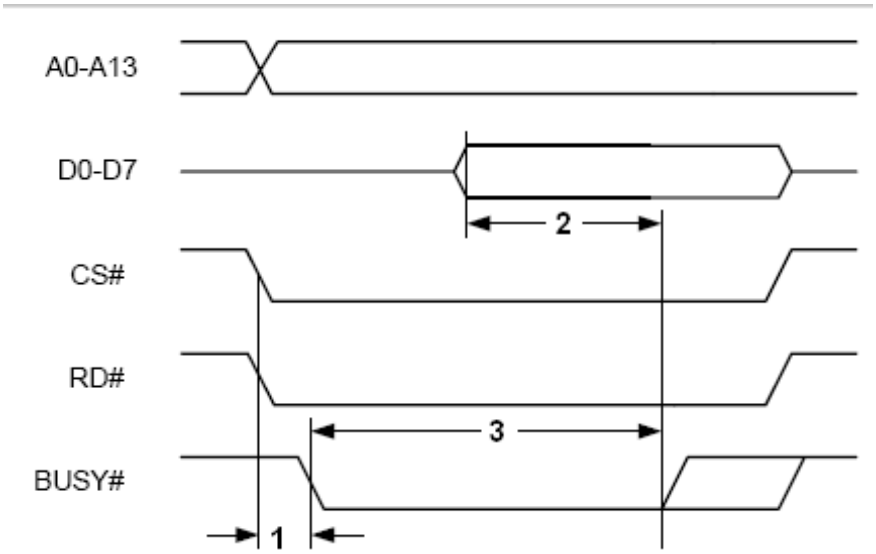


Abbildung 40: Timing-Diagramm des Lesezyklus des COM-Speichers [7, Seite 45]

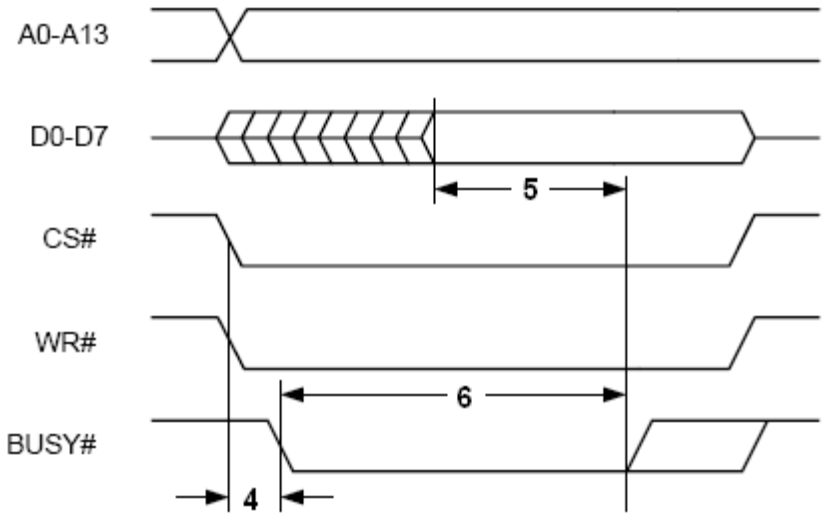


Abbildung 41: Timing-Diagramm des Schreibzyklus des COM-Speichers [7, Seite 45]

Tabelle 6: Ergänzung zu den Timing-Diagrammen

Nummer	Beschreibung	Min.	Max.	Units
1	Zeit, in der CS und RD low sind bis BUSY low wird	6		ns
2	Zeit, in der die Daten gelesen werden können bis BUSY high wird	12		ns
3	Zeit, in der BUSY low ist	0	3-7	CLK Zyklen
4	Zeit, ind der CS und WR low sind bis BUSY low wird	6		ns
5	Zeit, in der die zu schreibenden Daten anliegen müssen bis BUSY high wird	26		ns
6	Zeit, in der BUSY low ist	0	3-7	CLK Zyklen

5.4 Konfiguration der SPS

Für den Betrieb eines System mit einer SPS als Steuerungsrechner muss diese konfiguriert werden. Das Gateway wird als ein PROFIBUS-Gerät konfiguriert. An diesem PROFIBUS-Gerät sind die SERVOLINK 4-Geräte angeschlossen. Jedes SERVOLINK 4-Gerät ist in der SPS logisch als Modul mit zwei Modulteilen mit jeweils 16 Byte eingeteilt. Ein Modulteil stellt die Eingangsdaten und der andere Modulteil die Ausgangsdaten dar. Jeder Modulteil besitzt in der Konfiguration, die über PROFIBUS an das Gateway gesendet wird, eine eigene Kennung, an der erkannt wird, in welcher Reihenfolge und welche Art von Geräten, in diesem Fall Antriebsreglern, in der SPS eingerichtet wurden. Da es Einzel- und Doppelachsgeräte im SERVOLINK 4-Ring geben kann, muss zwischen diesen beiden Typen bei der Konfiguration für einen korrekten Kopiervorgang der Prozessdaten unterschieden werden.

Anhand von Abbildung 42 wird nun erklärt wie die Prozessdaten im Gateway aufbereitet werden. Bei der Gerätekonfiguration für den PROFIBUS können die Geräte nur direkt hintereinander gesetzt werden. Die Geräte selber besitzen 32 Byte große Speicher für die Daten jeder anzutreibenden Achse. Der Einzelachsantriebsregler benötigt in den Speicherblöcken für die Soll- und Istwerte nur jeweils 16 Byte effektive Nutzdaten. Es liegen hinter diesen 2x16 Byte Nutzdaten weitere 32 Byte (Nulldaten) die nicht genutzt werden (Soll|Ist|Null|Null). Eine Doppelachse legt die 16 Byte für die Soll- und Istwerte beider Achsen direkt hintereinander

ab (Soll|Ist|Soll|Ist). Werden nicht nur Doppelachsen, sondern auch Einzelachsen in einem SERVOLINK 4-Ring verwendet, entstehen „Lücken“ von 32 Byte, welche bei der Übertragung im SERVOLINK 4 berücksichtigt werden müssen. Damit die Daten an die richtigen Antriebe gesendet bzw. an die richtigen (Speicher-)Adressen kopiert werden, muss die Gerätereihenfolge bekannt sein.

Diese Information wird von der SPS an das COM-Modul gesendet und kann dort aus der Eingangsmailbox (HostMailbox) ausgelesen werden. Die beiden Module besitzen unterschiedliche Kennungen, ein Einzelachsmodul hat eine 2-Byte (0x9F|0xAF) und ein Doppelachsmodul eine 4-Byte (0x9Ff|0x9F|0xAF|0xAF) Kennung. Auf Grund dieser Kennungen wird ein Offset-Array berechnet, mit dessen Hilfe die Sollwerte an die richtigen Speicheradressen kopiert werden. Auch die Daten aus dem Istwert Speicherblock werden mit Hilfe des Offset-Array in die richtige Reihenfolge für die SPS gebracht.

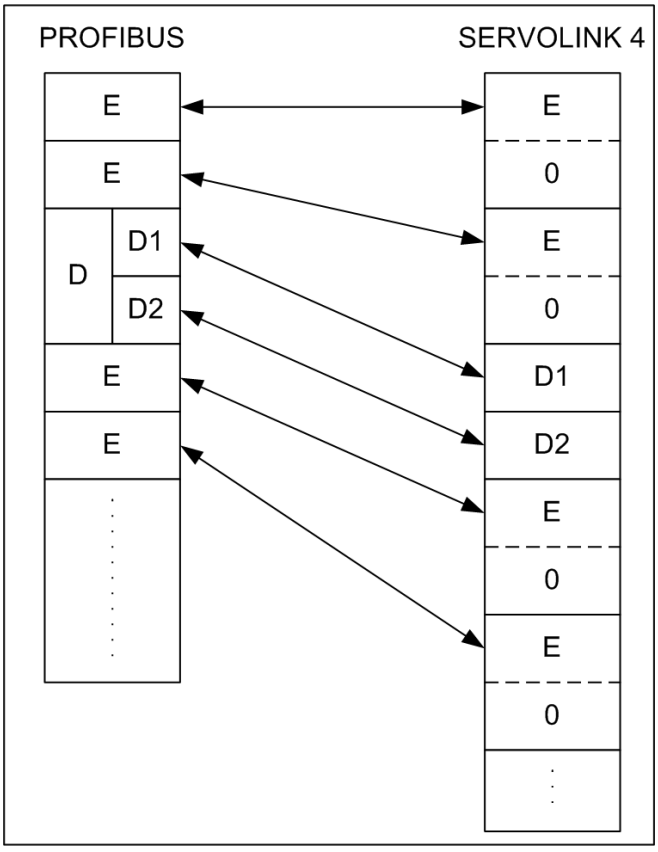


Abbildung 42: Reihenfolge der Daten in den beiden Bussystemen

5.5 Die Programmierung der MicroBlaze

5.5.1 Gesamtfunktionsablauf

Im Folgenden wird der prinzipielle Ablauf der Applikation des Gateways beschrieben.

Initialisierung und Start:

Zuerst wird eine minimale Initialiserung vorgenommen, bei der bekanntgegeben wird, an welchen Speicheradressen sich die Datenstrukturen der einzelnen Komponenten befinden. Als nächstes wird der Interrupt freigegeben und es folgt eine Endlosschleife, die von der Interrupt Service Routine (ISR) unterbrochen wird.

ISR:

In der ISR wird zuerst das Ergebnis der SERVOLINK 4 CRC-Prüfung ausgelesen, anschließend wird die Funktion *servolinkGatewayCycle* aufgerufen, in der die eigentliche ISR-Funktion implementiert ist. Dies ist in Abbildung 43 dargestellt.

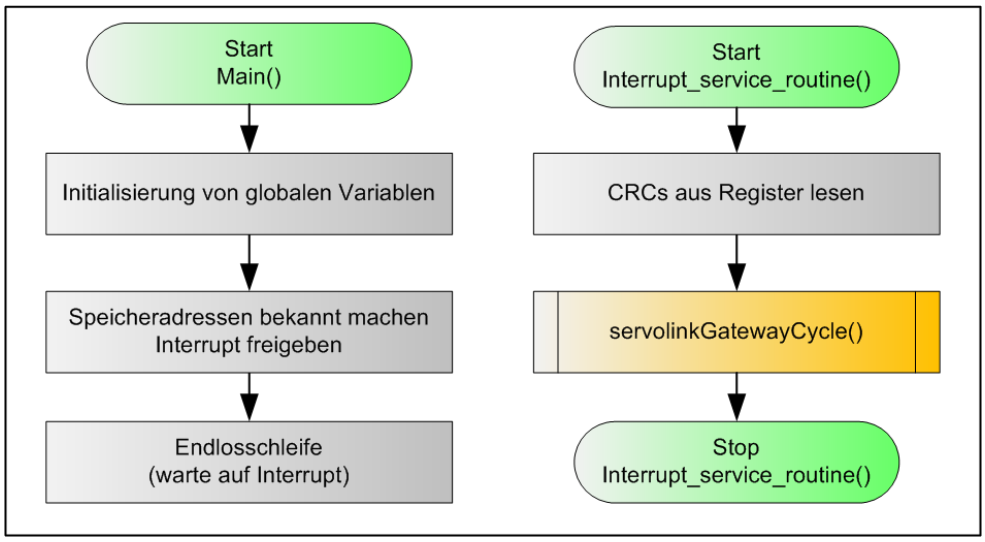


Abbildung 43: Oberste Ebene des Programmablaufplans Teil 2

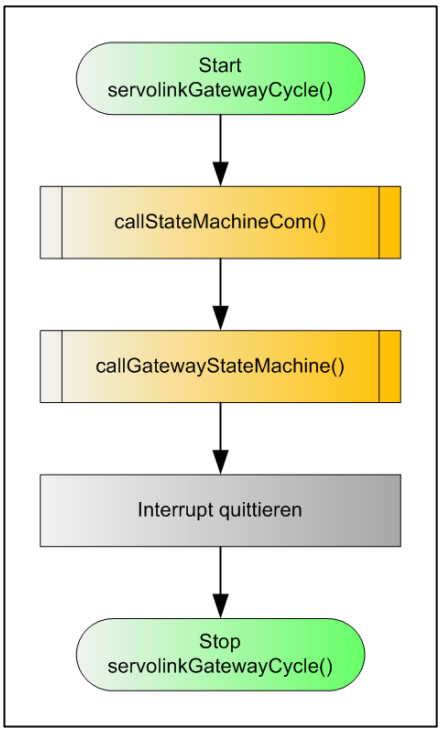


Abbildung 44: Oberste Ebene des Programmablaufplans Teil 1

Abarbeitung der Zustandsmaschinen:

In der Funktion *servolinkGatewayCycle* werden zwei Funktionen aufgerufen. Die erste ist die Funktion *callStateMachineCom*, die zweite ist *callGatewayStateMachine*.

callStateMachineCom:

In dieser Funktion wird der Zustand ermittelt, in dem sich das Hilscher COM-Modul momentan befindet. Zudem wird eine erste Konfiguration des COM-Moduls vorgenommen. Es wird festgelegt mit welcher Handshake-Methode das Modul arbeiten soll (Buffered, User controlled) und wie lang die Zeit für den Modul-Watchdog ist. Besonders wichtig ist, dass dem Modul mitgeteilt wird, dass es die Konfiguration annehmen soll, die die SPS über PROFIBUS sendet. So wird automatisch eine Verbindung zwischen dem Gateway und PROFIBUS aufgebaut. Die Zustandsmaschine des COM-Moduls kann sechs Zustände annehmen. Diese Zustände sind: START, SETUP, READY, RUN, RESET und FAULT, siehe Abbildung 45 auf Seite 79. Zusätzlich zu diesen Zuständen gibt es noch die Unterzustände Null bis Sieben (0-7), mit deren Hilfe die Parametrierung des COM-Moduls im Zustand SETUP durchgeführt wird. Die Parametrierung des COM-Moduls wird in mehreren Schritten über etliche Systemtakte abgearbeitet.

