VHDL-Entwurf eines I²C-Interfaces und Emulation mit einem Mikroprozessorkern auf FPGA

Daniel Bau, Prof. Dr. Dirk Jansen, ASIC Design Center
Fachhochschule-Offenburg, Badstrasse 24, D-77652 Offenburg
Tel. 0781/205-274

Ziel dieses Projektes ist es, unter Verwendung von VHDL ein Modul zum Betrieb des I²C-Busses zu realisieren. In Zusammenspiel mit dem Mikrokontrollersystem FHOP soll die I²C-Schnittstelle ansteuerbar sein und seriell Daten mit einem EEPROM austauschen.

1. Einleitung

Der I²C-Bus (Inter-Integrated-Circuit-Bus) ist eine Entwicklung der Firma Philips für den seriellen Datenaustausch zwischen einem Master und Slaves. Das I²C-Busprotokoll ist im Vergleich mit meheren anderen Bussystemen einfach verständlich und leicht anwendbar. Die Realisierung des Datenverkehrs zwischen zwei miteinander kommunizierenden Schnittstellen über das I²C-Bussystems, wurde bisher per reiner Software-Implementierung über Assembler bewältigt. Um den Datentransfer zu beschleunigen, wird ein in VHDL programmierter I²C-Schnittstellen-Code entworfen.

Nach erfolgreicher Realisierung wird im späteren Verlauf ein Hardmacro der I²C-Schnittstelleneinheit für den AMI 0.35µm Prozess entwickelt.
Mit Hilfe des Mikrokontrollersystems FHOP (First Homemade Operational Prozessor) soll der Datenaustausch über die I²C-Schnittstelle stattfinden. Der Mikrokontroller FHOP entstand im ASIC Design Center an der Fachhochschule Offenburg und findet seinen Einsatz in zahlreichen Anwendungen. Das Kontrollersystem besteht aus folgenden Komponenten: Prozessor FHOP, 8kByte großer Arbeitsspeicher, 4kByte großes ROM, serielle und parallele Schnittstelle, Chipselect-Einheit, Interruptcontroller, Watchdog, Timer und ein Acoustic Human Interface. Über die sich im ROM befindlichen BIOS-Routinen des Prozessors FHOP wird die I²C-Schnittstelle angesprochen, welche über zwei Busleitungen (SCL, SDA) seriell Daten mit Slaves bzw. I²C-Komponenten austauscht. Die Hauptkomponente soll hier ein serielles EEPROM sein. Bei einem Bootvorgang des Prozessors FHOP wird über den I²C-Bus das EEPROM angesprochen, die benötigten Anwendungsprogramme in den Arbeitsspeicher des FHOP's übertragen und dort ausgeführt. Abbildung 1-0 zeigt die grundsätzliche Master-Slave Kommunikation über das I²C-Interface.

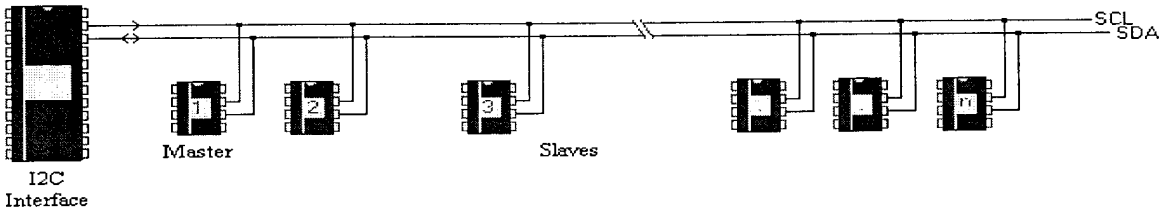

Abb. 1-0:
Master/Slave Implementierung über das I²C-Interface

2. I²C Grundlagen

Jedes I²C-Bauelement verfügt über eine eigene Adresse und kann in Abhängigkeit seiner Funktion als Sender (Transmitter) oder Empfänger (Receiver) arbeiten. Das I²C-Bussystem benutzt eine serielle Datenleitung (SDA, Serial-Data-Line) und eine serielle Taktleitung (SCL, Serial-Clock-Line) für den Datenaustausch. Beide Busleitungen sind bidirektional, was bedeutet, dass das I²C-Interface nicht nur als Treiber fungiert sondern auch von externen I²C-Komponenten getrieben werden kann.
Jedes I²C-Bauelement, das seine Kommunikation über die beiden Busleitungen aufnehmen will, muss über Open-Collector bzw. Open-Drain Ausgänge verfügen. Beide Busleitungen müssen über externe Pull-Up-Widerstände auf die positive Versorgungsspannung geschaltet sein.

