Partielle Rekonfiguration von FPGA basierten SoCs: Seitenblicke

Frank Opitz

Ausarbeitung

Fakultät Technik und Informatik
Department of Engineering and Computer Science

Frank Opitz

Partielle Rekonfiguration von FPGA basierten SoCs: Seitenblicke

Ausarbeitung eingereicht im Rahmen der Veranstaltung Anwendung 2
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

Die „Dynamische Partielle Rekonfigurations-Technologie“ (PR) ähnelt der aus der frühen Software Entwicklung bekannten „Overlay“ Technik. Aufgrund des knappen Haupt-Speichers wurden hier Teile eines Programms mit einer anderen Funktion überschrieben, sobald diese benötigt wurde (vgl. Abbildung 1). Während im frühen Verlauf des Programms, die Funktion „fb“ verwendet wird, so wird, der für diese Funktion genutze Speicher, im späteren Verlauf, mit der Funktion „fa“ überschrieben [Glatz (2010)]. Die zu überschreibenden Bereiche müssen vorher dem Linker bekanntgegeben werden und können, im Gegesantz zu PR, in beliebiger Tiefe geschachtelt werden. Während die „Overlay“ Technik mit geringeren Ressourcen, als eigentlich

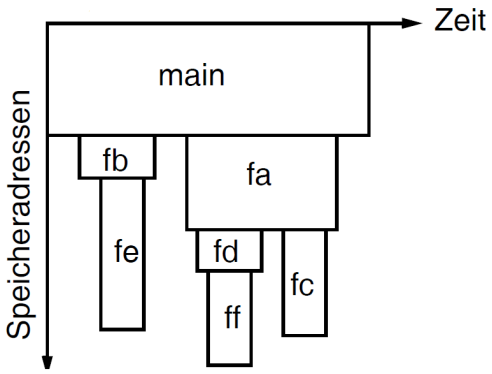

Abb. 1: *Speicherbelegung mit der Overlay Technik*

benötigt, auskommt, wird die Ausführung der Programme durch das Kopieren der Daten und Instruktionen verlangsamt. Im Gegensatz dazu wird bei der PR nur das rekonfigurierte Modul angehalten und ausgetauscht, die anderen Teile des Systems laufen ohne Einschränkungen weiter. So können auch FPGAs, die System kritische Module beinhalten, während des Betriebs Teile ihre Funktionalität austauschen.

Unter der Verwendung der PR wird die Funktion von FPGA basierten Systemen während des Betriebes erweitert oder an eine veränderte Umgebung angepasst. Weitere Anwendungen dieses Verfahrens sind beispielsweise:

- Teilen von Hardware-Modulen zwischen verschiedenen Anwendungen
- Wiederverwendbarkeit der Ressourcen in einem Design
- Wartung von entfernten Systemen
- Reduzieren des Energie verbrauchs

Diese Ausarbeitung beeinhaltet die Analyse verschiedener PR Projekte, welche aus einem breitgefächertem Anwendungsgebiet kommen. Vorgestellt wird das Thema des „Software defined Radio“ (SDR), welches einen großen Anklang in der Industrie findet (vgl. Kapitel 2.1). Kapitel 2.2 beschreibt eine Arbeit, welche die PR-Technologie bei Drohnen einsetzt, um Funktionalitäten innerhalb eines Schwarms von Drohnen auszutauschen. Ein Projekt zur Sicherung von Intellectual-Property (IP) beschreibt Kapitel 2.3. Die Kapitel 2.2 und 2.3 beinhalten zusätzlich jeweils eine Bewertung der Verwendung der PR Technologie. Anschließend folgt eine aktuelle Einschätzung der PR-Technologie.

2 Kommerzielle und wissenschaftliche Projekte

2.1 „Software defined Radio und Partielle Rekonfiguration“

In [Legat (2009)] wird die Anwendung der PR-Technologie in der Funk gestützten Kommunikation beschrieben. Hierbei wird eine Erweiterung des „Software defined Radio“ (SDR) erläutert.

In gegenwärtigen Funksystemen werden für die jeweiligen Standards dedizierte Lösungen verwendet, welche zumeist aus „Anwendungsspezifischen Integrierten Schaltungen“ (ASIC) bestehen. Diese beeinhalten die in Abbildung 2 dargestellten Schichten eines Funksystems. Durch diese Art der Implementierung ist der Wechsel eines Standards stark eingeschränkt. Ist das System in der Lage mehrere Standards zu verwenden, so werden die jeweiligen dedizierten Lösungen verwendet, wodurch z.B. der Energiebedarf sowie der Platzbedarf ansteigt .

