Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung

Marco Kirschke

Implementierungsansätze für ein FPGA basiertes Multiprozessorsystem

Fakultät Technik und Informatik
Department Informatik

Inhaltsverzeichnis

1 Einleitung **2**

2 FPGA basierte Systeme **3**
2.1 Anwendungsgebiete 3
2.2 System-on-Chip 4
2.3 Herstellerübersicht 5

3 Kriterien zur Entwicklung von MPSoC **6**
3.1 Anwendungsorientierter Systemaufbau 6
3.2 Ansätze für Implementierungsentwürfe 7
3.2.1 Eng gekoppelte Systeme 7
3.2.2 Lose gekoppelte Systeme 8

4 Nios II Multiprozessor System **9**
4.1 Hardwarekonfiguration 10
4.2 Software Komponenten 10

5 Zusammenfassung und Ausblick **11**

1 Einleitung

In den letzten Jahren hat die voranschreitende Entwicklung in der FPGA Technologie den Anwendungsbereich von FPGAs erweitert. Die stetig wachsende Anzahl an verfügbaren Logikblöcken, lässt die Implementierung ganzer System auf einem einzelnen FPGA zu. Dabei bietet sie den Entwicklern ein hohes Maß an Flexibilität, da zum einen Anwendungen mit hohen Echtzeitanforderungen oder Datenraten direkt durch digitale Systeme als Hardware Module implementiert werden können, und zum anderen der Einsatz von Softcore Prozessoren die Koordinierung von Datenflüssen und die Kommunikation mit externen Schnittstellen erleichtert(vgl. (Bilski u. a., 2007)). Diese Entwicklung ermöglicht zusätzlich den Einsatz von mehreren Prozessoren auf einem FPGA, wobei die einzelnen Prozessoren, im Gegensatz zur Verwendung von Mirkoprozessoren, gezielt auf bestimmte Aufgaben abgestimmt werden können.

Zudem hat sich während der letzten Jahre in der Entwicklung der Prozessortechnologie ein Wechsel vollzogen. Dabei wird zur Optimierung der Prozessorenleistung nicht mehr das alleinige Erhöhen der Taktraten vorangetrieben, sondern vielmehr der Einsatz von mehreren Prozessorkernen sowie die Verwendung von Multiprozessor Systemen priorisiert. Diese Entwicklung ist auf den vermehrten Einsatz der Prozessoren in mobilen Systemen und die damit einhergehenden Anforderungen an energieeffizienten und wärmereduzierenden Leistungsmerkmalen zurückzuführen (vgl. (Rauber und Rünger, 2007)).

Eine Verbindung dieser beiden Entwicklungen in der Erstellung von Multiprozessor System on Chip (MPSoC) bietet den Anbietern von spezialisierten Lösung im Marktsegment der *embbeded systems* ein hohes Maß an Flexibilität bei der Erstellung ihrer Produkte. Die Integration von Standardkomponenten in diese Systeme reduziert zusätzlich die Entwicklungskosten und beschleunigt die Einführungszeit der Produkte *(time to market)*; zudem steigt die Reaktionsfähigkeit auf marktbezogene Entwicklungen oder bei Änderung von Spezifikationen.

Als Zielsetzung der verschiedenen Projekt- und Anwendungsphasen im Masterstudiengang Informatik an der HAW ist eine sukzessive Einarbeitung und Vertiefung der Kenntnisse im Entwurf und der Implementierung von Multiprozessor Systemen auf FPGA Basis angestrebt. Im Bachelor Studiengang wurden bereits Kenntnisse in der Verwendung von SoC gesammelt und im Rahmen der Abschlussarbeit in einem System zur Echtzeitbildverarbeitung in einem Fahrspurerkennungssystem angewandt. Neben der technischen Implementierung soll auch der wirtschaftliche Aspekt bei der Erstellung eines MPSoC betrachtet werden. Dazu sollen die Auswirkungen von unterschiedlichen Entwicklungsentscheidungen miteinander verglichen und bewertet werden. Aus diesem Grund bietet sich zunächst die Erstellung von mehreren kleinen MPSoC an, bevor im Rahmen der Abschlussarbeit die Erfahrungen und Kenntnisse in ein umfangreicheres Anwendungsprojekt einfließen. Dazu gibt es eine Reihe von Projekten an der HAW, in deren Rahmen der Einsatz eines MPSoC geplant werden kann.

- Im Bereich des FAUST Projektes wird zur Zeit an der Entwicklung eines SoC basierten Fahrzeugs mit autonomen Funktionen gearbeitet. Der Einsatz eines MPSoC ist dabei für die Verarbeitung der verschiedenen Sensordaten und deren Auswertung zur Fahrzeugregelung geplant. Gleichzeitig kann ein solches System auch Komponenten zur Entwicklungsunterstützung, wie zum Beispiel Debuggingmodule, in das System integrieren.

