Entwicklung einer FPGA basierten Distributed Computing Plattform

Frank Opitz

Ausarbeitung

Fakultät Technik und Informatik
Department of Engineering and Computer Science

Frank Opitz

Entwicklung einer FPGA basierten Distributed Computing Plattform

Ausarbeitung eingereicht im Rahmen der Veranstaltung Seminar
im Masterstudiengang Informatik
im Studiendepartment Informatik
an der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Professor : Dr.-Ing. Bernd Schwarz

Gutachter : Prof. Dr. rer. nat. Kai von Luck
Gutachter : Prof. Dr. rer. nat. Gunter Klemke

Inhaltsverzeichnis

1 Einleitung **2**

2 Vorarbeiten zur Partiellen Rekonfiguration an der HAW-Hamburg **5**

3 Zielsetzung und Vorgehen **6**
3.1 Vorgehen beim Erstellen des SoC-Systems 6
3.2 Vorgehen beim Erstellen des Servers und der Projekt-Architektur 8
3.2.1 Aufbau des Projekt-Servers 8
3.2.2 Aufbau des Projekt-Clients 9

4 Risiken **9**
4.1 Risiken bei der Erstellung der Masterarbeit 9
4.2 Randbedingungen für das Projekt 10

5 Zusammenfassung **10**

Literatur **12**

Abbildungsverzeichnis **i**

A Ablaufdiagramm des DHPEC Projektes **ii**

1 Einleitung

Die stetig wachsende Komplexität von zu berechnenden Daten, erfordert den Bau sogenannter Supercomputer. Der zur Zeit schnellste Supercomputer, mit einer Rechenleistung von 2507 Tera FLOPS, ist der Tianhe-1A, welcher im National Supercomputer Center in China für die Berechnung von chemischen und physikalischen Prozessen verwendet wird, beispielsweise der Simulation von Flugzeugkomponenten ([NVIDIA (2010)],[Wikipedia (2010)]). Die Entwicklung dieses Supercomputers ist von 200 Wissenschaftlern innerhalb von zwei Jahren vollzogen worden und kostete etwa 88 Millionen USD [Raman (2010)]. Diese Kosten sowie die jährlichen Energiekosten, von etwa 2,7 Millionen USD, sind vom Träger des Projektes zu tragen. Für mittelständische Firmen sowie Universitäten sind Projektkosten dieser Größenordnung nicht zu finanzieren, so dass hier für komplexe Probleme die Rechnungen auf mehrere Rechner verteilt werden, wodurch ein virtueller Supercomputer entsteht.

Diese verteilten Berechnungen verbinden einzelne Rechner über eine bestehende Netzwerk Infrastruktur zu einem „Distributed Computing“ System. Zumeist bestehen derartige Systeme aus Freiwilligen, die die Rechenleistung ihrer PCs verschiedenen Projekten zur Verfügung stellen, wie es beim „Berkeley Open Infrastructure for Network Computing“ (BOINC) der Fall ist. Diese Plattform dient der Verwaltung der Ressourcen, wie CPUs und Speicher, und stellt sie wissenschaftlichen und kommerziellen Projekten zur Verfügung. Das BOINC ist aus dem SETI@Home Projekt entstanden, welches in den ersten Versionen des Projektes die wissenschaftliche Arbeit sowie die Verwaltung der Ressourcen durchgeführt hat. Die Aufeilung des eigentlichen Projektes in BOINC und den wissenschaftlichen Anteil, sorgte für einen Anstieg der interessierten Anwender, da die Art der Projekte stetig gewachsen ist [California (2010)]. Folgende Projekte geben einen kurzen Einblick in die Vielseitigkeit der wissenschaftlichen Themen.

- ClimatePrediction.net:
  Dieses Projekt umfasst die Berechnung von Wettermodellen der nächsten 50 bis 100 Jahre, um Aussagen über die globale Erwärmung zu treffen. Das ursprüngliche Projekt diente der Eingrenzung des Parameterraums, um Wettermodelle für die Zukunft erstellen zu können und wurde mit dem HadSM3-Modell durchgeführt. Dieses Modell erlaubt eine detaillierten Simulation der Atmosphäre, beinhaltet jedoch eine vereinfachtes Modell des Ozeans. Berechnet wurde das Modell in drei Schritten:

  1. Mit den gewählten Parametern wird berechnet, ob eine stabile Wetterlage erreicht wird.
  2. Die Werte des Ozeans werden verändert und der CO_2 Gehalt der Luft auf einen vorindustriellen Wert gesetzt.
  3. Der $C0_2$ Gehalt der Luft wird verdoppelt.

  Bei der Berechnung jedes Schrittes wird überprüft, ob mit diesen Parametern ein konstantes Klima erreicht werden kann. Ist dies nicht der Fall, so wird die Berechnung abgebrochen und die Berechnung mit neuen Parametern gestartet. Das zweite Experiment von ClimatePrediction.net startete 2005 mit einem neuen Klimamodell. Für dieses Modell wurden die Parameter gesucht, die auf das Wetter der Jahre 1950 - 2000 passen. 2006 startete das eigentliche Projekt, welches mit den zu berechneten Parametern eine Vorausberechnung für die Jahre 2000 bis 2100 erstellt [Oxford (2010)].