Somit ergibt sich eine Wired-OR-Verknüpfung aller Busteilnehmer, welche in Abbildung 2-0 dargestellt ist.
Findet keine Busaktivität statt, sind beide I²C-Bus Leitungen in hochohmigem Zustand.
Mit der OR-Verknüpfung über die SCL-Leitung findet eine synchronisierte Kombination der einzelnen Taktsignale statt, wodurch man unterschiedliche Taktraten am Bus verwenden kann. I²C-Komponenten können als Master, wobei mehrere Master über die Funktion der Bus-arbitration möglich sind, oder als Slaves in das Bussystem eingesetzt werden.

Mögliche Datenübertragungsraten von :
- 100 kbit/s im „Normal" Betrieb
- 400 kbit/s im „Fast" Betrieb
- 3,5Mbit/s im „High Speed" Betrieb

Für den High Speed Betrieb werden spezielle IO´s benötigt.

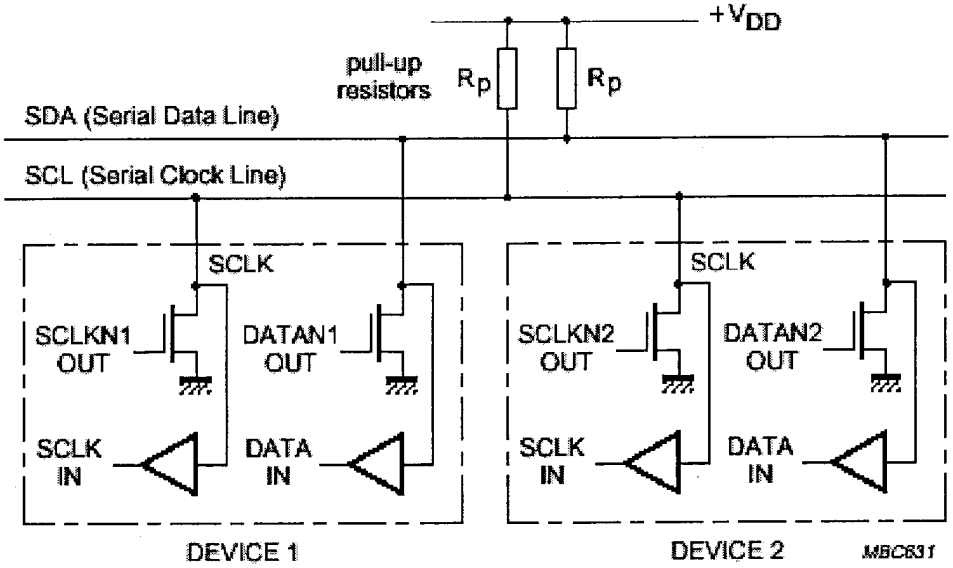

Abb. 2-0: Wired-OR-Verknüpfung der Busteilnehmer

2.1. Das I²C-Bus Protokoll

Die Datenübertragung zwischen einem Master und einem Slave erfolgt synchron zum SCL-Takt auf der SDA Leitung. Dabei wird byteweise übertragen, wobei jedes Datenbyte 8 Bit beinhaltet. Mit jedem Taktimpuls wird ein Bit übertragen, wobei das MSB zuerst auf die Datenleitung gegeben wird. Nach jedem übertragenem Datenbyte wird das Acknowledge-Bit abgefragt. Das Acknowledge-Bit teilt dem Empfänger mit, ob das übertragene Byte angekommen ist.
Jedes Bit wird während der High-Periode des SCL-Takt gespeichert. Der Zustand der Datenleitung darf nur während einer Low-Periode des SCL-Takts wechseln und muss, während sich SCL auf „High" befindet, stabil gehalten werden. Ein Zustandswechsel von SDA, während SCL „high" ist, wird als ein Start- oder Stopp-Kommando interpretiert.

