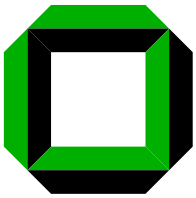

Universität Karlsruhe (TH)
Institut für
Betriebs- und Dialogsysteme
Lehrstuhl Systemarchitektur

Entwurf und Implementierung einer erweiterten Speicherkontrolleinheit

Bernd Ahues

Studienarbeit

Verantwortlicher Betreuer: Prof. Dr. Frank Bellosa
Betreuender Mitarbeiter: Dipl.-Inform. Raphael Neider

17. Dezember 2008

Hiermit erkläre ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Literaturhilfsmittel verwendet zu haben.

I hereby declare that this thesis is a work of my own, and that only cited sources have been used.

Karlsruhe, den 17. Dezember 2008

_____

Bernd Ahues

Abstract

Um die Energieeffizienz oder die Systemleistung zu erhöhen, ist es aus Sicht des Betriebssystems wünschenswert, Zugriffsprofile für Speicherbereiche erfassen und für die gezielte Migration dieser Bereiche in das für das Optimierungsziel beste Speichermodul verwenden zu können. Diese Arbeit behandelt den Entwurf und die Implementierung einer Hardwarekomponente, die dem Betriebssystem grundlegende Informationen und Methoden zur Verfügung stellen soll, die Aufteilung von Anwendungen und Daten auf Speichermodule zu verbessern. Die vorgestellte Komponente verwaltet sowohl mehrere Speichermodule als auch unterschiedliche Speichertypen und bietet zusätzlich Funktionen wie Speicher-Speicher-Transfers und Referenzzähler an. Damit wird es dem Betriebssystem möglich, zur Laufzeit Softwareteile zu identifizieren, sie in andere Speichermodule zu verschieben und somit die Energieeffizienz oder die Leistung des Systems zu erhöhen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Am Lehrstuhl für Systemarchitektur an der Universität Karlsruhe entsteht unter dem Namen OpenProcessor [1] eine Plattform, die den gemeinsamen Entwurf von Hardware und Betriebssystem und die Erforschung der Schnittstellen zwischen Hardware und Software ermöglichen soll. Mit dieser Plattform lassen sich neue Ansätze zur Vereinfachung und Unterstützung von Betriebssystemaufgaben durch die Hardware untersuchen.

Eine wichtige Aufgabe für heutige Betriebssysteme ist die Energieverwaltung, insbesondere da immer mehr Systeme mit begrenztem Energievorrat auskommen müssen.

Aktuelle Speichermodule bieten eine Reihe von Energiesparmodi, die jeweils unterschiedlichen Latenzen bei der Reaktivierung aufweisen. Es ist ungünstig, Module abzuschalten, die kurz danach wieder benötigt werden: Dies kann dazu führen, dass man sogar mehr Energie benötigt, als wenn man das Modul nicht in einen Energiesparmodus gesetzt hätte. Daher ist es wichtig zu wissen, auf welche Adressen weniger häufig zugegriffen wird, um diese Daten in einem Speichermodul zu sammeln und dieses länger in einen möglichst tiefen Energiesparmodus zu versetzen.

Die Information, welche Seiten wie häufig genutzt werden, ist jedoch nicht so einfach verfügbar, was fundierte Entscheidungen bezüglich der Aktivierung von Energiesparzuständen schwierig macht. Heutige Hardware bietet oft keine direkte Unterstützung, um Zugriffe auf physische oder virtuelle Seiten zu zählen. Es ist zwar möglich, über das Referenz-Bit Seitenzugriffe zu zählen, dies funktioniert aber nur, wenn es einen TLB-Fehler gab oder der TLB ein Referenz-Bit enthält, das regelmäßig zurückgesetzt wird. Auch können Speicherzugriffe, die nicht von der CPU ausgeführt werden und damit nicht über den TLB laufen, nicht gezählt werden.

Ein weiteres Problem birgt die Anordnung der Module im physischen Adressraum. Es müssen nicht zwangsläufig alle Module mit den gleichen Eigenschaften hintereinander im Adressraum liegen, was zum Teil aus anderen Gründen auch nicht möglich oder wünschenswert ist. Daher bietet die physische Adresse allein keine Informationen über die energetischen Eigenschaften des zuständige Speichermoduls; ohne diese Information aber kann das Betriebssystem keine sinnvolle Entscheidung über eine geeignetet Platzie-

rung (und ggf. Verschiebung) der Daten treffen.

Wenn die Zugriffshäufigkeit der Daten und der Energieverbrauch des Speichers bekannt sind, können die Daten bei Änderung des Zugriffsmusters oder der Speicherbelegung eventuell in ein anderes, günstigeres Modul verschoben werden. Diese Aufgabe kann natürlich die CPU erledigen, was aber gerade bei größeren Datenmengen relativ langsam und bezüglich Rechenleistung eher teuer ist. Aktuelle Systeme stellen deshalb einen *DMA-Controller* bereit, der Speicherzugriffe direkt erledigt, ohne den Prozessor zu beanspruchen.

Vor diesem Hintergrund verfolgt diese Arbeit das Ziel, eine erweiterte Speicherkontrolleinheit zu entwickeln, die grundlegende Funktionen zum Überwachen und Kopieren von Speicherbereichen bereitstellt und es dem Betriebssystem ermöglicht, verstreute Daten zu bündeln oder selten benötigte Daten in energieeffizientere Speichermodule zu verschieben.

Diese Arbeit gliedert sich wie folgt: In Kapitel 2 wird der Aufbau der OpenProcessor-Plattform beschrieben, auf der die erweiterte Speicherkontrolleinheit getestet und implementiert wurde. Kapitel 3 beschreibt die Aufgaben und den Entwurf der Speichereinheit und wägt Entwurfsalternativen ab, woraufhin Kapitel 4 auf einige Details der gewählten Implementierung eingeht. Eine Zusammenfassung der Ergebnisse und ein Ausblick auf künftige Entwicklungen findet sich in Kapitel 5.

Kapitel 2

Hintergrund

OpenProcessor ist eine einfache aber vollständige Plattform, die signifikante Merkmale heutiger Rechensysteme teilt. Sie beinhaltet einen MIPS-ähnlichen RISC-Prozessor (CPU), einen softwarekontrollierten Translation-Lookaside-Buffer (TLB), SRAM und DDR-RAM sowie Flash-Speicher und diverse Ein-/Ausgabegeräte. Wie in Abbildung 2.1 dargestellt, sind alle Systemkomponenten durch einen Systembus miteinander (und mit der CPU) verbunden.

Das System ist in Verilog [2] beschrieben und somit flexibel und erweiterbar. Die Ausführung kann mit einem Simulator getestet oder auf einem FPGA implementiert werden. Dazu wird aus der Verilogbeschreibung ein *Bitstrom* erzeugt, der die Konfiguration des FPGA bestimmt.