- START: Der Unterzustand wird auf 0 (Startzustand) gesetzt und es wird die Schnittstelle eingerichtet. Bei Auftritt eines Fehlers wird der Zustand FAULT angenommen, ansonsten SETUP.
- SETUP: In diesem Zustand werden die sieben Unterzustände durchgegangen und damit die Hardware betriebsbereit gemacht.
  - 0: Auf Betriebsbereitschaft der Hardware warten
  - 1: Parametrierung durchführen
  - 2: Warmstart des Gerätes auslösen
  - 3: Auf Ausführung des Warmstarts warten
  - 4: Warten bis das Gerät fertig ist
  - 5: Warten bis das Gerät im „run“-Modus ist
  - 6: Warten bis der Profibus-Task bereit ist
  - 7: Anwendung am Gerät anmelden

  Ist die Hardware eingerichtet, wird auf den Verbindungsaufbau gewartet. Dazu wird der Zustand READY angenommen.
- READY: Es wird auf die Kommunikation über PROFIBUS gewartet und dann in den Zustand RUN gewechselt. Sollte ein Fehler auftreten, wird der Zustand FAULT angenommen.
- RUN: Der Austausch von Daten findet statt. Wenn ein Fehler auftritt, wird in den Zustand FAULT gewechselt. Wird die Verbindung unterbrochen, wird der Zustand READY angenommen.
- RESET: Reset des Moduls durchführen und dann in den Zustand START wechseln.
- FAULT: Im Fehlerzustand wird gewartet, dass ein Fehlerreset stattfindet.

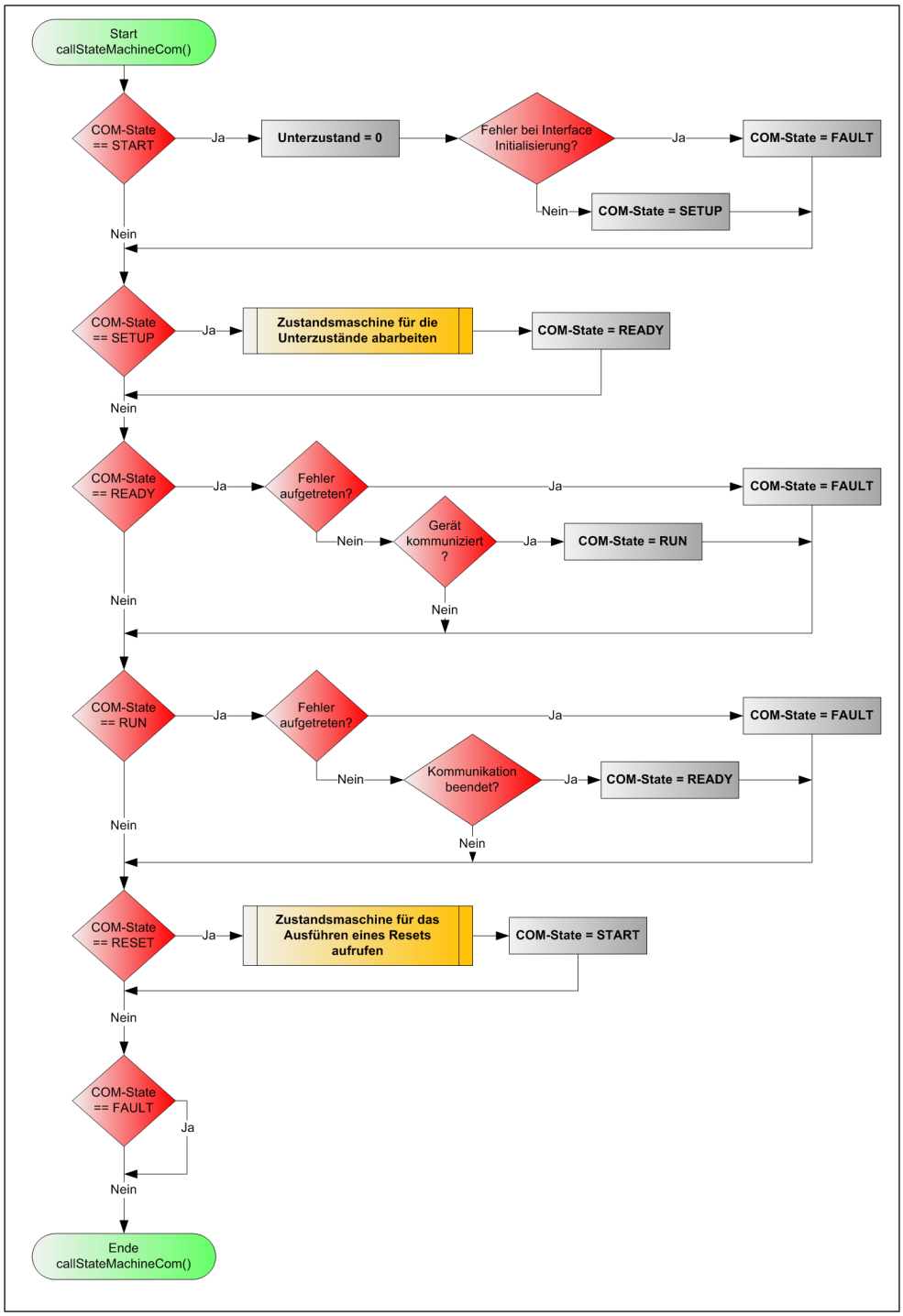


Abbildung 45: Programmablaufplan der Zustandsmaschine des COM-Moduls

callGatewayStateMachine:

In der Funktion *callGatewayStateMachine* wird auf Basis des Zustandes des COM-Modules der Zustand des gesamten Gateway ermittelt und es werden die Prozessdaten ausgetauscht. Auch in dieser Zustandsmaschine gibt es die Zustände START, SETUP, READY, RUN, RESET und FAULT. Bei jedem Aufruf dieser Funktion wird zuerst überprüft, ob sich das COM-Modul im Fehlerzustand befindet. Wenn dies zutrifft wird auch der Gateway-Zustand auf FAULT gesetzt. Siehe dazu Abbildung 13 auf Seite 36.

- START: Das Gerät wurde gestartet und das Gateway soll eingerichtet werden. Nun wird als erstes ein Reset ausgeführt, dazu werden die Zustände beider Zustandsmaschinen in den RESET-Zustand gesetzt.
- SETUP: Wenn das Gateway eingerichtet ist, soll die Hardware konfiguriert werden. Dazu wird gewartet, bis das COM-Modul den Zustand READY angenommen hat.
- READY: Der Speicherblock für die Sollwerte wird mit Nullen beschrieben, da in diesem Zustand des Gateways keine gültigen Daten für die Antriebsregler vorhanden sind. Hat das COM-Modul eine Verbindung über PROFIBUS aufgebaut, nimmt es den Zustand RUN an, und auch die Gateway Zustandsmaschine kann diesen Zustand annehmen.
- RUN: In diesem Zustand findet die zyklische Datenübertragung statt. Zuerst wird überprüft, ob neue Konfigurationsdaten von der SPS geschickt wurden. Neue Konfigurationsdaten werden ausgelesen und ausgewertet. Nun wird überprüft, ob die MicroBlaze Zugriff auf die Prozessdaten im COM-Modul hat, und ausschließt, dass der SERVOLINK 4-Ring unterbrochen ist. Wenn der Ring unterbrochen ist, werden die Sollwerte mit Nullen überschrieben, zudem werden Nullen an den Steuerungsrechner gesendet. Liegt keine Unterbrechung vor, werden die Prozessdaten aus dem Speicher des COM-Moduls in den Speicherblock für die Sollwerte und die Istwerte aus dem zugehörigen Speicherblock in den COM-Modul-Speicher kopiert. Sollte mehr als fünf mal direkt hintereinander ein CRC-Fehler eines Antriebs registriert werden, werden dessen Sollwerte im Speicherblock mit Nullen überschrieben. Für dieses Modul werden Einsen an den Steuerungsrechner gesendet, so kann dieser Fehler erkannt werden. Ist der Kopiervorgang abgeschlossen, wird das Handshakeflag für das COM-Modul gesetzt und der Interrupt wird quittiert. Sollte keine Kommunikation stattfinden, wird überprüft, ob das COM-Modul im FAULT Zustand ist. In diesem Fall wird das Gateway auch in diesen Zustand gesetzt. Ansonsten wird der Zustand READY angenommen.
- RESET: Es wird gewartet bis das COM-Modul einen Reset ausgeführt hat, dann wird der Zustand SETUP angenommen.
- FAULT: Im Fehlerzustand wird gewartet bis ein Fehlerreset stattgefunden hat.