Eine normale Standard Kommunikation besteht aus 4 Schritten:

1) Start-Signal generieren,
2) Slave-Adresse übertragen,
3) Daten übertragen,
4) Stopp-Signal generieren.

2.2. Dateiformat

Zusammenfassend ist hier in Abbildung 2-2 das Datentelegramm dargestellt. Eine Kombination aus Lese- und Schreibzugriffen ist ebenfalls möglich. Nach dem Stopp Signal folgt unmittelbar ein zweites Datentelegramm, welches ebenfalls dem Aufbau aus Abbildung 2-2 entspricht. Bei I²C-EEPROM´s ist zu beachten das zwischen einem Stopp und einem erneuten Start eine Ruhezeit von ca.10 ms einzuhalten ist, ansonsten sendet das EEPROM bei der Übertragung der Slave-Adresse kein
Acknowledge.

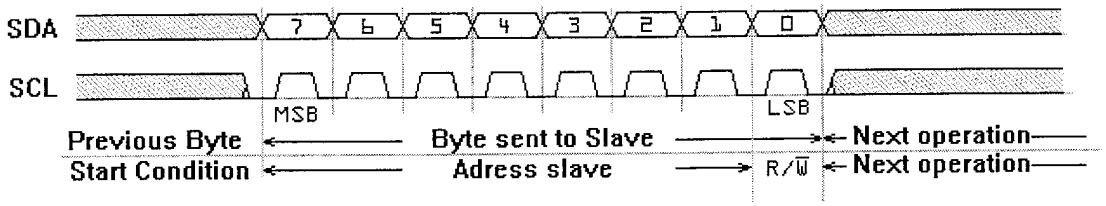

Abb. 2-1: Datenübertragung auf dem Bus

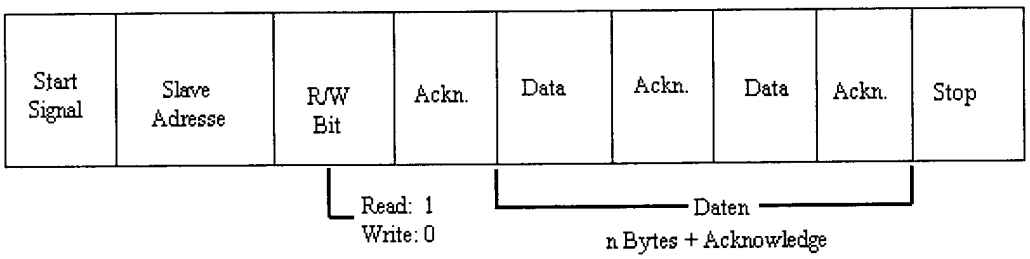

Abb. 2-2: Datenformat für Lese- und Schreiboperationen

3. Realisierung des I²C-Bus

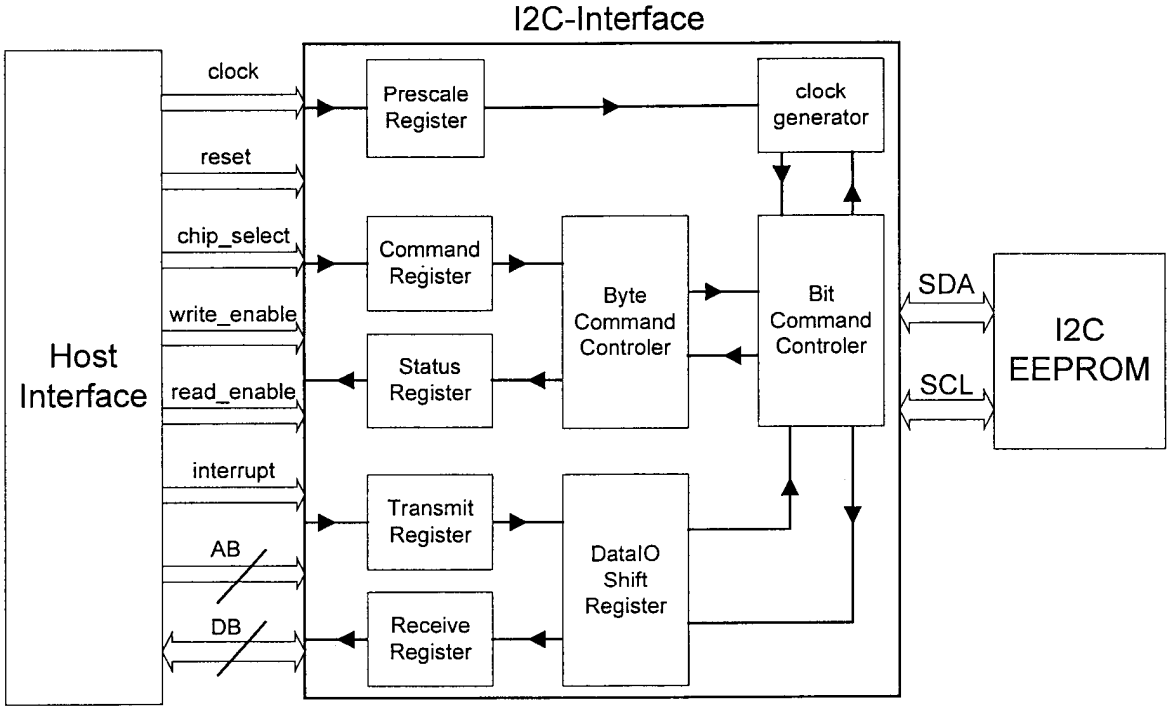

Abb. 3-0: Architektur des I2C-VHDL-Code

Das in Abbildung 3-0 gezeigte I²C-Modul ist in 4 primäre Blöcke untergeteilt:
- Takt Teiler
- Byte Command Controller
- Bit Command Controller
- Shift Register

Die anderen Blöcke beschreiben die Steuer-Register, um temporäre Werte abzuspeichern, welche über das Host Interface gesendet werden. Als Host-Interface wurde das Mikroprozessor-system FHOP verwendet, welches bereits als fertige Komponente vorlag.