Zur Durchführung dieser Arbeit stand ein *Memec Virtex-4 MB Development Kit* [3] zur Verfügung, das einen *Xilinx XC4VLX60*-FPGA [4] enthält. Auf diesem Board existieren verschiedene Möglichkeiten zur Ein- und Ausgabe von Daten:

- 10/100 MBit Ethernet
- RS232-konforme serielle Schnittstelle
- 2x16 LCD
- vier LED und vier Taster zur freien Verwendung
- Monitor über externen VGA-Adapter

Der FPGA enthält 360 KByte dedizierten SRAM, zusätzlich sind auf dem Memec-Board 64 MByte DDR-SDRAM und 4 MByte Flash-Speicher vorhanden. Diese Speicher werden wie die CPU und die Kontrollmodule der Ein-/Ausgabegeräte an den *Wishbone*-Bus [5] angebunden. Die Busstruktur ermöglicht den Anschluss mehrerer Master und mehrerer Slaves. Da zeitgleiche Zugriffe auf den Bus durch zwei (und mehr) Master verhindert werden müssen, findet eine Zuteilung des Busses durch einen Arbiter statt. Um

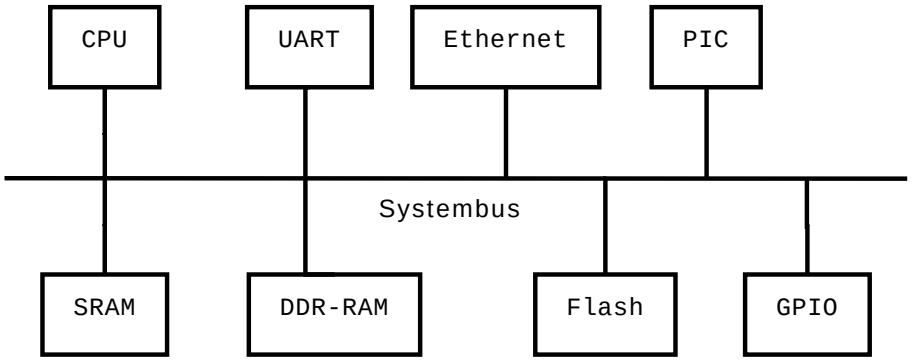

Abbildung 2.1: Überblick über die OpenProcessor-Plattform

die erneute Arbitrierung bei größeren Datentransfers zu vermeiden, können Master sog. Burst-Zugriffe initiieren und den Bus dadurch für mehrere aufeinanderfolgende Zyklen benutzen.

Weiter gibt es noch eine programmierbare Interruptkontrolleinheit, die mehrere Interruptquellen überwacht und gegebenenfalls eine Unterbrechungsanforderung an die CPU weiterleitet. Um Zeitscheiben im Betriebssystem zu unterstützen, steht ein Zeitgeber zur Verfügung, der periodisch eine Unterbrechungsanforderung auslösen kann. Außerdem ist es möglich, einige der CPU-generierten Signale mit einem Ereigniszähler zu messen. Rudimentäre Debugging-Möglichkeiten stellt die Trace-Einheit zur Verfügung, die die jeweils letzten 2000 Befehle der CPU mitsamt ihren Operanden protokollieren und bei Bedarf ausgeben kann. Um tiefere und bessere Einblicke in die CPU zu erhalten existiert eine Debug-Einheit, die im Rahmen einer früheren Studienarbeit [6] entwickelt wurde.

Um die Software von der Hardware zu entkoppeln wird nur ein rudimentäres Ladeprogramm in die FPGA-Speicher integriert, das beim Systemstart nach der Programmierung des FPGA das eigentliche Betriebssystem oder die zu testenden Software über die serielle Schnittstelle lädt.

Kapitel 3

Entwurf der erweiterten Speicherkontrolleinheit

Die erweiterte Speicherkontrolleinheit (ESKE) soll dem Betriebssystem Informationen und Methoden zur Verfügung stellen, mit denen es die Systemeigenschaften verbessern kann. In dieser Arbeit ergeben sich diese Eigenschaften durch die Zuordnung von Daten mit einem bestimmten Referenzmuster zu Speichermodulen, die unterschiedliche Zugriffslatenzen und Energiecharakteristiken besitzen. Die bereitzustellenden Informationen sind daher zum einen Zugriffshäufigkeiten auf die Daten und zum anderen die Eigenschaften der Speichermodule. Um die Systemeigenschaften zu verändern, müssen Daten in ein anderes Modul verschoben oder kopiert werden. Die Durchführung dieser Kopiervorgänge ist eine weitere Aufgabe von ESKE.

Da die Software mit virtuellen Adressen arbeitet, ist nach einem etwaigen Verschieben der Daten eine entsprechende Änderung der Zuordnung von virtuellen zu physischen Adressen notwendig. Dies ist dem Betriebssystem durch entsprechende Manipulationen des TLB leicht möglich. Die transparente Adressumsetzung durch die ESKE bietet Raum für weitere Arbeiten und wird in dieser Arbeit nicht weiter behandelt.

Um den Entwurf zu vereinfachen, werden im Weiteren drei voneinander größtenteils unabhängige Teilsysteme betrachtet: Zum Ersten wird eine Schnittstelle benötigt, die dem Betriebssystem die nötigen Informationen über die Eigenschaften der Speichermodule zur Verfügung stellt. Zum Zweiten wird eine Funktionseinheit benötigt, die Daten zwischen Speichermodulen kopieren/verschieben kann (DMA-Kontrolleinheit). Zum Dritten werden Referenzzähler benötigt, um dem Betriebssystem die Zugriffsmuster auf den Speicher zur Verfügung zu stellen.

Alle drei Aspekte sollen in den folgenden Abschnitten diskutiert werden, zunächst ist jedoch die Platzierung der Speicherkontrolleinheit im Gesamtsystem festzulegen.

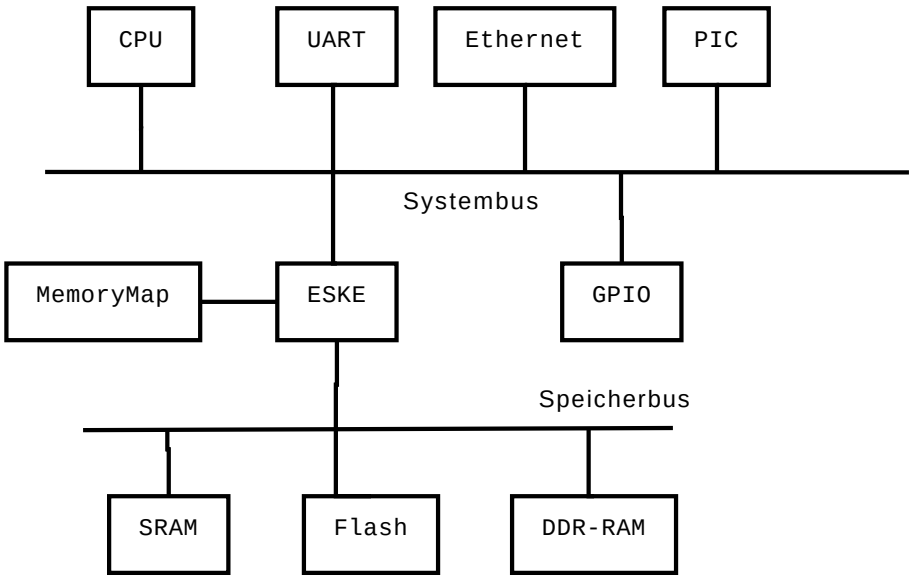

Abbildung 3.1: Möglicher Anschluss des Speichers an die Speicherkontrolleinheit

3.1 Platzierung der Speicherkontrolleinheit

Die Platzierung der erweiterten Speicherkontrolleinheit ist maßgebend sowohl für den Entwurf der enthaltenen Funktionseinheiten als auch für den Einfluss der Einheit auf die Systemleistung, weshalb hier nun zunächst die möglichen Alternativen abgewogen werden sollen.