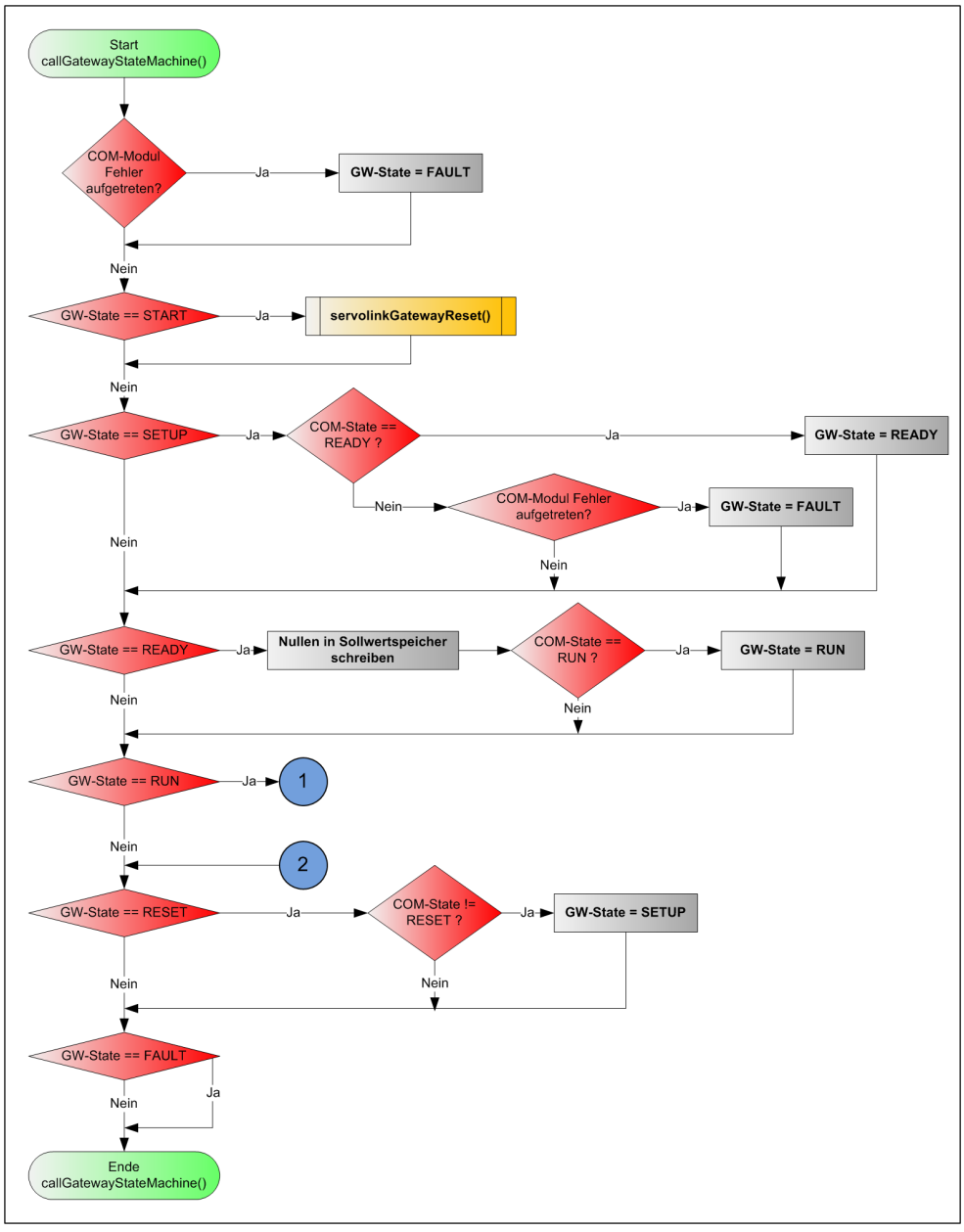


Abbildung 46: Programmablaufplan der Zustandsmaschine des Gateways

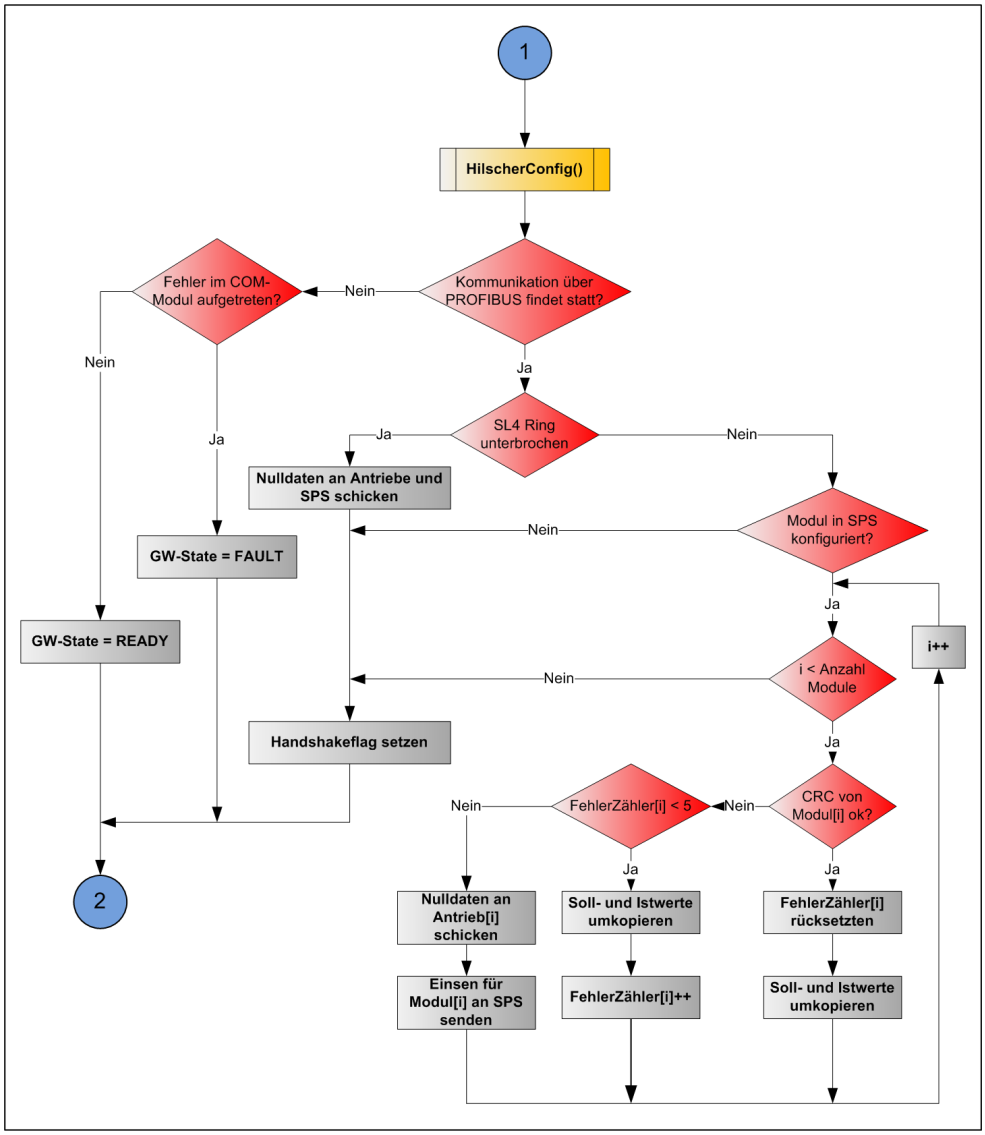


Abbildung 47: Der Zustand RUN der Gateway Zustandsmaschine

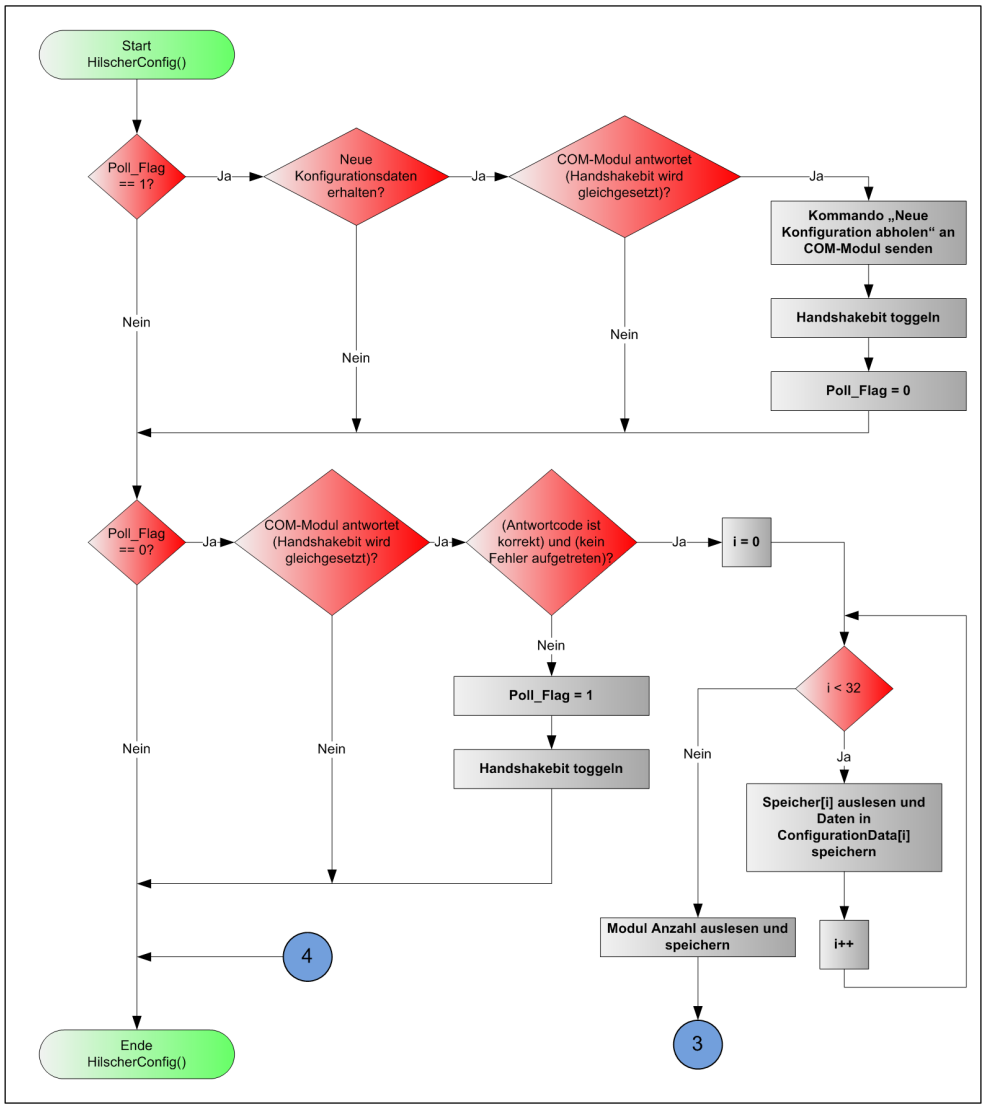


Abbildung 48: Die Funktion HilscherConfig() Teil 1

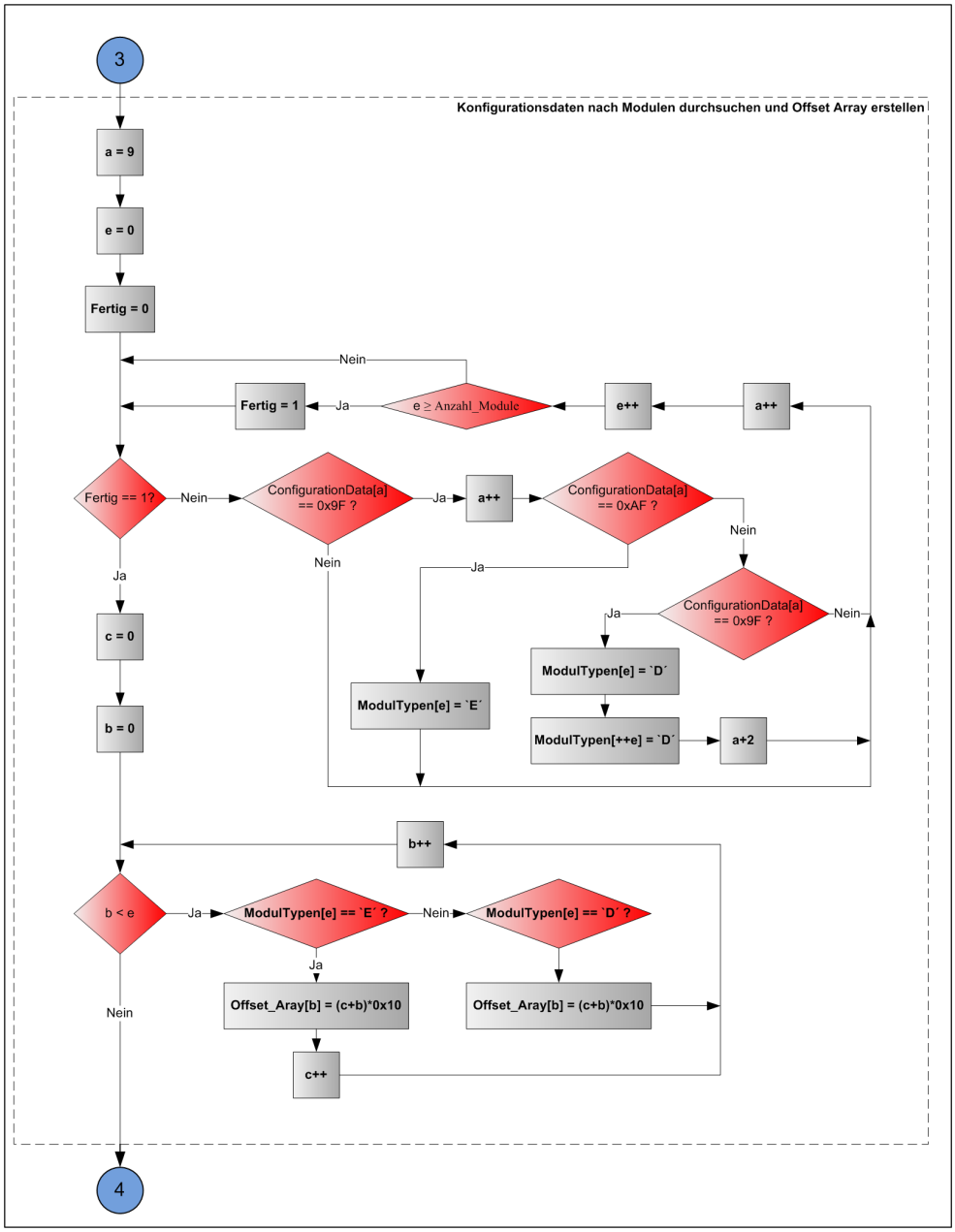


Abbildung 49: Die Funktion HilscherConfig() Teil 2

5.6 Hindernisse und Lösungen bei der Implementierung

Bei der Entwicklung des Gateways gab es viele, meist einfach zu lösende Probleme. Ein großes Problem zu Beginn der Entwicklung war, dass sich ein Update für die Entwicklungsumgebungen auf dem vorgesehenen Rechner zuerst nicht richtig installieren ließ. Wie sich später herausstellte, funktionierte der Compiler nicht korrekt. Der UART z.B. ließ sich zwar einbinden und übersetzen, funktionierte aber nicht. An anderer Stelle funktionierte die Schnittstelle zum Hilscher-Speicher in ihren Grundfunktionen, aber es konnte keine Veränderung des Timing der WRITE, READ und sonstigen Befehle vorgenommen werden. Dieses Problem wurde einfach dadurch behoben, dass die Entwicklung des Gateways auf einem anderen Rechner fortgesetzt wurde, auf dem die Entwicklungssoftware einwandfrei funktionierte.

Ein Hindernis bei der Entwicklung war, dass die MicroBlaze nur 32-Bit-Schreib- und Lesezugriffe macht. Da das SERVOLINK 4-Modul aber mit 8-Bit Datenblöcken arbeitet, musste in der Hardware erst ein Daten Demultiplexer für die Ausgangsseite und ein Multiplexer für die Eingangsdaten geschrieben werden. Die Lösung ist in Abschnitt 5.1.4 auf Seite 60 erklärt. Ansonsten traten nur wenige und triviale Probleme beim Entwickeln der Hardware bzw. Software auf, z.B. das Verdrehen von Datenvektoren auf Grund der Handhabung von Big- und Little-Endian-Strukturen in der MicroBlaze.

6 Einbindung und Test des Gateways

6.1 Überblick über Simulations-, Test- und Debugmöglichkeiten des Systems

Ein eingebettetes System wie dieses ist schwer in seiner Performance bezüglich der Signale, die im FPGA intern verlaufen, zu testen. Daher musste die entwickelte Hardware von Beginn der Entwicklung an kontinuierlich getestet und simuliert werden, um Fehler schnellst möglich aus dem System zu entfernen. Die Möglichkeiten dieser Fehlersuche sind im Folgenden erläutert.

6.2 Simulation der Hardware

Um die in VHDL geschriebene Hardware zu testen, wird der zu testende VHDL-Code mittels ModelSIM simuliert. Dazu wird eine Testbench erzeugt, in welcher der Anwender das zeitliche Verhalten von Signalen und Eingängen vorgeben kann. Es wurden Takte erzeugt, mit denen die Frequenzen des Clock Generators und damit das Timing im Gateway simuliert wurden. In dieser Arbeit wurde vor allem getestet, wie sich das SERVOLINK 4-Modul verhält, um es in das System optimal einbinden zu können. So wurde ein Datenstrom auf dem Lichtwellenleiter simuliert und überprüft, ob dieser richtig dekodiert wird und ob die resultierenden Write- und Interrupt-Befehle für den Ablauf des Programmcodes zu den richtigen Zeitpunkten erzeugt werden. Zudem wurde die Simulation benutzt, um das SERVOLINK 4-Modul zu analysieren, da dieses Modul zwar schon existierte, für den Einsatz in diesem Projekt jedoch angepasst werden musste.

6.3 Test der Hardware

Die Hardware konnte in der MicroBlaze nur bedingt getestet werden. Es wurde vor allem über die XMD-Konsole der Schreib- und Lesezugriff auf Speicher und Register getestet. So wurden z.B. im Hardwaredesign konstante Werte an einige Adressen geschrieben und mit

der XMD-Konsole überprüft, ob diese Werte an den bekannten Adressen ausgelesen werden konnten. Dadurch wurde getestet, ob Fehler bei der Anbindung von Big-Endian- an Little-Endian-Vektoren gemacht wurden. Der Unterschied ist in Abbildung 50 zu erkennen.

Eine weitere Möglichkeit des Testens ist der Einbau eines Chipscopes. Ein Chipscope ist ein VHDL-Baustein, der über die ISE in ein System eingefügt werden kann. Dort kann der Anwender sich die Signale im FPGA auf einer Oszilloskopoberfläche anzeigen lassen. Es wurde kurzzeitig ein Chipscope eingebaut, um zu prüfen, ob es für weitere Tests nützlich ist. Der Aufwand ein Chipscope einzubauen und sinnvoll einzustellen war zu groß. Es wurde deswegen nicht weiter genutzt.