- Einstein@home:
  Nach Gravitationswellen, welche entstehen, wenn stark verdichtete, schnell rotierenden Sterne die Raumzeit um sich herum deformieren, sucht dieses Projekt. Die These dazu wurde von Einstein in der allgemeinen Relativitätstheorie aufgestellt. Der Beweis, dass diese existieren, fehlt aber bis heute. Ausgewertet werden hier Datensätze, die bei der Messung mit einem Michelson-Interferometern entstehen. Hier werden mit Lasern die Längen von rechtwinklig angeordneten Vakuumröhren gemessen, was bis auf eine Genauigkeit von ca. 10^{-18} m erfolgt [Max Planck (2010)].
- FightAIDS@Home:
  FightAIDS@Home ist ein Projekt aus dem Pharmazie Bereich. Hier wird nach einem Medikament gesucht, welches zur Bekämpfung des HI-Virus am geeignetsten ist. Der HI-Virus unterliegt einer starken Mutation, welche simuliert wird, um für möglichst viele Mutationen ein passendes Medikament zu finden. Genauer wird ein Molekül gesucht, welches möglichst viele verschieden HIV Proteasen, welche an die menschlichen Zellen andocken, blocken kann [Olson-Laboratory (2010)].

Ist die Bearbeitung von Daten in einer Echtzeit Umgebung erforderlich, so werden FPGAs zur Berechnung eingesetzt. Diese haben die Eigenschaft, dass die implementierten Algorithmen, anders als in einer CPU, nebenläufig ablaufen. So wird ein höherer Datendurchsatz bei einer geringen Frequenz erreicht. In Tabelle 1 ist der Vergleich des Datendurchsatzes, von einem Virtex2 FPGA und einem Itanium 2 Prozessor dargestellt. Gezeigt wird, dass bei einer, um den Faktor zehn, geringeren Frequenz die vierfache Anzahl an Berechnungen auf dem FPGA durchgeführt werden.

Tabelle 1: *Vergleich eines Virtex 2 FPGAs mit einem Itanium 2 Microprozessor [Xilinx (2006)].*

	Microprozessor Itanium 2	FPGA Virtex 2VP100
Technology	0,13 Micron	0,13 Micron
Frequenz	1,6 GHz	180 MHz
Durchsatz des Internen Speichers	102 GBytes/s	7,5 TBytes/s
Rechen Einheiten	5 FPU(2MACs + 1FPU) + 6 MMU + 6 Integer Unit	212 FPU oder 300+ Integer Units u.v.m.
Verbrauch	130 Watt	15 Watt
Performanz (Spitze)	8 GFLOPs	38 GFLOPs
Performanz (Mittel)	2 GFLOPs	19 GFLOPs
Durchsatz von I/O-Ports	6,4 GBytes/s	67 GBytes/s

Bisherige FPGA-basierte Systeme waren durch die Größe des FPGAs begrenzt und wurden beim Systemstart konfiguriert. Hier wurde die komplette Konfiguration beim Start in den FPGA geschrieben, so dass die Größe des FPGAs die Anzahl und Komplexität der Module bestimmt. Überschreitet die Anzahl der benötigten Ressourcen die Kapazität des FPGAs, so wurde die Anzahl der FPGAs erhöht oder eine komplette Rekonfiguration des FPGAs vorgenommen, wobei in den einzelnen Konfigurationen unterschiedliche Module integriert waren. Die Rekonfiguration des FPGAs erfolgt hierbei, je nach Größe des FPGAs und der Module, in mehreren

Millisekunden, in denen der FPGA reaktionsunfähig ist, was besonders in Sicherheitskritischen-Systemen nicht akzeptabel ist. Zur Verkürzung der Reaktionsunfähigkeit wird aktuell mit der Partiellen Rekonfiguration nur ein Teil des FPGAs rekonfiguriert. In dieser Zeit kann der FPGA aber weiterhin nicht verwendet werden. Die Weiterentwicklung der Partiellen Rekonfiguration, die „Dynamische partielle Rekonfiguration“ (DPR) erlaubt es, dass sicherheitskritische Module ihre Arbeit fortsetzen, während Teile des FPGAs rekonfiguriert werden. Dazu ist die Unterteilung des FPGAs in verschiedene Bereiche notwendig. Zum einen sind hier die dynamischen Bereiche, sogenannte „Partial Reconfiguration Regions“ (PRR), zum anderen eventuell vorhanden statische Bereiche zu definieren. Diese statischen Bereiche sind in der Lage über einen „Internal Configuration Acces Point“ (ICAP) den FPGA selbst zu rekonfigurieren [Xilinx (2008)].

Das Projekt „Entwicklung einer FPGA basierten Distributed Computing Plattform“ befasst sich mit einer Kombination aus dem „Distributed Computing“ und der „Dynamischen partiellen Rekonfiguration“ (vgl. Abbildung 1). Angestrebt wird ein Netzwerk aus System on Chips (SoC), welches sich an der BOINC Architektur orientiert und zur Berechnung komplexer Daten verwendet wird. So ist es nicht mehr notwendig eine größere Anzahl an FPGAs für ein Projekt anzuschaffen, was in bisherigen Systemen der Fall war [Güneysu u. a. (2007)].

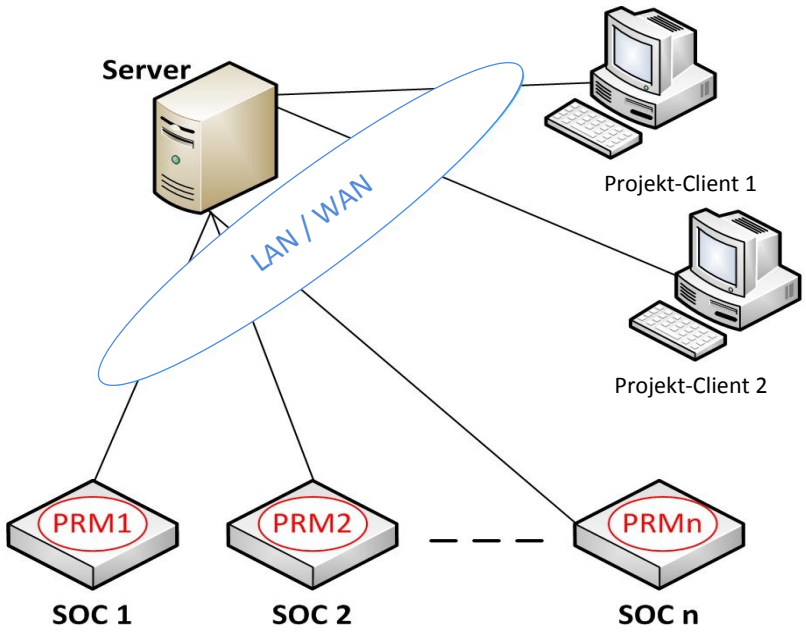

Abb. 1: *Architektur des Distributed Computing Systems mit den einzelnen Komponenten und der Kommunikation untereinander*

