Linux auf FPGAs

Inhaltsverzeichnis

1 GNU/Linux in eingebetteten Systemen

1.1 Einsatzort

Einsatzort

Gerät	**Beispiel**
Router	Linksys WRT54G
Navigationssysteme	TomTom
Access Points	FON2100
Mobiltelefone	Motorola RAZR2 V8
VoIP Telefone	Siemens Gigaset C455 ip

GNU/Linux wird in verschiedenen Bereichen betrieben ohne dass wir uns dies zwingend[1] bewusst sind.

1.2 Vorteile

Vorteile von Linux

- Herstellerunabhängig
- Einfach zum portieren
- Treiberunterstützung[2]
- effiziente Softwareentwicklung[3]
- skalierbar

[1] Microsoft verwendet einem Gerücht nach Access Points mit GNU/Linux in Redmond.
[2] Linux ist der Kernel mit der grössten Treiberunterstützung.
[3] Software kann auf Desktopcomputer mit einer anderen Architektur entwickelt werden.

1.3 Distribution

Freie Metadistributionen

Eine Auswahl:

- μClinux
- Gentoo Embedded
- OpenEmbedded

Es gibt eine Vielzahl proprietärer Hersteller im Embedded-Linux Bereich. Da diese mit proprietären Patches und Werkzeugen arbeiten, besteht wie bei der Verwendung von proprietären Betriebssystemen die Gefahr des *Vendor lock-in*.

Falls Linux auf FPGAs installiert werden soll, sind μClinux oder Petalinux wegen den geringen Anforderungen (keine Speicherverwaltungseinheit (MMU) nötig), der guten Dokumentation und der einfachen Übernahme der Hardware-Definition interessant. Leider ist μClinux veraltet und Petalinux proprietär. Für die Weiterentwicklung von Gecko3 wird eine angepasste OpenEmbedded-Metadistribution verwendet.

2 Einführung Programmable Logic Devices

Was sind Programmable Logic Devices?

Definition 1 (Die Idee)**.** Die Hardware passend für die Anwendung zu programmieren und nicht Software auf einer fixen Hardware zu programmieren. So etwas wie ein *General Purpose Chip*. [4]

Diese Idee und die Technik dazu ist nicht neu. Die erste Variante PLDs, genannt PLA (Programmable Logic Array), kam schon Mitte der 70er auf den Markt. Auch die FPGAs, um die es sich hier dreht und die mittlerweile sehr populär sind, stammen aus den 80ern.

Entwicklungsgeschichte Überblick

1970 ROMs optimiert als Look-up-table

1973 FPLA von Signetics

1978 PAL

198? CPLD

1984 FPGA

1998 Erster 1 Mio. Gatter FPGA

2.1 Was macht diese Chips so attraktiv?

Praktischer Vergleich der verschiedenen PLD Typen

- PAL typischerweise als Adressdecoder.
- CPLD z.B. als Bus Koppler, Speicher Kontroller oder Antriebsregler.
- FPGA: Video Encoder, Software Defined Radio, Neuronale Netzwerke, Ein-Chip-Systeme, etc

Die PLDs die einem Entwickler heute zur Verfügung stehen, erlauben es, komplette digitale Systeme in einem einzigen Chip zu realiseren. Die Chipkosten fallen bei kleinen bis mittleren Stückzahlen nicht ins Gewicht.

Altera CPLDs.

Family	*Max7000 (B, AE, S)*	*MAX3000 (A)*	*MAX II (G)*
Macrocells/ LUTs	32–512 macrocells	32–512 macrocells	240–2,210 LUTs (192–1,700 equiv. macrocells)
System gates	600–10,000	600–10,000	
I/O pins	32–512	34–208	80–272
Max. internal clock freq.	303 MHz	227 MHz	304 MHz (I/O limited)
Supply voltage	2.5 V (B), 3.3 V (AE), 5 V (S)	3.3 V	1.8 V (G), 2.5 V, 3.3 V
Interconnects	EEPROM	EEPROM	Flash + SRAM
Static current	9 mA–450 mA	9 mA–150 mA	2 mA–50 mA
Technology	0.22 u CMOS EEPROM 4-layer metal (7000 B)	0.3 u, 4-layer metal	0.18 u, 6-layer metal

Abbildung 1: Vergleichstabelle für CPLDs von Altera [4]

Xilinx CPLDs.