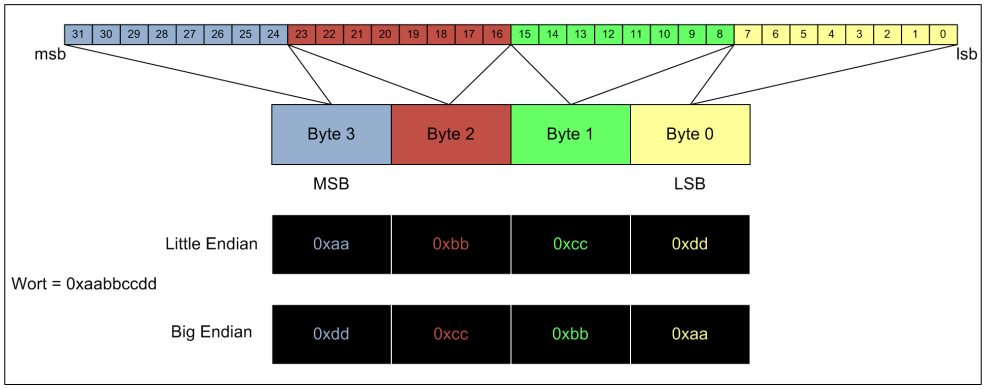


Abbildung 50: Unterscheidung von Big und Little Endian

6.4 Debug

Neben dem Quellcode für die Hardware musste auch noch der Programmcode der Software getestet werden. Dazu gibt es zwei Möglichkeiten:

- Debuggen mittels XMD-Konsole
- Debugfunktion der SDK-Umgebung

6.4.1 Debug mittels XMD-Konsole

Die erste Art des Testens des Programmcodes ist das Debuggen mittels XMD-Konsole, siehe Abbildung 51. Die Konsole wird unter XPS gestartet und greift während des laufenden

Betriebs der MicroBlaze auf die Speicherinhalte des Mikroprozessorsystems zu. So können die Speicherblöcke für den Datenaustausch mit dem SERVOLINK 4-Modul ausgelesen und beschrieben werden. Ebenso kann auf den Hilscher DPRAM und alle anderen adressierten Speicherblöcke der MicroBlaze zugegriffen werden.

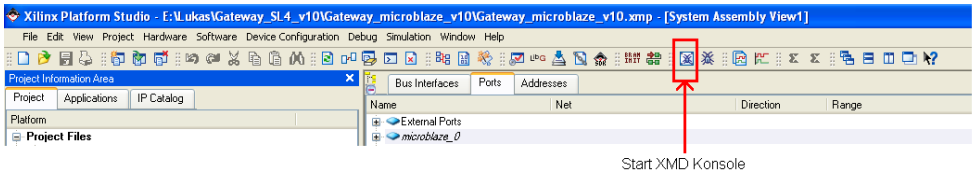


Abbildung 51: Starten der XMD-Konsole unter XPS

In Abbildung 52 ist ein Lesezugriff auf die Servolink-Register abgebildet. Es kann überprüft werden, ob ein Fehler im SERVOLINK 4-Bus vorliegt oder wie viele Antriebsregler das Gateway gefunden hat. Mit dem Befehl „help“ bekommt der Anwender eine Auswahl an verschiedenen Befehlen, die in weitere Unterkategorien geordnet sind. Mit diesen Befehlen kann der Anwender manuell die Befehle geben, die im nächsten Abschnitt (6.4.2) automatisch vom Debugger ausgeführt werden, wie Schreib- und Lesebefehle oder das Setzen von Breakpoints. Es wurden vorwiegend die Speicher über die Konsole ausgelesen und mit den erwarteten Werten verglichen, um auf diese Weise Fehler im Programmcode zu finden.

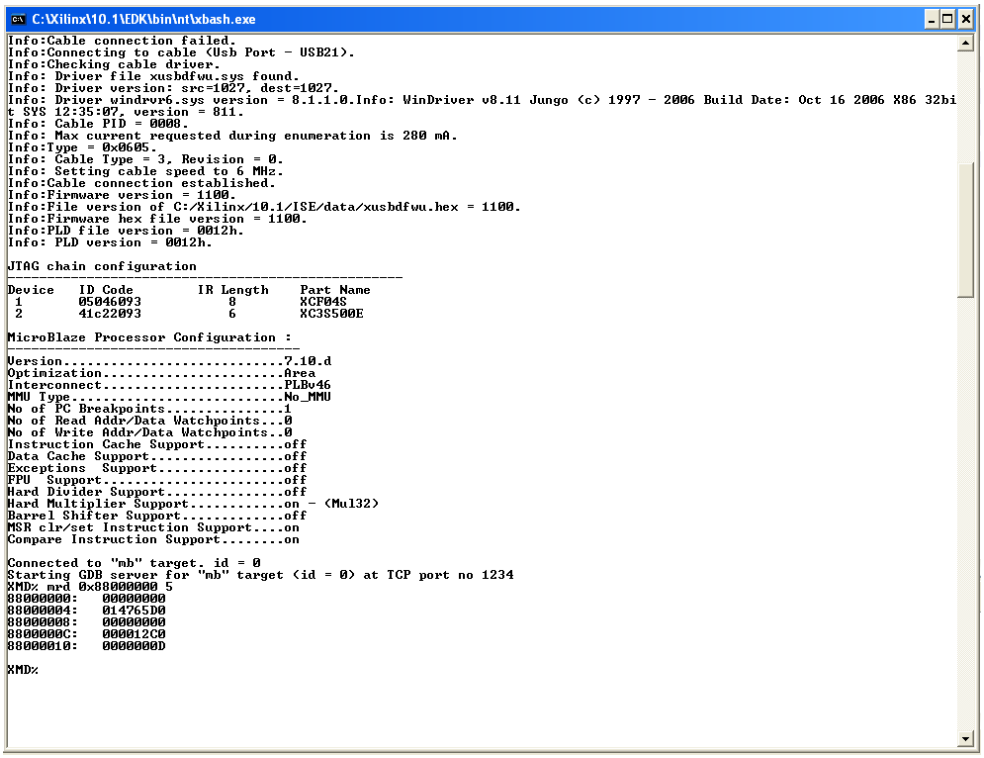


Abbildung 52: Lesezugriff auf die Servolink-Register mit XMD-Konsole

6.4.2 Debuggen des Programmcodes

Die zweite Art des Testens der Programmcodes funktioniert über die Debugfunktion der SDK-Umgebung. Dabei wird der auszuführende Programmcode unterbrochen und der Debugger startet bei der ersten Zeile des Anwendercodes. Nun kann der Anwender schrittweise durch den Programmcode durchgehen oder sich an bestimmten Stellen im Code Breakpoints setzen und warten bis der Prozessor diesen Breakpoint erreicht.

In Abbildung 53 ist das Debuggen mittels der Debugfunktion der SDK Umgebung dargestellt. Das große Hauptfenster zeigt den Programmcode und an welcher Stelle sich der Debugger momentan befindet. Im rechten Fenster kann der Anwender die Werte der Variablen zu diesem Debug-Zeitpunkt sehen. Oben rechts werden alle Breakpoints angezeigt, ob aktiv oder inaktiv. Zudem kann der Benutzer sehen, in welcher Datei an welcher Stelle sich ein Breakpoint befindet, und mit einem Doppelklick auf den Breakpoint kann er zu diesem springen. In diesem Fenster kann der Benutzer Breakpoints bequem an und ausschalten, ohne die Datei wechseln zu müssen. Im oberen linken Fenster kann der Benutzer sehen, in welcher Funktion

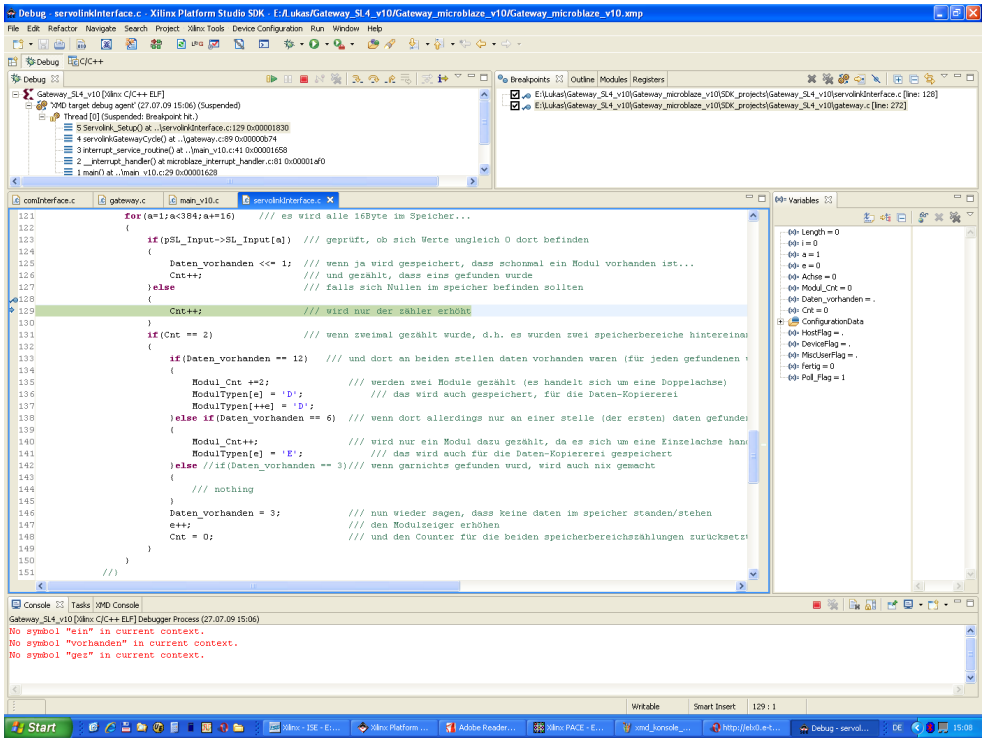


Abbildung 53: Debug mittels des Debuggers in SDK

der Prozessor momentan gestoppt hat und wie diese Funktion im übrigen Quellcode hierarchisch gelegen ist.

6.5 Weitere Testmöglichkeiten, Hindernisse und Lösungen beim Testen

Die in den vorigen Sektionen beschriebenen Arten des Auffindens von Fehlern reichen nicht aus. Es gibt Fälle in denen man den Debugger nicht einsetzen kann, da dieser den Funktionsablauf des Gateways erheblich stört. Der Debugger unterbrach den Ablauf des Programms zum Teil an sehr kritischen Stellen, so dass ein Neustart des Systems erforderlich war. An diesen kritischen Stellen stand das System zwischen zwei Zuständen und konnte keinen der beiden annehmen. Dieses Problem konnte nicht gelöst werden. So wurde z.B. die RS232-Schnittstelle genutzt, um Ausgaben auf ein Terminal zu geben, die zeigten, in welchem Zustand sich das Gateway gerade befand. So konnte festgestellt werden, ob es unter gewissen Bedingungen zu Störungen kam und das Gateway in einen erwarteten Zustand ging oder aber an ganz anderer Stelle im Programmcode stehen blieb. Diese Testmöglichkeit wurde besonders für die Fehlersuche in der Software eingesetzt.

Weitere mit dem Debugger nicht zu testende Funktionalitäten waren z.B. der Anlauf des Gateways und die automatische Konfiguration bei schnellem Aus- und wieder Einschalten des Gateways, dabei werden keine gültigen Konfigurationsdaten vom PROFIBUS gelesen. Das Gateway muss längere Zeit ausgeschaltet sein bevor es nach dem Einschalten die automatische Konfiguration korrekt durchführt. Dieser Vorgang konnte nur von außerhalb überprüft werden.

Ein weiteres Problem beim Testen waren Störungen auf dem PROFIBUS Kabel wodurch sich das System immer mit einer Fehlermeldung abschaltete. Dies konnte durch den Austausch des Kabels behoben werden.

Um die Hardware nach Fehlern zu durchsuchen wurde ein Oszilloskop verwendet. Da auf der Platine keine Messpunkte in Form von Lötösen oder Ähnlichem vorhanden sind, musste entweder ein Adapter auf den Hilscher Verbinder gesteckt werden, an dem dann Signale aus dem FPGA herausgeführt und gemessen werden konnten oder es musste Silberlackdraht an die FPGA-Pins direkt angelötet werden. Mit dem Oszilloskop wurde überprüft, ob die Systemtakte stimmen oder ob die Signale zum Schreiben oder Lesen der Speicher und der Interrupt zeitlich richtig zueinander verlaufen und mit der Simulation übereinstimmen.

6.6 Testaufbau

In Abbildung 54 ist der Aufbau zum Testen des Gateways dargestellt. Als PROFIBUS-Master dient eine SPS. Die Daten auf dem PROFIBUS werden mittels eines Busmonitors (Amproly-

zer) beobachtet. Gleichzeitig wurde mittels eines PCs und einer Verbindung von diesem über RS232 zu den sich im SERVOLINK 4-Ring befindenden Antriebsreglern überprüft, welche Daten als Soll- und Istwerte in den Antrieben stehen und in welchen Zuständen sich die einzelnen Antriebe befinden. Mit dem Laptop wird der gesamte FPGA-Inhalt, d.h. Hardware und Software, entwickelt, übersetzt und in den FPGA geschrieben. Zudem wird mit dem Laptop die Software getestet und "´gedebuggt“.

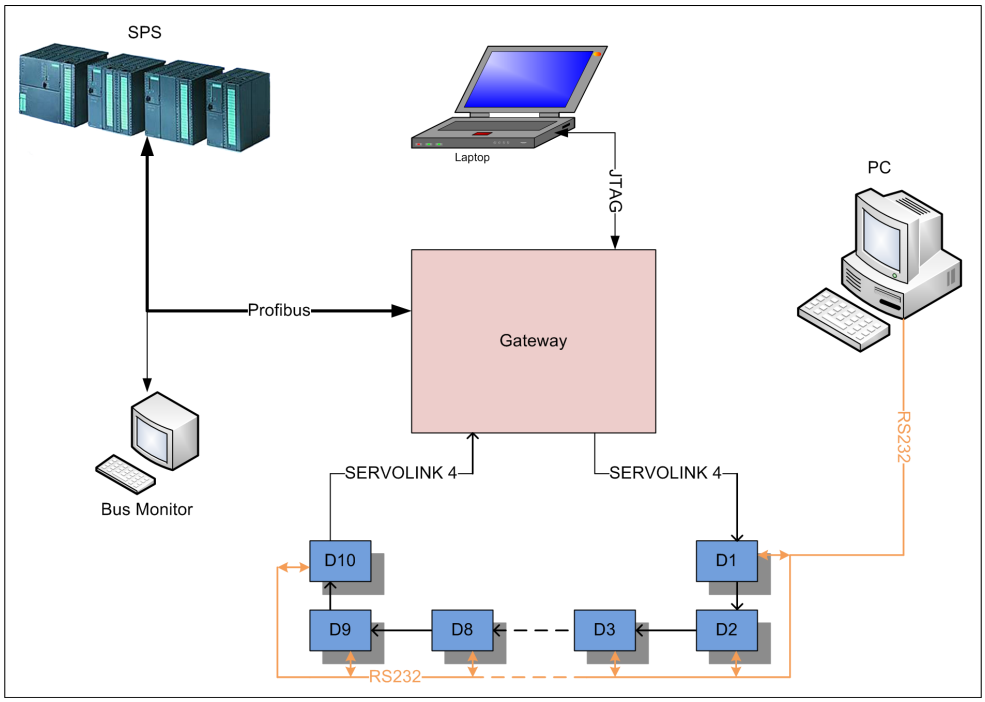


Abbildung 54: Prinzipzeichnung des Testaufbaus

Im folgenden wird ein Teil des Testaufbaus gezeigt. Die Busmonitore für PROFIBUS und SERVOLINK 4 werden nicht gezeigt, genauso wie die SPS, da dies nicht in dem benötigten Umfang dargestellt werden kann. In Abbildung 55 sieht man 3 Einzelachsantriebe (auf dem Tisch stehend), einen Doppelachsantrieb (rechts am Reck hängend) und die Powereinheit (links am Reck hängend). Ganz links im Bild sieht man die Gateway-Platine auf dem Tisch liegen.