Diese Ausarbeitung beschreibt in Kapitel 2, die bisherigen Projekte, an der HAW-Hamburg, mit der DPR und einem selbstrekonfigurierbaren System. Kapitel 3 stellt das Vorgehen und die Zielsetzung für die Masterarbeit dar. Die Erstellung des SoC Systems sowie der Aufbau des Software-Servers finden hier besondere Beachtung. Kapitel 4 zeigt die Risiken für die Masterarbeit sowie das Gesamtprojekt auf. Anschließend folgt eine Zusammenfassung dieser Ausarbeitung.

2 Vorarbeiten zur Partiellen Rekonfiguration an der HAW-Hamburg

Eine Evaluierung des Design-Flows zur DPR mit Xilinx ISE 9 fand in Mamegani (2010) statt. Es wurde ein FPGA-basiertes SoC aufgebaut, mit dem sich mehrere SW-Anwendungen durch zur Laufzeit geladene HW-Module beschleunigen lassen. Das System besteht aus einem MicroBlaze basiertem SoC und einer Petalinux Distribution. Die Petalinux Distribution beinhaltet eine Zusammenstellung eines Linux-Kernels und Treiber für MicroBlaze-Systeme, mit der die Ausführung mehrerer SW-Anwendungen verwaltet wird. Dem Linux-Kernel wurde zusätzlich ein Treiber zur Anbindung der internen Rekonfigurationsschnittstelle des FPGAs (Internal Configuration Access Port) hinzugefügt. Die HW-Konfiguration wird aus einem statischem und mehreren dynamischen Teilen gebildet. Der statische Teil des Systems setzt sich aus allen Modulen zusammen, deren Funktionalitäten zur Laufzeit immer aktiv sind. Der dynamische Teil lässt sich neu konfigurieren, ohne den statischen Teil des Systems in seinem Betrieb zu unterbrechen. Die PRR ist ein lokaler Bereich des FPGAs, in dem rekonfigurierbare Module (PRM) platziert werden. Die PRMs implementieren verschiedene Funktionalitäten und werden je nach Anforderung durch die Anwendung dynamisch geladen [Mamegani (2010)].

Aufbauend auf der Arbeit von Mamegani wurde im „Projekt 1“ der überarbeitete Design Flow mit ISE 12 und Plan Ahead evaluiert. Es wurde eine SoC Plattform zum Test der DPR aufgebaut und die Eigenschaften der Rekonfiguration ermittelt (vgl. Abbildung 2). Die Steuerung des selbstrekonfigurierenden Systems erfolgt mit dem μC/OS-2 Echtzeitbetriebssystems anstelle des Petalinux. Zur Analyse des Zeitverhaltens wurde beispielswiese die Rekonfigurationszeit der PRMs gemessen [Opitz (2010)].

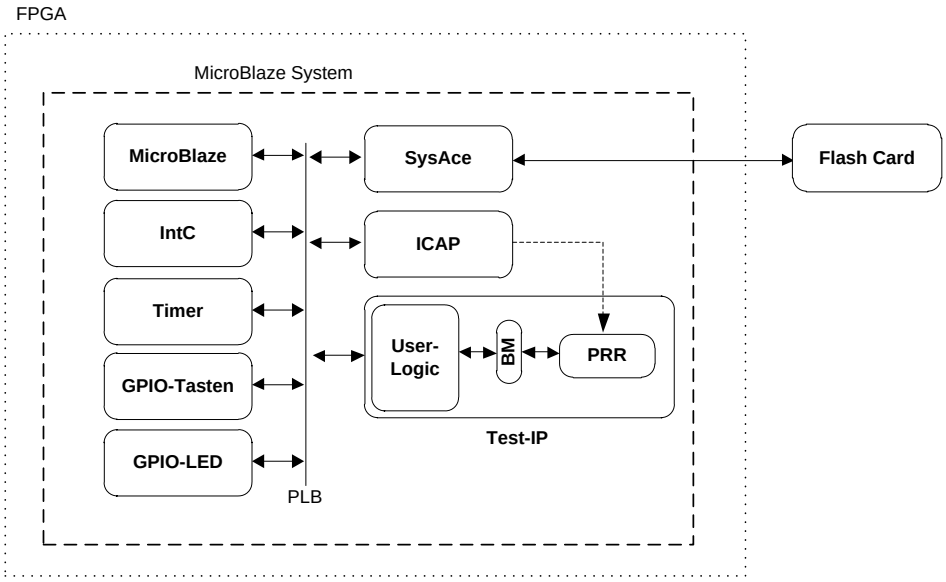

Abb. 2: *SoC-System mit MicroBlaze und der „Partiel Rekonfiguration Region“, mit Kommunikation über Bus Makros zur Synchronisation der Signale. Das System besteht aus einem MicroBlaze, der zur Ausführung des μC/OS-2 verwendet wird, sowie beispielsweise Interrupt Controllern (IntC) oder Timern. Die PRM werden über das SysACE Modul von einer CompactFlash Karte gelesen und vom ICAP in den FPGA geschrieben [Opitz (2010)].*