- Ein weiteres Forschungsgebiet an der HAW konzentriert sich auf die Simulation von Audioquellen in einem drei dimensionalem Raum. Dazu liegen bereits Vorschungsergebnisse vor, die ein SoC zur Erzeugung von Signalquellen im 3D Raum behandeln. Um die Erzeugung von mehreren Signalquellen bereitzustellen oder ganze Szenarien simulieren zu können, ist der Einsatz eines MPSoC geplant, welches die entsprechenden Berechnungen vornimmt.

Im zweiten Kapitel dieser Ausarbeitung werden zunächst die Einsatzgebiete von SoCs behandelt. Neben einer Übersicht zu den marktführenden Herstellern, enthält dieses Kapitel eine Darstellung der grundlegenden Struktur von FPGA basierten SoCs. Das 3. Kapitel soll einen Einblick in die Rahmenbedingungen für die Implementierung von MPSoC geben und einige Entwicklungsmethoden im Kontext von Hardware-Software Codesign darstellen. Als initaler Ausgangspunkt für die kommenden Semester wurde ein MPSoC erstellt, das mit Hilfe von drei Nios II Softcore Prozessoren ein erstes Erprobungsbeispiel implementiert. Die Erstellung dieses Beispielsystems wird im vierten Kapitel vorgestellt. Eine Zusammenfassung soie einen Ausblick auf die nächsten Schritte und Entwicklungen liefert das fünfte Kapitel.

2 FPGA basierte Systeme

2.1 Anwendungsgebiete

Im Marktsegment für elektronische Geräte mit mittleren oder kleinen Stückzahlen werden FPGA basierte Systeme eingesetzt, da sie im Gegensatz zu auf ASIC[1] Technologie basierenden Lösungen im Bereich der Entwicklungs- und Herstellungskosten rentabler sind (vgl. (Rowen, 2005)). Die kürzere Entwicklungszeit bietet zudem eine wesentlich schnellere Marktreife der Produkte. Zusätzlich kann auf neue Standards und Erweiterungen von Spezifikationen durch die Rekonfigurierbarkeit der FPGAs direkt reagiert werden. FPGAs werden in einer Vielzahl von Bereichen eingesetzt; zum Beispiel zur Implementierung digitaler Filter, Signalverarbeitungsanwendungen, kryptografischen Operationen oder der Verarbeitung von Video- und Audiostreams. Der Einsatz hardwarespezifischer Abbildungen von Algorithmen auf RTL[2]-Strukturen ermöglicht die parallele Verarbeitung einer Vielzahl von Additionen und

[1] ASIC - Application Specific Integrated Circuit
[2] RTL - Register Transfer Level

Multiplikationen, die zudem über Pipelining Verfahren optimiert werden können. Diese Vorteile begünstigen den Einsatz in speziellen Systemen, die mit hohen Datenraten sowie zeitkritischen Anforderungen umgehen müssen. Der traditionelle Ansatz zur Implementierung von eingebetteten Systemen ist die Verwendung von Mikroprozessoren, da diese erstens kostengünstig zu beziehen sind und es zweitens eine Vielzahl an Entwicklern gibt, die mit der Softwareentwicklung vertraut sind. Ein Nachteil dieses Ansatzes ist, dass die Architekturen der meisten Mirkoprozessoren, wie zum Beispiel beim *ARM*, *PowerPC* oder *x86* an herkömmliche General-Purpose Prozessoren für Desktop PCs angelehnt sind und sie somit auf die Bearbeitung von allgemeinen Anwendungen ausgelegt sind (vgl. (Rowen, 2005)).

2.2 System-on-Chip

Der Entwicklungsprozess für SoC auf FPGA Basis wird durch die Tatsache erschwert, dass für den Entwurf[3] der Einsatz von speziell ausgebildeten Entwicklern erforderlich ist; diese müssen Kenntnisse in Hardwarebeschreibungssprachen, wie VHDL oder Verilog, haben, den Umgang mit den speziellen Synthese und Place&Route Tools beherrschen und zusätzlich mit dem Entwurf digitaler Systeme vertraut sein. Die steigende Anzahl von Logikblöcken in den FPGAs ermöglicht jedoch, die Implementierung von mehreren eingebetteten Microcontrollerkonfigurationen mit Beschleunigungskomponenten auf einem einzigen FPGA, welche CPUs, Speicherblöcke, Bussysteme und Peripheriekomponenten beinhalten. Ein solches System-on-Chip ermöglicht ein Hardware-Software Codesign und weist neben den Vorteilen der hardwarespezifischen RTL-Logik auch die Vorteile der Softwareentwicklung auf, da auf den Softcore Prozessoren nicht nur C/C++ Programme, sondern auch angepasste Betriebssysteme laufen können. Dabei werden die Komponenten, die IPs[4], zur Implementierung eines solchen Systems auch von den Herstellern der FPGAs bereitgestellt und können durch die Entwicklungsumgebungen in das entsprechende System integriert werden. Zusätzliche Peripherieanbindungen, Speicherzugriffsmodule, optimierte Bussysteme oder Kommunikationsschnittstellen werden als Bausteine in eine Systemkonfiguration integriert. Dadurch umfasst der Aufgabenbereich des Entwicklers neben der Konfiguration und dem Systemdesign auch die Erstellung von IPs, die zudem mit Matlab oder Kodegeneratoren entworfen werden können. Zusätzlich erleichtert der Einsatz von standardisierten Softwaretechniken den Entwicklungsprozess.