Zum Austesten der I²C-Funktionalität wurde ein serielles I²C-EEPROM verwendet, welches über die beiden Busleitungen des I²C-Buses Daten aus-tauscht. Um die Bidirektionalität der Busleitungen zu realisieren, werden die Datenleitung SDA und die Taktleitung SCL über Tristatetreiber angesteuert.

3.1. Beschreibung der Funktionsblöcke

Takt Teiler:
Der Takt Teiler (clock generator) generiert aus einem 5-fach höherem Takt ein internes Enable-Signal, welches alle synchronen Elemente im Bit Command Controller triggert. Über das Prescale Register wird der Takt Teiler gesteuert und wird somit zur Einstellung der SCL-Takt Geschwindigkeit benutzt.

DataIO Shift Register:
Über das Shift Register werden die empfangenen bzw. gesendeten Bits je nach Übertragungsmodus eingelesen und in dem entsprechendem Register abgelegt. So werden z.B. während eines Lese-Vorgangs die Daten vom Datenbus SDA bitweise dem Shift Register übergeben. Nachdem das komplette Byte gelesen ist, wird der Inhalt des Shift Registers in das Empfangsregister kopiert.

Byte Command Controller:
Über den Byte Command Controller wird der „I²C-Verkehr" byteweise geregelt. Über das Setzen entsprechender Bits im Command Register wird der Byte Controller gesteuert, welcher je nach gesetztem Bit eine Byte-Sequenz überträgt.
Als Beispiel wird im Command Register das Start-, Write und Stopp Bit gesetzt. Der Byte Command Kontroller generiert nun ein Start Signal, überträgt ein Byte zum Slave und generiert das Stopp Signal. Dabei wird jedes Byte in seperate Bits umgesetzt, welche dem Bit Command Controller zur Übertragung übergeben werden.