Zum einen könnte die Speicherkontrolleinheit wie ein ganz normaler Busteilnehmer an den Systembus angeschlossen werden. Dieser Ansatz hat jedoch den Nachteil, dass Speicher-Speicher-Transfers den Systembus belegen und somit mit CPU-Zugriffen konfligieren und die Systemleistung mitunter signifikant negativ beeinträchtigen.

Alternativ kann die ESKE selbst eine Verbindungsstruktur anbieten, an die alle von ihr zu verwaltenden Speichermodule angeschlossen sind (siehe Abbildung 3.1). So wird der Systembus für Speicher-Speicher-Transfers nicht genutzt und die CPU kann über den Systembus zumindest auf E/A-Geräte weiterhin ungehindert zugreifen. Ausserdem ist es so möglich, eine vom Systembus abweichende Verbindungsstruktur zu benutzen.

Aufgrund der genannten Vorteile wird von nun an von der zweiten Platzierungsalternative ausgegangen.

3.2 Speicherkontrolleinheit

An die Speicherkontrolleinheit werden alle Speichermodule angeschlossen, so dass alle Anfragen an den Speicher über diese Einheit an das jeweilige Modul geleitet werden.

Zusätzlich werden hier die Metadaten über die angeschlossenen Speichermodule verwaltet und dem Betriebssystem zugänglich gemacht.

3.2.1 Anschluss der Speichermodule

Da schon alle verfügbaren Speichercontroller Verilogbeschreibungen zum Anschluss an Wishbone mitbringen, bietet es sich an, diese Schnittstelle auch weiterhin für den Anschluss der Speichermodule an die erweiterte Speicherkontrolleinheit zu verwenden. Wishbone unterstützt neben dem Bus auch Kreuzschienen-Verteiler, Datenfluss- und Punkt-zu-Punkt-Verbindungen als Verbindungsstrukturen.

Um den Anschluss der Module mit einer Datenfluss-Struktur zu realisieren, müsste man die einzelnen Module um eine Schnittstelle zur Weiterleitung der Anfrage und Antwort erweitern. Dies würde dem Vorteil entgegenlaufen, der durch die Übernahme von Wishbone erreicht wurde.

Die Verwendung einer Punkt-zu-Punkt-Struktur, bei der jedes Modul direkt an die Kontrolleinheit angeschlossen ist, ist ebenfall nicht sinnvoll: Zwar erlaubt sie die parallele Verwendung aller Module und maximiert dadurch die Anzahl paralleler Zugriffe, erzwingt aber gleichzeitig die Anpassung der Speicherkontrolleinheit, wenn mehr Speichermodule als vorgesehen angeschlossen werden sollen. Darüberhinaus kann die mögliche Parallelität nicht voll ausgenutzt werden, da nur so viele Zugriffe erfolgen können, wie Master an die Kontrolleinheit angeschlossen sind.

Der ebenfalls von Wishbone unterstützte Kreuzschienenverteiler lässt auch mehrere gleichzeitige Zugriffe auf verschiedene Module zu. Daher ist er relativ anspruchsvoll bezüglich der Anzahl der erforderlichen Verbindungen zwischen Mastern und Slaves. Wie bei der Punkt-zu-Punkt-Struktur kann von der möglichen Parallelität nur bedingt profitiert werden, weshalb die hohen Kosten (durch viele Verbindungen) nicht zu rechtfertigen sind.

Eine einfachere Struktur ist der Bus. Der Anschluss mehrerer Master ist auch hier möglich, jedoch kann immer nur einer von ihnen auf die Slaves zugreifen. Die maximale Anzahl der Slaves (hier: Speichermodulen) ist zumindest durch die logische Busstruktur nicht begrenzt, eine Erweiterung des Systems erfordert hier im Gegensatz zum Punkt-zu-Punkt-Fall auch keine Anpassung der Kontrolleinheit.

Aufgrund der geringen Kosten, einfachen Wartbarkeit und ausreichender Leistung wird in dieser Arbeit ein (vom Systembus getrennter) Bus zur Anbindung der einzelnen Speichermodule verwendet.

3.2.2 Verwaltung der Speicher-Metadaten

Weiter muss die ESKE dem Betriebssystem eine Möglichkeit bieten, herauszufinden, hinter welcher Adresse was für eine Art von Speicher liegt. Informationen wie Startadresse, Länge und Typ der einzelnen Module können dezentral bei jedem Modul oder zentral in der Betriebssystem-Schnittstelle der ESKE abgelegt werden.

Der dezentrale Ansatz hat den Vorteil, dass er beliebig viele Module unterstützt. Allerdings müssen hierfür in jedem Modul Logik und Leitungen hinzugefügt werden, die die Übertragung der Information regeln und die einzelnen Speichercontroller komplexer werden lassen.

Alternativ ist es möglich, die Daten in einer zentralen *memory map* in der Schnittstelle selbst zu speichern. Die einzelnen Speichermodule müssen hierfür nicht angepasst werden, allerdings müssen die Informationen schon zur Übersetzungszeit in die *map* geschrieben werden, da sonst wieder eine Übertragung zwischen Speicherkontrolleinheit und Modul stattfinden muss.

Schließlich gibt es noch weitere Entwurfsfreiheiten bei der Abfrage der Daten aus der memory map. Eine einfache Lösung wäre, eine Adressse an die memory map zu schicken und den Speichertyp zusammen mit einer Längenangabe des Moduls als Antwort zu erhalten. Alternativ wäre es möglich, die Informationen in den Speicher zu schreiben und wie normalen physischen Speicher lesbar (und ggf. auch schreibbar) zugänglich zu machen. Da hier nur Informationen über Module bereitgestellt werden, muss dann vom Betriebssystem bestimmt werden zu welchem Modul eine Adresse gehört.

3.3 DMA-Kontrolleinheit

Die DMA-Einheit soll primär Speicher-Speicher-Transfers durchführen. Dafür muss die Einheit die Daten von der Quelle abholen und, wenn der Bus wieder frei ist, an das Ziel übertragen.

Spätere Anwendungen der OpenProcessor-Plattform können jedoch von einer DMA-Einheit profitieren, die auch auf Geräte zugreifen kann. Deshalb bietet es sich an diese Einheit auch für Speicher-Geräte-Transfers auszulegen. Hier erweist sich die Ansprechbarkeit der Geräte über physische Adressen als Vorteil, da sich daduch die Kopiervorgänge für Geräte und Speicher nicht unterscheiden und somit nur unwesentlich mehr Logik für die erweiterte Aufgabe erstellt werden muss.

Um alle erforderlichen Adressen erreichen zu können, benötigt die Kontrolleinheit zumindest einen Anschluss am Systembus.

Eine Einheit die ausschließlich am Systembus angeschlossen ist, würde diesen jedoch für längere Zeiträume belegen und somit entweder die CPU-Zugriff behindern oder selbst von dieser blockiert werden.

Denkbar wäre deshalb eine zweite DMA-Einheit einzubinden, die nur für Speicher-Speicher-Transfers zuständig ist und diese ohne Benutzung des Systembusses z. B. über die DMA-Schnittstelle der Speicherkontrolleinheit realisiert. Dies erfordert jedoch im Betriebssystem für jede DMA-Operation eine Fallunterscheidung, je nachdem ob die jeweilige Busadresse zu einem Gerät oder zu einem Speichermodul führt. Außerdem ist dieser Ansatz ineffizient, da die komplette DMA-Logik doppelt vorhanden ist.

Eine einzige Kontrolleinheit, die sowohl an den Systembus als auch an die Speicherkontrolleinheit angeschlossen ist, bietet den Vorteil, dass der Systembus für Transfers zwischen Speicher und Geräten kürzer belegt ist.

