Redesign eines USB- Controllers in VHDL und Emulation mit einem Mikroprozessorkern auf FPGA

Artur Kurz, Prof. Dr. Dirk Jansen
FH-Offenburg, Badstrasse 24, 77652 Offenburg

Das Ziel dieses Projektes war, eine USB-Schnittstelle für das Mikrocontrollersystem FHOP zu realisieren. Der FHOP wurde an der Fachhochschule Offenburg entwickelt und wird in verschiedenen Anwendungen eingesetzt. Das USB-Modul ist in der VHDL-Sprache zu programmieren. Später soll es zusammen mit dem Mikroprozessor FHOP synthetisiert und auf einem FPGA-Board getestet werden. Für den Funktionsnachweis soll zusätzlich die Firmware des Mikrocontrollers um mehrere USB-Routinen erweitert sowie eine Applikation auf der PC-Seite in Delphi erstellt werden.

1. Einleitung

Vor etwa zehn Jahren wurde im ASIC Design Center an der Fachhochschule Offenburg der Mikroprozessor FHOP (First Homemade Operational Prozessor) entwickelt. Seitdem wurde der Prozessor in mehreren Studien- und Diplomarbeiten zu einem umfangreichen Mikrocontrollersystem weiterentwickelt, das in Anwendungen verschiedenster Art eingesetzt wird. Im aktuellen Stand besteht das System aus folgenden Komponenten: Serielle und parallele Schnittstellen, Chipselect-Einheit, Interrupt-Controller, Watchdog, Timer, Acoustic Human Interface. Für die Kommunikation mit der Außenwelt sind neben der seriellen RS232-Schnittstelle auch andere Schnittstellen realisiert worden. Dazu zählt PSK (Phase Shift Keying) sowie RFID (Radio Frequency Identification), wobei sich RFID noch in der Entwicklungsphase befindet.

Für die Kommunikation mit dem PC wird standardmäßig die serielle RS232-Schnittstelle benutzt. Sie ist nach mehreren Jahrzehnten gut dokumentiert und sehr einfach zu bedienen. Die RS232-Schnittstelle hat aber auch ihre Schwächen: Zum einen kann man ihr nur umständlich und sehr begrenzt Energie entnehmen, um die Applikationsschaltung damit zu versorgen, und zum anderen muss das Gerät vor dem Booten des PCs angeschlossen sein, also ist es keine Plug & Play-Anwendung realisierbar.

Diese Nachteile kennt eine USB-Schnittstelle nicht. Darüber hinaus hat Universal Serial Bus noch viele weitere Vorteile wie z. B. leichte Anwendbarkeit, schnelle und zuverlässige Datenübertragung, Flexibilität, geringe Kosten und niedriger Stromverbrauch.

Aus diesem Grund hat man sich entschieden, für den FHOP ein USB-Modul zu entwickeln, das später in das System integriert werden kann. In der Abbildung 1-1 ist das Blockbild vom FHOP mit der zugehörigen Peripherie dargestellt, so wie es später im ASIC realisiert werden kann.

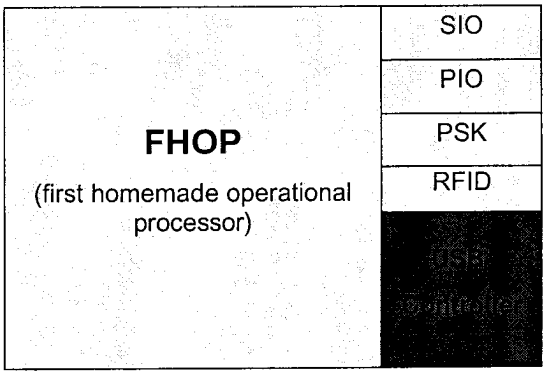

Abbildung 1-1: Blockbild für die Realisierung im ASIC

2. USB Grundlagen

Der USB ist ein serielles Bus-System (Universal Serial Bus), an das mehrere Geräte angeschlossen und adressiert werden können. USB ist wesentlich komplexer als RS232 und mit bis zu 1,5 MBit pro Sekunde (Low Speed - Geräte) oder 12 MBit pro Sekunde (Full Speed - Geräte) auch viel schneller. Dazu kommt, dass man bei der Entwicklung des USB großen Wert auf Einfachheit und Sicherheit der Bedienung gelegt hat. Erst mit dem USB hat Plug & Play seine eigentliche Bedeutung eingeholt. Denn ein USB-Gerät wird im Normalfall einfach nur