Bit Command Controller:
Der Bit Command Controller schaltet die aktuellen Daten, das Start-Signal, das Signal des wiederholten Starts und das Stopp-Signal auf die beiden Busleitungen. Der Byte Controller teilt dem Bit Controller mit, welche Operation er durchführen soll. Um z.B. ein Byte einzulesen, erhält der Bit Controller acht separate Kommandos.

4. Simulation und Synthese

Der I²C -Bus wurde mit der Software „ModelSim SE 5.5f" der Firma Model Technology Inc programmiert und simuliert.
Damit die Simulation unter realen Bedingungen erfolgt, wurde das Mikrokontrollersystem FHOP und ein I²C-EEPROM stets mitsimuliert. Der Prozessor FHOP wird im I²C-Busverkehr als Master eingesetzt und enthält die BIOS Routinen, um das I²C-Interface anzusteuern.

Die BIOS-Routinen werden in Assembler programmiert und in das ROM an die Adresse „F000h" geschrieben. Das compilierte Hex-File der BIOS-Routinen wird dann in den VHDL-Code miteingebunden. Der Simulationsausschnitt in Abbildung 4-0 zeigt den Datenverkehr, zwischen einem EEPROM und dem Prozessor FHOP, auf den beiden I²C-Busleitungen SDA und SCL.
Kommunikationsablauf:
- Generierung des Start-Signal (rote Makierung)
- Übertragung der Slave-Adresse (gelbe Makierung)
- 1 Byte Datenübertragung aus dem Arbeitsspeicher
  des Prozessors FHOP ins EEPROM (weiße Makierung)
- Generierung des Stopp-Signal um die Kommunikation zu beenden (rosa Makierung)

Die blauen Makierungen zeigen die Empfangsbestätigung des EEPROM´s (Acknowledge).

Nach erfolgreicher Simulation wurde der VHDL-Code mit dem Synthese-Programm „Leonardo Spectrum Level 3" synthetisiert. Die Syntheseanweisungen werden in einer Script-Datei festgelegt. Im Syntheseprogramm wird dann das erstellte TCL-Script aufgerufen, welches die Anweisungen automatisch abarbeitet.
Platzieren und Routen der mit Leonardo synthetisierten VHDL-Files wird mit dem Programm „Quartus II" durchgeführt. Mithilfe des TCL-Scripts erfolgt der Programmaufruf und die Durchführung des Platzierens und Routens automatisch. Nachdem Quartus das Platzieren und Routen erfolgreich durchgeführt hat, erhält man als Ergebnis ein sogenanntes sof-File. Über das sof-File lässt sich das Layout der synthetisierten VHDL-Files auf den FPGA aufspielen.

Als Zieltechnologie wurde hier der FPGA-Chip der Firma Altera Cyclone „EP1C12" verwendet.