Um die Auslastung der DMA-Einheit mit mehreren Anschlüssen zu erhöhen, ist es möglich Kopieroperationen, die nicht auf den gleichen Anschluss zugreifen, parallel auszuführen. Einen Weg dazu bieten mehrere DMA-*Kanäle*, wie sie schon in früheren Entwürfen wie z. B. [7] genutzt wurden. Sie verwenden Teile der DMA-Hardware gemeinsam und erhöhen so die Auslastung und Funktionalität der Hardware, ohne soviel zu Kosten wie eine zweite DMA-Einheit.

Jeder Kanal benötigt die Basisadresse des Quell- und des Zielbereichs sowie eine Längenangabe. Eine Programmierung über Speicher, der in den physischen Adressraum eingeblendet wird, ist einer Programmierung über separate Leitungen vorzuziehen, da letztere eine Anpassung der CPU erfordert.

Wenn der Controller mehrere Anschlüsse bietet, muss der Kanal Lese- und Schreibzugriffe auf den korrekten Anschlüssen durchführen. Diese Auswahl kann kanal-intern oder extern getroffen werden.

Die externe Lösung bietet sich bei mehreren Kanälen an, wenn es schon eine Kontrollinstanz gibt, die Konflikte zwischen den einzelnen Kanälen verhindern soll. Diese kann zusätzlich eine Verbindung mit dem korrekten Anschluss herstellen.

Intern kann die Auswahl nur erfolgen, wenn der Kanal von mehreren Anschlüssen weiß, also entweder eine interne Repräsentation oder externe Leitungen zum Umschalten zwischen den Ausgängen besitzt. Diese Lösung erweitert jeden Kanal um die notwendige Logik und ist deshalb gut für eine kleinere Anzahl von Kanälen geeignet.

Bei der Übertragung der Daten kann es vorkommen, dass der sendende Anschluss bereits durch einen konkurierenden Zugriff belegt ist. Somit benötigt man einen Puffer, der mindestens ein Datenwort halten kann. Größere Puffer machen die Nutzung von sogenannten *Burst-Transfers* möglich, wenn der Zielbus anderweitig belegt ist.

Eine wichtige Funktion von DMA-Controllern ist auch die Benachrichtigung des Betriebssystems nach Abschluss des Kopiervorgangs. Neben wiederholtem Abfragen eines Registers (sog. *polling*) besteht auch die Möglichkeit, von Seiten der DMA-Einheit eine Unterbrechung an die Interruptkontrolleinheit zu senden. Ein üblicher Mittelweg ist, das Betriebssystem bei der Programmierung der DMA-Einheit über ein Bit im Statuswort entscheiden zu lassen, welche Art der Benachrichtigung (*polling* oder Unterbrechung) verwendet werden soll. Der Einfachheit halber wird die DMA-Einheit Benachrichtigungen nur über *polling* unterstützen.

3.4 Referenzzähler

Der Referenzzähler soll Zugriffe auf Speicherbereiche zählen, die durch eine Startadresse und eine Bereichslänge definiert werden. Hierzu benötigt er einen Anschluss an die

Adressleitungen der Speichermodule bzw. deren Verbindungstruktur um Anfragen an den Speicher zu sehen. Mit Hilfe dieser Informationen wird dann verglichen, ob die Adresse auf dem Bus im beobachteten Bereich liegt:

$BUSADRESSE \geq STARTADRESSE$ und
$BUSADRESSE \leq STARTADRESSE + LAENGE$

Diese Art der Überprüfung ist in Hardware aufwendig zu realisieren, da Vergleich und Addition verhältnismäßig komplexe Operationen sind. Auf der OpenProcessor-Plattform wird deshalb die Infomation über die Länge in Form einer *MASKE* so kodiert, dass die Überprüfung mittels

$(BUSADRESSE \ \& \ MASKE) == STARTADRESSE$

stattfinden kann.

Hierbei wird nur ein 32 Bit breites logisches UND und eine logische Äquivalenz (ebenfalls 32 Bit) benötigt. Diese Form der Längenkodierung verlangt, dass die Startadresse bei ganzzahligen Vielfachen der Größe des Moduls liegt (alle Bereiche sind *natürlich ausgerichtet*).

Dem Vorgang der Längenkodierung in OpenProcessor folgend wäre der einfachste Ansatz zur Referenzzählung, die Startadresse und die Maske abzuspeichern und den Zählerwert bei positiver Überprüfung der Busadresse zu erhöhen. Diese Art der Referenzzählung ist jedoch nicht sehr effizient, denn sie verbraucht pro überwachtem Bereich drei Speicherplätze (je 32 Bit Startadress und Maske plus n Bit Zähler).

Ein besserer Ansatz ist, *Blöcke* zu zählen, die aus einer bestimmten Anzahl von aneinandergrenzenden Bereichen gleicher Größe bestehen. Jeder Bereich erhält einen separaten Speicher für den Zählwert, aber Adresse, Maske und Additions-Logik werden nur einmal für den Block bereitgestellt. Die Maske kodiert die Länge eines Bereichs. So ergibt sich die Größe des Blocks aus der Anzahl mal der Größe der Bereiche. Diese Art von Referenzzähler ist im Vergleich zum vorherigen Entwurf weniger flexibel, jedoch gut geeignet um ganze Speichermodule mit bestimmten Bereichsgrößen zu beobachten.

Ein dritter Weg wäre, dass sich mehrere Bereiche nur die Additions-Logik teilen. Dadurch bleibt die Flexibilität größtenteils erhalten, jedoch auf Kosten von etwas mehr Hardware für separaten Speicher von Startadresse und Maske. Ähnlich wie beim Blockzähler lässt sich hier noch etwas optimieren, wenn man die Adressen und Masken in dedizierten Speichern ablegt.

Kapitel 4

Implementierung

Dieses Kapitel beschreibt die Implementierung der erweiterten Speicherkontrolleinheit, die im vorhergehenden Kapitel entworfen wurde. Einen Überblick über die Struktur der OpenProcessor-Plattform mit der eingefügten ESKE zeigt Abbildung 4.1.

Wie schon im Entwurf wird auch die Implementierung in drei Teilen behandelt. Zuerst wird auf die Implementierung der Speicherkontrolleinheit eingegangen.Danach werden einige interressante Details des DMA-Controllers vorgestellt. Der letzte Teil behandelt zwei verschiedene Implementierungen des Referenzzählers.

4.1 Speicherkontrolleinheit

Mit der Speicherkontrollheit sind alle Speichermodule über einen Bus verbunden. Der Bus wurde gewählt, weil er mit geringem Leitungsaufwand umzusetzen ist. Da auf dem FPGA keine Tristate-Treiber vorhanden sind, werden für Datenein- und -ausgang der Module verschiedene Leitungen benutzt. Um diesem Umstand gerecht zu werden, muss die Speicherkontrolleinheit eingehende Leitungen für die Datenausgabe der einzelnen Module bereitstellen. Deshalb ist die Speicherkontrolleinheit entgegen der ursprünglichen Intention auf eine feste Anzahl von Speichermodulen festgelegt. Eine Anpassung der Schnittstelle muss nicht zwangsläufig erfolgen, wenn man Multiplexer-Einheiten verwendet, die es erlauben, mehrere Module an einer externen Schnittstelle zu betreiben. Die Implementierung bietet Anschlüsse für acht externe Module, wovon vier bereits durch den vorhandenen Speicher belegt sind.