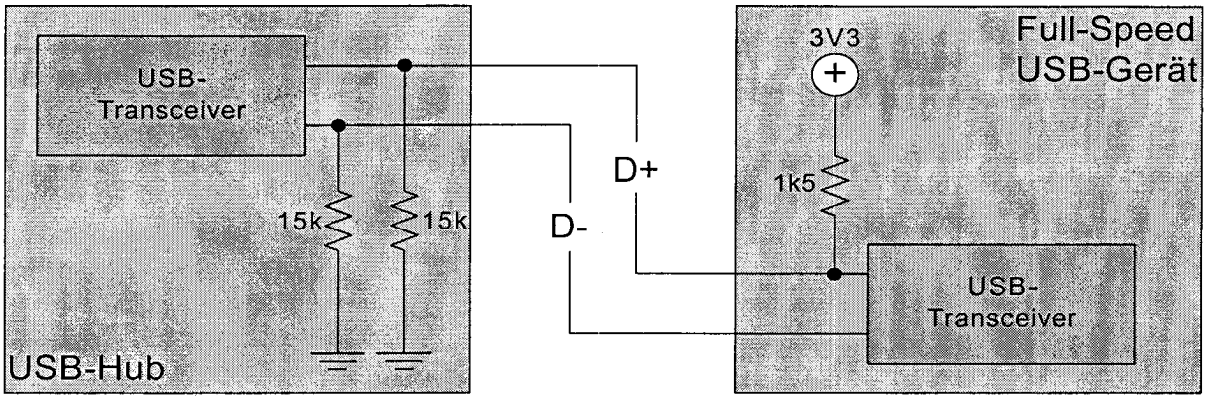

Abbildung 2-0: Gerätetyp Erkennung (Full-Speed)

bei abgeschaltetem oder laufendem PC angeschlossen und initialisiert sich dann automatisch. Das lästige Problem mit doppelt belegten Interrupts, falschen Adressen und fehlenden Treibern ist damit gelöst. Die Vereinfachung für den Anwender bedeutet aber auch einen wesentlich größeren Aufwand für den Entwickler.

USB Geräte werden mittels eines vieradrigen Kabels miteinander verbunden. Das Kabel hat folgende Belegung:

PIN #	1	2	3	4
Bezeichn.	+5V	Data –	Data +	GND

Tabelle 2-1

Der Anschluss stellt eine Betriebsspannung von 5 V zur Verfügung, die ohne weiteres bis zu 100 mA belastet werden kann. Bei einer besonderen Konfiguration kann das Gerät bis zu 500 mA entnehmen. Die Signale auf den beiden Leitungen D+ und D- sind Differenzsignale mit Spannungspegeln 0 V und 3,3 V.

Der USB ist ein Bus mit nur einem Master, alle Aktivitäten gehen vom PC aus. Daten werden in Paketen von 8 bis zu 256 Bytes versandt und empfangen. Der PC kann Daten von einem Gerät anfordern. Umgekehrt kann kein Gerät von sich aus Daten absenden. Der gesamte Datenverkehr hat einen Rahmen von exakt einer Millisekunde. Innerhalb eines Rahmens können nacheinander Datenpakete für mehrere Geräte verarbeitet werden. Dabei können Low-Speed- und Full-Speed- Pakete zusammen vorkommen. Wenn mehrere Geräte angesprochen werden, sorgt ein Bus-Verteiler (Hub) für die Verteilung.

Low-Speed-Geräte arbeiten mit einer Datenrate von 1,5 Mb/s, ein Bit ist also 666,7 ns lang. Full-Speed-Verbindungen verwenden 12 Mb/s entsprechend 83,33 ns. Die Geschwindigkeit wird allein vom Master vorgegeben. Die Slaves müssen sich auf den Datenstrom synchronisieren. Da kein getrenntes Taktsignal übertragen wird, muss der Takt aus dem Datensignal gewonnen werden. Man verwendet dazu das NRZI-Verfahren (Non Return to Zero Inverted). Daten-Nullen führen hierbei zu einem Pegelwechsel, Einsen lassen den Pegel unverändert. Zusätzlich wird beim USB Bit-stuffing benutzt. Bei diesem Verfahren wird nach sechs aufeinander folgenden Einsen eine Null eingefügt. Damit wird maximale Zeitdauer ohne einen Signalwechsel auf sechs Takte begrenzt.