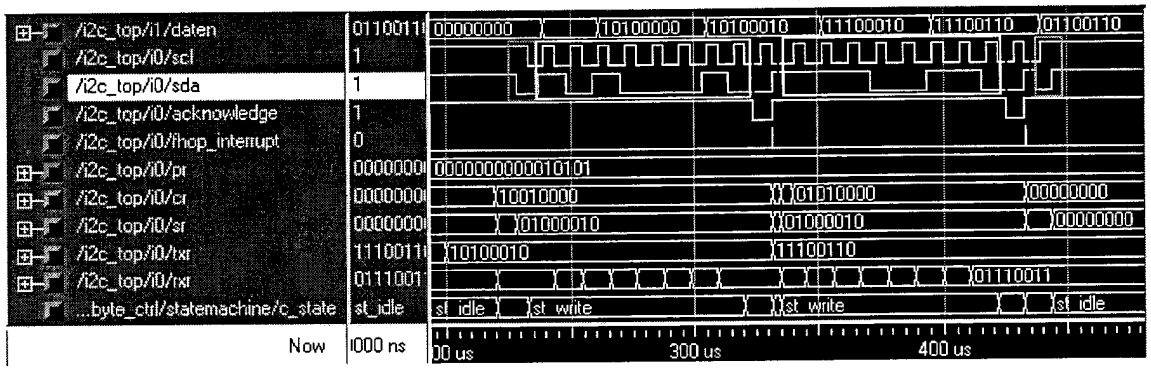

Abb.4-0:
Simulationsausschnitt: Sendet ein Byte an das EEPROM

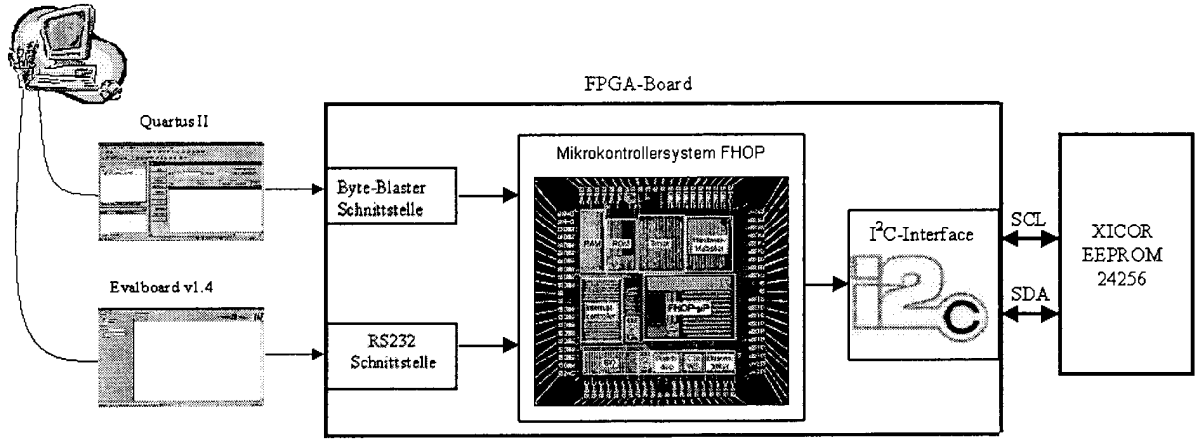

Abb.5-0: Testaufbau

5. Funktionsnachweiß

Abbildung 5-0 zeigt den Testaufbau des FPGA-Boards. Das synthetisierte Design des Mikrokontrollersystem FHOP und des I²C-Interface wurde mit dem Programm „Quartus II" auf den FPGA mit Hilfe des „Byte-Blasters" programmiert. Der FPGA wird über die PC-Schnittstelle (RS232) angesteuert. Dabei werden die zur Ansteuerung des I²C-Interface benötigten BIOS-Routinen über die Monitor-Software „Evalboard" in das RAM des Prozessors FHOP geschrieben und dort ausgeführt.Mit Hilfe der I²C-Bios-Routinen ist es nun möglich, einzelne Bytes von einem EEPROM zu lesen und zu schreiben oder ganze Programmseiten (256 Byte) vom EEPROM in den Programmspeicher des FHOP´s zu kopieren.
Beim Testaufbau wurde das von XICOR entwickelte EEPROM „24256" verwendet, welches sich nicht auf dem FPGA-Board befindet und extern dazuge-schaltet wurde. Die 5kΩ Pull-Up-Widerstände, welche die Taktleitung SCL und Datenleitung SDA benötigen, befinden sich auf dem externen XICOR-EEPROM. Die beiden Busleitungen zum EEPROM hin, wurden mit einem Osziloskop aufgezeichnet und dokumentiert. Der Oziloskopausdruck in Abbildung 5-1 zeigt, wie ein Byte ins EEPROM geschrieben wird. Dabei stellt das untere Signal die Datenleitung SDA dar, das obere Signal die Taktleitung SCL.