Die memory map wird zentral in der Speicherkontrolleinheit mit Hilfe eines 2 KByte großen SRAM gespeichert. Sie ist wie die anderen Module an den Speicherbus angeschlossen, verwendet aber keinen der externen Anschlüsse. Die enthaltenen Informationen sind nur lesbar und können über den Bus abgefragt werden.

Pro Modul sind 12 Byte vorgesehen, die sich in je 32 Bit für Adresse, Maske und Modultyp aufteilen. Die Informationen über die Module werden in der Verilogbeschreibung

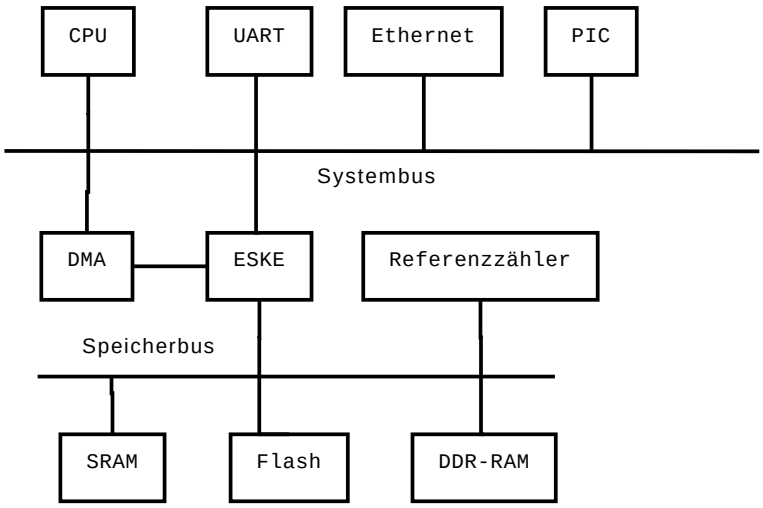

Abbildung 4.1: OpenProcessor-Plattform mit integrierter ESKE

übergeben und hintereinander in der Memory-Map abgelegt.

Aufgrund des begrenzten Speichers kann nur eine bestimmte Anzahl von Modulen in der Memory-Map abgebildet werden. Danach ist eine Erweiterung und Neuübersetzung der Verilogbeschreibung nötig. Dies ist aber kein Problem, da die Plattform ohnehin jedes Mal neu übersetzt werden muss, wenn neue Module hinzugefügt werden.

Die Einheit bietet außerdem noch eine Schnittstelle zum Anschluss eines weiteren Masters. Diese wird vom DMA-Controller genutzt, um Datentransfers vom und zum Speicher durchzuführen. Da die Module über einen Bus mit der Speicherkontrolleinheit verbunden sind, darf immer nur entweder die CPU oder der DMA-Controller zugreifen. Um Konflikte auszuschließen, ist in die Speicherkontrolleinheit ein Arbiter integriert. Der CPU wird von diesem eine höhere Priorität zugewiesen, so dass keine Blockierung durch den DMA-Controller stattfindet.

4.2 DMA-Kontrolleinheit

Die DMA-Kontrolleinheit (Abbildung 4.2) nutzt sowohl den Anschluss der Speicherkontrolleinheit als auch einen Anschluss als Master am Wishbone-Bus. So lassen sich Speicher-Speicher-Transfers ohne Umweg über den Systembus realisieren.

Die Implementierung enthält nur einen DMA-Kanal, der aber Transfers zwischen beiden Anschlüssen unterstützt. Daneben sind im Controller noch zwei Wishbone-Master-Brücken vorhanden, die die Kommunikation mit dem System- und Speicherbus übernehmen. Die Verwendung der Wishbone-Brücke macht es möglich, den DMA-Controller auch weiterhin zu nutzen, wenn die unterliegende Verbindungsstruktur nicht mehr Wishbone-kompatibel ist. Die Brücke wird auch in der CPU genutzt und muss bei

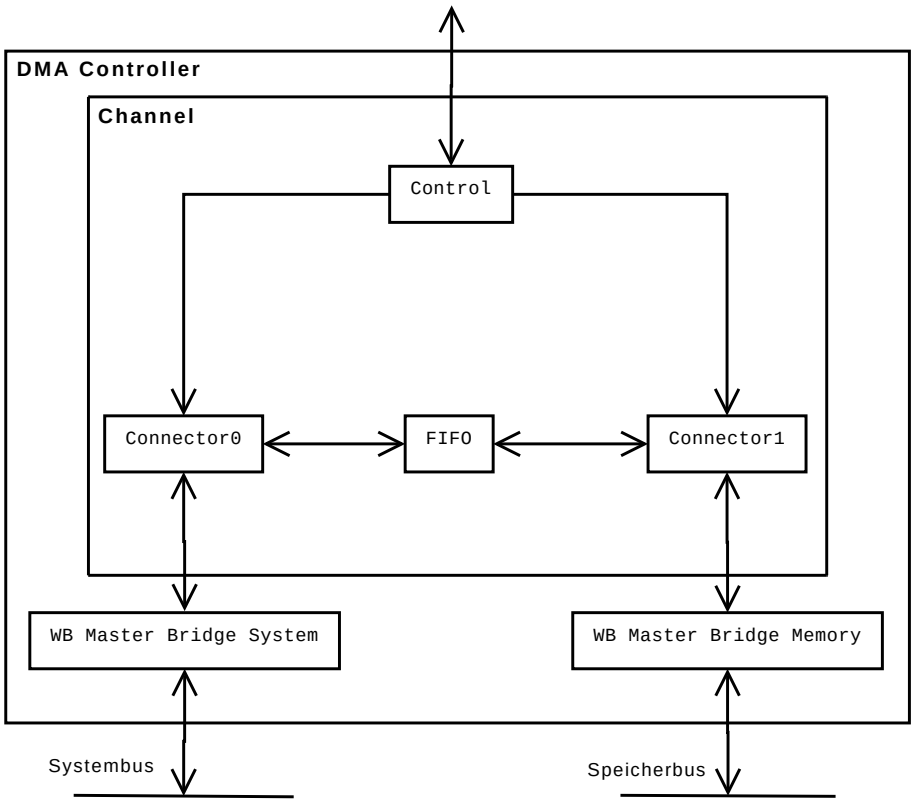

Abbildung 4.2: Aufbau des DMA-Controllers ohne Steuerleitungen

einem Umstieg nur einmal geändert werden.

Die Control-Einheit in Abbildung 4.2 enthält Register, die in den physischen Adressraum eingeblendet werden. Über diese insgesamt 12 Byte Speicher wird die Einheit gesteuert, außerdem werden hierüber Statusmeldungen ausgegeben. Aufgrund der Breite des Adressbusses werden für Quelle und Ziel je 32 Bit Speicher benötigt. Weitere 16 Bit werden für das sogenannte *Limit* verwendet, das die um 1 verringerte Anzahl zu kopierender 32-Bit-Worte angibt. Die zwei höchsten Bytes enthalten das Kontrollwort, das benutzt wird, um den Status des Kanals abzufragen und zu setzen. Hier wird aktuell nur ein Bit verwendet, mit dem das Betriebssystem der Einheit anzeigt, dass die Programmierung abgeschlossen ist und der Transfer beginnen kann.

Da der Wishbone-Bus 32-Bit-Worte übertragen kann, wird diese Breite auch voll ausgenutzt. Kleinere Datenwörter können nicht übertragen werden. Daraus folgt, dass maximal 2^{16} Worte oder 256 KByte in einem DMA-Transfer bewegt werden können.