[3] Bei der Programmierung eines FPGAs handelt es sich vielmehr um eine Konfiguration, die aus dem erstellten VHDL/Verilog Code durch das Synthese Werkzeug in RTL-Logik erstellt wird.

[4] IP - Intellectual Property: Werden entweder vom Hersteller einer Komponente bereitgestellt oder können selbst durch RTL-Strukturen erstellt werden

2.3 Herstellerübersicht

Neben den beiden Marktführern Xilinix und Altera, die sich fast 90% des FPGA Marktes teilen, gibt es eine Reihe von kleineren Anbietern, deren FPGAs auf bestimmte Anwendungsbereiche spezialisiert sind. Als Beispiele seien hier die low cost/power FPGAs der Firmen *Lattice* und *Actel* genannt. Im Produktrahmen von *Lattice* (Lattice, 2009) befindet sich zudem die ispXPGA Familie, die mit Hilfe einer Flash basierten Architektur den Einsatz von zusätzlichen Speichermodulen zur Konfiguartion des Systems vermeidet. *Actel* FPGAs werden vermehrt in der Raumfahrttechnik eingesetzt, da diese eine besondere Resistenz gegenüber Strahlungen aufweisen (Actel, 2009). *Xilinx* und *Altera* hingegen bieten ähnliche Produktfamilien, die sowohl auf low cost/power als auch auf high performance Anwendungen abgezielt sind; Abbildung 1 zeigt die Produktpaletten der beiden Marktführer (vgl. (Xilinx, 2009) und (Altera, 2009)).

Produktpalette	Xilinx	Altera
High Performance FPGA		
	VIRTEX Family	Stratix Series
Low Cost/Power FPGA		
	SPARTAN Family	Cyclone Series
Synthese/ Place&Route		
	ISE Design Suite	Quartus
System on Chip		
Hardware *Software*	Embedded Dev. Kit Software Dev. Kit	SOPC Builder/Quartus NIOS IDE
Processor Cores ***Softcore Processors***	PPC 440, PPC405, **MicroBlaze**	C68000, ARM Cortex, **NIOS2**

Abbildung 1: Übersicht zu den Produktpaletten von *Xilinx* und *Altera*

Neben der Erforschung, Entwicklung und Herstellung von FPGAs bieten die Anbieter eine umfangreiche Produktpalette zur Erstellung, Verifikation und Optimierung von FPGA basierten Systemen. Darunter fallen die Synthesetools, die aus dem in einer Hardwarebeschreibungssprache erstellten Quellcode, unter Berücksichtigung der bausteinspezifischen Ressourcen, eine Netzliste erstellen. Diese beinhaltet die Funktion der Anwendung in Form einer digitalen Schaltung. Auch Entwicklungsumgebungen zur Erstellung von SoC werden von den Herstellern angeboten. Dazu stehen eine Reihe von IPs zur Verfügung, die eine Integration von Komponenten in das System ermöglichen; das können zum einen Hardware Beschleuniger sein, die auf die Verarbeitung von z.B. Video- oder Audiostreams optimiert sind, oder zum anderen die Schnittstellen für den Zugriff auf Hardwaremodule, wie Speicher, bereitstellen. Das Hardware-Software Codesign wird mit Hilfe spezieller Werkzeuge voneinander getrennt, so dass der Entwickler zunächst eine Hardware Konfiguration anlegt, um die Struktur des Microcontrollers auf dem FPGA festzulegen. In einem nächsten Schritt wird dann die anwendungsspezifische Software für das erzeugte Hardwareprofil erstellt. Somit

besteht die Möglichkeit, die im Hardware Design angelegten Ressourcen bei der Programmierung der Anwendung effizient auszunutzen.