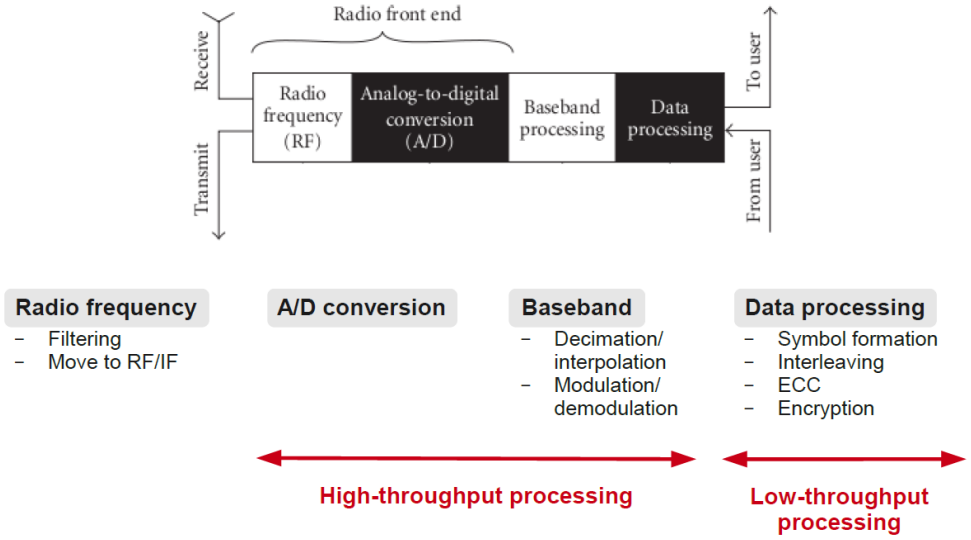

Abb. 2: *Darstellung der Funktionalen Blöcke, welche durchlaufen werden, um aus den Funkwellen, Informationen zu erhalten, welche vom Menschen verarbeitet werden können [Legat (2009)]*

Einen hoher Grad an Flexibilität, wie er z.B. bei der Kommunikation mit ständig wechselnden Teilnehmern benötigt wird, wird durch das verwenden des SDR-Verfahrens, anstelle der dedizierten Lösungen, erreicht. Bei diesem Verfahren, werden soviele Schichten wie möglich in Software realisiert. In der Regel sind dies die „Baseband“ und „Data processing“ Schichten. So ist es möglich einen „General Purpose Processor“ (GPP) für die Kommunikation und die weitere Verarbeitung zu verwenden, wodurch die Komplexität sowie der Energiebedarf des Systems abnimmt. Die Verarbeitung der Daten in Software erfordert im Gegenzug eine Erhöhung des Prozessortaktes, um einen gleich bleibenden Datendurchsatz zu erhalten, wodurch der Energiebedarf durch den GPP wieder ansteigt.

Die Verwendung des SDR erlaubt es mit Systemen zu kommunizieren, welche in der Standard-Konfiguration nicht kompatibel sind. Um dies zu tun sind während des Betriebes folgende Schritte durchzuführen:

1. Verschicken von Informationen über den Standard:
   Das Verschicken des Standards, wie z.B. GSM oder GPRS, erfolgt in der Regel verschlüsselt und beinhaltet alle Informationen, die zur Verarbeitung der Signale benötigt werden.

2. Programmieren der Funksysteme:
   Je nachdem, auf welchen Standard sich geeinigt wurde, müssen die entsprechenden Geräte programmiert werden, um die Kommunikation untereinander zu ermöglichen.

Eine Interoperabilität der Systeme wird durch die Verwendung eines entsprechenden Standards erreicht. Einer dieser Standards ist die „Software communication architecture“, welche von der „Joint Tactical Radio System“-Abteilung des „Department of Defense“ erstellt wurde . Dieser Standard unterteilt das System in zwei Hauptkomponenten (vgl. Abbildung 3)[Gailliard u. a. (2007)].

- „Application Layer“:
  Auf der Application Schicht läuft die vom Software der Kommunikation. Der Entwickler ist von der Hardware losgelöst und die SDR-Module sind untereinander austauschbar.
- „Platform Layer“:
  Die Plattform Schicht kapselt das eigentliche System vor dem Anwender und ist wiederum in zwei Schichten unterteilt.
  - „Hardware Layer“:
    Die Hardware Schicht besteht aus den Komponenten, welche zur Ausführung der Software oder der Wandlung der Analogen Funksignale in digitale verwendet werden, wie z.B. einem GPP oder AD/DA-Wandler. Aufgrund der „Abstraction“ Schicht ist der genaue Aufbau vor dem Anwender verborgen und kann so je nach System variieren.
  - „Abstraction Layer“:
    Die Abstraction Schicht kapselt grundlegende Software, wie z.B. ein „Real Time Operating System“ (RTOS) vor dem Anwender und stellt für jedes System gleiche Schnittstellen zur Verfügung.

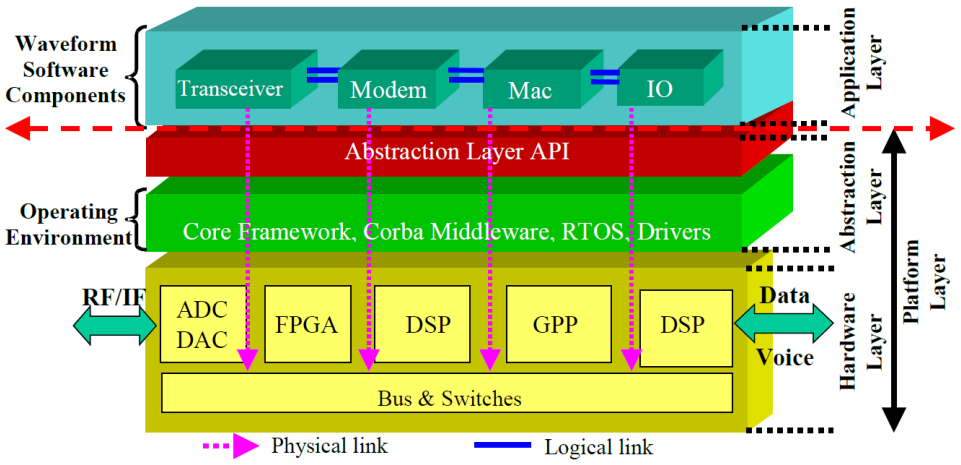

Abb. 3: *Schichten des SCA Standards mit den enthaltenen Modulen [Gailliard u. a. (2007)]*