3 Zielsetzung und Vorgehen

Das Ziel der Masterarbeit ist die Entwicklung eines „FPGA basierten Distributed Computing Systems“. Dieses System hat die Aufgabe, die Berechnungen, die für Projekte notwendig sind, auf FPGAs auszulagern und so die Anschaffung größerer Rechenzentren oder FPGA-Cluster zu vermeiden. Das System besteht dabei aus folgenden Komponenten, welche zu entwickeln und implementieren sind:

- SoC-System:
  Das SoC-System stellt die FPGA-Ressourcen zur Verfügung und ist für die Übertragung der PRM sowie der berechneten Daten zuständig. Wurde ein Modul zur Berechnung der Daten empfangen, so wird dieses in die PRM geschrieben. Ist dies erfolgt, werden die vom Server gesendeten Daten mit dem empfangenden Modul berechnet und die Ergebnisse werden zurück an den Server gesendet.
- Projekt-Client:
  Hier werden die zu berechnenden Daten und die dafür verwendeten Module verwaltet und nach der Anmeldung am Server den SoCs zur Verfügung gestellt.
- Server zur Verwaltung der SoC sowie der Projekte:
  Der Server stellt die Kommunikation zwischen den SoCs sowie den Projekten sicher und verteilt die Aufgaben der Projekte an die zur Verfügung stehen SoC-Systeme.

Abbildung 1 stellt die Architektur des Systems dar. Diese Architektur ist auf die SoCs ausgelegt, die nur eine geringe Anzahl an Sockets verwalten können, welche mit dieser Architektur auf ein Minimum reduziert sind. Die Kommunikation zwischen den Komponenten ist in Anhang A dargestellt.

Die Berechnungen in diesem Distributed Computing System werden, anders als in bestehenden Systemen, auf SoC-Systemen durchgeführt. Hierzu ist zusätzlich zu einem PC-Server ein SoC-System zu erstellen. Bei der Entwicklung und Implementierung ist bei beiden Systemen auf unterschiedliche Schwerpunkte zu achten, so ist beispielsweise das SoC-System so auszulegen, dass viele Ressourcen für die PRR, welche die Projekte beinhaltet, zur Verfügung stehen. Im Gegensatz dazu, ist der Server so zu gestalten, dass eine große Anzahl an SoC-Clients sowie Projekten gleichzeitig verwaltet werden.

3.1 Vorgehen beim Erstellen des SoC-Systems

Das SoC ist für die Verwaltung der FPGA Ressourcen sowie zum Versenden und Empfangen der Konfiguration und Projektdaten zuständig. Unterteilt wird das System in ein statisches und ein dynamisches Subsystem (vgl. Abbildung 3).

Das statische System wird zu Beginn in den FPGA geschrieben, dies erfolgt über die JTAG Konfigurationsschnittstelle oder über das SystemAce Modul beim Anschalten des FPGA Boards. Mit den enthaltenen Hardware Komponenten (vgl. Abbildung 3, statisches System) wird eine Verbindung zu dem Server aufgebaut und die Projekt-IP angefordert. Ist dieses empfangen, so wird das dynamische Subsystem geschrieben. Nach der Konfiguration, werden die Projektdaten empfangen und die Berechnung gestartet. Parallel zur ausgeführten Berechnung werden weitere Daten geladen und nach Abschluss einer Berechnung das Ergebnis zum Server übertragen. Die jeweiligen Subsysteme bestehen aus den folgenden Komponenten:

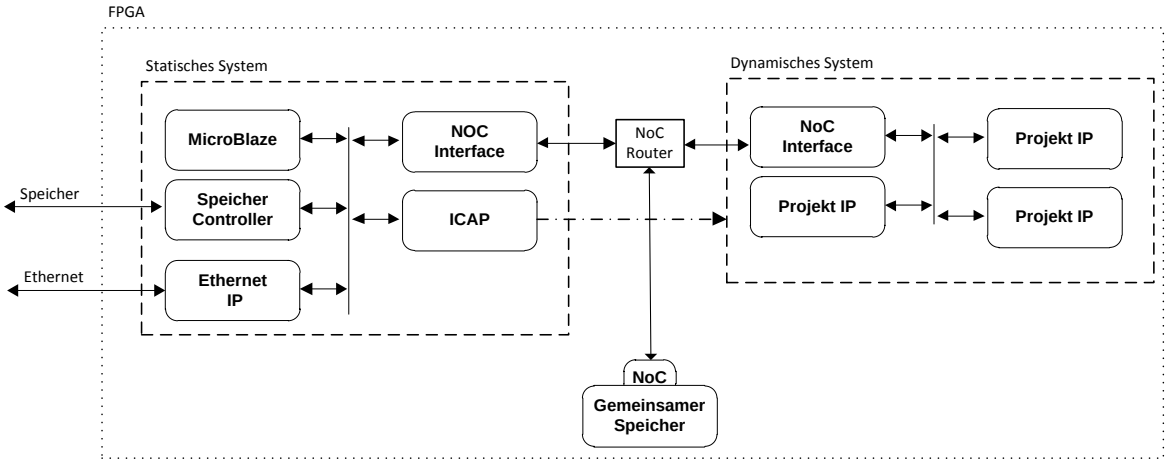

Abb. 3: *SoC des Distributed Computing Systems mit statischem und dynamischem Subsystem*

- Statisches Subsystem:

  – MicroBlaze:
  Der MicroBlaze, ein 32 Bit RISC Prozessor, ist für die Steuerung des Systems zuständig. Er ist für die Ausführung des verwendeten „Real-time operating systems“ (RTOS) zuständig. Dieses übernimmt das Verwalten der Konfigurationsdateien sowie der zur Berechnung verwendeten Daten. Es wird zusammen mit dem statischen System geladen und aus einem der angeschlossenen Speicher ausgeführt.