In Abbildung 56 sind drei Motoren zu sehen, die an die Antriebsregler angeschlossen sind.

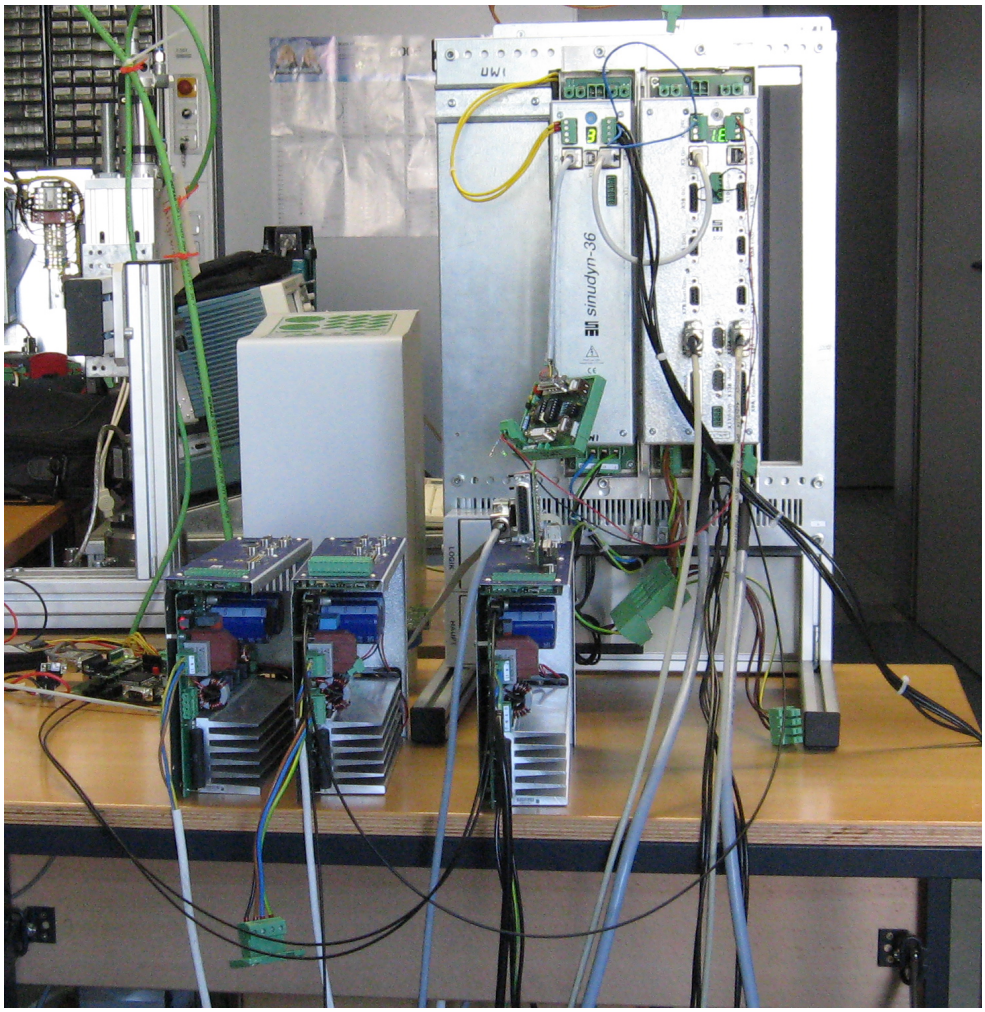


Abbildung 55: Der Testaufbau Teil 1

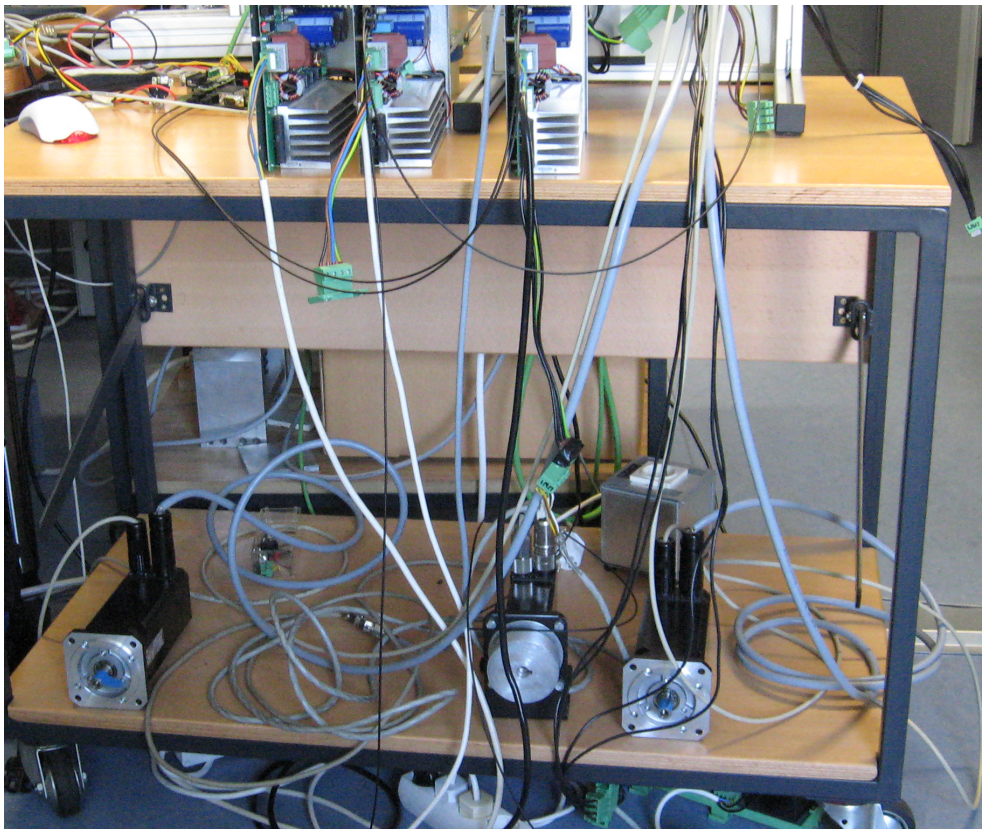


Abbildung 56: Der Testaufbau Teil 2

6.7 Testergebnisse

Die ersten Testergebnisse brachte die Simulation der Hardware. Diese bildeten die Grundlage für die weiteren Entwicklungsschritte. Der Zeitpunkt zu dem die Daten im Servolink-Eingangsspeicher gültig sind, ist ausschlaggebend für den Aufruf der Interrupt Service-Routine und damit für den gesamten Vorgang des Datenaustauschs. In den Abbildungen 57 und 60 ist die Simulation eines SERVOLINK 4-Zyklusses zu sehen. Die Simulationsergebnisse zeigten, dass das Konzept des Systems aufging.

Nach der Simulation und dem Test der Hardware, die größtenteils funktionierte und Daten zwischen den Antrieben und den Speicherblöcken in der MicroBlaze erfolgreich austauschte, wurde nach und nach die Software implementiert. Zuerst wurde der bestehende Softwarestand mit den Zustandsmaschinen ohne Interrupt des SERVOLINK 4 getestet. Das Gateway baute erfolgreich eine Verbindung über PROFIBUS zu der SPS auf, wobei zu diesem Zeitpunkt die Geräteanzahl noch vorgegeben wurde und keine automatische Konfiguration des COM-Moduls stattfand. Nun wurden nach und nach das Abfangen von Verbindungsfehlern und die automatische Konfiguration des COM-Moduls eingebaut. Das ganze System lief nun auch interruptgesteuert und tauschte erfolgreich Daten aus. Das System verhielt sich so, wie es zu Beginn der Entwicklung konzipiert wurde. Abbildung 57 zeigt einen Überblick über eine Simulation und dient nicht zur analytischen Betrachtung.

In den Abbildungen 59 und 58 ist zu sehen, dass der Interrupt nach dem 24ten Antrieb auftritt (es wird von 0 bis 15 und dann von -15 bis -9 gezählt). Abbildung 61 zeigt wie ein Write Enable für ein Byte aus der Speicheradresse erzeugt wird.

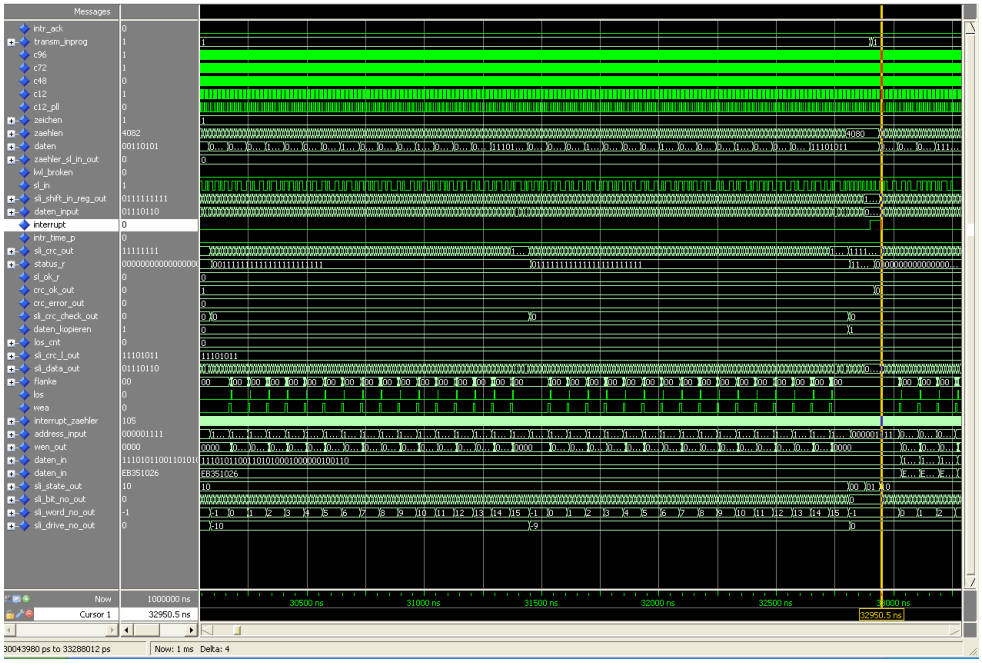


Abbildung 57: Simulation des SERVOLINK 4-Moduls - Interrupt

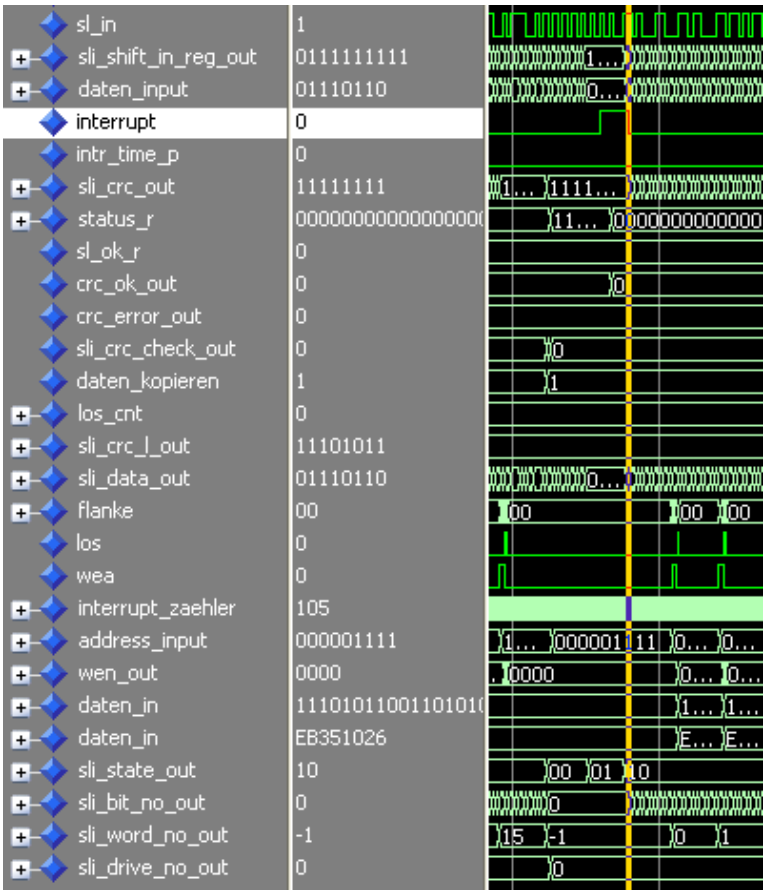


Abbildung 58: Simulation des SERVOLINK 4-Moduls - Interrupt (vergrößert)