3 Kriterien zur Entwicklung von MPSoC

3.1 Anwendungsorientierter Systemaufbau

Die Verbreitung von kommerziell oder öffentlich erhältlichen IP's, lässt eine immer größere Funktionsvielfalt bei der Konfiguration von FPGAs zu. Der Einsatz von SoC in hardwarebezogenen Systemen optimiert den Entwicklungsprozess durch eine vereinfachte Integration von peripheren Komponenten. In MPSoC wird dieser Ansatz zusätzlich noch erweitert, indem die einzelnen Prozessoren auf bestimmte Aufgaben spezialisiert werden (vgl. (Rotenberg und Anantaraman, 2005)). Gleichzeitg können Methoden und Funktionalität aus der Softwareentwichlung in den Entwicklungsprozess aufgenommen werden; der Einsatz von imperativen Programmiersprachen und die Verwendung von Betriebssystemen mit standardisierten Prozesskommunikationsmethoden erleichtert zudem die Einbindung von Programmierern, die keine spezielle Kenntnisse von Hardwarebeschreibungssprachen besitzen (vgl. (Tanenbaum, 2006)). In Abbildung 2 wird die Spezialisierung eines FPGA basierten Systems von der grundlegenden RTL-Logik (a), über ein SoC (b), hin zu einem MPSoC (c) in einer schematischen Darstellung gezeigt.

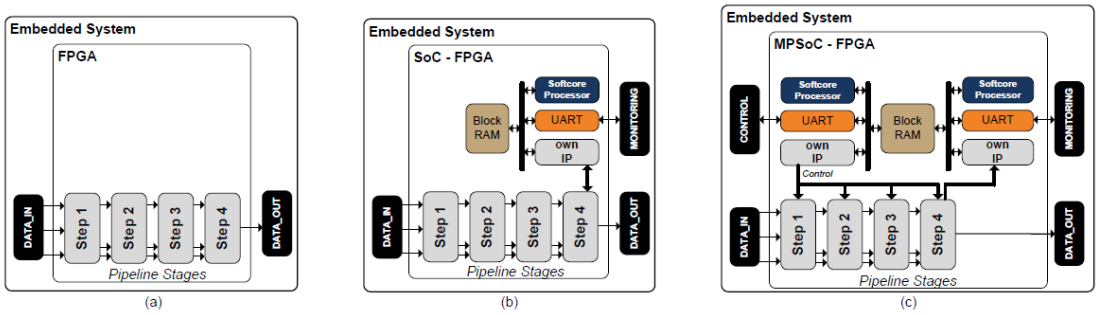

Abbildung 2: Entwicklungsstufen eines FPGA basierten Systems

Für die Erstellung eines MPSoC ist im Vorfeld eine detaillierte Systemanalyse durchzuführen, die klärt, welche Komponenten direkt in RTL-Hardware implementiert werden und welche Komponenten auf Prozessoren laufen. Darüberhinaus müssen Entscheidungen über die Art der Kommunikation zwischen den Prozessoren und deren Zugriff auf gemeinsame Ressourcen getroffen werden. Diese Entscheidungen müssen unter Berücksichtigung von Entwicklungsaufwand, -dauer und -kosten sowie der Wiederverwendbarkeit der erstellten Komponenten getroffen werden. Dabei stellt sich die Frage, ob es sich lohnt ein bestimmtes Modul in Hardware anzulegen, und damit einen gesteigerten Entwicklungsaufwand zu

leisten, dagegen aber die Möglichkeit zu besitzen, die Schnittstellen des Moduls selbst definieren zu können, und somit eine Verwendung durch andere Module zu erleichtern. Es gibt eine Reihe weiterer Entscheidungskriterien, die für verschiedene Implementierungsansätze sprechen; im Folgenden werden einige dieser Ansätze in Bezug auf die Hardware- bzw. Softwareerstellung betrachtet.

3.2 Ansätze für Implementierungsentwürfe

Die Kenntnis über die Kooperation zwischen hardware- und softwarebasierten Komponenten eines MPSoC ist von elementarer Bedeutung für deren Hardware-Software Codesign. Dabei kann der Entwicklungsprozess je nach Bedarf auf die Priorisierung von Hardware- bzw. Softwareanteilen abgestimmt werden. Während die Entwicklung von Beschleunigungsmodulen zunächst einen erhöhten Aufwand und spezielles Fachwissen erfordert, kann so jedoch die spätere Softwareentwicklung entlastet und ggf. die Wiederverwendbarkeit von Komponenten sichergestellt werden. Zunächst muss jedoch die Grundarchitektur des MPSoC festgelegt werden. Anhand der Erkenntnisse aus der Analyse der Systemanforderungen kann entschieden werden, ob für die Architektur des MPSoC ein homogener oder heterogener Ansatz gewählt werden soll (vgl. (Tanenbaum, 2006) und (Rauber und Rünger, 2007)). Eine homogene Multiprozessor Architektur umfasst die Verwendung von baugleichen Prozessoren, welche die Bearbeitung einer bestimmte Aufgabe gemeinsam erledigen und dabei auf die selben Systemressourcen zugreifen. Diese Eigenschaft macht homogen basierte MP Architekturen skalierbar, erfordert jedoch eine besondere Behandlung der Ressourcenzugriffe. In heterogenen MP Architekturen werden verschieden Prozessorvarianten eingesetzt, die auf die Bearbeitung einer speziellen Aufgabe optimiert sind (Rowen, 2005). Eine weitere Unterscheidung von Prozessoranordnungen in MPSoC kann über die Art der Speicheraufteilungen und Zugriffstechniken unternommen werden.