  – Ethernet IP:
  Das Ethernet IP stellt die physikalische Anbindung zum Netzwerk zur Verfügung.

  – Speicher Controller:
  Der Speicher Controller ist für die Verbindung zum Speicher zuständig. Auf dem verwendeten Entwicklungsboard ist dies ein DDR3 Speichermodul.

  – ICAP:
  Der ICAP konfiguriert den dynamischen Teil des Systems. Die Daten werden vom MicroBlaze vom Speicher Controller angefordert und anschließend dem ICAP übergeben.

  – NoC-Interface:
  Die Kommunikation zwischen dem dynamischen und statischen Subsystem erfolgt über ein „Network on Chip“. Es verbindet weiter einen gemeinsamen Speicher mit beiden Systemen, über den die Daten für die Berechnung ausgetauscht werden.

- Dynamisches Subsystem:
  Das dynamische Subsystem ist für die Berechnung der Projektdaten zuständig. Es wird vom Projekt-Betreiber zur Verfügung gestellt. Das System ist durch die Größe der PRR begrenzt und enthält ein Interface zum Anbinden an das statische Subsystem.

  – NoC-Interface:
  Das Interface ist fester Bestandteil des dynamischen Subsystems. Die Kommunikation erfolgt ausschließlich über die Signale des NoC, da diese bei der Synthese des statischen Subsystems bekannt sein müssen.

– Projekt IP:
Das Projekt IP wird vom Ersteller des Projektes zur Verfügung gestellt und beinhaltet die Logik zum Berechnen der Daten.

3.2 Vorgehen beim Erstellen des Servers und der Projekt-Architektur

Die Kommunikation zwischen dem SoC-Client und dem Projekt erfolgt über einen Verwaltungsserver [vgl. Abbildung 1]. Dieser ist für die Koordination der SoCs sowie der Projekte zuständig. Bei der Anmeldung der SoC-Clients werden diese kategorisiert und je nach Anforderung auf die Projekte verteilt. Die Projekte geben ihre Anforderungen beim Anmelden bekannt und erhalten die entsprechenden Ressourcen bei ihrer Verfügbarkeit.

Steht dem Projekt mindestens ein SoC-Client zur Verfügung, so stellt das Projekt diesem die Daten zur Berechnung sowie die PRM zur Verfügung. Die Daten werden vom Projekt verwaltet und vom Server entsprechend weiter geleitet. Ist die Berechnung der Daten abgeschlossen, so liegt es an dem Projekt-Client die Daten zu verifizieren und auf Korrektheit zu überprüfen (vgl. Abbildung 4).

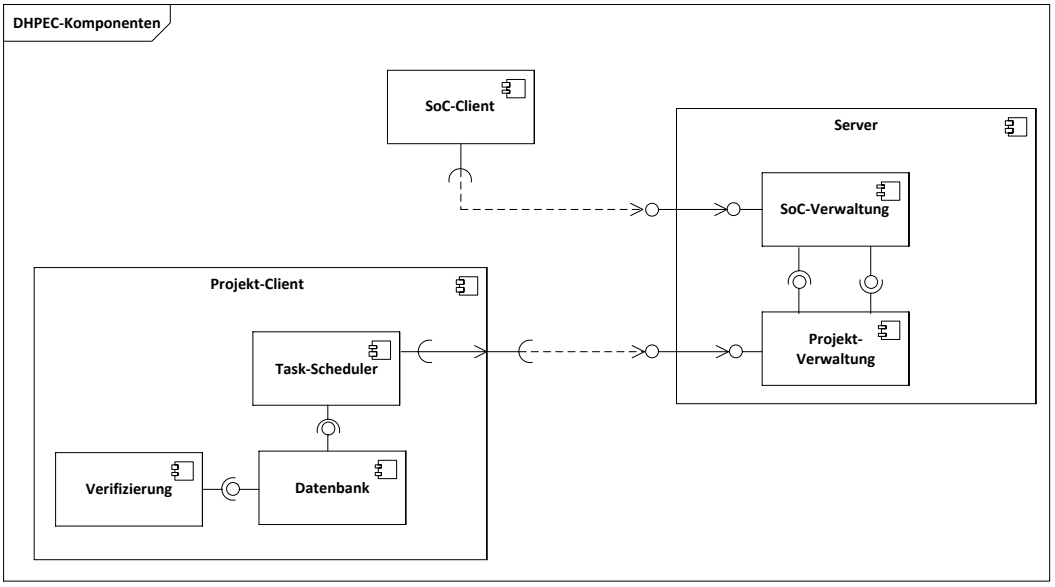

Abb. 4: *Komponenten Diagramm der Software des DHPEC Projekts. Der SoC-Client besteht dabei aus dem RTOS und der Software zum Übertragen der Daten.*

3.2.1 Aufbau des Projekt-Servers

Der Server steht den SoCs und Projekten als Single Point of Contact zur Verfügung, dies ist für die SoCs erforderlich, da deren Socket Anzahl stark eingeschränkt ist. Er besteht aus den folgenden Komponenten (vgl. Abbildung 4):

- SoC-Verwaltung:
Hier werden die Anfragen der SoCs angenommen und entsprechend weitergeleitet. Meldet sich ein SoC an, so wird dessen Konfiguration abgefragt und das System entsprechend des FPGA-Typs oder PRR-Größe eingeordnet.

- Projekt-Verwaltung:
  Bei der Anmeldung des Projektes gibt dieses bekannt, für welche SoC Konfiguration PRMs bereitliegen. Die Projekt-Verwaltung teilt den Projekten mit, ob für ihre Konfiguration entsprechende SoCs angemeldet sind und teilt den Projekten, je nach Verfügbarkeit, die Ressourcen zu.