Bei einem nicht verwendeten USB-Anschluss sind beide Signalleitungen low und haben einen Innenwiderstand von 15 kO. Jedes USB-Peripheriegerät besitzt einen Widerstand von 1,5 kO, der eine der beiden Signalleitungen mit +3,3 V verbindet. Bei einem Full-Speed-Gerät wird D+ mit 3,3 V verbunden (Abbildung 2-0), bei einem Low-Speed-Gerät D-. Der PC-Host erkennt daran den Typ des Geräts und kann eine Datenverbindung mit der entsprechenden Übertragungsrate aufbauen. Bei dem Enummerieren[1] eines Geräts wird automatisch der passende Treiber geladen. Umgekehrt erkennt das Betriebssystem auch, wenn ein Gerät vom Bus getrennt wird. In diesem Fall wird der Treiber wieder aus dem Speicher entfernt.

[1] Enumeration: Konfigurationsvorgang nach der Herstellung der Verbindung

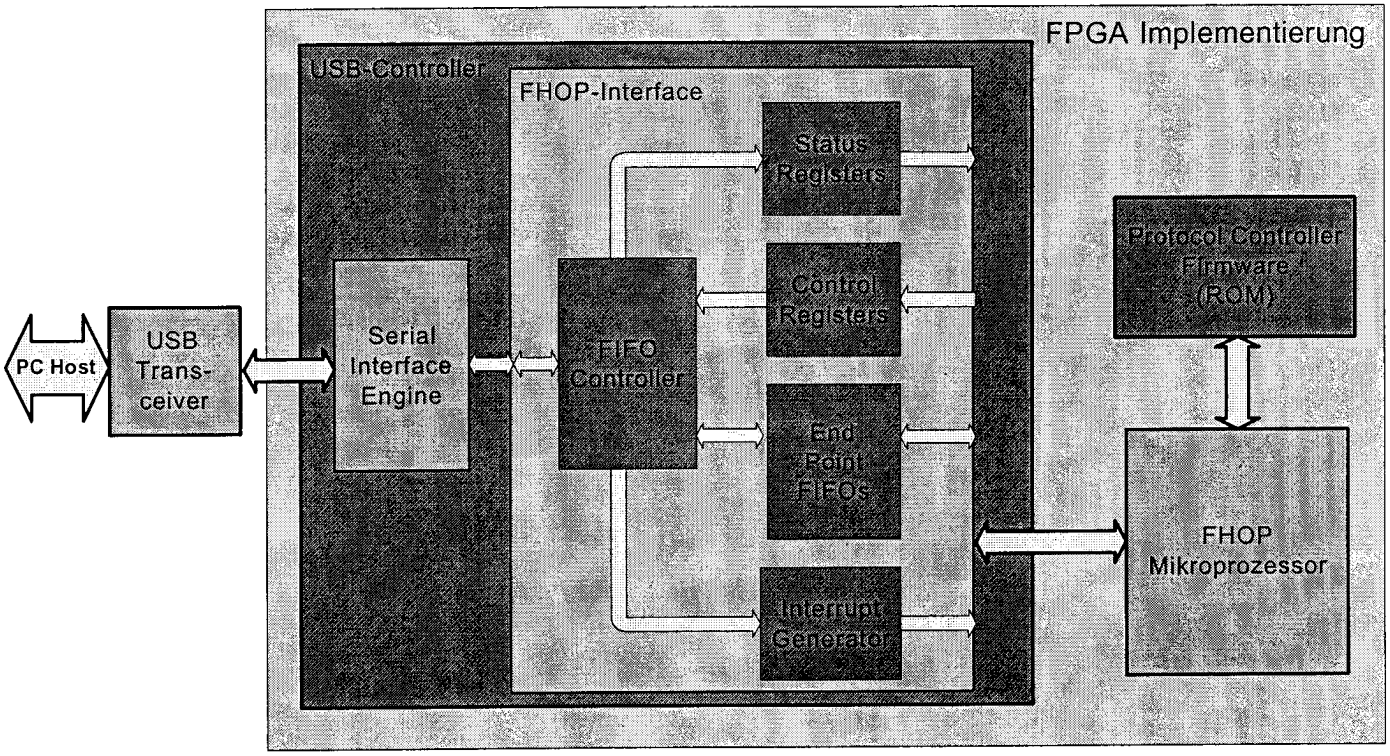

Abbildung 3-0: System-Übersicht

3. Realisierung des USB-Moduls

Eine Vorarbeit für dieses Projekt wurde von Dipl.-Ing. Gopal Pudasaini im Rahmen einer Master Thesis geleistet. Das Arbeitsergebnis war der VHDL-Code für den USB-Controller. Der Code besteht aus zwei Hauptblöcken:

- Serial Interface Engine
- FHOP-Interface.