3.2.1 Eng gekoppelte Systeme

Die eng gekoppelten Systeme verwenden eine gemeinsam genutzten Speicher (vgl. Abbildung 3 (a)), wobei die Zugriffe auf diesen Speicher über bestimmte Verfahren synchronisiert werden müssen. Unter Berücksichtigung verschiedener Konsisentzmodelle lässt sich hier eine weitere Unterteilung in verschiedene Kategorien vornehmen: UMA (Uniform Memory Access), NUMA (NonUniform Memory Access) und COMA (Cache Only Memory Access). Aufgrund der Tatsache, dass in eng gekoppelten Systemen oftmals mehrere Speichermodule an verschiedenen Orten verfügbar sind, ergeben sich unterschiedliche Zugriffszeiten. Der UMA Ansatz garantiert dabei eine konstante Zugriffszeit auf jeden Speicherbereich im System, wobei ggf. schnellere Zugriffszeiten verzögert werden. Dagegen verfügt ein NUMA-Multiprozessor über einen Speicherbereich bzw. Cache auf den schneller zugegriffen werden kann, was eine spezielle Verteilung von Daten und Instruktionen erfordert, um optima-

le Leistungen zu erzielen. Trotzdem kann es bei dieser Variante zu erheblich verzögerten Zugriffszeiten kommen. Mit Hilfe des COMA Ansatzes wird versucht auch diese zu minimieren, indem der gesamte Speicherbereich in Cache Zeilen aufgeteilt wird (vgl. (Tanenbaum, 2006)). Diese Ansätze bieten dem Hardwareentwickler eine Reihe von Möglichkeiten das MPSoC spezielle auf die benötigten Anforderungen anzupassen. Bei der Erstellung eines entsprechenden Hardwaremoduls zum Speicherzugriff muss er sicherstellen, dass die Systemvorgaben entsprechend erfüllt werden. Allerdings erfordert diese Implementierung auch genaue Kenntnisse über die Auswirkungen auf die Softwareentwicklung. Eng gekoppelte Systeme haben diesbezüglich den Vorteil, dass sie den eigentlichen Vorgang des Speicherzugriffes für den Softwareentwickler kapseln. Über einfache LOAD und STORE Befehle kann dieser auf die benötigten Speicherbereiche zugreifen. Desweiteren sollte die Implementierung des Hardwareentwicklers vorsehen, dass eine Koordinierung der Speicherzugriffe, durch bestimmte Schedulingverfahren eines entsprechenden Betriebssystems unterstützt werden kann, um den Softwareentwickler weiter zu entlasten.

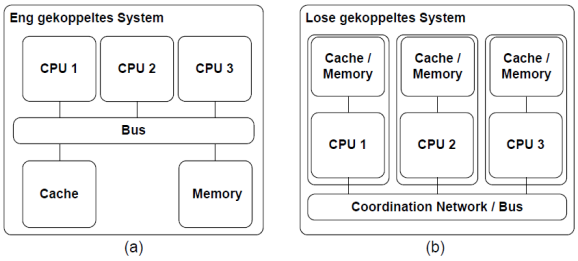

Abbildung 3: Darstellung eines eng - (a) und eines lose gekoppelten Systems (b)

3.2.2 Lose gekoppelte Systeme

In lose gekoppelten Systemen besitzt jeder Prozessor einen eigenen Speicher, auf den nur er selbst zugreifen kann. Der Austausch von Informationen unter den Prozessoren erfolgt über Nachrichten und spezielle Verbindungsnetze (vgl. Abbildung 3(b)). Dabei können verschiedene Topologien zum Einsatz kommen, die auf die Verteilung selbst, Skalierung des Systems oder Sicherheit des Nachrichtenaustausches optimiert sind. Beispiele für diese Topologien, wie Stern-, Baum-, Ring-, oder Würfeltopologien sind aus der Netzwerktechnik bekannt und können auf den Bereich der MPSoC übertragen werden. Das Hardwaremodul muss die Eigenschaften eines Routers übernehmen, um die Nachrichten zwischen den Prozessoren verteilen zu können. Bei der Verwendung von lose gekoppelten Systemen steigt darüberhinaus der Entwicklungsaufwand des Softwareentwicklers. Um bestimmte Informationen unter den Prozessoren verteilen zu können, müssen Mechanismen zum Nachrichtenaustausch über SEND/RECEIVE Methoden eingehalten werden. Zusätzlich können Vorgaben zu synchroner bzw. asynchroner Übertragung, dem Anlegen von Eingangs- bzw. Ausgangsbuffern

oder Einhaltung von definierten Formaten der Nachrichtenpakete zu erfüllen sein (vgl. (Tanenbaum, 2006)).