3.2.2 Aufbau des Projekt-Clients

Der Aufbau des Projekt-Clients ist je nach Projekt zu entscheiden. Der in Abbildung 4 dargestellten Aufbau enthält empfohlene Komponenten, welche die Strukturierung des Projekt-Clients vereinfachen (vgl. Abbildung 4).

- Task-Scheduler:
  Der Scheduler verteilt die anstehenden Berechnungen auf die zur Verfügung stehen SoCs. Das Senden der Daten zur Berechnung sowie das Empfangen der fertigen Daten unterliegt der Kontrolle des Schedulers.
- Verifizierung:
  Die Verifizierung der Daten erfolgt je nach Projekt. Eine Variante ist hier beispielsweise, die Berechnungen auf verschiedenen SoCs durchführen zu lassen, um redundante Ergebnisse zu erhalten. Weiter besteht die Möglichkeit, dem SoC die Berechnung von bekannten Ergebnissen durchführen zu lassen und die Ergebnisse zu vergleichen.
- Datenbank:
  Die Datenbank enthält die zu berechnenden Daten sowie die Ergebnisse. Stehen die PRMs für mehrere Systeme bereit, so sind diese ebenfalls hier zu speichern.

4 Risiken

Die Risiken für die Durchführung dieser Arbeit sowie dem Erfolg des Projektes sind in den folgenden zwei Kapiteln dargestellt.

4.1 Risiken bei der Erstellung der Masterarbeit

Die Risiken, die bei der Erstellung der Masterarbeit auftreten können, sind unterteilt in die zwei Haupt-Komponenten des Projektes, in die SoC- und PC-Seite (vgl. Tabelle 2).

Die Einschätzung des Risikos, zur Erstellung der HW-Plattform, erfolgt aufgrund der Komplexität des NoC Themas sowie Erfahrungen aus vorherigen Projekten bei der Verwendung von neuen Technologien. Bei der Entwicklung der NoC Infrastruktur ist ein Router zu entwickeln, der auch Projekt intern verwendet werden kann, um einen einfachen Austausch von Projekt Modulen zu gewährleisten. Weiter ist eine Anbindung vom NoC an den MicroBlaze zu erstellen.

Die Kommunikation zwischen dem Server und dem SoC ist mit einer hohen Risiko Einschätzung versehen. Diese Einschätzung ist durch den Aufbau des TCP/IP Stacks sowie des verwendeten Betriebssystems gegeben. Beide sind durch ihre Optimierung auf Mikrocontroller Systeme in der Funktionalität stark eingeschränkt. Weiter ist bei der Implementierung die Kontrolle des Projekt-IPs zu beachten. Hier ist zu entscheiden, ob ein Standard-Treiber für alle Projekte

Tabelle 2: *Einschätzung der Risiken bei dem Erstellen der Masterarbeit.*

Kategorie	Tätigkeit	Risiko	Einschätzung
SoC	HW-Plattform	NoC, Speicheranbindung	Mittel/Hoch
	Software	Anbindung an Server, Projekt Treiber	Mittel/Hoch
	Beispiel-Projekt	Implementierung	Mittel
PC	Server	Architektur	Niedrig
	Projekt-Client	Architektur Implementierung	Mittel

realisiert wird oder die Projekte eigene Treiber verwenden, die entsprechend vom Server geladen werden.

Die Implementierung eines Beispiel-Projektes ist aufgrund der variablen Komplexität mit einer mittleren Risikoeinschätzung versehen. Hier kann beispielsweise ein Modul für Bruteforce-Angriffe auf Cäsar oder AES Verschlüsselungen implementiert werden.

4.2 Randbedingungen für das Projekt

Zusätzliche zu den Risiken bei der Erstellung der Arbeit, sind für die Etablierung des Projektes, aufgrund der Technologien, weitere Risiken einzuschätzen. Ein erhöhtes Risiko hierbei ist, dass für jeden FPGA Hersteller, eigene Synthese-Werkzeuge sowie ein eigenes SoC-System notwendig, sind da die jeweiligen Tools und IP-Module nur für die eigenen FPGAs lizenziert sind. Weiter ist für jeden FPGA-Typ ein Synthesedurchlauf notwendig, was je nach Umfang der Module einige Stunden in Anspruch nehmen kann. Diese Schritte können leicht automatisiert werden, so dass die Synthese am Anfang des Projektes für alle verfügbaren FPGA-Typen durchgeführt wird, wodurch die PRMs anschließend für alle Systeme zur Verfügung stehen.

Ein weiteres Risiko sind die erhöhten Kosten für die Lizenzen zur Partiellen Rekonfiguration. So betragen die Kosten, bei Xilinx, für eine Universitäts-Lizenz etwa 1000€, Firmen Lizenzen sind deutlich teurer. Hinzu kommt, dass die Entwicklungszeit bei einer reinen Software Lösung kürzer ist, als bei einer Hardware basierten Lösung. Gegen diese Zeiten ist der erreichte Durchsatz zu stellen und je nach Projekt zu entscheiden, welche Implementierung die vorzuziehende Variante darstellt.

5 Zusammenfassung

Der Bau von Supercomputern ist für die stetig wachsende Komplexität von zu berechnenden Daten notwendig. Die Konstruktion und der Aufbau des zur Zeit schnellsten Supercomputers, des Tianhe-1A, kosteten ca. 88 Millionen USD. Diese Kosten sowie die jährliche Unterhaltung, von ca. 2,7 Millionen USD sind, vor allem für universitäre Projekte, nicht für alle finanzierbar, so dass hier alternative Rechenplattformen zu schaffen sind.