Aktuelle Systeme, die die SDR Technologie Einsetzen, verwenden in der Regel eine kombination aus GPP und Digital Signal Processor (DSP). Diese Kombination ist nicht energieeffizient, da beide Komponenten mit sehr hohen Taktraten betrieben werden und stößt bei der Verarbeitung mehrerer Funk-Standards, die parallel verarbeitet werden, schnell an seine Grenzen. Eine bessere Energieeffizienz wird durch die Kombination von GPP und FPGA oder durch eine SoC

Architektur, wie der Verwendung eines MicroBlaze Softcore auf einem FPGA, erreicht. Bedingt durch die parallelen Rechenressourcen eines FPGAs lassen sich im Vergleich zu DSPs und GPPs hohe Signalverarbeitungsraten mit niedrigeren Taktfrequenzen realisieren.

Unter der Verwendung der PR-Technologie werden die Kommunikationsstandards in SDR-Plattformen wieder in Hardware verarbeitet. Anstelle der Software werden hier IPs zur Berechnung versendet und auf der Zielplattform ausgeführt.

2.2 „Simulation einer Partiellen Rekonfiguration in einer mobilen Agenten Umgebung“

Der Artikel [Kearney und Jasiunas (2006)] befasst sich mit der Verteilung von Hardware-Modulen innerhalb eines Schwarms von Drohnen oder der Portierung eines Drohnen-Zustandes auf eine andere, wenn diese ausgetauscht werden. Hierbei wird insbesondere auf kleine „Unpiloted Airborne Vehicles“ (Drohnen, UAV), die in der Ausstattung und der Energieversorgung sehr eingeschränkt sind, eingegangen.

Der Einsatz von Drohnen wird in den nächsten 20 Jahren einen Großteil der Flugzeug-Industrie beschäftigen und der Forschungsetat ist in den Jahren 2003-2013 auf 5,5 Milliarden Euro geschätzt [SpaceDaily (2004)]. Die Datenverarbeitung von Kameras und kleinen RF-Sensoren ist in den meisten Fällen die Aufgabe dieser Drohnen. Die Verarbeitung der Daten erfolgt oft in FPGAs, da diese die Daten parallel und mit einem determiniertem Zeitverhalten abarbeiten, was mit einer CPU, auf der auch ein OS beziehungsweise kritische Applikationen laufen, nicht möglich ist.

Damit eine größere Genauigkeit der Daten erreicht wird, sind mehrere Drohnen gleichzeitig zu verwenden, so wird z.B. der Fehler einer Ortung um mehr als 80% verringert. Die Verwendung eines Drohnen Schwarms wird in dem Artikel in zwei Gruppen unterschieden.

- „Single Mission“:
  Es sind für die Mission N Aufgaben zu erfüllen und es stehen N Drohnen zur Verfügung. Hierbei wird angenommen, das die Aufgaben so verteilt sind, dass die Drohnen gleichzeitig zur Basis zurückkehren. Hierbei ist es das Ziel, die Zeit der Ausführung zu maximieren.
- „Continuous Mission“:
  In diesem Szenario existieren mindestens $N + 1$ Drohnen für N Aufgaben. Hierdurch können Drohnen mit geringem Energievorrat durch eine voll aufgeladene Drohne ersetzt werden. Bei Aufgaben, die die Drohnen in unterschiedliche Zustände versetzen, sind die Zustände auf die übernehmende Drohne zu übertragen so dass eine Mobilität der einzelnen Applikationen vorauszusetzen ist.

Diese Mobilität wird z.B. über das Agenten Paradigma erreicht [Wooldridge und Jennings (1995)]. Insbesondere die mobilen Agenten sind in der Lage, während des Betriebes zwischen Host-Systemen zu migrieren, wobei jeweils eine spezielle Umgebung zur Ausführung der Agenten verwendet wird. Diese Umgebung stellt hier das OS dar. Dieses OS hätte zusätzlich, die Verwaltung der Hardware (HW) Ressourcen im FPGA zu übernehmen und die HW-Module der Agenten an die freien Ressourcen des FPGAs anzupassen. Dies ist auf grund, der aktuellen Beschränkungen der FPGAs, wie festgelegte Bereiche zur PR, aber noch nicht möglich. Aus diesem Grund wird die Partielle Rekonfiguration simuliert, welches mit Hilfe eines „Checkpointing“ Verfahrens implementiert wird.