4 Nios II Multiprozessor System

An der HAW werden im Rahmen des Bachelorstudienganges die FPGA Serien und Entwicklungstools von *Xilinx* verwendet; für die Projekte des Masterstudienganges werden daher gezielt die Produkte von *Altera* eingesetzt, um so einen Einblick in diese Entwicklungsumgebungen bzw. Synthesetools zu erlangen und die beiden Hauptanbieter mit einander vergleichen zu können. Für die Implementierung wurde das DE2-70 Entwicklungsboard der Firma *terasic* verwendet, welches mit einem *Altera* Cyclone II 2C70 FPGA bestückt und zudem mit einer Vielzahl von peripheren Multimedia- und Kommunikationskomponenten ausgestattet ist (vgl. Abbildung 4). In Hinblick auf die Projektarbeiten der nächsten Semester kann das erstellte MPSoC durch die Einbindung dieser Komponenten stetig erweitert werden. Die Programmierung des FPGA erfolgt über den *jtag_uart* der über die USB Blaster Schnittstelle direkt mit dem Entwicklungsrechner verbunden wird. Diese Schnittstelle kann zusätzlich darauf konfiguriert werden Standard Ein- und Ausgaben zu Debugging Zwecken vom und zum Entwicklungsrechner zu übertragen.

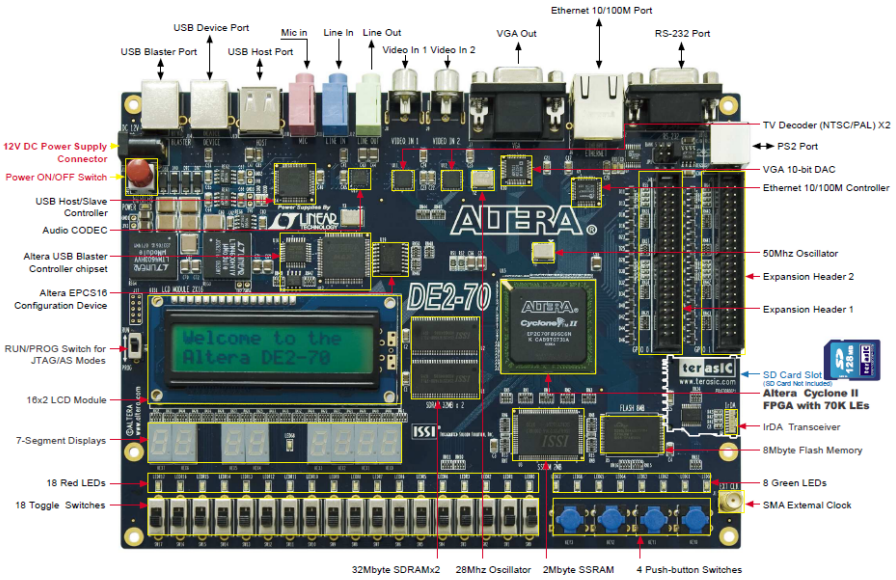

Abbildung 4: Das terasic DE2-70 Entwicklungsboard

Um sich mit den Entwicklungstools *Quartus II*, *SOPC Builder* und der *Nios II IDE* vertraut zu machen. wurde ein Beispielsystem erstellt, in dem drei Nios II Prozessoren in unterschiedlichen Zeitabständen Statusmeldungen in einen definierten Speicherbereich des

OnChip Memorys ablegen. Der Speicherbereich ist durch einen Hardwaremutex gesichert, so dass nur ein Prozessor zur Zeit Zugriff auf diesen Speicherbereich hat. Der erste Prozessor *cpu1* hat zusätzlich die Aufgabe, die im Speicher liegenden Meldungen zu lesen und über die Standardausgabe an den Entwicklungsrechner auszugeben.