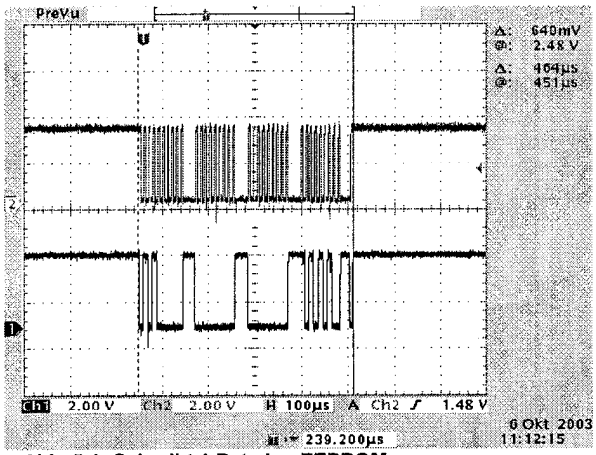

Abb. 5-1: Schreibt 1 Byte ins EEPROM (rotes Signal = SCL , blaues Signal = SDA)

Folgendes Zeitverhalten wurde bei der Datenübertragung mit einem XICOR 24256 EEPROM bei einer Übertragungs-Taktgeschwindigkeit von 100kHz gemessen:

Beschreibung	Übertragene Bytes	Durchlaufzeit
Speichert ein Byte im EEPROM ab	4	464 µs
Liest ein Byte vom EEPROM ein	5	574 µs
ladet eine Page vom EEPROM	260	26,7 ms
speichert eine Page im EEPROM	280	53,6 ms

6. Zusammenfassung

In dieser Arbeit wurde ein I²C-Interface für ein bestehendes Mikrokontrollersystem FHOP in der Hardwarebeschreibungssprache VHDL realisiert Dabei wurde die Spezifikation des von der Firma Philips patentierten I²C-Busses realisiert. Die Takt-Geschwindigkeit des I²C-Taktsignals SCL ist variabel über ein Prescale-Register einstellbar. Dabei wurden Datenübertragungsraten von bis zu 400kbit/s erzielt. Beim VHDL-Entwurf wurde ein streng synchrones Schaltungsdesign verfolgt. Als Master auf dem I²C-Bus wurde der an der Fachhochschule Offenburg entwickelte Prozessor FHOP eingesetzt. Mithilfe der in Assembler erstellten I²C-BIOS-Routinen aus dem ROM des Prozessors wird das I²C-Interface gesteuert. Es wurden einzelne Bytes oder ganze Pages in beide Richtungen erfolgreich übertragen. Bei 100kHzTakt- frequenz ergibt beim Auslesen einer Page von 256 Byte aus dem EEPROM eine Übertragungszeit von 26,7 msec. Das Schreiben einer Page ins EEPROM benötigt wegen der internen Programmvorgänge doppelt so lang (53,6 msec).
Das komplette Design, bestehend aus dem Mikrokontrollersystem FHOP und I²C-Interface, wurde auf dem von Altera entwickelten FPGA Cyclone „EP1C12" getestet. Dabei stand ein FPGA-Board zur Verfügung, welches zum Testen eine RS232 Schnittstellen beinhaltet. Mithilfe der Monitor-Software „Evalboard", wurde vom PC aus mit dem Mikrokontrollersystem FHOP und dem I²C-Interface kommuniziert.
Zur Funktionsprüfung des I²C-Interface wurde dabei das RAM des Prozessors mit den I²C-BIOS-Routinen beschrieben und diese dort ausgeführt. Dabei wurde der Datenverkehr auf dem I²C-Bus mit einem Osziloskop ausgemessen und protokolliert.

Die I²C-Busschnittstelle wird zum Booten des Prozessors, zum Ablegen von Programmen und Daten im EEPROM verwendet und kann zur Ansteuerung weiterer I²C-Bus-Komponenten wie A/D–Wandler effektiv eingesetzt werden.

7. Literaturverzeichnis

[1] Handbuch der Electronic Design Automation Herausgegeben von Dirk Jansen

[2] Netzliste vom FHOP-DESIGN-KIT V2.1

[3] Philips Semiconductor „The I²C-Bus Specification"

[4] Bussysteme : Parallele und serielle Bussysteme, lokale Netze, Herausgegeben von Georg Färber und verfasst von Bernd Wiemann

[5] http://www.esacademy.com/faq/i2c I²C-Info-Homepage