Die Control-Einheit ist auch verantwortlich dafür die Connector-Einheiten zu programmieren, die den eigentlichen Datentransfer erledigen. Einen Ausschnitt des Zu-

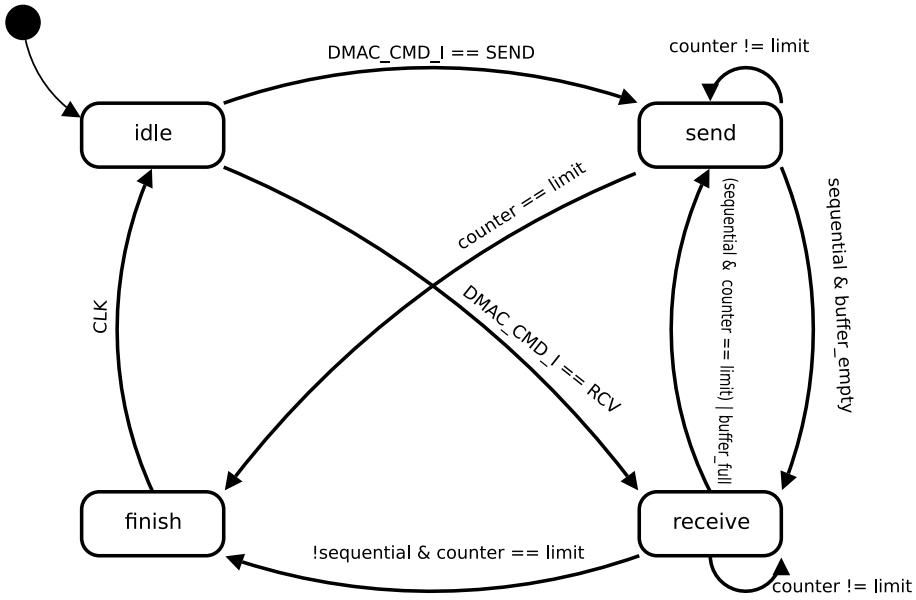

Abbildung 4.3: Vereinfachtes Zustandsdiagramm des DMA-Connnectors

standsautomaten zeigt Abbildung 4.3.

Die Connector-Einheiten sind direkt mit den Wishbone-Brücken verbunden, um eine arbitrierte Verbindungsstruktur zu vermeiden. Diese wurde nicht implementiert, da sie für nur einen Kanal nicht benötigt wird. Hieraus ergibt sich jedoch, dass die Connectoren nicht mehr jeden Anschluss erreichen können sondern nur entweder am Systembus oder am Speicherbus angeschlossen sind. Die Control-Einheit darf die Connectoren somit nur deren Möglichkeiten entsprechend programmieren.

Dem Kanal wird zur Übersetzungszeit der erreichbare Adressbereich jeder Schnittstelle mitgeteilt. So kann die Control-Einheit, nach der Programmierung durch das Betriebssystem, entsprechend wählen, welcher Connector welche Daten holen oder senden soll.

Zwischen den Connector-Einheiten befindet sich ein Puffer. Er wurde mit Hilfe eines Block-RAM implementiert und kann bis zu 2 KByte an Daten zwischenspeichern. Die Größe des Speichers ermöglicht es, Burst-Zugriffe auch dann durchzuführen, wenn ein Ende des Kanals nicht bereit ist.

Bei Funktionstests hat sich gezeigt, dass, sofern der Bus frei ist, nur jeweils ein 32-Bit-Wort im Puffer liegt und somit auch ein Puffer dieser Größe ausreichen würde.

Nachdem der Datentransfer durchgeführt wurde, setzt die Control-Einheit das Kontrollwort wieder auf 0. Eine Benachrichtigung über Interrupts ist in der vorliegenden Implementierung nicht vorgesehen.

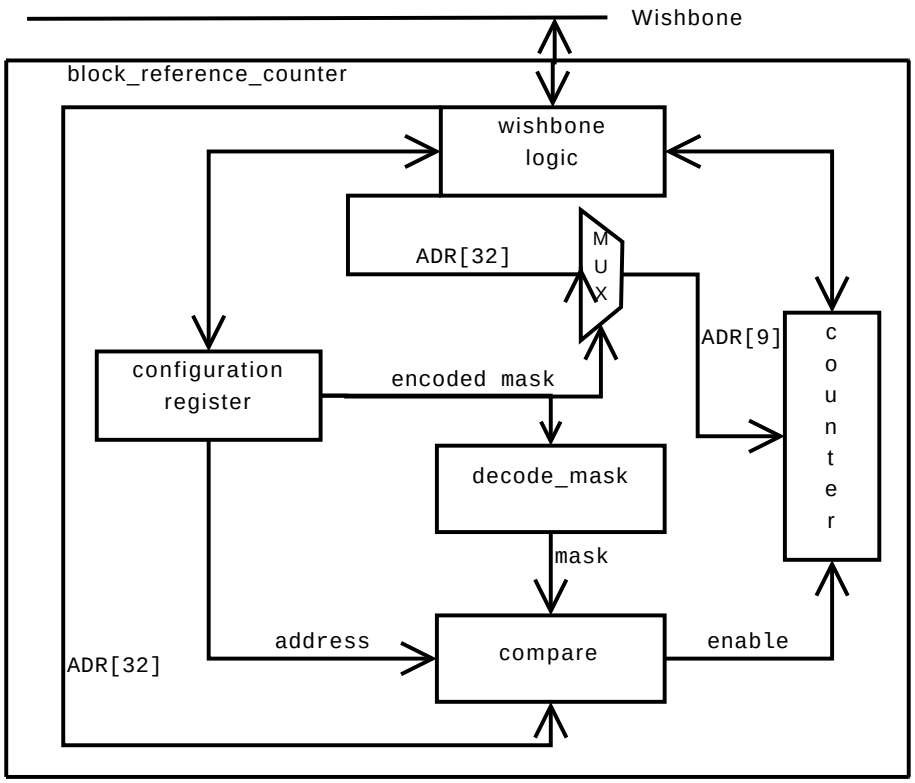

Abbildung 4.4: Aufbau des Block-Referenzzählers

4.3 Referenzzähler

Es wurden zwei verschiedene Referenzzähler implementiert. Zum einen der Block-Referenzzähler, der mit wenig Hardware auskommt und gut geeignet ist, Zugriffe auf gleichgroße Bereiche für ganze Module zu zählen. Zum anderen ein flexiblerer, aber dafür teurerer Zähler, der viele auch auseinanderliegende Bereiche unterschiedlicher Größe überwachen kann.

4.3.1 Block-Referenzzähler

Der Aufbau des Block-Referenzzählers ist in Abbildung 4.4 dargestellt. Er wird nur an den Wishbone-Bus angeschlossen. Alle 512 Zählerwerte und das Konfigurationsregister können über Wishbone-Zugriffe gelesen werden. Während das Konfigurationsregister mit frei wählbaren Daten beschrieben werden kann, werden Zählerwerte durch einen Schreibzugriff immer auf 0 zurückgesetzt.

Das Konfigurationsregister enthält die obersten 22 Bits der Startadresse des Blocks, drei weitere Bits zur Kodierung der Maske und weitere sieben Bits, die für spätere Zwecke

reserviert sind.

Da für die Startadresse nur 22 Bit vorgesehen sind, kann ein Block nur an ganzen Vielfachen von 1 KByte anfangen. Da außerdem aus den Bereichslängen eine Maske für den Block berechnet wird, gelten für die Platzierung des Blocks im physischen Adressraum alle Einschränkungen, die durch eine Adressierung durch Masken entstehen.