Bei der Simulierten Partiellen Rekonfiguration wird jeweils der komplette FPGA mit der neuen Konfiguration überschrieben, welche vorher synthetisiert wurde und als Bitstream bereit liegt. Hierbei gehen alle Zustände des Systems verloren. Um dies zu verhindern, wird das „Checkpointing“ eingesetzt. Dieses sichert die Zustände der einzelnen HW-Module und der Software und wird in zwei unterschiedlichen Varianten eingesetzt.

- „Cooperative“:
  Bei dieser Variante teilt das OS den Tasks mit, dass eine Rekonfiguration nötig ist und diese Sichern darauf hin die eigenen Zustände. Ein Vorteil dieses Verfahrens ist, dass während des normalen Betriebes keine Rechenzeit für die Sicherung der Zustände verwendet wird. Nachteilig ist, dass nach der Nachricht und der Sicherung aller Zustände eine lange Zeitspanne verstreicht, in welcher die Zustände erst gesichert werden.
- „Preemptive“:
  Die einzelnen Tasks sichern periodisch ihre Zustände, so dass sie jeweils bei dem letzten „Checkpoint“ nach einer Rekonfiguration wieder starten. Der Vorteil dieser Variante liegt darin, dass eine Rekonfiguration durchgeführt werden kann, sobald diese angefordert wird. Nachteilig wirkt sich die zur Sicherung benötigte Rechenzeit auf die Effektivität aus, da immer wieder eine Sicherung durchgeführt wird, welche nicht immer gebraucht wird.

In dem vorgestellten Projekt wird das „Preemptive Checkpointing“ eingesetzt. Dazu wurden die Applikationen in logische Blöcke unterteilt, welche als atomare Einheiten behandelt werden. Diese Atomaren Einheiten werden dann als Zustände in einem Automaten betrachtet und die Variablen bei jedem Zustandsübergang gesichert, so dass jederzeit eine Rekonfiguration stattfinden kann.

Zusätzlich zu der Kommunikation zwischen den einzelnen Drohnen und der Migration der Agenten behandelt das Projekt die Verteilung von FPGA externen Ressourcen wie Speicher. Um diese an die einzelnen Module zu verteilen, sind ein Netzwerk, ein Arbiter und eine Strategie zur Zuteilung der Ressourcen entscheidend (vgl. Abbildung 4).

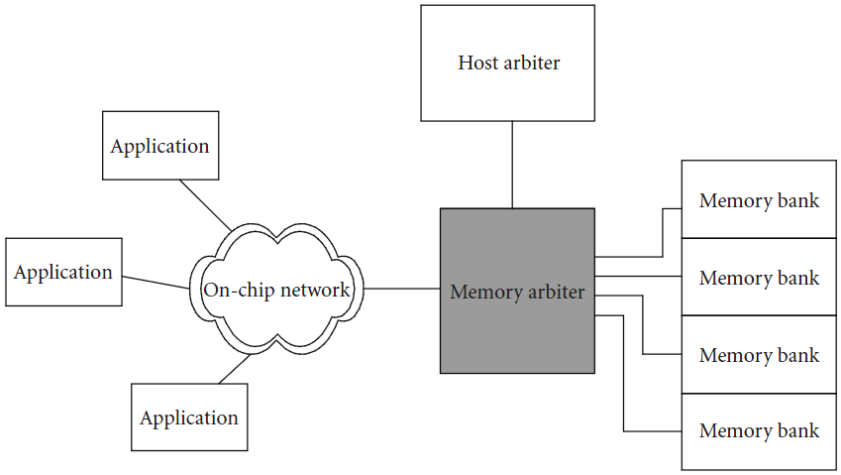

Abb. 4: *SoC Netzwerkarchitektur bestehend aus dem Netzwerk mit den HW-Module sowie dem Memory-Arbiter. Der Host-Arbiter wird durch einen GPP Implementiert. [Kearney und Jasiunas (2006)]*

Bei der Wahl der Netzwerk Architektur wurde auf folgende Punkte besonderen Wert gelegt.

- Implementierung:
  Wie einfach lässt sich die jeweilige Architektur implementieren.
- Routing Kosten:
  Wieviele Pfade sind notwendig, um eine neue Applikation an das vorhandene System anzukoppeln.
- Nebenlaufigkeit:
  Wie weit wird Nebenläufigkeit unterstützt.
- Latenz:
  Wie verhält sich die Latenz bei verschiedenen Größen des Netzwerks.
- Skalierbarkeit:
  Wie Skaliert das Netzwerk bei einer großen Anzahl an Applikationen.

Der Vergleich der zur Auswahl standen Netzwerk-Topologien ist in Tabelle 1 zu finden. Aufgrund der hohen Anforderung von Drohnen an die Nebenläufigkeit sowie die Latenz wird hier die „Star“ Topologie verwendet. Die geringe Skalierbarkeit ist hier vernachlässigbar, da nur eine kleine Anzahl an Modulen auszutauschen ist. Die Verteilung des zur Verfügung stehen Speichers erfolgt statisch, d.h. jede Applikation bekommt einen definierten Speicherbereich und es ist somit keine dynamische Strategie erforderlich. Der Arbiter selbst ist nach dem Round-Robin Prinzip aufgebaut und verbindet Applikationen mit dem Speicher sowie anderen HW-Ressourcen.