Family	*XC9500 (XV, XL, –)*	*CoolRunner XPLA3*	*CoolRunner II*
Macrocells	36–288	32–512	32–512
System gates	800–6,400	750–12,000	750–12,000
I/O pins	34–192	36–260	33–270
Max. internal clock frequency	222 MHz	213 MHz	385 MHz
Building block	GAL 54V18 (XV, XL) GAL 36V18 (–)	PLA block	PLA block
Supply voltage	2.5 V (XV), 3.3 V (XL), 5 V	3.3 V	1.8 V
Interconnects	Flash	EEPROM	
Technology	0.35 u CMOS	0.35 u CMOS	0.18 u CMOS
Static current	11–500 mA	<0.1 mA	22 uA–1 mA

Abbildung 2: Vergleichstabelle für CPLDs von Xilinx [4]

Actel FPGAs.

Family	*Accelerator*	*ProASIC*	*MX*	*SX*	*eX*
Logic modules	2,016–32,256	5,376–56,320	295–2,438	768–6,036	192–768
System gates	125 k–2 M	75 k–1 M	3 k–54 k	12 k–108 k	3 k–12 k
I/O pins	168–684	204–712	57–202	130–360	84–132
Flip-flops	1,344–21,504	5,376–26,880	147–1,822	512–4,024	128–512
Max. internal frequency	500 MHz	250 MHz	250 MHz	350 MHz	350 MHz
Supply voltage	1.5 V	2.5 V, 3.3 V	3.3 V, 5 V	2.5 V, 3.3 V, 5 V	2.5 V, 3.3 V, 5 V
Interconnects	Antifuse	Flash	Antifuse	Antifuse	Antifuse
Technology	0.15 u 7-layer metal CMOS	0.22 u 4-layer metal CMOS	0.45 um 3-layer metal CMOS	0.22 u CMOS	0.22 u CMOS
SRAM bits	29 k–339 k	14 k–198 k	2.56 k	n.a.	n.a.

Abbildung 3: Vergleichstabelle für FPGAs von Actel [4]

Xilinx FPGAs.

Family	*Virtex II Pro (X)*	*Virtex II*	*Virtex E*	*Virtex*	*Spartan 3*	*Spartan IIE*	*Spartan II*
Logic blocks (CLBs)	352–11,024	64–11,648	384–16,224	384–6,144	192–8,320	384–3,456	96–1,176
Logic cells	3,168–125,136	576–104,882	1,728–73,008	1,728–27,648	1,728–74,880	1,728–15,552	432–5,292
System gates		40 k–8 M	72 k–4 M	58 k–1.1 M	50 k–5 M	23 k–600 k	15 k–200 k
I/O pins	204–1,200	88–1108	176–804	180–512	124–784	182–514	86–284
Flip-flops	2,816–88,192	512–93,184	1,392–64,896	1,392–24,576	1,536–66,560	1,536–13,824	384–4,704
Max. internal frequency	547 MHz	420 MHz	240 MHz	200 MHz	326 MHz	200 MHz	200 MHz
Supply voltage	1.5 V	1.5 V	1.8 V	2.5 V	1.2 V	1.8 V	2.5 V
Interconnects	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
Technology	0.13 u 9-layer copper CMOS	0.15 u 8-layer metal CMOS	0.18 u 6-layer metal CMOS	0.22 u 5-layer metal CMOS	0.09 u 8-layer metal CMOS		
SRAM bits (Block RAM)	216 k–8 M	72 k–3 M	64 k–832 k	32 k–128 k	72 k–1.8 M	32 k–288 k	16 k–56 k

Abbildung 4: Vergleichstabelle für FPGAs von Xilinx [4]

Attraktiv weil...

- Design Wiederverwendung (IP Cores)
- Kleine Entwicklungsrisiken verglichen mit ASIC
- Sehr gut test- und verifizierbar
- An zukünftige Anforderungen anpassbar

Wie auch in der Software Entwicklung üblich, wird ein Design in Teil Systeme und Funktionsblöcke unterteilt. Diese einzelnen Blöcke sind weiterverwendbar in anderen Projekten. Solche Blöcke (genannt Cores) können eingekauft/verkauft werden oder als Opensource (Opencores) veröffentlicht werden. Ein reichhaltiger Satz solcher *IP*[4] *Cores* sind verfügbar und erlauben eine rasche Entwicklung kompletter System-on-Chips.