4.1 Hardwarekonfiguration

Die Konfiguration des MPSoC wurde mit Hilfe des SOPC Builders und dessen Component Library erstellt. Dazu wurden drei Nios II Prozessoren konfiguriert und für jeden ein Timer zur Software Steuerung eingerichtet. Als gemeinsam genutzter Speicher wurde ein 1kB großer Speicherbereich des Flash Speichers alloziert [*message_buffer_ram*]. Zur Gewährleistung des geregelten Zugriffs auf diesen Speicherbereich, wurde ein Hardwaremutex eingerichtet [*message_buffer_mutex*], und wie Abbildung 5 zeigt, zusammen mit dem Speicherbereich über die *connection matrix* an die Datenbusse der Prozessoren angeschlossen. Zur Syn-

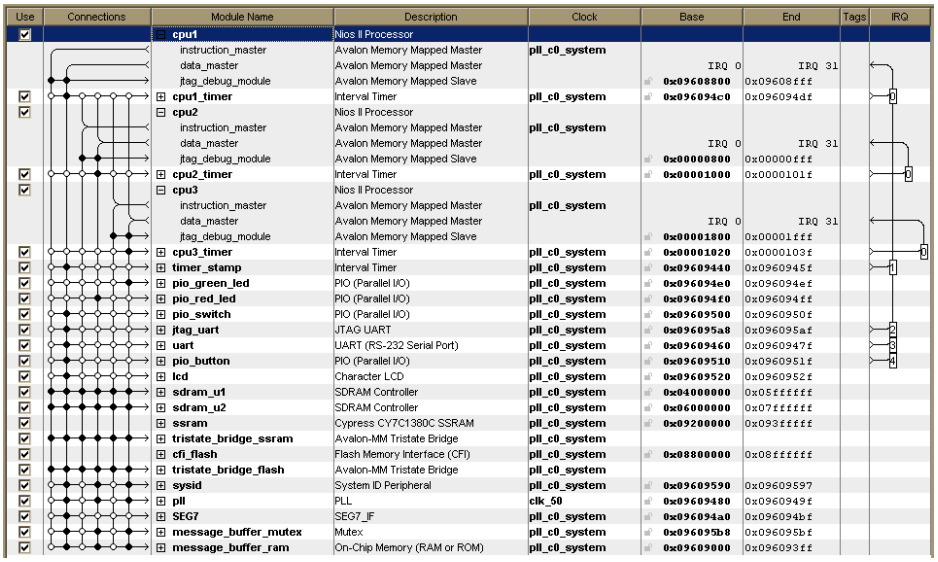

Abbildung 5: Ansicht des SOPC Builders mit der *connection matrix*

these der so erstellten Systemkonfiguration wurde das *Quartus II* Synthesetool verwendet, welches auch die entsprechenden Prozessorkonfigurationsdateien erstellt, die für die Erstellung der Softwareprojekte nötig sind.

4.2 Software Komponenten

Die Software für das MPSoC ist in der *Nios II IDE* erstellt worden. Für jeden Prozessor wurde ein eigenes Softwareprojekt erstellt und die entsprechenden vom *Quartus II* erzeugten prozessorspezifischen Konfigurationsdateien importiert. Jedes Softwareprojekt enthält

ein eigenes C Programm, welches die Aufgabe des Prozessors implementiert. Da in diesem Beispielprojekt alle Prozessoren die gleichen Aufgabe haben, genügte es lediglich eine Quellcode Datei, *hello_world_multi.c*, anzulegen, die in alle Projekt importiert wurde. Innerhalb des Programmes wird in einer Endlosschleife

- überprüft, ob der Timer zum Erstellen der Nachricht abgelaufen ist
- versucht den Hardware Mutex zu belegen
- bei Erhalt des Mutexes, eine Nachricht in den gemeinsamen Speicher geschrieben
- und anschließend der Mutex wieder freigegeben.

Zusätzlich wird in einer Abfrage (vgl. Listing 1) überprüft, ob das Programm auf dem ersten Prozessor *cpu1* läuft, um die Statusnachrichten durch Verwendung der Standardausgabe über den USB Blaster auf dem Entwicklungsrechner auszugeben.

Listing 1: Abfrage zur Ausgabe über den USB Blaster

```
#ifdef JTAG_UART_NAME {
        if(message->flag == MESSAGE_WAITING){
        printf("%s", message->buf);
        message->flag = NO_MESSAGE;
      }}
#endif /* JTAG_UART_NAME */
```