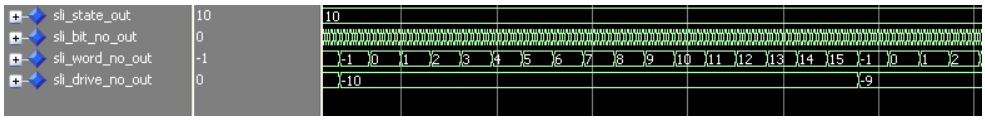


Abbildung 59: Simulation des SERVOLINK 4-Moduls - Antriebszähler

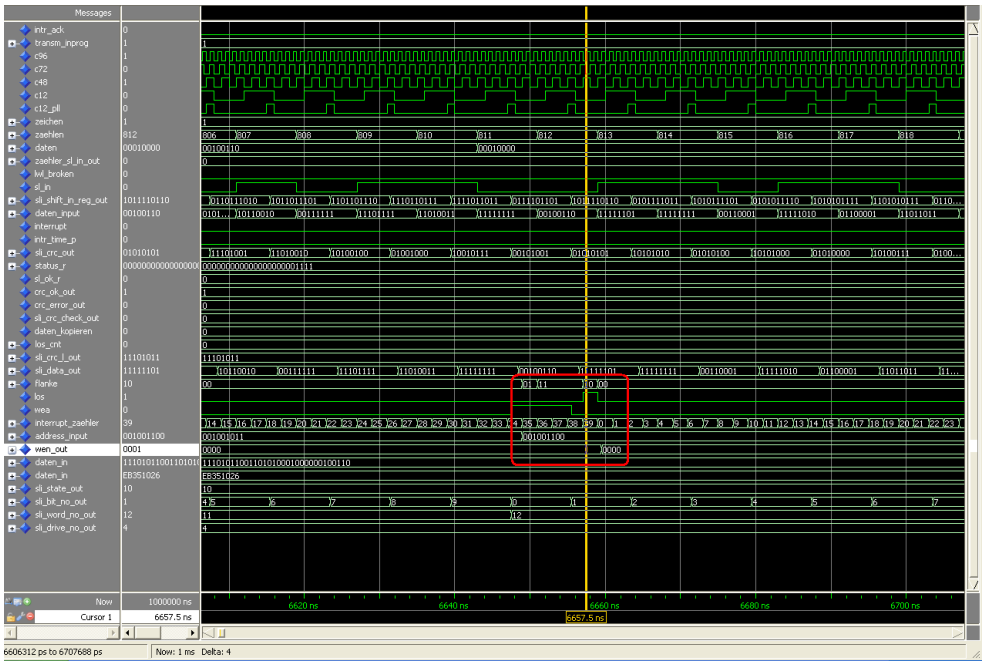


Abbildung 60: Simulation des SERVOLINK 4-Moduls - Write Enable für den Speicherblock der Istwerte

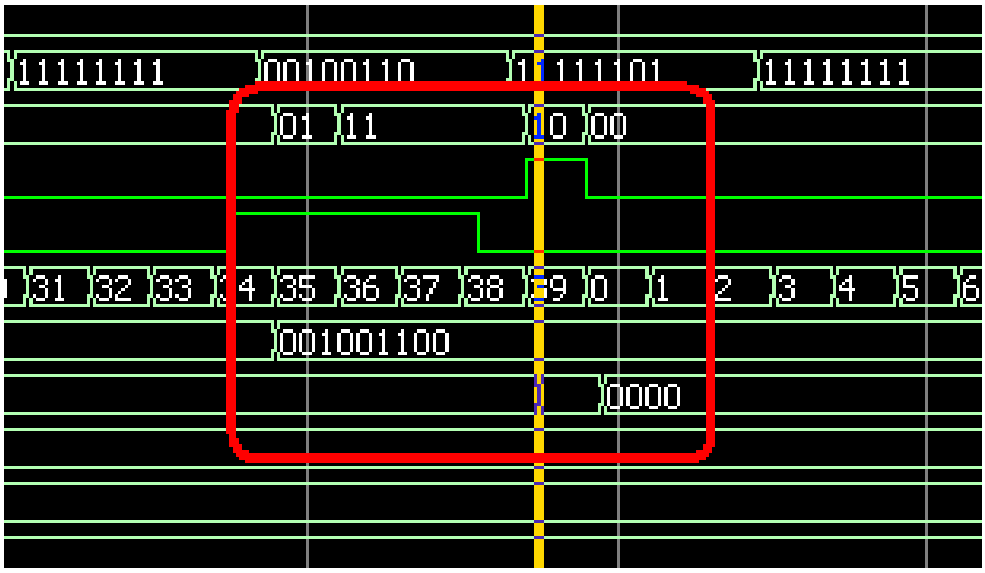
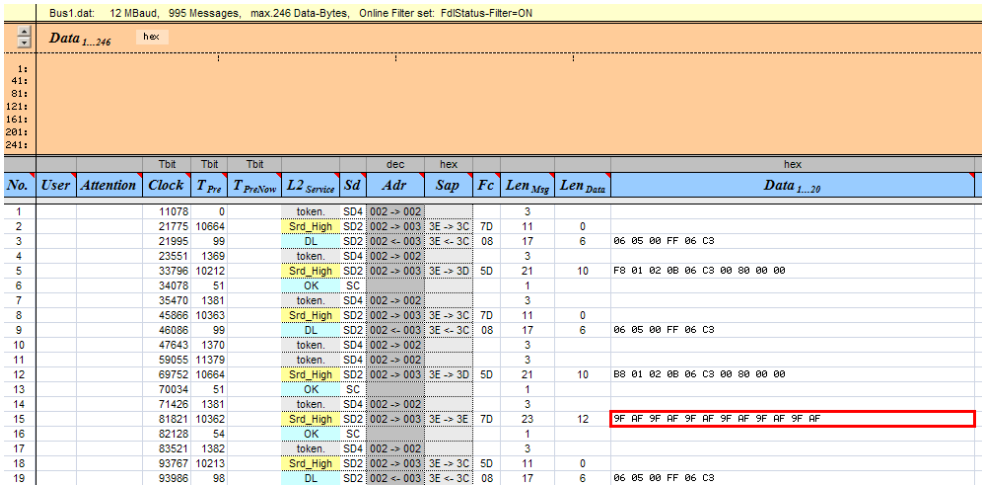


Abbildung 61: Simulation des SERVOLINK 4-Moduls - Write Enable für den Speicherblock der Istwerte (kleinerer Ausschnitt)

Die Abbildungen 62 und 63 zeigen die Daten, die über PROFIBUS gesendet und vom Busmonitor aufgezeichnet wurden. In Abbildung 62 sind die Konfigurationsdaten zu sehen, wie sie in der SPS konfiguriert sind. Abbildung 63 zeigt, wie die Prozessdaten zwischen der SPS und dem Gateway ausgetauscht werden. Die blau markierten Daten sind die Sollwerte von der SPS und die rot markierten Daten sind die Istwerte von den Antriebsreglern.

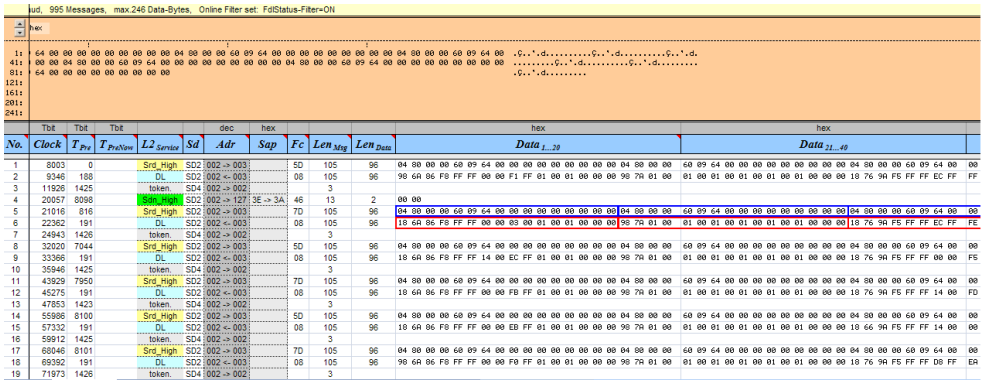


Bus1.dat: 12 MBaud, 995 Messages, max.246 Data-Bytes, Online Filter set: FdlStatus-Filter=ON

Data 1...246 hex

No.	User	Attention	Clock	T Pre	T PreNow	L2 Service	Sd	Adr	Sap	Fc	Len Msg	Len Data	Data 1...20
			Tbit	Tbit	Tbit			dec	hex				hex
1			11078	0		token.	SD4	002 -> 002			3		
2			21775	10664		Srd_High	SD2	002 -> 003	3E -> 3C	7D	11	0	
3			21995	99		DL	SD2	002 <- 003	3E <- 3C	08	17	6	06 05 00 FF 06 C3
4			23551	1369		token.	SD4	002 -> 002			3		
5			33796	10212		Srd_High	SD2	002 -> 003	3E -> 3D	5D	21	10	F8 01 02 0B 06 C3 00 80 00 00
6			34078	51		OK	SC				1		
7			35470	1381		token.	SD4	002 -> 002			3		
8			45866	10363		Srd_High	SD2	002 -> 003	3E -> 3C	7D	11	0	
9			46086	99		DL	SD2	002 <- 003	3E <- 3C	08	17	6	06 05 00 FF 06 C3
10			47643	1370		token.	SD4	002 -> 002			3		
11			59055	11379		token.	SD4	002 -> 002			3		
12			69752	10664		Srd_High	SD2	002 -> 003	3E -> 3D	5D	21	10	B8 01 02 0B 06 C3 00 80 00 00
13			70034	51		OK	SC				1		
14			71426	1381		token.	SD4	002 -> 002			3		
15			81821	10362		Srd_High	SD2	002 -> 003	3E -> 3E	7D	23	12	9F AF 9F AF 9F AF 9F AF 9F AF 9F AF
16			82128	54		OK	SC				1		
17			83521	1382		token.	SD4	002 -> 002			3		
18			93767	10213		Srd_High	SD2	002 -> 003	3E -> 3C	5D	11	0	
19			93986	98		DL	SD2	002 <- 003	3E <- 3C	08	17	6	06 05 00 FF 06 C3

Abbildung 62: Busmonitor - Konfigurationsdaten werden gesendet



ud, 995 Messages, max.246 Data-Bytes, Online Filter set: FdlStatus-Filter=ON

hex

```
1:   64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00  .ç..'.d.........ç..'.d.........ç..'.d.
41:  00 00 04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00  ......ç..'.d.........ç..'.d.....
81:  64 00 00 00 00 00 00 00 00 00                                                                        .ç..'.d.....
121:
161:
201:
241:
```

No.	Clock	T Pre	T PreNow	L2 Service	Sd	Adr	Sap	Fc	Len Msg	Len Data	Data 1...20	Data 21...40
	Tbit	Tbit	Tbit			dec	hex				hex	hex
1	8003	0		Srd_High	SD2	002 -> 003		5D	105	96	04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00	60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00
2	9346	188		DL	SD2	002 <- 003		08	105	96	98 6A 86 F8 FF FF 00 00 F1 FF 01 00 01 00 00 00 98 7A 01 00	01 00 01 00 01 00 01 00 01 00 00 00 18 76 9A F5 FF FF EC FF FF
3	11926	1425		token.	SD4	002 -> 002			3			
4	20057	8098		Srd_High	SD2	002 -> 127	3E -> 3A	46	13	2	00 00	
5	21016	816		Srd_High	SD2	002 -> 003		7D	105	96	04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00	60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00
6	22362	191		DL	SD2	002 <- 003		08	105	96	18 6A 86 F8 FF FF 00 00 03 00 01 00 01 00 00 00 98 7A 01 00	01 00 01 00 01 00 01 00 01 00 00 00 18 76 9A F5 FF FF EC FF FE
7	24943	1426		token.	SD4	002 -> 002			3			
8	32020	7044		Srd_High	SD2	002 -> 003		5D	105	96	04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00	60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00
9	33386	191		DL	SD2	002 <- 003		08	105	96	18 6A 86 F8 FF FF 14 00 EC FF 01 00 01 00 00 00 98 7A 01 00	01 00 01 00 01 00 01 00 01 00 00 00 18 76 9A F5 FF FF 00 00 F5
10	35946	1425		token.	SD4	002 -> 002			3			
11	43929	7950		Srd_High	SD2	002 -> 003		7D	105	96	04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00	60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00
12	45275	191		DL	SD2	002 <- 003		08	105	96	18 6A 86 F8 FF FF 00 00 FB FF 01 00 01 00 00 00 98 7A 01 00	01 00 01 00 01 00 01 00 01 00 00 00 18 76 9A F5 FF FF 14 00 FD
13	47853	1423		token.	SD4	002 -> 002			3			
14	55986	8100		Srd_High	SD2	002 -> 003		5D	105	96	04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00	60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00
15	57332	191		DL	SD2	002 <- 003		08	105	96	18 6A 86 F8 FF FF 00 00 EB FF 01 00 01 00 00 00 98 7A 01 00	01 00 01 00 01 00 01 00 01 00 00 00 18 66 9A F5 FF FF 14 00 00
16	59912	1425		token.	SD4	002 -> 002			3			
17	68046	8101		Srd_High	SD2	002 -> 003		7D	105	96	04 80 00 00 60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00	60 09 64 00 00 00 00 00 00 00 00 00 04 80 00 00 60 09 64 00 00
18	69392	191		DL	SD2	002 <- 003		08	105	96	98 6A 86 F8 FF FF 00 00 F0 FF 01 00 01 00 00 00 98 7A 01 00	01 00 01 00 01 00 01 00 01 00 00 00 18 76 9A F5 FF FF D8 FF EA
19	71973	1426		token.	SD4	002 -> 002			3			

Abbildung 63: Busmonitor - Prozessdaten werden ausgetauscht

6.8 Ausblick

In Zukunft soll das Gateway noch zusätzliche Funktionen erhalten. Es werden zwei Status-LEDs angebracht, die den Zustand des Gateways anzeigen. Zudem wird sehr wahrscheinlich die USB-Schnittstelle dazu verwendet werden, anstatt eines PROFIBUS-Masters, einen PC als Steuerungsrechner einzusetzen. Dies könnte ebenso über die serielle Schnittstelle realisiert werden. Die serielle Schnittstelle könnte auch für den Anschluss eines Terminals genutzt werden, über das dem Arbeiter an der Maschine Informationen und aufgetretene Fehler/Probleme angezeigt werden könnten.
Mit der neuesten Distribution der Entwicklungsumgebungen von ISE und XPS sollte es möglich sein, das Design der MicroBlaze zu optimieren, so dass in dieser Version des Gateways verschwendete Ressourcen für weitere Implementierungen zur Verfügung stehen würden. Mit der genutzten Version der Entwicklungsumgebungen lässt sich z.B. kein kleinerer DPRAM Block als 8 kByte einfügen. Mit der neuen Distribution ließen sich die DPRAM Blöcke auf exakt 384 Byte einstellen, wodurch mehr Platz für eine andere Logik zur Verfügung stehen würde.