Tabelle 1: *Eigenschaften von Netzwerk Topologien in der Implementierung. (+ = Gut, - = schlecht, +/- = neutral)*

	East of implementation	Wire routing cost	Concurrency	Latency	Scalability
Bus	$++$	$-$	$--$	$-$	$--$
Star	$++$	$--$	$++$	$++$	$--$
Mesh	$--$	$--$	$+$	$+$	$+$
Ring	$++$	$+$	$-$	$+/-$	$+/-$
Tree	$+/-$	$++$	$+/-$	$+$	$+$
Fat tree	$-$	$-$	$+$	$+$	$+$

Bei der Verteilung der Applikationen zwischen den Drohnen wird das Agenten Paradigma verwendet. Unterschieden wird zwischen statischen Agenten, die z.B. eine Ressource wie eine Kamera darstellen, und mobilen Agenten, welche die austauschbaren Applikationen repräsentieren. Anders als andere Paradigmen erlauben es Agenten, dass nicht nur die Ausführung zwischen den Host-Plattformen wechselt, sondern auch der jeweilige Zustand, wobei für jeden Agenten spezielle Regeln festgelegt werden können. Die einzelnen Agenten sind hier über ein einzigartiges Tupel im jeweiligen Netzwerk adressierbar. Dieses Tupel besteht aus:

- „sequence number“ und „home node“ : Eine einzigartige Identifikationsnummer und dem eigentlichen Heimatknoten.
- „class“ : Der Typ des Agenten
- „current node“ : Der Knoten auf dem der Agent aktuell ausgeführt wird.
- „ability list“ : Eine Liste an Fähigkeiten, die der Agent besitzt.

Damit die einzelnen Agenten zwischen den Drohnen migrieren können, stellt die Netzwerkumgebung folgende Dienste bereit:

- Finden von anderen Agenten
- Kommunikation mit anderen Agenten
- Abfragen von Informationen über anderen Drohnen
- Migration der Agenten zwischen den Drohnen

Damit die Agenten die anderen adressieren können, erstellt jede Drohnen periodisch einen Snapshot mit ihren aktuellen Agenten. Dieser wird anschließend den anderen Drohnen mitgeteilt.

Die Migration der Agenten ist ein teurer Prozess, im Bezug auf Energie sowie Durchsatz (um die Downtime des Agenten so gering wie möglich zu halten), daher benötigt der Agent vorher Informationen über das Zielsystem wie freie FPGA Ressourcen oder den aktuellen Energieverbrauch. Sind diese Information vorhanden und der Agenten möchte migrieren, läuft dieser Prozess wie folgt ab:

1. Der Agent schickt eine Anfrage an die Ziel-Drohne
2. Die Ziel Drohne erstellt eine Instanz des Agenten im „Sleep Mode“
3. Die neue Instanz bekommt eine temproräre Adresse, welche der anfragenden Instanz zugeschickt wird.
4. Der Agent stoppt seine Berechnungen und schickt die Daten des erreichten Checkpoints an das Ziel
5. Die neue Instanz übernimmt die Zustände der alten und beginnt die Arbeit, während die alte Instanz abgeschaltet wird.

2.2.1 Bewertung

Die Autoren dieses Projektes haben sich dazu entschieden die eigentliche PR durch ein statisches Verfahren zu simulieren. Dies erfolgt, in dem jede mögliche Belegung der FPGAs synthetisiert wurde. Findet dann eine Rekonfiguration statt, so wird der komplette FPGA rekonfiguriert. Dieses Verfahren ist für die Praxis ungeeignet, da jeweils die Zustände aller Module gesichert werden müssen und zweitens auch sicherheitskritische Module, die auf dem FPGA laufen, kurzzeitig abgeschaltet und neu gestartet werden.

Vermieden werden könnte dies durch das Einsetzen der PR-Technologie. Durch die festgelegten Bereiche wäre zwar ein Verlust der Flexibilität der Module, welche während der Synthese auf ein einen Bereich fest zu legen sind und ein zusätzlicher Verbrauch der FPGA Ressourcen in Kauf zu nehmen, dies wird aber durch die Schnelligkeit der Rekonfiguration sowie die Praxistauglichkeit wieder ausgeglichen. Dies ist dadurch zu erreichen, dass nur Teile des FPGA rekonfiguriert werden und systemkritische Module weiter arbeiten.