Beide Teile wurden einzeln erstellt und simuliert. Die Simulation des gesamten Systems zusammen mit dem Mikroprozessor wurde damals aus Zeitgründen nicht mehr durchgeführt. Deshalb musste zuerst dieser Schritt ausgeführt werden. Die ersten Simulationsergebnisse des gesamten Systems haben gezeigt, dass die FHOP Interface Unit komplett überarbeitet werden musste. Serial Interface Engine war fehlerfrei und wurde nicht mehr geändert.

In der Abbildung 3-0 ist das gesamte System in einem Blockschaltbild dargestellt. Die grün dargestellten Funktionsblöcke (USB-Transceiver, SIE und FHOP Mikroprozessor) waren fertige Komponenten und mussten nicht mehr modifiziert werden.

Über ein USB-Kabel ist der PC mit dem USB-Transceiver verbunden, der die elektrischen Anforderungen auf dem Bus erledigt. Der Transceiver ist mit einem externen Baustein MAX3340E realisiert. Alle anderen Komponenten sind in VHDL-Sprache formuliert worden und können somit alle zusammen nach der Synthese in einem FPGA-Chip emuliert werden.

3.1. Serial Interface Engine (SIE)

SIE ist einer der wichtigsten Funktionsblöcke in diesem Design. Dieser wurde freundlicherweise von der Firma Intel im VHDL-Format zur Verfügung gestellt. Die SIE erledigt alle paketbezogenen Aufgaben. Im Folgenden sind die wichtigsten Aufgaben der Serial Interface Engine aufgelistet:

- Paket Erkennung,
- Detektierung und Generierung von Start of Paket (SOP), End of Paket (EOP), und Reset Signalen,
- Synchronisierung des Takts,
- NRZI-Daten Codierung/Decodierung,
- CRC-Generierung/Prüfung,
- Paket-ID (PID) Generierung/Prüfung,

- Umwandlung seriell in parallel und umgekehrt.

Die Funktion der SIE ist abhängig von der Datenrichtung. Es werden serielle Daten vom Bus empfangen, decodiert und mittels CRC auf die Gültigkeit überprüft. Die gültigen Daten werden byteweise an die FHOP Interface Unit weitergeleitet. In umgekehrter Richtung muss SIE vom FHOP empfangene Bytes in Pakete packen, CRC16 bzw. CRC5 generieren und als seriellen Bitstrom im USB-Format (NRZI) auf dem Bus senden.

3.2. FHOP Interface Unit

Sowohl die SIE als auch der FHOP Prozessor besitzen einen 8 Bit breiten Bus. Die FHOP Interface Unit wird als Systemschnittstelle zwischen den beiden Komponenten eingesetzt und muss damit die Übertragungsspezifikation der beiden Seiten unterstützen. Die FHOP Interface Unit besteht intern aus folgenden Funktionsblöcken:

- Address Decoder,
- Status Register,
- Control Register,
- USB Address Register,
- FIFO-Controller,
- FIFOs (4x),
- Interrupt Generator.

Der Address Decoder hat als Eingang eine Chip-Select Leitung (CS_N) und vier Adressleitungen (A0 bis A3), die vom FHOP Prozessor kommen. Er hat die Aufgabe, die Adresse verschiedener Register und FIFOs eindeutig zu decodieren. Über die Status Register werden die Statussignale vom USB-Modul an den Prozessor weitergeleitet.

Die Control Register werden zur Steuerung des USB-Controllers benutzt. Hier kann der FHOP festlegen, wie sich das USB-Modul verhalten soll, wenn das nächste USB-Paket empfangen wird.

Im USB-Address Register wird die Geräteadresse gespeichert. Diese wird jedem Gerät nach dem Anschließen an den Bus vom PC zugewiesen.