6.8.1 Hilfe bei falschem Systemaufbau

In einem Betrieb, in dem das Gateway eingesetzt wird, können Fehler auftreten. Einer dieser Fehler ist die unterschiedliche Konfiguration von Gerätetypen und Gerätereihenfolge in der SPS-Oberfläche und den tatsächlichen Garäten und deren Reihenfolge im SERVOLINK 4-Ring. Um dem Bediener an der Maschine, in diesem Fehlerfall eine Hilfe zu geben, könnte neben dem Auslesen der Konfigurationsdaten, die über PROFIBUS gesendet werden noch die Reihenfolge und die Gerätetypen im SERVOLINK 4-Ring automatisch vom Gateway erkannt werden. Dazu existiert eine Funktion, die den Servolink-Empfangsspeicher nach Daten ungleich Null durchsucht. Je nachdem an welcher Position Daten stehen, wird erkannt, ob es sich um eine Einzel- oder Doppelachse handelt. Diese Informationen werden in einem Array gespeichert und es kann die vom Gateway erkannte Gerätereihenfolge mit der in der SPS konfigurierten verglichen werden, siehe dazu Abschnitt 5.4. Dieser Unterschied im Aufbau könnte mittels der Status-LEDs angezeigt oder bei zukünftigem Einsatz eines Terminals ausgegeben werden. Ein Einsatz dieser Funktion würde die Fehlersuche stark erleichtern.

7 Fazit

In dieser Arbeit wurden zuerst die auf der Platine vorhandenen Bauteile und die an das Gateway gestellten Anforderungen analysiert und aus den gewonnenen Erkenntnissen eine Systemstruktur für die MicroBlaze und die dazugehörige Logik/Hardware in VHDL und ein Funktionsablauf für die Software entwickelt. Das gesamte System besteht aus einer MicroBlaze, die Daten zwischen Speicherbänken umkopiert. Diese Daten werden von zwei weiteren Modulen, dem Hilscher COM-Modul und dem SERVOLINK 4-Modul über die angeschlossenen Bussysteme versendet und empfangen. Das Kopieren der Daten wird durch Software realisiert, ebenso werden die Status- und Kontrollflags des Hilscher COM-Moduls zur Datenübertragung über PROFIBUS mittels Software kontrolliert. Die Datenübertragung über SERVOLINK 4 wird vollständig mit VHDL-Logik realisiert. Die Quittierung des Interrupts, der von dem SERVOLINK 4-Modul ausgelöst wird, wird in Software realisiert, ist aber für den SERVOLINK 4-Prozess nicht unbedingt relevant. Der Anwender muss den Interrupt quittieren, das SERVOLINK 4-Modul kann auch ohne diesen Daten übertragen. Aber um zu gewährleisten, dass alle Daten rechtzeitig und korrekt übertragen werden, wird diese Quittierung angewendet. Die Komponenten des Gateways, ob innerhalb oder außerhalb des FPGAs, sind alle über definierte Schnittstellen miteinander vernetzt. Das Gateway arbeitet selbstständig, sofern es korrekt über PROFIBUS parametriert wurde.

Die Entwicklung dieses Gateways im Rahmen dieser Arbeit bildet die Basis für weitere Forschungen und Entwicklungen. Bei der Firma SIEB & MEYER können die in dieser erworbenen Kenntnisse für verschiedene Einsatzgebiete genutzt und weiterentwickelt werden.

Literaturverzeichnis

[1]

[2] BRECHMANN, DZIEIA, HÖRNEMANN, HÜBSCHER, JAGLA und PETERSEN: *Elektrotechnik - Tabellen Kommunikationselektronik*. Nummer ISBN 3-14-225037-9. Westermann, 3 Auflage, 2005. Enthält kurze knappe Informationen über allerlei Technisches.

[3] FELSER, MAX: *PROFIBUS Handbuch*. http://profibus.felser.ch/technik/PROFIBUS_Technik.pdf, August 17 2009.

[4] FIRMA SIEB & MEYER, ENTWICKLER DER. Nicht publiziert, 2008. Interne Dokumentationen und Konzepte der SIEB & MEYER AG.

[5] HILSCHER: *Hilscher COM-CA-DPS Produktübersicht*. http://de.hilscher.com/products_details_hardware.html?p_id=P_4063200a88918&bs=8, 2009. Datenblatt der Hilscher COM-Module; Online; Stand 9. September 2009.

[6] HILSCHER: *Hilscher Firmen Profil*. http://de.hilscher.com/home_aboutus.html, 2009. Die Firma Hilscher stellt sich vor; Online; Stand 9. September 2009.

[7] HILSCHER, H.J.: *Design Guide COM and COMX - Communication Modules COM-A, COM-B, COM-C and COMX-C*. http://uk.hilscher.com/files_manuals/COM_ABC_COMX_C_DesignGuide_en.pdf, November 17 2007. Technische Daten des COM-Moduls.

[8] JESMAN, ROD, FERNANDO MARTINEZ VALLINA und JAFAR SANIIE: *MicroBlaze Tutorial Creating a Simple Embedded System and Adding Custom Peripherals Using Xilinx EDK Software Tools*. http://ecasp.ece.iit.edu/mbtutorial.pdf. Wie erstellt man eine MicroBlaze und eigene IP-Cores mittels Wizard in XPS?

[9] KATZ, MARIANNE, AXEL FUNKE, GERHARD BIWER, YUE LI, THOMAS SEBASTIANY, GERD GÜLDENPFENNING, HORST ALTSTÄDT und KLAUS BENDER: *PROFIBUS - Der Feldbus für die Automation*. Nummer ISBN 3-445-16170-8. Carl Hanser Verlag München Wien, 1990.

[10] MÄDER, ANDREAS: *VHDL Kompakt*. http://tams-www.informatik.uni-hamburg.de/vhdl/doc/kurzanleitung/vhdl.pdf.

[11] WIKIPEDIA: *MicroBlaze — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=MicroBlaze&oldid=63862836, 2009. [Online; Stand 15. September 2009].

[12] WIKIPEDIA: *Speicherprogrammierbare Steuerung — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Speicherprogrammierbare_Steuerung&oldid=64317518, 2009. [Online; Stand 15. September 2009].

[13] WIKIPEDIA: *Very High Speed Integrated Circuit Hardware Description Language — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Very_High_Speed_Integrated_Circuit_Hardware_Description_Language&oldid=64310536, 2009. [Online; Stand 15. September 2009].

[14] XILINX: *MicroBlaze Software Reference Guide*. http://narong.ece.engr.tu.ac.th/embedded/document/swref.pdf.

[15] XILINX: *Xilinx Firmen Profil*. http://www.xilinx.com/company/about.htm. Die Firma Xilinx stellt sich vor.

[16] XILINX: *Block RAM (BRAM) Block (v1.00a)*. http://www.xilinx.com/support/documentation/ip_documentation/bram_block.pdf, August 13 2004. Xilinx BRAM Dokumentation.

[17] XILINX: *EDK Concepts, Tools and Techniques - A Hands-On Guide to Effective Embedded System Design*. http://www.xilinx.com/support/documentation/sw_manuals/edk10_ctt.pdf, September 18 2008. Anleitung zum Entwerfen von Embedded Systems.

[18] XILINX: *MicroBlaze Processor Reference Guide - Embedded Development Kit 10.1i*. http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf, January 12 2008. Xilinx MicroBlaze Dokumentation.

[19] XILINX: *Platform Flash PROM User guide*. http://www.xilinx.com/support/documentation/user_guides/ug161.pdf, October 17 2008. Xilinx PROM Anleitung.

[20] XILINX: *Spartan-3E FPGA Family: Complete Data Sheet*. http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf, April 18 2008. Xilinx FPGA Datenblatt.

[21] XILINX: *Using Xilinx XCF02S/XCF04S JTAG PROMs for Data Storage Applications*. http://www.xilinx.com/support/documentation/application_notes/xapp544.pdf, January 11 2008.

[22] XILINX: *XPS Block RAM (BRAM) Interface Controller (v1.00a)*. http://www.xilinx.com/support/documentation/ip_documentation/xps_bram_if_cntlr.pdf, July 25 2008. Xilinx BRAM Interface Controller Dokumentation.

[23] XILINX: *Digital Clock Manager (DCM) Module*. http://www.xilinx.com/support/documentation/ip_documentation/dcm_module.pdf, April 24 2009. Xilinx DCM Dokumentation.

[24] XILINX: *MicroBlaze Debug Module (MDM) (v1.00f)*. http://www.xilinx.com/support/documentation/ip_documentation/mdm.pdf, June 24 2009. Xilinx MDM Dokumentation.

[25] XILINX: *XPS Multi-CHannel External Memory Controller (XPS MCH EMC) (v3.00a)*. http://www.xilinx.com/support/documentation/ip_documentation/xps_mch_emc.pdf, April 24 2009. Xilinx EMC Dokumentation.

A Signaltabellen

A.1 MicroBlaze (VHDL-Komponente)

Tabelle 7: Externe Ports der MicroBlaze Teil 1

Signal/Port Name	Netz-/Signalname	I/O	Typ
External_Memory_DQ_pin	External_Memory_DQ	IO	std_logic_vector[0:7]
External_Memory_A_pin	External_Memory_A_pin	O	std_logic_vector[0:31]
External_Memory_WEN_pin	External_Memory_WEN	O	std_logic
External_Memory_OEN_pin	External_Memory_OEN	O	std_logic_vector[0:0]
External_Memory_CEN_pin	External_Memory_CEN	O	std_logic_vector[0:]
Output_BRAM_Din_B_pin	Output_BRAM_Din_B	O	std_logic_vector[0:31]
Output_BRAM_Addr_B_pin	Output_BRAM_Addr_B	I	std_logic_vector[0:31]
Output_BRAM_WEN_B_pin	Output_BRAM_WEN_B	I	std_logic_vector[0:3]
Output_BRAM_EN_B_pin	Output_BRAM_EN_B	I	std_logic
Output_BRAM_Clk_B_pin	Output_BRAM_Clk_B	I	std_logic
Output_BRAM_Rst_B_pin	Output_BRAM_Rst_B	I	std_logic
Input_BRAM_Dout_B_pin	Output_BRAM_Din_B	I	std_logic_vector[0:31]
Input_BRAM_Addr_B_pin	Output_BRAM_Addr_B	I	std_logic_vector[0:31]
Input_BRAM_WEN_B_pin	Output_BRAM_WEN_B	I	std_logic_vector[0:3]
Input_BRAM_EN_B_pin	Output_BRAM_EN_B	I	std_logic
Input_BRAM_Clk_B_pin	Output_BRAM_Clk_B	I	std_logic
Input_BRAM_Rst_B_pin	Output_BRAM_Rst_B	I	std_logic

Tabelle 8: Externe Ports der MicroBlaze Teil 2

Signal/Port Name	Netz-/Signalname	I/O	Typ
sl_reg_INTERRUPTPOS_pin	sl_reg_INTERRUPTPOS	O	std_logic_vector[12:0]
sl_reg_Intr_Ackn_pin	sl_reg_Intr_Ackn	O	std_logic
sl_reg_WriteEnable_pin	sl_reg_WriteEnable	I	std_logic
sl_reg_Intr_Time_Pos_pin	sl_reg_Intr_Time_Pos	I	std_logic
sl_reg_Cylce_start_pin	sl_reg_Cyle_start	I	std_logic
sl_reg_NULLTR_cnt_pin	sl_reg_NULLTR_cnt	I	std_logic_vector[15:0]
sl_reg_Servolink_ok_pin	sl_reg_Servolink_ok	I	std_logic
sl_reg_TRANSM_INPROG_pin	sl_reg_TRANSM_INPROG	I	std_logic_vector[0:0]
sl_reg_Status_Bits_pin	sl_reg_Status_Bits	I	std_logic_vector[23:0]
sl_reg_los_pin	sl_reg_los	I	std_logic
sl_reg_lwl_broken_pin	sl_reg_lwl_broken	I	std_logic
sl_reg_daten_kopieren_pin	sl_reg_daten_kopieren	I	std_logic
sl_reg_Anzahl_Module_pin	sl_reg_Anzahl_Module	O	std_logic_vector[31:0]
microblaze_0_INTERRUPT_pin	microblaze_0_INTERRUPT	I	std_logic
dcm_locked	Dcm_all_locked	I	std_logic
sys_rst_pin	sys_rst_s	I	std_logic
sys_clk_pin	sys_clk_s	I	std_logic
fpga_0_RS232_TX_pin	fpga_0_RS232_TX	O	std_logic
fpga_0_RS232_RX_pin	fpga_0_RS232_RX	I	std_logic

A.2 XPS Multi-CHannel External Memory Controller

Tabelle 9: XPS MCH EMC Signale

Signal/Port Name	Interface	I/O	Beschreibung
MCH_PLB_CLK	System	I	MCH/PLB Takt
RdClk	System	I	Takt mit dem die Daten gelesen werden
MCH_PLB_Rst	System	I	MCH/PLB Reset
MEM_CEN[0:C_NUM_BANKS_MEM-1]	External Memory	O	Memory chip enable (low aktiv)
MEM_OEN[0:C_NUM_BANKS_MEM-1]	External Memory	O	Memory output enable
MEM_WEN	External Memory	O	Memory write enable
MEM_A[0:C_MCH_SPLB_AWIDTH-1]	External Memory	O	Memory address bus
MEM_DQ[0:C_MAX_MEM_WIDTH-1]	External Memory	IO	Daten Ein- und Ausgang (Tristate)

Tabelle 10: XPS MCH EMC Parameter Teil 1

Parameter Name	erlaubte Werte	VHDL Typ	Beschreibung
C_SPLB_DWIDTH	32, 64, 128	Integer	Datenbusbreite für MCH und PLB in Bit
C_MCH_SPLB_AWIDTH	32	Integer	Adressbusbreite für MCH und PLB in Bit
C_SPLB_SMALLEST_MASTER	32, 64, 128	Integer	Datenbusbreite des kleinsten Masters in Bit
C_MCH_NATIVE_DWIDTH	32	Integer	Datenbusbreite des Slave in Bit
C_TCEDV_PS_MEM_x[2]	Wert in Pikosekunden	Integer	Zeit, in welcher im Lesezyklus das ChipSelect Signal low-Pegel haben muss bis die Daten gültig sind
C_TAVDV_PS_MEM_x[2]	Wert in Pikosekunden	Integer	Zeit, in welcher die Adresse im Lesezyklus gültig sein muss bevor die Daten gültig sind
C_THZCE_PS_MEM_x[2]	Wert in Pikosekunden	Integer	Zeit, in welcher im Lesezyklus das ChipSelect Signal high-Pegel haben muss, bevor der Datenbus hochohmig wird
C_THZOE_PS_MEM_x[2]	Wert in Pikosekunden	Integer	Zeit, in welcher im Lesezyklus das OutputEnable(RD) Signal high-Pegel haben muss, bevor der Datenbus hochohmig wird
C_TWC_PS_MEM_x[2]	Wert in Pikosekunden	Integer	Schreibzykluszeit

Tabelle 11: XPS MCH EMC Parameter Teil 2

Parameter Name	**erlaubte Werte**	**VHDL Typ**	**Beschreibung**
C_TWP_PS_MEM_x[2]	Wert in Pikosekunden	Integer	minimale Dauer des WriteEnable Pulses
C_TLZWE_PS_MEM_x[2]	Wert in Pikosekunden	Integer	Zeit, in welcher das WriteEnable im Schreibzyklus high-Pegel haben muss, bevor der Datenbus hochohmig wird

[2] Gilt für Speicher Bank x (x = Wert von 0 bis 3).