Hier bieten sich insbesondere verteilte Berechnungen an, wie das „Distributed Computing“. Dieses verbindet PCs über eine Netzwerk-Infrastuktur zu einem virtuellen Supercomputer. Das

bekannteste dieser Netzwerke ist das BOINC, welches die Infrastruktur für vielseitige wissenschaftliche Projekte liefert. Beispielsweise sind hier Projekte aus den Bereichen der Klimaforschung, der Astrologie und der Pharmazie vertreten.

Die Berechnungen, die für diese Projekte durchgeführt werden, werden auf handelsüblichen PCs durchgeführt. Eine hohe Effizienz dieser Systeme ist durch die serielle Abarbeitung sowie die Unterbrechung durch anderes Threads oder Interrupts nicht gegeben. Hier bietet es sich an, zuzüglich zu den bisherigen Systemen, FPGAs als Beschleuniger-Plattform zur verwenden. Mit der Partial Reconfiguration Technologie ist es hier möglich ein statisches SoC-System zu erstellen, welches Teile des FPGAs während des Betriebes neu programmiert. Durch die Anbindung solcher SoC-Systeme an einen Server, welcher verschiedene Projekte verwaltet, wird ein FPGA basiertes Distributed Computing System geschaffen.

Der Aufbau des Systems basiert auf der Server/Client Architektur. Die SoCs sowie die Projekt-Clients stellen zu Beginn eine Verbindung zu einem Server her, welcher die einzelnen Teilnehmer verwaltet. Diese Architektur wird vorwiegend durch die Eigenschaften des verwendeten TCP/IP-Stacks auf dem SoC bestimmt, welcher nur eine begrenzte Anzahl an Verbindungen verwalten kann. Der FPGA selbst ist in zwei Subsysteme aufgeteilt, dem statische und dem dynamischen. Die Funktionalität des dynamischen Subsystems wird hierbei vom Projekt bestimmt, während das statische Subsystem die Kontrolle des SoCs übernimmt. Das hier entwickelte Komzept basiert auf Xilinx FPGAs und wird mit dem MicroBlaze Mikroprozessor gesteuert. Die Verbindung zum Server erfolgt über einen Ethernet-Controller. Zu dem enthält das statische Subsystem einen Memory-Controller zum Speichern der Projekt-Module, sowie der zur Berechnung verwendeten Daten und der Ergebnisse.

Der Server ist unterteilt in eine Verwaltungseinheit für die SoCs sowie eine Einheit für die Projekte. Die SoCs teilen dem Server bei der Anmeldung ihre Konfiguration mit und werden entsprechend kategorisiert. Bei der Anmeldung der Projekten teilen diese dem Server mit, für welche Konfiguration sie Module bereithalten und der Server teilt die Ressourcen den einzelnen Projekten anschließend zu. Für den Aufbau des Projekt-Clients werden drei Komponenten empfohlen, zum einen der Task-Scheduler, welcher die Verbindung zum Server verwaltet und die Aufgaben auf die SoCs verteilt. Zum anderen besteht es aus der Datenbank, die die Module sowie die Daten und Ergebnisse verwaltet, und der Verifizierung, welche die Korrektheit der Daten überprüft.

Die Risiken bei der Erstellung der Masterarbeit werden vorwiegend durch die Erstellung des SoC-Systems bestimmt. Die Anbindung des dynamischen an das statische Subsystem sowie den Speicher stellt hier ein Risiko dar. Weiter ist die optimale Verwendung des Betriebssystems mit dem TCP/IP-Stack ein zeitliches Risiko. Im Bereich der PC-seitigen Komponenten des Projektes stellt die Architektur des Projekt-Clients ein erhöhtes Risiko dar, da hier eine geeignete Beispiel- Implementierung gefunden werden muss.

Literatur

[California 2010] CALIFORNIA, University o.: *Open-Source Software für Volunteer Computing und Grid Computing*. 2010. – Verfügbar Online unter `http://boinc.berkeley.edu/`; 15.12.10

[Güneysu u. a. 2007] GÜNEYSU, Tim ; RUPP, Andy ; SPITZ, Stefan: *Cryptanalytic Time-Memory Tradeoffs on COPACOBANA*. 2007. – Verfügbar Online unter `http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/publications/conferences/copa_tmto.pdf`; 12.01.11

[Mamegani 2010] MAMEGANI, Armin J.: *Erprobung und Evaluierung von Methoden zur partiellen und dynamischen Rekonfguration eines SoC-FPGAs*, HAW-Hamburg, Masterarbeit, 2010

[Max Planck 2010] MAX PLANCK, Geselschaft: *Einstein@Home*. 2010. – Verfügbar Online unter `http://einsteinathome.org/`; 22.12.10

[NVIDIA 2010] NVIDIA: *NVIDIA Tesla GPUs Power World's Fastest Supercomputer*. 2010. – Verfügbar Online unter `http://pressroom.nvidia.com/easyir/customrel.do?easyirid=A0D622CE9F579F09&version=live&prid=678988&releasejsp=release_157`; 15.12.10

[Olson-Laboratory 2010] OLSON-LABORATORY: *FightAIDS@Home*. 2010. – Verfügbar Online unter `http://fightaidsathome.scripps.edu/`; 2.01.11

[Opitz 2010] OPITZ, Frank: *Projektbericht: Dynamic Reconfiguration of Accelerators in System on Chips*, HAW-Hamburg, Diplomarbeit, 2010

[Oxford 2010] OXFORD, University: *The world's largest climate forecasting experiment for the 21st century.* 2010. – Verfügbar Online unter `http://climateprediction.net/`; 15.12.10