Zur Zwischenspeicherung der Daten werden vier FIFOs benutzt. Zwei FIFOs sind für Control-Transfer zuständig und die anderen zwei für den Datentransfer. Sobald das Modul ein USB-Paket erfolgreich empfängt und ins FIFO speichert, wird vom Interrupt Generator ein Interrupt ausgelöst. Dieser signalisiert dem Prozessor, dass die Daten zur Weiterverarbeitung bereit stehen.

Alle diese Register, FIFOs und Interrupt Generator werden vom FIFO-Controller gesteuert. Dieser Block ist als eine Statemachine realisiert worden. Jedes Paket auf dem Bus wird von der SIE empfangen und an den FIFO-Controller weitergeleitet. Der FIFO-Controller hat die Aufgabe, zu überprüfen, ob das Paket für das Gerät bestimmt ist. Das geschieht, indem er die Adresse im Token Paket mit der Adresse im USB Address Register vergleicht. Stimmen beide Adressen überein, werden die Daten in das betreffende FIFO gespeichert. Anderenfalls wird das aktuelle Paket ignoriert. Hiermit wird der Mikroprozessor entlastet, denn er braucht nicht mehr zu überprüfen, ob das Paket gültig und für ihn bestimmt ist.

3.3. Protocol Controller

Der Protocol Controller ist in der Software realisiert worden und muss alle protokollspezifischen Aufgaben erledigen. Die Software ist in Assembler geschrieben und besteht aus drei Interrupt Routinen und USB-Descriptor Daten. Je nach Art des empfangenen Pakets (Control, Daten IN / OUT) löst die FHOP Interface Unit den entsprechenden Hardware Interrupt aus. Die Endpoint-0 Interrupt Routine ist für den Control Transfer zuständig und erledigt alle im USB spezifizierten Standard Requests. Sie ist zuständig für das richtige Enummerieren des Geräts beim Anschließen an den Bus.

Die beiden anderen Routinen steuern den Daten Transfer (Endpoint-1 IN- und Endpoint-2 OUT-Daten). Damit das Gerät richtig konfiguriert werden kann, fragt das Betriebssystem von der neu angeschlossenen Hardware bestimmte Informationen in Form von Descriptoren ab. Dabei handelt es sich um genau definierte Datenblöcke von wenigen Bytes, die im ROM abgelegt sind.

Für das vollständige Beschreiben des Geräts gibt es fünf Arten von Descriptoren:

- Device-,
- Configuration-,
- Interface-,
- Endpoint- und
- String- Descriptor.

Die Control Requests vom Host werden analysiert, um die Art der Anfrage zu erkennen. Der Mikroprozessor liest die entsprechenden Daten aus dem ROM und schreibt sie ins Ausgabe-FIFO. Von diesem werden sie an den Host zurückgesandt.