Heute kosten kleine CPLDs unter 2$ und ein FPGA mit 1 Mio. Gattern unter 30$.

Die Simulations- und Verifikationstools sind die gleichen wie in der ASIC Entwicklung. Dies stellt das funktional korrekte Verhalten eines Systems sicher. Dazu kommen FPGA spezifische Möglichkeiten, die eine sehr schnelle flexible Fehlersuche erlauben. Zu diesen Möglichkeiten gehören Logic-Analyser die mit in den FPGA integriert werden und Messungen an jedem beliebiegen internen Signal erlauben.

Für komplexe und teure Systeme kann es von grossem Vorteil sein, wenn die Hardware flexibel, reprogrammierbar und anpassbar ist, da dadurch eine massiv grössere Einsatzdauer der Systeme erreicht werden kann. Ein grosser Vorteil für alle Kunden, da ihre Investitionen so länger geschützt werden. Ein gutes Beispiel sind Mobilfunktstationen, die einfach per *Softwareupdate* auf GPRS oder EDGE aufgerüstet werden.

2.2 Einsatzbereich FPGA

Einsatzbereich FPGA

Typische Produktbereiche sind:

- Telekommunikation
- RADAR, SONAR
- Bild-/Videoverarbeitung
- High Performance Computing
- Kryptographie
- Luft und Raumfahrt

2.3 Wann lohnt sich der Einsatz von FPGAs?

Dem Entwickler stehen diverse Lösungswege offen, typischerweise als reine Softwarelösung (CPU), starkt optimierte Softwarelösung auf spezialisierter Hardware (DSP, Cell, GPU), reine Hardwarelösung mit flexibler Hardware (FPGA), Hardwarelösung mit spezialisierter Hardware (ASIC).
Eine reine Softwarelösung ist ca. doppelt so schnell entwickelt wie eine reine Hardwarelösung. Diese hat jedoch eine ca. zehnmal höhere Ausführungsgeschwindigkeit. Ebenso ist ein solches System in der Massenherstellung wesentlich günstiger und energieeffizienter.

Der Vorteil von Hardware ist die echte Parallelverarbeitung. Wird mehr Systemleistung gefordert, kann die Hardware skaliert werden z.B. durch massive Parallelisierung, noch längeren Pipelines oder höhere Verarbeitungsbusbreiten. Dies zeigt auch schon, dass Hardwarelösungen bei strukurierten grossen Mengen von Daten und parallelisierbaren Algorithmen ihre grössten Stärken ausspielen.

3 freie Software - freie Hardware

- FPGAs ermöglichen freie Hardware zu erstellen.
- FPGAs sind aber unfrei.

[4] IP steht für *Geistiges Eigentum* (Englisch: intellectual property). Achtung: Dieser Begriff ist irreführend, da dadurch nicht definiert ist, welche Rechte gemeint sind.

3.1 Datenbusse

Datenbusse

Bus	Organisation	Lizenz
Wishbone	OpenCores	Public Domain
Coreconnect	IBM	Proprietär, frei nutzbar
AMBA	ARM	Proprietär, frei nutzbar

Für eine genauere Gegenüberstellung der verschiedenen Busse siehe [6].

3.2 Prozessoren

Prozessoren

Prozessor	Bus	Lizenz
OpenRISC	Wishbone	LGPL
AEMB	Wishbone	LGPL
S1 (T1)	Wishbone	GPL
LEON	AMBA	GPL dual

Eine detailierte Übersicht von softcore Prozessoren aus dem Jahre 2007 befindet sich auf der Webseite der *University of Illinois*.

3.3 Verschmelzen von Hard- und Software

DSP, GPU, Cell	→	CPU	→	PLD
Optimierte Software		Portierbarer Code		Optimierte Hardware

Zur Beschleunigung eines Algorithmus kann die Software für eine optimierte Recheneinheit geschrieben (wenig portabel) oder den Algorithmus direkt in Hardware zu implementiert werden. [7]

Einerseits besteht die Möglichkeit, dass der Algorithmus in *dezidierter Hardware* implementiert ist und die Software allein die Steuerung übernimmt(Beispiel Videoencoder [3] [2]). Andererseits den *Befehlssatz der* CPU *zu erweitern* und mittels inline Assembler anzusprechen. Die Verwendung *Wrapper* ist es ebenfalls möglich, Funktionen der Software transparent in Hardware auszulagern (Beispielsweise Kryptoalgorithmen des Kernels).