2.3 „Sicherung von HW-Modulen vor unauthorisierter Benutzung“

Der Artikel [Hori u. a. (2008)] beschäftigt sich mit der Sicherung von IP-Modulen, die bei der Rekonfiguration vom FPGA aus einem externen Speicher gelesen werden. Diese Module liegen in einem bekannten Format, z.B. als .bit bei Xilinx, vor und könnten, nach dem Auslesen mit einem Logic-Analyser, analysiert und weiter verwendet werden. Einige FPGAs besitzen einen HW-Block, der einen Bitstream entschlüsseln kann, diese können aber nur komplette Konfigurationen verarbeiten, wodurch Rekonfigurationsmodule weiterhin unverschlüsselt übertragen und gespeichert werden. Ein weiteres Problem der ungesicherten Speicherung bzw. Übertragung ist, dass ein fehlerhafter Bitstream das System beschädigen kann. Dies kann z.B. durch eine fehlerhafte Beschaltung der I/O-Pins erfolgen. Aus diesem Grund wird eine Kombination aus Verschlüsselung und Authentifizierung („authenticated encryption“, AE) angestrebt. Erreicht wird das durch die Kombination des „Advanced Encryption Standard“ (AES) und dem „Galois/Counter Mode“ (GCM). Diese beiden Algorithmen zeichnen sich weiter durch die Eigenschaft aus, dass sie pipelinefähig und parallelisierbar sind, wodurch sie sich für die Implementation in Hardware eignen. Damit ein Vergleich zwischen dem AES-GCM Verfahren und anderen besteht, wurde außerdem ein System mit AES-CBC und SHA-256 implementiert.

Der verwendete Verschlüsselungs-Algorithmus AES wurde 2001 als Nachfolger des „Data Encryption Standard“ bekanntgegeben und im Rahmen einer Ausschreibung des „National Institute of Standards and Technology“ (NIST) ermittelt [of Standards und Technology (2001)]. AES verwendet eine symmetrische Blockchiffre Verschlüsselung. Im Gegensatz zu dem eigentlichen Rijndael Algorithmus beschränkt AES die Blockgröße auf 128 Bit, erlaubt aber weiterhin eine Schlüssellänge von 128, 192 oder 256 Bit. Für die Verwendung von Blockchiffre existieren verschiedene Modi wie z.B. „Electronic Code Book“, „Cipher Block Chaining“ oder „Counter Modus“. Das Verwendete GCM ist eine Erweiterung des „Counter Modus“. Hierbei wird ein Initialisierungsvektor (Nonce) mit einem Zähler konkatiniert, welche dann mit dem Schlüsel und dem Klar-Text XOR-verknüpft wird (vgl. Abbildung 5).

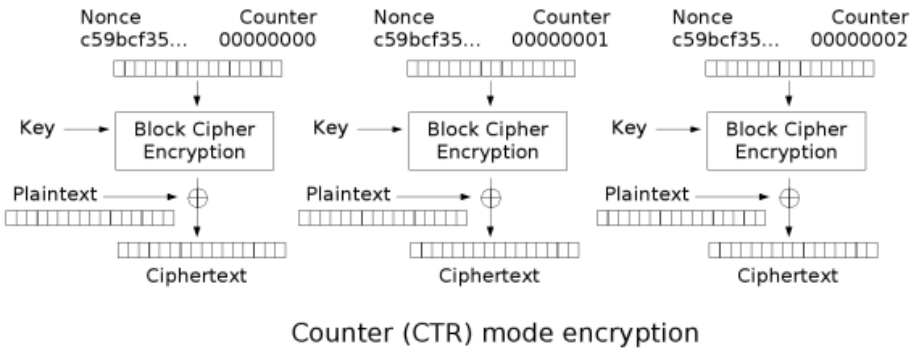

Abb. 5: *Funktionsweise des Counter Modus beim Blockchiffreverfahren*

Der Aufbau des Systems zur Entschlüsselung und Authentifizierung besteht aus drei Pfaden (vgl. Abbildung 6). Der erste Pfad ist der Verschlüsselungspfad, welcher aus dem AES-CTRL und AES-GCM Modul besteht. Der zweite Pfad beeinhaltet die Module zur Speicherung der Daten sowie das ICAP. Der letzte Pfad besteht aus der PRR und den Datenverarbeitungsmodulen des Systems. Die Entschlüsselung erfolgt mit einem Schlüssel, der in das System integriert ist und nicht geändert werden kann. Wird ein Datenstrom mit Konfigurationsdaten empfangen, so wird dieser entschlüsselt, während die restlichen Daten empfangen werden und anschließend

im Block-Ram gespeichert. Ist der ganze Datenstrom empfangen, so wird die „Partial Reconfiguration Region“(PRR) über den „Internal Configuration Access Port“ (ICAP) beschrieben.

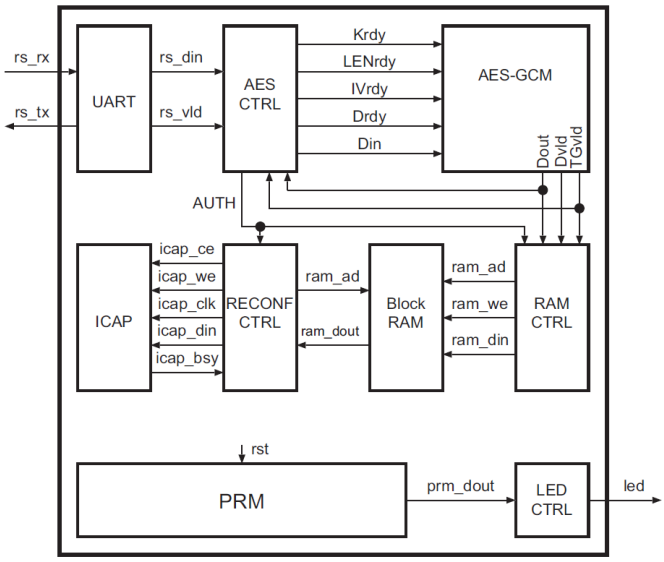

Abb. 6: *Aufbau des AE-Systems bestehend aus dem Verschlüsselungs Pfad, dem Rekonfigurationspfad sowie der PRR [Hori u. a. (2008)]*