Mit den drei Masken-Bits (msk) lassen sich insgesamt acht verschiedene Bereichslängen kodieren. In der vorliegenden Implementierung lässt sich die Länge der Bereiche mit Hilfe der folgenden Formel aus msk berechnen:

$$\text{Bereichslaenge} = 2^{msk*2^{msk[2]}+10}$$

Zu beachten ist die besondere Bedeutung des höchsten Maskenbits $msk[2]$, das, wenn es gesetzt ist, den Wert von msk verdoppelt und somit deutlich größere Werte für die Bereichslängen erlaubt. Ohne diese Multiplikation wären maximal 128 KByte-große Bereiche zählbar, so aber liegen die erreichbaren Längen zwischen 1 KByte und 16 MByte.

Decode_mask erzeugt aus den Bits im Konfigurations-Register die Bereichslänge und multipliziert diese mit der Anzahl der Bereiche um die Blocklänge zu berechnen. Hieraus wird eine Maske generiert, die wiederrum der compare-Einheit zugeführt wird.

In dieser Einheit findert der Vergleich mit der auf dem Bus anliegenden Adresse statt. Eine positive Überprüfung wird dem Zähler über das enable-Signal mitgeteilt. Dem Zähler wird ausserdem durch 9 Bit übermittelt, auf welchen Bereich zugegriffen wurde. Diese Bits werden durch encoded_mask mit Hilfe von Multiplexern je nach kodierten Bereichsgröße aus der Busadresse gewählt. Damit die Bereiche richtig auf die Zählerspeicher zugewiesen werden, ist es notwendig die 9 Bit so auszuwählen, dass das niedrigste Bit an der Stelle $msk * 2^{msk[2]} + 10$ liegt.

Wenn als Bereichsgrößen keine Zweierpotenzen vorgegeben wären, müsste man hier mit deutlich mehr Aufwand eine Übersetzung der physischen Adresse auf die Bereichsnummer durchführen. Im ungünstigsten Fall müssten also alle 512 Adressen mittels einer Tabelle in die entsprechende Adressen im Zähler überführt werden.

Da in der Implementierung Bereiche nur in der Länge von Zweierpotenzen möglich sind, muss keine weitere Übersetzung der ausgewählten Bits stattfinden.

4.3.2 Flexibler Referenzzähler

Der zweite Referenzzähler (Abbildung 4.5) ist dem Block-Referenzzähler sehr ähnlich. Es sind wieder 512 Bereiche zählbar, die jedoch nicht mehr hintereinander liegen müssen. Sowohl der Wert der Zähler als auch die Konfigurationsdaten sind über den Wishbone-Anschluss zugänglich.

Im Unterschied zum Block-Referenzzähler sind nun für jeden Bereich 32 Bit zur Konfiguration verfügbar. Diese Daten sind wie beim Blockzähler kodiert und werden aus Effizienzgründen in einem Block-RAM zusammengefasst.

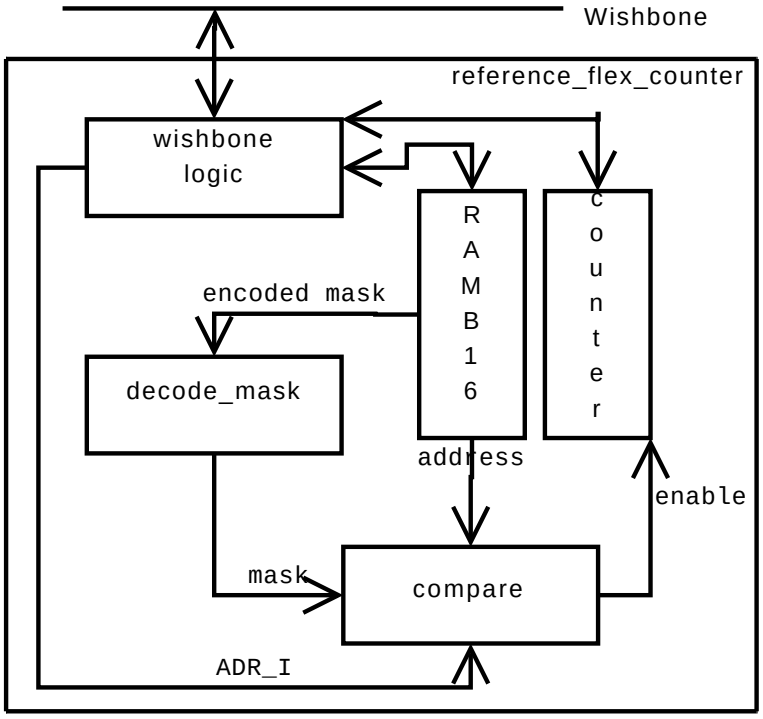

Abbildung 4.5: Aufbau des flexiblen Referenzzählers

Die Adressierung der Konfigurationsdaten wie auch der Zählerwerte erfolgt durch Bits aus der Busadresse. Es bietet sich an, hierfür die höchsten Bits der Busadresse zu nutzen, um die maximale Bereichslänge zu erzielen. Zur Adressierung der 512 Konfigurationssätze werden 9 der 32 Adressbits benötigt, sodass die größte zählbare Seite 2^{23} Byte oder 8 MByte lang sein darf.

Höhere Werte lassen sich hier nur erzielen, wenn man weniger Bits der Busadresse zur Auswahl der Daten nutzt. Dies würde bedeuten, dass man nicht mehr alle Daten im Block-RAM ansprechen kann. Es sei denn man nutzt mehrere Speicherbausteine, deren Adressleitungen teilweise auf feste Werte gestellt sind. Dies würde pro Modul einen Vergleicher benötigen und somit die Effizienz des Zählers senken.

Größere Bereiche lassen sich trotzdem zählen, wenn man mehrere Zähler mit kleineren Größen programmiert.

Im Gegensatz zum Blockzähler lassen sich, unabhängig von der Kodierung, Bereiche kürzer als 1 KByte nicht mehr zählen, da die niederwertigsten 10 Bit der Startadresse nicht gespeichert werden und ein Vergleich dieser mit der Busadresse folglich nicht möglich ist.

Um falsche Zählerwerte durch alte Werte zu verhindern, wird der Zählerwert bei jedem Schreibzugriff auf die Konfigurationsdaten gelöscht. Eine manuelle Löschung lässt sich durch einen Schreibzugriff auf die Adresse des Zählerwertes erreichen.