4 Demonstration

4.1 Hardware

Als Hardware Plattform für die Demonstration verwenden wir das GECKO3main FPGA Modul.Dieses Modul ist als Basis für FPGA basierte Entwicklungen gedacht und kann dank seiner kompakten Abmessungen auch im fertigen Endgerät eingesetzt werden.

Das GECKO3 ist ein Projekt des Mikroelektronik Labors der Berner Fachhochschule (Schweiz). Für weitere Informationen steht die Projekthomepage zur Verfügung:
http://labs.ti.bfh.ch/gecko/

GECKO3 Openhardware Plattform

GECKO3main Blockdiagramm

4.2 System-on-Chip

Aufbau des Demonstrationssystems

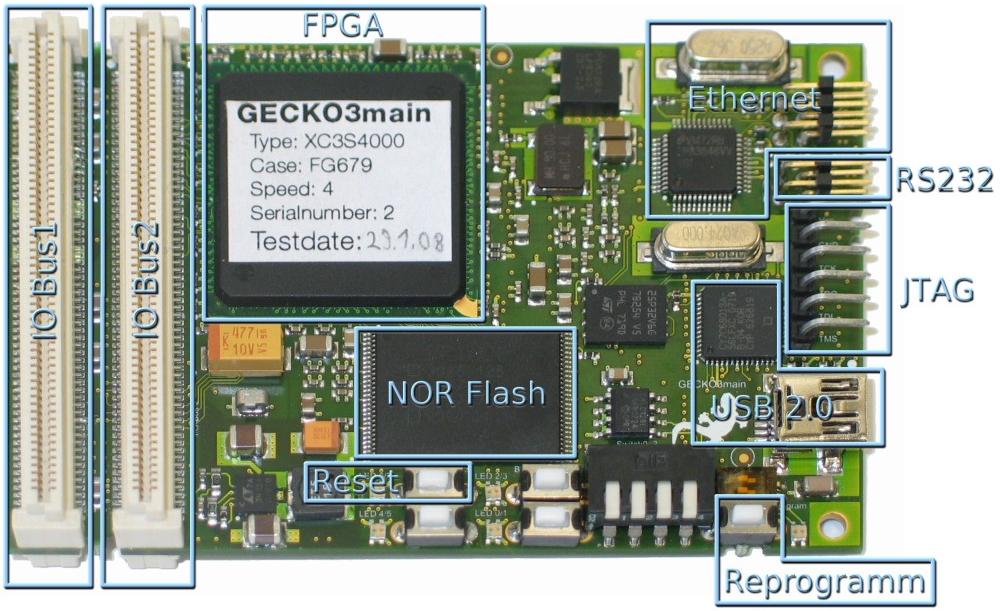


Abbildung 5: Das GECKO3main Modul

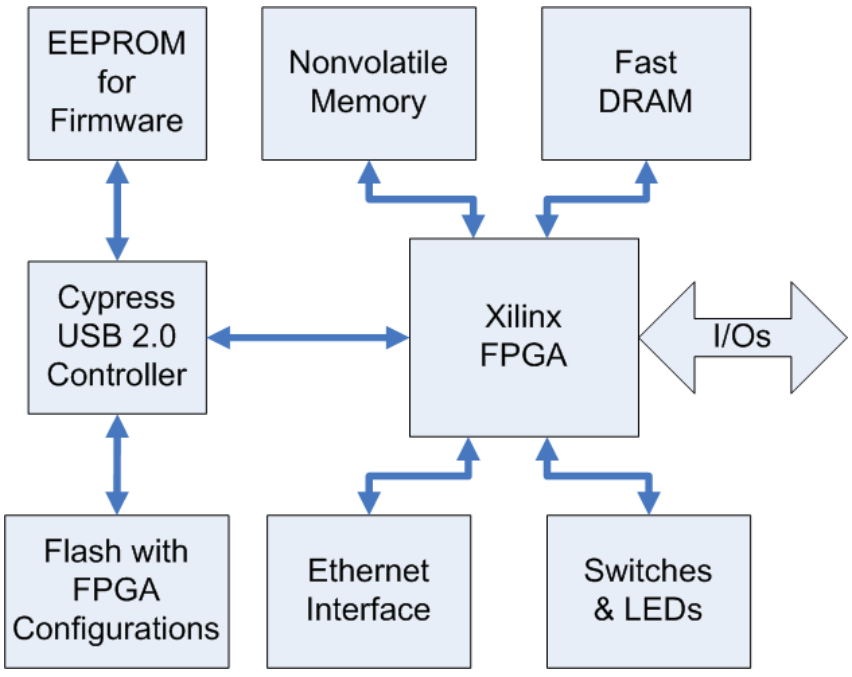


Abbildung 6: Blockdiagramm des GECKO3main Moduls

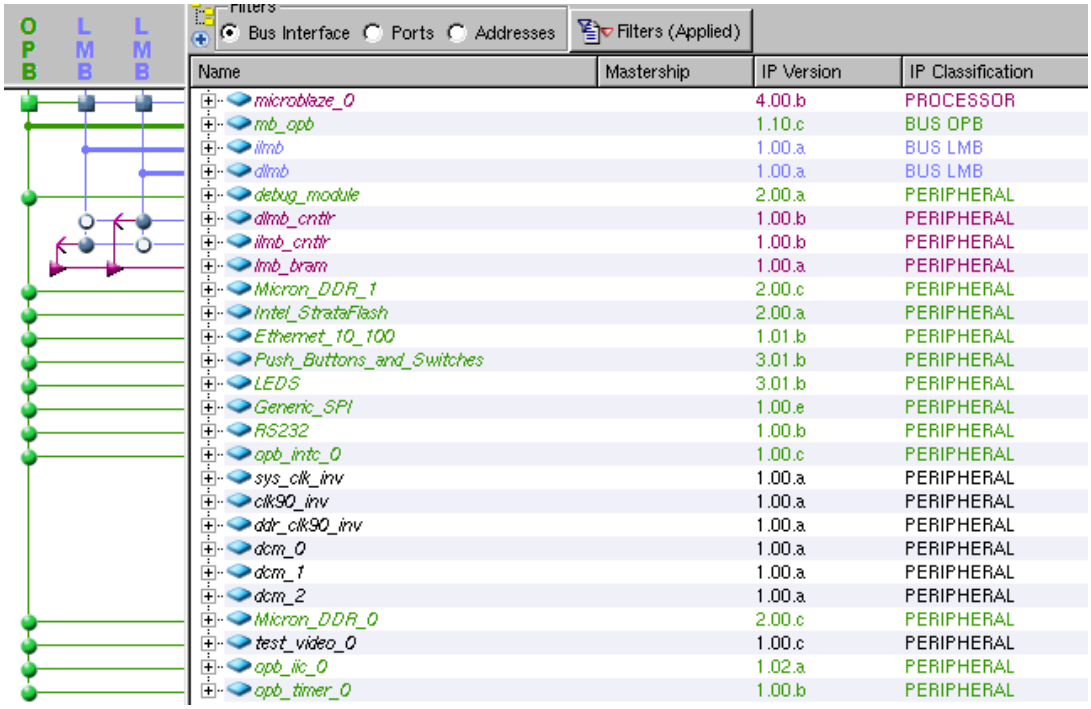
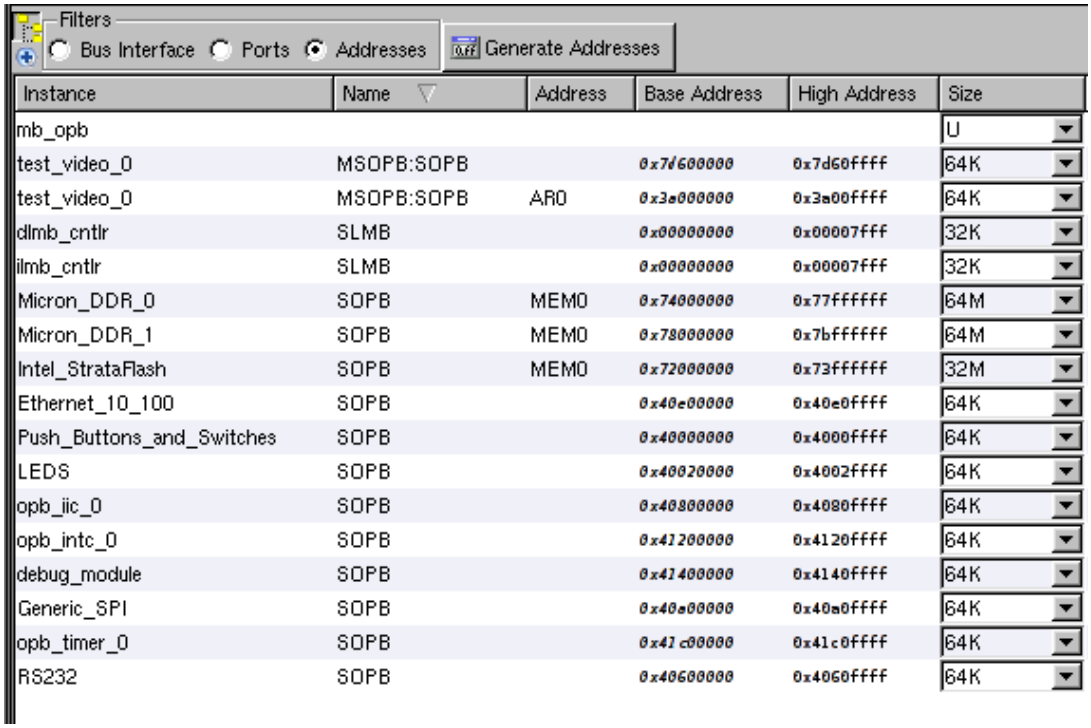