[Raman 2010] RAMAN, Manikandan: China Makes World's Fastest Supercomputer. In: *International Business Times* (2010). – Verfügbar Online unter `http://www.ibtimes.com/articles/76731/20101028/tianhe-1a-tianhe-supercomputer-fastest-supercomputer-china-us-nvidia-am` `htm`; 17.12.10

[Wikipedia 2010] WIKIPEDIA: *Supercomputer — Wikipedia, Die freie Enzyklopädie*. 2010. – Verfügbar Online unter `http://de.wikipedia.org/w/index.php?title=Supercomputer&oldid=82613882`; 15.12.10

[Xilinx 2006] XILINX: *Programming Modern FPGAs*. 2006. – Verfügbar Online unter `http://www.xilinx.com/univ/mpsoc2006keynote.pdf`; 15.12.10

[Xilinx 2008] XILINX: *Partial Reconfiguration Software Training*. 2008

Abbildungsverzeichnis

1 Architektur des Distributed Computing Systems mit den einzelnen Komponenten und der Kommunikation untereinander 4
2 SoC-System mit MicroBlaze und der „Partiel Rekonfiguration Region“, mit Kommunikation über Bus Makros zur Synchronisation der Signale. Das System besteht aus einem MicroBlaze, der zur Ausführung des μC/OS-2 verwendet wird, sowie beispielsweise Interrupt Controllern (IntC) oder Timern. Die PRM werden über das SysACE Modul von einer CompactFlash Karte gelesen und vom ICAP in den FPGA geschrieben [Opitz (2010)]. 5
3 SoC des Distributed Computing Systems mit statischem und dynamischem Subsystem 7
4 Komponenten Diagramm der Software des DHPEC Projekts. Der SoC-Client besteht dabei aus dem RTOS und der Software zum Übertragen der Daten. . . . 8
5 Ablaufdiagramm des DHPEC Projektes am Beispiel eines Projektes und eines SoCs ii

A Ablaufdiagramm des DHPEC Projektes

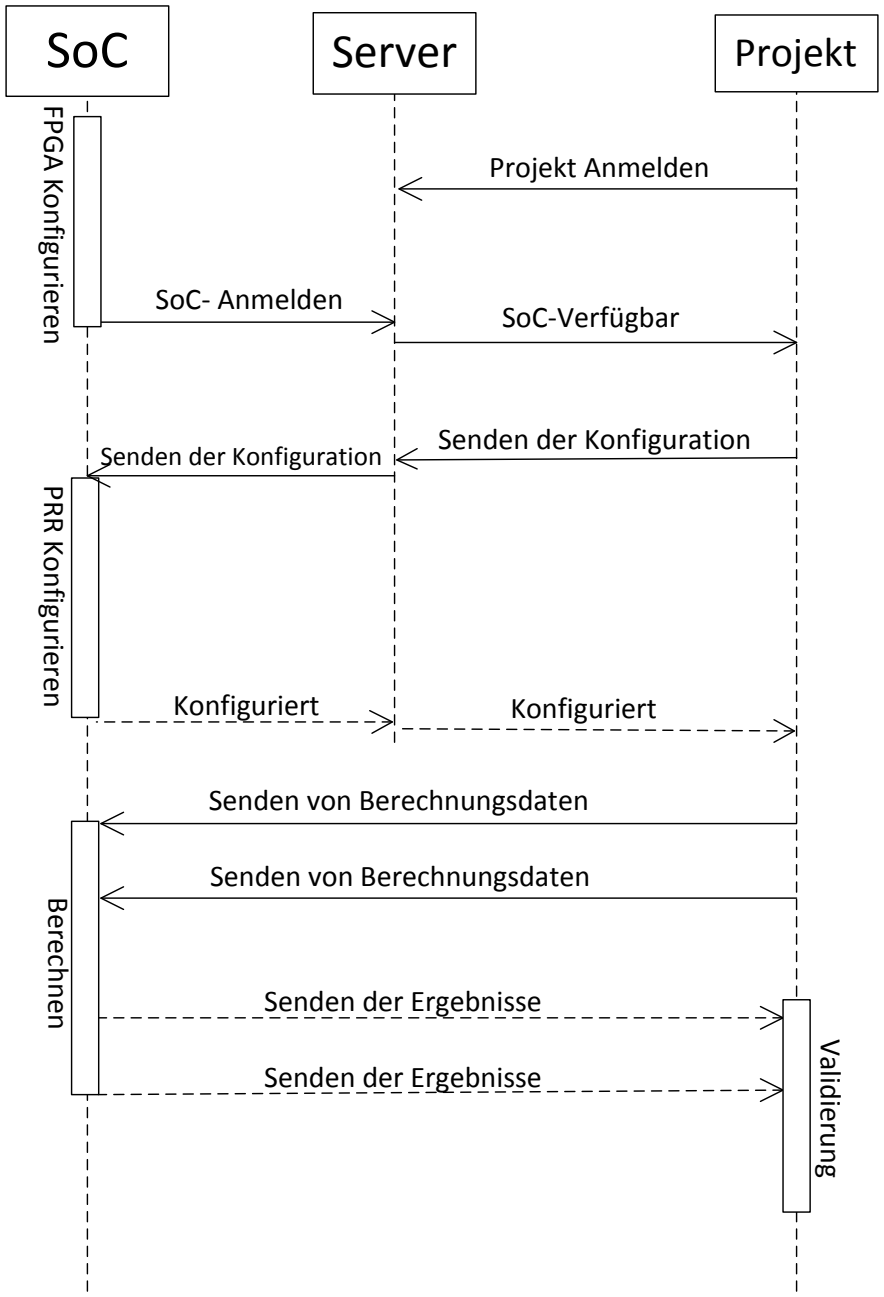

Abb. 5: *Ablaufdiagramm des DHPEC Projektes am Beispiel eines Projektes und eines SoCs*