AES-GCM entschlüsselt, in dieser Implementation, einen 128 Bit Block in 12 Takten wobei nach dem letzten Block 10 Takte hinzukommen, um die Authenzität der nachricht zu ermitteln. Tabelle 2 zeigt den Datendurchsatz der AES-GCM und der AES-SHA256 Implementation. Weiter sind diesen Hardwarelösungen Softwarelösungen zu AE von PR-Modulen gegenübergestellt. Der Vergleicht zeigt, dass die AES-GSM Lösung bis 16087 mal Schneller ist als die anderen Implementationen.

Tabelle 2: *Datendurchsatz bei der Verschlüsselung und Authentifizierung der verschiedenen Implementationen*

System	Slice[1]	Authentifizierung	Entschlüsselung	Configuration	Gesamt	Ratio
PR-AES-GCM	2,687	106.43 μ s 1067 Mbps		35.3 μ s 3200 MBPS	141.73 μ s 797 Mbps	1
PR-AES-SHA256	2,730	160.97 μs 701 Mbps	97.14 μs 1164 Mbps	35.3 μs 3200 Mbps	196.27μs 575 Mbps	1.28
PowerPC[2]	1,334 [3]	139 ms 812 kbps	208 ms 543 kbps	56 ms 2016 kbps	403 ms 280 kbps	2843
MicroBlaze[2]	1,706	776 ms 145 kbps	1472 ms 77 kbps	32 ms 3528 kbps	2280 ms 50 kbps	16087

[1]Zum vergleich sind die Slices angegeben die auf einem Virtex-2 gebraucht werden, anstelle des im Projekt verwendetem Virtex-5

[3][Zeineddini und Gaj (2005)]

[2]Beinhaltet nur den Konfigurationskontroller

2.3.1 Bewertung

Die Implementationen der beiden AE Systeme haben eine Ressourcenbedarf von bis zu 22% der Slices sowie des Block-Rams (BRAM) (vgl. Tabelle 3). In der vorgestellten Implementierung werden diese Module immer im FPGA gehalten. Dieses ist, vorallem bei einer geringen Anzahl an Rekonfigurationen und somit einer geringen Verwendung der AE-Module nicht effizient. Hier bietet sich ein System an, welches in [Bossuet und Gogniat (2006)] vorgestellt wird. In diesem wird das Entschlüsselungsmodul auch als PRM verwendet. Im Gegensatz zu der hier vorgestellten Implementation wird der Schlüssel nicht im Modul gespeichert, da dieser sonst aus dem Bitstream gelesen werden kann. Die statische Version ist uneffizient bei Systemen, in denen eine hohe Rate an Rekonfiguration stattfindet, da jeweils erst das AE-System geladen wird und anschließend das gewünschte Modul. So ist je nach Anwendung zu entscheiden, welche Version die effizientere ist.

Tabelle 3: *Ressourcenbedarf der unterschiedlichen AE Implementationen auf einem Virtex 5(XC5VLX50T)*

Module	Register	(%)	LUT	(%)	Slice	(%)	BRAM	(%)
AES-CGM	2,5166	7	3,040	10	1,390	19	13	21
AES-SHA	2,327	8	3,661	12	1,592	22	13	22

3 Einschätzung der aktuellen Implementation der Partiellen Rekonfiguration

Die Analyse der Suchergebnisse zur Partiellen Rekonfiguration in den Datenbeständen der IEEE und ACM haben ergeben, dass die Verwendung dieser Technologie in der Wissenschaft bereitwillig aufgenommen und verwendet wird. In der Industrie wird, zum aktuellen Zeitpunkt, noch zurückhaltend reagiert. Gründe hierfür sind z.B.:

- Implementierung ist Herstellerspezifisch:
  Der Design-Flow für die Implementierung von PR-Systemen hängt vom jeweiligen Hersteller des FPGAs und den verwendeten Implementierungswerkzeugen ab.
- Keine „Online“-Synthese:
  PR-Module (PRM) sind während des Designs auf festgelegte Bereiche begrenzt und können nicht während des Betriebes auf die freien Ressourcen des FPGAs gelegt werden [Kao (2005)].
- Einschränkung in der Form der Module:
  Die Bereiche in denen die PRM liegen, werden z.B. zwischen zwei Slices aufgespannt. Mit dieser Vorgabe ergibt sich jeweils eine rechteckige Form der Module und eine an den Verbrauch der Module angepasste Form lässt sich nicht erstellen [Kao (2005)].

Die Integration des PR Design Flows in die Standard Xilinx Tools, sowie die Verfügbarkeit von PR Möglichkeiten in den FPGAs anderer Hersteller lassen aber erkennen, dass ein großer Bedarf an dieser Technologie vorhanden ist [Altera (2010)][Xilinx (2010)].