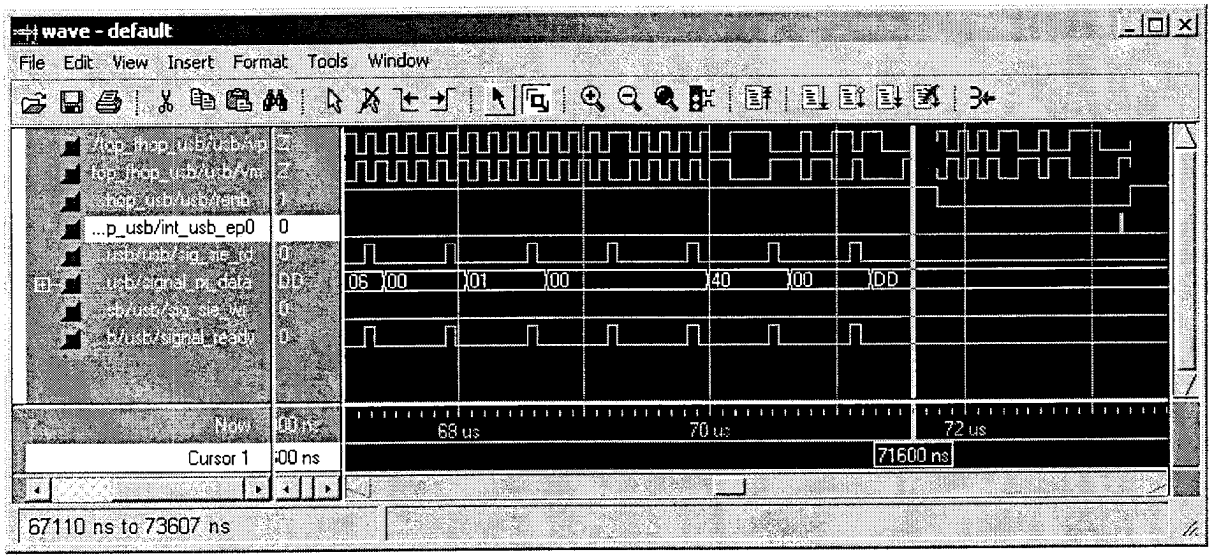

Abbildung 4-0: Ausschnitt aus der Simulation

4. Simulation und Synthese

Zum Simulieren des VHDL-Programms wurde die Software „ModelSim SE 5.7c" der Firma Model Technology Incorporated benutzt. Im ersten Schritt der Simulation wurden die Funktionsblöcke einzeln auf die korrekte Funktion getestet. Dabei wurden das Auslesen und Beschreiben der Register bzw. FIFOs simuliert. An dieser Stelle war es sehr wichtig, die Spezifikation des Zugriffs auf den FHOP-Datenbus streng einzuhalten. Denn die Fehler beim Buszugriff würden fatale Folgen verursachen.

Im nächsten Schritt wurde das komplette USB-Modul simuliert. Dazu wurden die USB-Pakete, so wie sie auf dem Bus benutzt werden, konstruiert. Die daraus entstandene serielle Bitfolge muss noch nach der USB-Spezifikation in NRZI codiert werden. Mittels einer Makro-Datei wird das Signal dann am Eingang des USB-Controllers simuliert. Hiermit kann man diverse Varianten des Datentransfers durchspielen und die Reaktion des Moduls am Ausgang überprüfen. Damit wurde unter anderem getestet, ob die Daten richtig decodiert werden, in das betreffende Endpunkt-FIFO geschrieben werden und der Interrupt an den Mikroprozessor ausgelöst wird. Die Abbildung 4-0 zeigt einen kleinen Ausschnitt aus der Simulation. Man sieht hier zunächst die beiden differenziellen Signale „vp" und „vm", die vom USB-Kabel kommen. Das Signal „signal_rx_data" ist der Ausgang von Serial Interface Engine (ein 8-Bit breiter Bus). Hier sieht man die Datenbytes in der Form, wie sie in das FIFO gespeichert werden. Zusätzlich ist hier dargestellt, wie das empfangene Paket (vor der Zeitmarkierung) durch ein ACK-Paket bestätigt wird. Nachdem das Paket erfolgreich übertragen wurde, wird ein Interrupt auf der „int_usb_ep0" ausgelöst.

Zuletzt wurde der Mikroprozessor FHOP mit der zugehörigen Peripherie in die Simulation eingebunden. Um diese Simulation zu ermöglichen, wurden auch die USB-Treiber-Routinen für den FHOP geschrieben. Die Treiber bestehen aus drei Interrupt-Routinen und übernehmen die Weiterverarbeitung der Daten auf der höheren Protokollschicht. Der kompilierte Assembler Code wird in das für Simulationsprogramme verständliche Format (.mif -Datei) konvertiert und in die Simulation eingebunden. Damit wurde das komplette Design getestet so wie es später auf FPGA emuliert wird.

Nach der erfolgreichen Simulation wurde der VHDL-Code mit dem Programm „Leonardo Spectrum" synthetisiert. Als Zieltechnologie wurde hier der FPGA-Chip der Firma Altera EP20K1500EBC652-3 eingesetzt. Mit 1,5 Mio. Gatter ist der Chip einer der größten dieser Art. Der Chip wurde zusammen mit einem Entwicklungsboard der Firma El Camino erworben. Das Board hat die Größe eines DIN A4 Blatts und beinhaltet alle notwendigen Komponenten für eine komfortable Entwicklung. In der nachfolgenden Tabelle 4-1 sind die Ergebnisse der Synthese zusammengestellt.

Device Utilization for EP20K1500EBC652			
Resource	Used	Avail	Utilization

IOs	30	488	6.15%
LCs	4220	51840	8.14%
Memory Bits	12800	442368	2.89%

Tabelle 4-1: Synthese Ergebnis

1,5 Mio. Gatter entsprechen ca. 50000 Altera spezifischen Logik Zellen. Davon sind 8,14% für dieses Design genutzt worden.