5 Zusammenfassung und Ausblick

Die Verwendung von FPGAs zur Implementierung von *embedded systems*, die speziellen Anforderungen genügen müssen, besitzt den Vorteil, dass der Entwicklungsprozess durch den Einsatz von MPSoC optimiert werden kann, wobei deren Komplexität mit der steigenden Anzahl an Logikblöcken auf den FPGAs wächst. Die Erweiterung von SoC zu MPSoC ermöglicht eine konkretere Abstimmung auf die jeweiligen Hardwarebestandteile des Systems. Dazu unterstützen die FPGA Hersteller die Entwickler mit einer Reihe von Entwicklungstools zum Hardware-Software Codesign. Die Aufgabe des Entwicklers ist es, eine geeignete Hardware-Software Partitionierung auf dem FPGA zu erstellen, die den Anforderungen an das eingebettete System entspricht. Zur Kommunikation und Synchronisation der Prozessoren gibt es verschiedene Ansätze, die beim Systemdesign anhand der Aufgabenstellung festgelegt werden. Für den Entwurf eines MPSoC wird zunächst eine Hardware Konfiguaration erstellt, welche die verfügbaren Systemressourcen einbindet. Im zweiten Schritt wird die Software für diese spezielle Konfiguartion erstellt. Somit sind für den Entwurf von MPSoC sowohl Kenntnisse in der Erstellung von hardwarenahen Systemen als auch

Erfahrungen in der Implementierung von Softwareanwendungen. Nachdem das in Kapitel 4 beschriebene System einen ersten Ansatz zum Umgang mit MPSoC bietet, kann dieses in den kommenden Semestern erweitert werden, um die Aufgabentrennung der Prozessoren sowie die Kommunikation zwischen diesen weiter zu verdeutlichen. Für das Entwicklungsboard ist neben einem 5 Mega Pixel Digital Camera Module *(TRDB_D5M)* auch ein 4.3"LCD Touch Panel Modul *(TRDB_LTM)* verfügbar. Diese sollen für das nächste Projekt in Betrieb genommen und ein entsprechender Systemaufbau zur Verwendung dieser Bauteile durch unterschiedliche Prozessoren entwickelt werden. Ein weiterer Nios II Prozessor soll die Kommunikation mit dem Entwicklungsrechner und einem weiteren VGA Bildschirm bearbeiten. Desweiteren ist zu analysieren, wie schnell eine funktionierende Konfiguration mit Hilfe der vom Hersteller bereitgestellten Komponenten und Entwicklungsumgebungen zu erzielen ist, bzw. welche Auswirkungen auf den Entwicklungsprozess verschiedene Optimierungsansätze hervorrufen. Daneben sind die technologischen Kenntnisse zu vertiefen und die Strukturen und Methoden für die Erstellung von Multiprozessor Systemen zu untersuchen. Das behandelt neben der Partitionierung und Anordnung der Prozessoren auf dem FPGA sowie deren Kommunikations- und Synchronisationsmechanismen auch die auf diese Aspekte angepasste Softwareentwicklung. Dazu ist eine Einarbeitung in die Methodik der parallelen Programmierung und deren Auswirkungen erforderlich.

Literatur

[Actel 2009] ACTEL: *System solutions*, 2009. – URL `http://www.actel.com/products/solutions/default.aspx`. – from companies webpage [Nov 2009]

[Altera 2009] ALTERA: *Altera Devices*, 2009. – URL `http://www.altera.com/products/devices/dev-index.jsp`. – from companies webpage [Nov 2009]

[Bilski u. a. 2007] BILSKI, Göran ; MOHAN, Sundararajarao ; WITTIG, Ralph: *Designing Soft Processors for FPGAs*. Kap. 17. In: *Customizable Embedded Prozessors*. San Fransisco, CA : Morgan Kaufman, 2007. – ISBN 0-12-369526-0

[Lattice 2009] LATTICE: *FPGA Solutions*, 2009. – URL `http://www.latticesemi.com/products/fpga/index.cfm?source=topnav`. – from companies webpage [Nov 2009]

[Rauber und Rünger 2007] RAUBER, Thomas ; RÜNGER, Gudula: *Parallele Programmierung*. Heidelberg : Springer, 2007. – ISBN 3-540-46549-2

[Rotenberg und Anantaraman 2005] ROTENBERG, Eric ; ANANTARAMAN, Aravindh: *Architecture of Embedded Microprocessors*. Kap. 4. In: *Multiprocessor System-on-Chips*. San Fransisco, CA : Morgan Kaufman, 2005. – ISBN 0-12-385251-X

[Rowen 2005] ROWEN, Chris: *Performance and Flexibility of Multiple-Processor SoC Design*. Kap. 5. In: *Multiprocessor System-on-Chips*. San Fransisco, CA : Morgan Kaufman, 2005. – ISBN 0-12-385251-X

[Tanenbaum 2006] TANENBAUM, Andrew S.: *Parallele Rechnerarchitekturen*. Kap. 8. In: *Computerarchitektur*. München : Pearson Studium, 2006. – ISBN 3-8273-7151-1

[Xilinx 2009] XILINX: *Silicon Devices Overview*, 2009. – URL `http://www.xilinx.com/products/devices.htm`. – Overview from companies webpage [Nov 2009]