4 Zusammenfassung

Diese Ausarbeitung stellt unterschiedliche Projekte aus dem Bereich der Partiellen Rekonfiguration dar, welche aus industriellen und wissenschaftlichen Bereichen kommen. Diese Projekte nutzen jeweils eine Teilmenge der Vorteile der PR-Technlogie, wie z.B. Erweiterung des Funktionsumfanges oder der Wiederverwendung der FPGA Ressourcen in einem System.

Vorgestellt wurde unter anderem das „Software Defined Radio“ Verfahren. Hier wird, anders als in regulären Funksystemen, die De-/Kodierung etc. in Software realisiert. Eine einheitliche Plattform für die Hersteller der Funksysteme ist durch die vorgestellte „Software communication architecture“ gegeben, welche die HW-Plattform von den in Software realisierten Funkstandards trennt. Bei der Verwendung der PR-Technologie in einem SDR System, wird anstelle der Software ein Konfigurations-Bitstream ausgetauscht und der FPGA entsprechend konfiguriert.

Im zweiten Projekt wurde die Verwendung der Simulation einer PR-Technologie in einem Schwarm von Drohnen vorgestellt. Aufgrund der aktuellen Einschränkungen der PR Technologie haben sich die Autoren dazu entschieden diese zu simulieren, in dem jede mögliche Kombination aus Modulen synthetisiert und der komplette FPGA rekonfigueriert wurde. Das „Checkpointing“ Verfahren wurde verwendet, um die Zustände der einzelnen Teile zu sichern, welche bei einer Migration der Module auf eine andere Drohne mit versendet werden. Weiter wurden verschiedene Netzwerk Topologien evaluiert und die Stern Topologie als bestes Netzwerk für diese Verwendung befunden.

Das dritte Projekt beschreibt wie IPs geschützt werden können. Dies erfolgt aus einer Kombination von Authentifizierung und Verschlüsselung. Verwendet wird dazu der AES Algorithmus mit der GCM Erweiterung, welche dafür sorgt, dass der FPGA nur mit authentifizierten Modulen beschrieben wird, wodurch eine Beschädigung des Systems vermieden wird. Gezeigt wird, dass dieses Verfahren bis zu 16087 mal schneller ist als andere Implementationen eines AE Verfahrens.

In der Wissenschaft wird die PR Technologie bereitwillig aufgenommen. Die Industrie reagiert noch zurückhaltend auf diese Technologie durch die Beschränkungen wie z.B. herstellerspezifische Implementierungen sowie Einschränkungen in der Form der Module.

Literatur

[Altera 2010] ALTERA: *Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs*. 2010. – Verfügbar Online unter `http://www.altera.com/products/devices/stratix-fpgas/stratix-v/overview/partial-reconfiguration/stxv-part-reconfig.html`; 8.08.10

[Bossuet und Gogniat 2006] BOSSUET, L. ; GOGNIAT, G.: Dynamically confgurable security for SRAM FPGA bitstreams. In: *Int. J. Embedded Systems* 2 (2006), S. 73–85

[Gailliard u. a. 2007] GAILLIARD, Grégory ; NICOLLET, Eric ; SARLOTTE, Michel: *Transaction Level Modelling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM*. 2007

[Glatz 2010] GLATZ, Eduard: *Betriebssysteme - Grundlagen, Konzepte, Systemprogrammierung*. dpunkt.verlag, 2010. – ISBN 978-3-89864-678-9

[Hori u. a. 2008] HORI, Yohei ; SATOH, Akashi ; SAKANE, Hirofumi ; TODA, Kenji: BITSTREAM ENCRYPTION AND AUTHENTICATION WITH AES-GCM IN DYNAMICALLY RECONFIGURABLE SYSTEMS. (2008)

[Kao 2005] KAO, Cindy: Benefits of Partial Reconfiguration. In: *Xcell Journal* (2005)

[Kearney und Jasiunas 2006] KEARNEY, David ; JASIUNAS, Mark: Using Simulated Partial Dynamic Run-Time Reconfiguration to Share Embedded FPGA Compute and Power Resources across a Swarm of Unpiloted Airborne Vehicles. In: *'EURASIP Journal on Embedded Systems* (2006)

[Legat 2009] LEGAT, Jean-Didier: *Dynamically reconfigurable architectures for SDR in professional embedded systems*. 2009

[SpaceDaily 2004] SPACEDAILY: Both Civil and Military Needs Driving European UAV Market. In: *SpaceDaily* (2004). – Verfügbar Online unter `http://www.spacedaily.com/news/uav-04a.html`; 11.07.10

[of Standards und Technology 2001] STANDARDS, National I. of ; TECHNOLOGY: Announcing the advanced encryption standard (AES). In: *FIPS PUB 197* (2001)

[Wooldridge und Jennings 1995] WOOLDRIDGE, M. ; JENNINGS, N.: Intelligent agents: theory and practice. In: *Knowledge Engineering Review* 10 (1995), Nr. 2, S. 115– 152

[Xilinx 2010] XILINX: *Partial Reconfiguration User Guide (UG702)*. 2010. – Verfügbar Online unter `http://www.xilinx.com/tools/partial-reconfiguration.htm`; 8.08.10

[Zeineddini und Gaj 2005] ZEINEDDINI, A. S. ; GAJ, K.: Secure partial reconfiguration of FPGAs. In: *ICFPT* (2005), S. 155–162

Abbildungsverzeichnis