Abbildung 7: Aufbau des Demonstrationssystems.
Zu sehen sind die verwendeten Cores und die Struktur der Busse

Aufbau des Demonstrationssystems

Dieses FPGA Design wurde mit dem Xilinx EDK (Embedded Development Kit) erstellt. Es ist eine komerzielle Entwicklungsumgebung für System-on-Chip Designs.
Für die Entwicklung, Simulation und Verifikation von Cores stehen gute opensource Programme zur Verfügung. Opencollector ist eine Datenbank mit quelloffener Software für Elektronik- und Hardwareentwicklung.
Um ein System auf einem FPGA zu Implementieren gibt es keine offenen Programme! Die meisten Hersteller bieten kostenlose, in Teilen eingeschränkte, Entwicklungsumgebungen bereit.



Instance	Name	Address	Base Address	High Address	Size
mb_opb					U
test_video_0	MSOPB:SOPB		0x7d600000	0x7d60ffff	64K
test_video_0	MSOPB:SOPB	AR0	0x3a000000	0x3a00ffff	64K
dlmb_cntlr	SLMB		0x00000000	0x00007fff	32K
ilmb_cntlr	SLMB		0x00000000	0x00007fff	32K
Micron_DDR_0	SOPB	MEM0	0x74000000	0x77ffffff	64M
Micron_DDR_1	SOPB	MEM0	0x78000000	0x7bffffff	64M
Intel_StrataFlash	SOPB	MEM0	0x72000000	0x73ffffff	32M
Ethernet_10_100	SOPB		0x40e00000	0x40e0ffff	64K
Push_Buttons_and_Switches	SOPB		0x40000000	0x4000ffff	64K
LEDS	SOPB		0x40020000	0x4002ffff	64K
opb_iic_0	SOPB		0x40800000	0x4080ffff	64K
opb_intc_0	SOPB		0x41200000	0x4120ffff	64K
debug_module	SOPB		0x41400000	0x4140ffff	64K
Generic_SPI	SOPB		0x40a00000	0x40a0ffff	64K
opb_timer_0	SOPB		0x41c00000	0x41c0ffff	64K
RS232	SOPB		0x40600000	0x4060ffff	64K

Abbildung 8: Aufbau des Demonstrationssystems. Zu sehen sind die Adressbereiche der Cores

Literatur

[1] Matthias Zurbrügg, Christoph Zimmermann: *Gecko3 - New generation of the Microlab HW/SW co-design Platform*. Berne University of Applied Sciences, Dezember 2006.

[2] Marc-André Beck: *Embedded Streaming Software*. Berne University of Applied Sciences, November 2007.

[3] Marco Spirig: *Hardware Video Encoder*. Berne University of Applied Sciences, November 2007.

[4] Volnei Pedroni: *Circuit Design with VHDL*. The MIT Press, 2004.

[5] Marc Leeman, Bruno Vandevelde, Ronny Dewaele: *Development and Deployment of High Performance Linux-Controlled Streaming Video Decoder Boards*. FOSDEM, 2005.

[6] Rudolf Usselmann: *SoC Bus Review*. OpenCores, 2001.

[7] Eric Blossom, Matt Ettus: *GNU Radio & the Universal Software Radio Peripheral*. Chaos Communication Camp, 2007.