5. Funktionsnachweis

Beim Anschließen eines USB-Geräts an den Bus startet der Enummerierungsvorgang, bei dem der PC notwendige Daten von der neu angeschlossenen Hardware anfordert. Nach der erfolgreichen Enummerierung wird das Gerät im Geräte-Manager eingetragen. Hiermit kann der erste Funktionsnachweis durchgeführt werden. Abbildung 5-1 zeigt den Eintrag des FHOP-Controllers im Geräte-Manager nachdem er erfolgreich konfiguriert wurde. Hier sieht man auch, dass das Gerät als USB-HID (Human Interface Device) konfiguriert wurde. Das ermöglicht die Benutzung von Standard Windows Treibern bei der Applikationsprogrammierung.

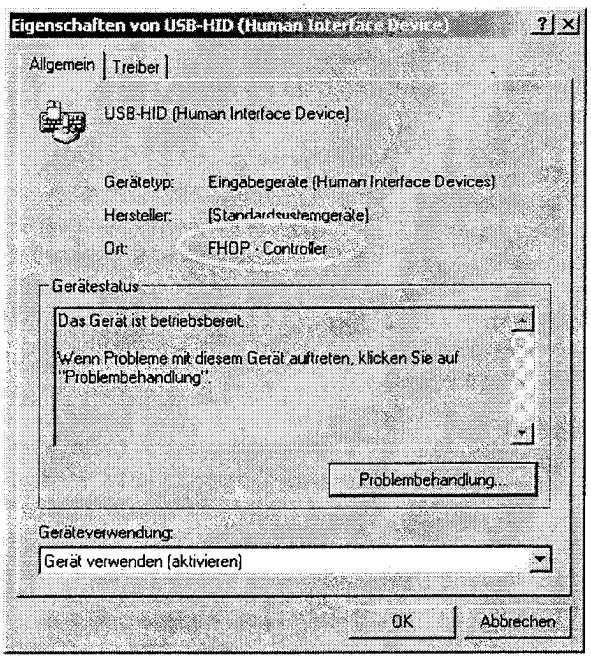

Abbildung 5-1: Eintrag im Geräte-Manager

Für den vollständigen Funktionsnachweis wurde auch eine Applikation programmiert. Die Applikation wurde in Delphi erstellt und realisiert eine Datenübertragung in beide Richtungen (PC → Gerät und Gerät → PC). Mittels 8 Schaltern, die an den parallelen Port des FHOPs angeschlossen sind, wird ein Byte definiert. Das Byte wird in periodischen Abständen (10 ms) über die USB Datenpipe an den PC übertragen und auf dem Monitor visuell dargestellt. Ähnlich funktioniert es auch in die andere Richtung. Mittels 8 Buttons in der Applikation kann man die an den FHOP angeschlossenen LEDs steuern.

Damit man die HID-Kommunikation in Delphi realisieren kann, muss ein Package zusätzlich installiert werden. Das Package beinhaltet „Interface for Microsoft Human Input Device (HID) API" und wurde von Robert Marquardt im Rahmen des Jedi Projekts erstellt. Das Package ist als Freeware auf der Homepage http://www.delphi-jedi.org verfügbar.

6. Zusammenfassung

In dieser Arbeit wurde eine USB-Schnittstelle für ein bestehendes Mikrocontroller System FHOP realisiert. Im aktuellen Stand funktioniert das Design zuverlässig in Low Speed Konfiguration. Im Full Speed gibt es noch einige Schwierigkeiten, denn die Kommunikation bricht nach einigen Paket-Transfers zusammen. Durch das Emulieren des Designs auf FPGA wurde die Funktion nachgewiesen. Die nächste Aufgabe wird sein, die Hardware zu optimieren, damit das USB-Modul auch im Full Speed zuverlässig funktioniert. Zusätzlich wird die Software auf der PC Seite optimiert, um höhere Übertragungsraten zu erzielen.

7. Literatur

[1] John Hyde, „USB Design by Example", A Practical Guide to Building I/O Device

[2] Jan Axelson, „USB Complete", Everything You Need to Develop Custom USB Peripherals

[3] Universal Serial Bus Specification, Revision 1.1 http://www.usb.org/developers/docs/