A.3 XPS Block RAM Interface Controller

Tabelle 12: BRAM Interface Controller Parameter

Parameter Name	erlaubte Werte	VHDL Typ	Beschreibung
C_BASEADDR	0 bis 0xffffffff	std_logic_vector	BRAM Basisadresse von C_SPLB_AWIDTH
C_HIGHADDR	0 bis 0xffffffff	std_logic_vector	BRAM Endadresse von C_SPLB_AWIDTH
C_SPLB_NATIVE_DWIDTH	32, 64 oder 128	Integer	Legt die benötigte Breite des BRAM Interfaces fest
C_SPLB_AWIDTH[3]	32, 36[4]	Integer	PLB Adressbus Breite
C_SPLB_DWIDTH[3]	32, 64 oder 128	Integer	PLB Datenbus Breite
C_SPLB_NUM_MASTERS[3]	1 - 16	Integer	Anzahl der Master

[3] Diese Parameter werden, während der Erstellung des Systems automatisch durch die SDK XPS Tools ermittelt und festgelegt

[4] Xilinx SDK begrenzt die Adressen auf 32 Bit

A.4 XPS Block DPRAM Block

Tabelle 13: BRAM Block Signale

Signal/Port Name	Interface	I/O	Beschreibung
BRAM_Rst_A	Port A	I	BRAM Reset, High aktiv
BRAM_Clk_A	Port A	I	BRAM Clock
BRAM_EN_A	Port A	I	BRAM Enable, High aktiv
BRAM_WEN_A	Port A	I	BRAM Write Enable, High aktiv
BRAM_Addrv_A (0:C_PORT_AWIDTH-1)	Port A	I	BRAM Address
BRAM_Din_A (0:C_PORT_AWIDTH-1)	Port A	O	BRAM Data Output
BRAM_Dout_A (0:C_PORT_AWIDTH-1)	Port A	I	BRAM Data Input
BRAM_Rst_B	Port B	I	BRAM Reset, High aktiv
BRAM_Clk_B	Port B	I	BRAM Clock
BRAM_EN_B	Port B	I	BRAM Enable, High aktiv
BRAM_WEN_B	Port B	I	BRAM Write Enable, High aktiv
BRAM_Addr_B (0:C_PORT_AWIDTH-1)	Port B	I	BRAM Address
BRAM_Din_B (0:C_PORT_AWIDTH-1)	Port B	O	BRAM Data Output
BRAM_Dout_B (0:C_PORT_AWIDTH-1)	Port B	I	BRAM Data Input

Tabelle 14: BRAM Block Parameter

Parameter Name	erlaubte Werte	VHDL Typ	Beschreibung
C_PORT_AWIDTH	9 - 17	Integer	Port A und B Adressbreite
C_PORT_DWIDTH	32, 64	Integer	Port A und B Datenbreite
C_NUM_WE	1, 2, 4, 8	Integer	Anzahl an Write Enables (Byte Enables zum Schreiben)
C_FAMILY	spartan2,spartan2e, spartan3,virtex, virtexe,virtex2, qvirtex2,qrvirtex2, virtex2p,virtex4	String	Ziel FPGA Familie für den BRAM Block
C_MEMSIZE	512 - 131072	Integer	Größe des BRAM Blocks in Byte

A.5 Servolink Register

Tabelle 15: Register Signale

Signal/Port Name	Netz-/Signalname	I/O	Typ
INTERRUPTPOS	sl_reg_INTERRUPTPOS	O	std_logic_vector[12:0]
Intr_Ackn	sl_reg_Intr_Ackn	O	std_logic
daten_kopieren	sl_reg_daten_kopieren	I	std_logic
Anzahl_Module	sl_reg_Anzahl_Module	O	std_logic_vector[31:0]
lwl_broken	sl_reg_lwl_broken	I	std_logic
los	sl_reg_los	I	std_logic
WriteEnable	sl_reg_WriteEnable	I	std_logic
Intr_Time_Pos	sl_reg_Intr_Time_Pos	I	std_logic
Cycle_start	sl_reg_Cycle_start	I	std_logic
NULLTR_cnt	sl_reg_NULLTR_cnt	I	std_logic_vector[15:0]
Servolink_ok	sl_reg_Servolink_ok	I	std_logic
TRANSM_INPROG	sl_reg_TRANSM_INPROG	I	std_logic_vector[0:0]
Status_Bits	sl_reg_Status_Bits	I	std_logic_vector[23:0]

A.6 Debug Modul

Tabelle 16: System Signale des Debug Moduls

Signal/Port Name	I/O	Beschreibung
Interrupt	O	Interrupt vom UART
Debug_SYS_Rst	O	Debug System Reset
EXT_BRK	O	Externer Interrupt
EXT_NM_BRK	O	Externer nichtmaskierbarer Interrupt

A.7 FPGA

Tabelle 17: Entity Ports (Signale am FPGA)

Signal/Port Name	I/O	VHDL Typ	Beschreibung
CLK_in	I	std_logic	CLK_in ist der Takt vom Quarz außerhalb des FPGA
RST_in	I	std_logic	RST_in kommt vom Watchdog außerhalb des FPGA
WatchDog_control	O	std_logic	Ausgang, der den Watchdog ruhig stellt
RD	O	std_logic_vector[0:0]	READ Signal für das COM-Modul
WR	O	std_logic	WRITE Signal für das COM-Modul
CS	O	std_logic_vector[0:0]	ChipSelect Signal für das COM-Modul
Hilscher_Address	O	std_logic_vector[13:0]	Adressleitungen des COM-Modul Speichers
Hilscher_Daten_IO	I/O	std_logic_vector[7:0]	Datenleitungen des COM-Modul Speichers
RS232_in_X0	I	std_logic	UART Eingangsleitung
RS232_out_X0	O	std_logic	UART Ausgangsleitung
SL_in	I	std_logic	SERVOLINK Eingangsleitung
SL_out	O	std_logic	SERVOLINK Ausgangsleitung

A.8 Clock Generator

Tabelle 18: Clock Generator Signale

Signal/Port Name	I/O	VHDL Typ	Beschreibung
CLKIN_IN	I	std_logic	Eingangstakt aus dem die Systemtakte gewonnen werden
RST_IN	I	std_logic	Reset Eingang
CLKIN_IBUFG_OUT	O	std_logic	CLK_IN gepuffert als Ausgangssignal
CLKFX_OUT	O	std_logic	Ausgangstakt der über Parameter eingestellt werden kann
CLK0_OUT	O	std_logic	Ausgangstakt mit gleicher Frequenz wie Eingangstakt
LOCKED_OUT	O	std_logic	LOCKED_out gibt an, wenn alle Takte phasengleich und eingeschwungen sind
CLKDV_OUT	O	std_logic	Eingangstakt durch 2 geteilt
CLK2X_OUT	O	std_logic	Doppelter Eingangstakt

Tabelle 19: Clock Generator Parameter

Parameter Name	**erlaubte Werte**	**VHDL Typ**	**Beschreibung**
C_CLKDV_DIVIDE	1.5 - 16.0	Real	Wert durch den der Eingangstakt für CLKDV_OUT geteilt wird
C_CLKIN_PERIOD		Real	Periodendauer des Eingangstaktes in Pikosekunden
C_CLKFX_DIVIDE	1 - 32	Integer	Wert durch den der Eingangstakt für CLKFX_OUT geteilt wird
C_CLKFX_MULTIPLY	1 - 32	Integer	Wert mit dem der Eingangstakt für CLKFX_OUT multipliziert wird
C_CLKOUT_PHASE_SHIFT	None, Fixed, Variable	String	Gibt an, in welcher Form eine Phasenverschiebung gemacht werden soll
C_CLK_FEEDBACK	1x, 2x	String	Gibt an, welcher Takt als Feedback für die Synchronisation benutzt wird
C_DUTY_CYCLE_	TRUE, FALSE	Boolean	Gibt an, ob eine Korrektur des Tastverhältnisses stattfinden soll
C_PHASE_SHIFT	-255 bis 255	Integer	Wert für die Phasenverschiebung
C_STARTUP_WAIT	TRUE, FALSE	Boolean	Gibt an, ob es eine Startverzögerung gibt

A.9 SERVOLINK 4-Modul

Tabelle 20: SERVOLINK 4-Modul Signale Teil 1

Signal/Port Name	I/O	VHDL Typ	Beschreibung
SL_in	I	std_logic	Digitales Abbild des SERVOLINK 4-Eingangs-Lwls
SL_out	O	std_logic	Digitales Abbild des SERVOLINK 4-Ausgangs-Lwls
RST	I	std_logic	Reset Eingang vom SERVOLINK 4-Modul
C96	I	std_logic	Systemtakt mit 96 MHz
C12	I	std_logic	Systemtakt mit 12 MHz
interrupt	O	std_logic	Interruptsignal vom SERVOLINK 4-Modul
TRANSM_INPROG	O	std_logic_vector	Gibt an, ob eine SERVOLINK 4-Übertragung stattfindet
Daten_Output	I	std_logic_vector	Daten zum Versenden über SERVOLINK 4
Daten_Input	O	std_logic_vector	Daten, die über SERVOLINK 4 empfangen werden
Address_Output	O	std_logic_vector	Adresse, von der die zu versendenden Daten gelesen werden
Address_Output	O	std_logic_vector	Adresse, an die die empfangenen Daten geschrieben werden

Tabelle 21: SERVOLINK 4-Modul Signale Teil 2

Signal/Port Name	I/O	VHDL Typ	Beschreibung
wea	O	std_logic	Schreibsignal wenn Daten (1 Byte)über SL empfangen wurden
SL_ok	O	std_logic	Gibt an, ob Servolink OK ist
CYCLE_start	O	std_logic	Zeigt an, wann ein neuer SL Zyklus beginnt
NULLTR_cnt	O	std_logic_vector	Gibt an, wie viele Nulldaten Telegramme gesendet wurden
STATUS_r	O	std_logic_vector	Gibt den CRC-Status jedes Antriebsreglers an
lwl_broken	O	std_logic	Zeigt an, ob ein LWL an den SL Eingang angeschlossen ist
INTERRUPTPOS	I	std_logic_vector	Gibt eine Position im SL Zyklus vor, zu der ein Interrupt kommen soll
INTR_TIME_P	O	std_logic	Interrupt, der durch INTERRUPTPOS vorgegeben wird
INTR_ACKN	I	std_logic	Quittiert den Interrupt

A.10 Hilscher COM-Modul

Tabelle 22: Hilscher COM-Modul Pins

Pin	Signal	I/O	FPGA Port
21	Reset, low aktiv	I	RST_in
24	Read, low aktiv	I	RD
25	Write, low aktiv	I	WR
26	Chip select, low aktiv	I	CS
27 - 40	Addressline 13 bis 0	I	Hilscher_Address[13:0]
41 - 48	Dataline 7 bis 0	I/O	Hilscher_Daten_IO[7:0]

B Grafiken

B.1 Verweis auf CD

Auf den folgenden Seiten stehen Verweise zu Grafiken, die das Kapitel 5 ergänzen. Diese Grafiken sind zu groß, um sie gut erkennbar in das Dokument einzufügen. Diese Grafiken sind auf der zu dieser Diplomarbeit zugehörigen CD enthalten und bei Prof. Ulfert Meiners einzusehen.

B.2 External Ports

Diese Grafik heißt „bsp_external_ports2.png“ und zeigt wie die Signale/Ports der IP-Cores zu externen Ports in der VHDL-Komponente MicroBlaze geleitet werden (oberer Teil, XPS-Oberfläche) und wie diese dann im Toplevel als Ports einer Instanz weiter verbunden werden (unterer Teil, ISE VHDL Editor).

B.3 MicroBlaze

Die Grafik heißt „Signalanbindung_MicroBlaze_Komponenten.png“ und zeigt den gesamten Zusammenhang zwischen den einzelnen Komponenten des Gateways und wird durch Abbildung 35 auf Seite 67 ergänzt.

C Programmcodes

Der gesamte Quellcode befindet sich auf der zu dieser Diplomarbeit zugehörigen CD.

Der Hardware Quellcode, d.h. die VHDL-Dateien, sowie die MicroBlaze System-Dateien befinden in dem Ordner „VHDL + System Files“. Die Zeilen des Quellcodes sind teilweise mit Kommentaren versehen.

Der Software-Quellcode ist in mehreren Ordnern verteilt. Die Ordner heißen „C Files zu Beginn des Projektes“, „Eigene, selbst erstellte C Files“ und „MicroBlaze Library C Files“.

In „C Files zu Beginn des Projektes“ befinden sich die Dateien, die von dem Entwickler des Gateway-Prototypen geschrieben wurden. Nur die Datei „gateway.c“ enthält einen Teil, der in dieser Arbeit in den bestehenden Quellcode eingefügt wurde und der für die Funktion des Gateways sehr wichtig ist. In dieser Datei wurden die Zeilen 29-67, 110/111, 147 und 159-217 hinzugefügt.

Der Ordner „Eigene, selbst erstellte C Files“ enthält neu erstellten Quellcode. „MicroBlaze Library C Files“ beinhaltet Dateien, die von der XPS Umgebung aus den Systemdaten erstellt wurden. Diese Dateien gehören in einem gewissen Sinne auch zu den selbst erstellten Dateien.