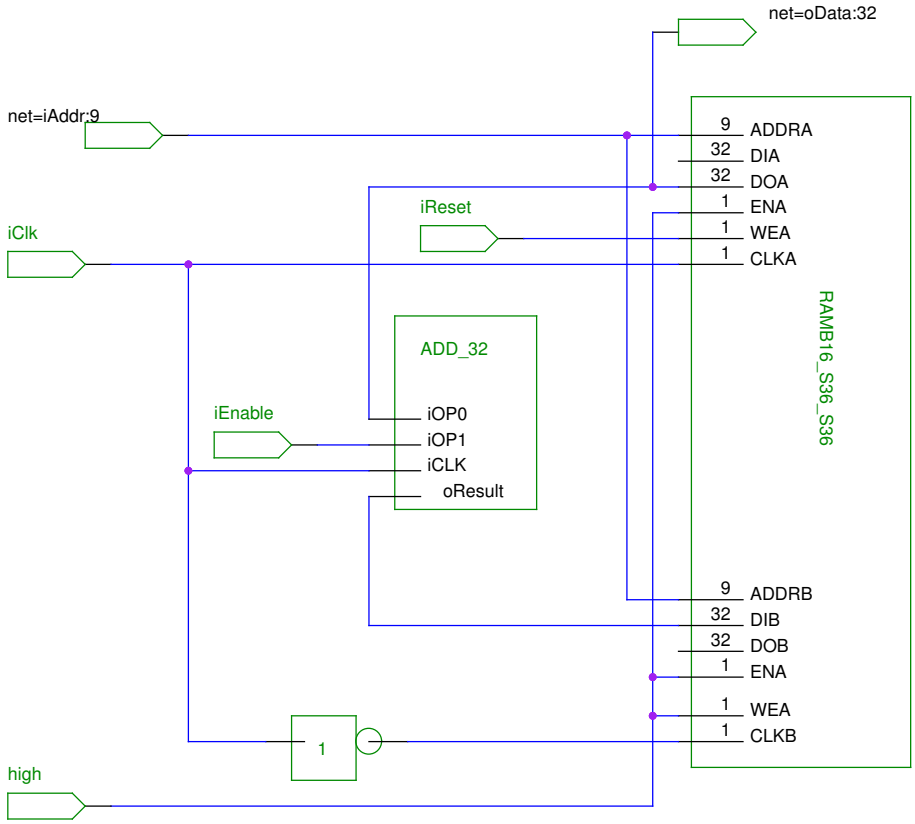

Abbildung 4.6: Schaltbild der Zähleinheit

4.3.3 Zähleinheit

Beiden Referenzzählern ist die eigentliche Zähleinheit (Abbildung 4.6) gemein, die sowohl das Speichern als auch das Inkrementieren der Zählerwerte übernimmt.

Die Auswahl des Speichers erfolgt über von außen zugeführte Adressleitungen. Falls der Zählerwert erhöht werden soll, müssen zuerst die Daten an der Adresse ausgelesen, inkrementiert und anschließend zurückgeschrieben werden. Dies muss innerhalb eines Bustaktes geschehen, da im nächsten Takt wieder eine neue Adresse auf dem Bus anliegen kann.

Auf der OpenProcessor-Plattform ist es nicht möglich den Zähler mit dem doppelten Bustakt zu betreiben, da es keine schnelleren Taktsignale gibt. Deshalb wurde als Datenspeicher ein Dual-Port-RAM verwendet, um den Vorgängen Auslesen und Zurückschreiben jeweils einen eigenen Port zu geben.

Der erste Port ist nur für die Datenausgabe und das Rücksetzen der Zählerwerte zuständig. Er bekommt zur steigenden Taktflanke eine Adresse und gibt die zugehörigen Daten aus. Ist das Reset-Signal gesetzt, werden die Zählerwerte zudem auf Null gesetzt.

Der zweite Port ist für das Zurückschreiben der gegebenenfalls erhöhten Werte vorge-

sehen, die der Addierer erzeugt hat. Dieser bekommt den ursprünglichen Wert des Zählers vom Datenausgang des ersten Ports und erhöht den Wert bei gesetztem enable-Signal um eins.

Um Konflikten zwischen den Ports vorzubeugen, wird der zweite Port mit einem Takt betrieben, der gegenüber dem Takt des ersten Ports um einen halben Zyklus versetzt ist (180° phasenverschoben).

Kapitel 5

Zusammenfassung

Im Rahmen dieser Arbeit wurde eine erweiterte Speicherkontrolleinheit (ESKE) entwickelt und implementiert, die dem Betriebssystem zusätzliche Möglichkeiten gibt, den Energieverbrauch und die Leistungsfähigkeit des ausführenden Systems hinsichtlich der Speichermodule zu bewerten und zu beeinflussen.

Es ist dem Betriebssystem jetzt möglich, über eine Schnittstelle herauszufinden, welche physischen Adressen von welchem Speichermodul belegt sind. Ebenso lassen sich Zugriffmuster auf Speicheradressen durch die Verwendung der vorgestellten Referenzzähler bestimmen und somit feststellen, welche Daten es sich in andere Module zu verschieben lohnt.

Der dazu nötige Kopiervorgang kann von einem ebenfalls hier implementiertem DMA-Controller durchgeführt werden. Durch einen abgekoppelten Speicherbus und einen separaten DMA-Anschluss hieran wird der Systembus entlastet, sodass die CPU parallel auf Gerätecontroller zugreifen kann. Eine Blockierung der CPU bei gleichzeitigem Speicherzugriff durch CPU und DMA wird durch eine entsprechende Arbitrierung des Speicherbusses verhindert, die Zugriffe von Seiten der CPU bevorzugt.

Die durch Datenmigration notwendige Anpassung der Übersetzung von virtuellen auf physische Adressen kann durch die bereits vorhandene Manipulationsmöglichkeit des TLB vorgenommen werden.

Mit diesen Hilfsmitteln ist es möglich, Programme und Daten zu identifizieren, durch deren Verschieben in ein anderes Speichermodul ein geringerer Energieverbrauch oder höhere Leistung erzielt werden kann.

5.1 Ausblick

Eine Abkoppelung der Gerätecontroller in Verbindung mit einer weiteren Schnittstelle für den DMA-Controller würde es ermöglichen, DMA-Transfers vollständig ohne den Systembus durchzuführen. In diesem Szenario wäre es auch die Einführung weiterer

DMA-Kanäle in der DMA-Einheit sinnvoll, um parallele Transfers zu unterstützen. Da hier Konflikte auftreten können, ergibt sich die Notwendigkeit einer Arbitrierung, die in dieser Arbeit aufgrund der höheren Komplexität nicht in Betracht gezogen wurde.

Im Moment werden virtuelle Adressen fast ausnahmslos direkt auf physische Adressen abgebildet. Mit der Entwicklung von weiterer Software wird das Betriebssystem zur Optimierung Anpassungen am TLB vornehmen. Je nach Charakteristik der einzelnen Programme kann diese Aufgabe sehr zeitintensiv werden, so dass es besser ist diese Aufgabe an eine Einheit wie in [8] zu delegieren.

Da die Verbindungsstruktur der Speicher durch die ESKE vom Systembus abgekoppelt ist, können für erstere auch Strukturen verwendet werden, die es erlauben, parallel auf Speichermodule zuzugreifen oder nur so viele Adressleitungen haben, wie es für die Adressierung des existierenden Speichers notwendig ist.

Um genauere Zugriffsmuster zu bekommen, ist es zum Beispiel vorstellbar, separate Zählerwerte für Lese- und Schreibzugriffe einzuführen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Raphael Neider. *OpenProcessor v1*. Universität Karlsruhe.

[2] Donald E. Thomas and Philip R. Moorby. *The Verilog hardware description language*. Kluwer, 5th edition, 2002.

[3] MemecBoard. *Virtex-4 MB Development Board User's Guide*. Calgary, Alberta, Canada, December 2005.

[4] Xilinx. *Virtex-4 User Guide*, March 2006.

[5] Richard Herveille (Steward). *Specification for the WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores*. OpenCores Organization, September 2002. Revision B.3.

[6] Stefan Bach. Implementing a debug unit for the openprocessor. Study Thesis, January 2008.

[7] Intel. *8237/8237-2 high performance programmable DMA controller*. `http://www.datasheetarchive.com/pdf-datasheets/DataBooks/Book261-1091.pdf`.

[8] Frank Bellosa. When physical is not real enough. In *Proceedings of the 11th ACM SIGOPS European Workshop*, page 25, New York, NY, USA, 2004. ACM.