Anforderungen an den Arbeitsspeicher und die Speicheranbindung

Handout zum 3. Aufgabenblatt

Hardwarepraktikum SoSe 17

Dieses Dokument definiert die Anforderungen an die Speicheranbindung und den Arbeitsspeicher einer einfachen ARMv4-Implementierung für das Hardwarepraktikum. Darüber hinaus werden die Speicherbussignale festgelegt sowie die Realisierung des Speichers mit FPGA-Ressourcen beschrieben.

Anforderungsdefinition

Für die Ausführung von Anwendungen auf dem ARM-Prozessor ist zumindest ein kleiner, möglichst direkt und schnell angebundener Speicher notwendig. Infrage kommen ein SRAM-Cache mit Anbindung an größeren DRAM oder ein kleiner *Tightly Coupled Memory* (TCM) auf SRAM-Basis ohne Anbindung an einen größeren Speicher. Im Idealfall sollten Speicherzugriffe innerhalb eines Taktes ausgeführt werden, damit sie den Prozessorkern nicht länger als unbedingt notwendig belegen. Lesezugriffe auf den Speicherbussen sollten deshalb zumindest asynchron *erscheinen*: Der Speicher legt ein Datum auf den Bus, sobald der Prozessor eine gültige Adresse und die notwendigen Steuersignale erzeugt hat. Dadurch kann das gelesene Datum noch im selben Takt in den Prozessor übernommen werden. Es sind zwei grundverschiedene Zugriffe auf den Speicher zu unterscheiden:

- Lesen von Instruktionen als ersten Schritt der Instruktionsverarbeitung.
- Lesen und Schreiben von Nutzdaten eines Programms durch Datentransferinstruktionen

ARM-Prozessoren können sowohl in von-Neumann- als auch als in Harvard-Architektur realisiert werden, verfügen also über einen oder zwei Speicherbusse. Allerdings verwendet sie immer einen gemeinsamen 32-Bit-Adressraum für Programm und Daten. Auch bei der Realisierung mit zwei Speicherbussen (*Instruktions-* und *Datenbus*) können daher auf dem Datenbus Zugriffe in den Instruktionsspeicher stattfinden. Ohnehin muss eine Möglichkeit bestehen, ein Programm in den Instruktionsspeicher zu laden. Wie wir noch sehen werden, kann dies über eine zusätzliche Komponente am Datenbus mit begrenztem Mehraufwand geschehen. Tatsächlich müssen Speicherzugriffe also auf einem gemeinsamen Speicher erfolgen, auch wenn dafür verschiedene Busse verwendet werden.

Das verwendete FPGA verfügt über 20 dedizierte, jeweils 16 kibibit[1] große Speicher (*Blockram*) mit je zwei getrennten Schnittstellen (*Port* A und B). Es bietet sich daher an, einige dieser Komponenten zu einem größeren Speicher mit gemeinsamem Adressraum und zwei Speicherbussen zu verbinden. Dementsprechend verfügt unsere ARM-Implementierung über einen *Instruktions-* und einen *Datenbus* (Harvard-Architektur). Die Möglichkeit, parallel einen Datentransfer durchzuführen und die nächste Instruktion aus dem Speicher zu lesen, wirkt sich positiv auf den Befehlsdurchsatz aus.

Zu jedem Speicherbus gehören mindestens 32 Adress- und 32 Datenleitungen. Hinzu kommen, je nach Bedarf, verschiedene Steuersignale. Es wird in diesem Zusammenhang häufig von Adressbus, Datenbus und Steuerbus gesprochen, wobei jeweils ein funktional zusammengehörendes Leitungsbündel gemeint ist. Diese Bezeichnungen sollten nicht verwechselt werden mit *Instruktions-* und *Datenbus*, die jeweils über Leitungen für Adressen, Daten und Steuerung verfügen.

Auf dem Instruktionsbus kommunizieren nur Prozessorkern und Speicher. An der Kommunikation auf dem Datenbus sind hingegen weitere Geräte beteiligt, weil Ein-/Ausgabe-Geräte in der ARM-Architektur memory-mapped angebunden werden. Da Instruktionen immer 32 Bit umfassen, erfolgen Lesezugriffe auf dem Instruktionsbus immer wortweise. Instruktionen sind immer an Wortadressen ausgerichtet. Schreibzugriffe in den Speicher finden auf dem Instruktionsbus nicht statt.

ARMv4 umfasst Befehle für den byte-, halbwort-, und wortweise lesenden und schreibenden Zugriff auf den Datenbus, die Zugriffsadressen sind entsprechend ausgerichtet und die Art und Richtung des Zugriffs muss durch Steuersignale angezeigt werden. Absolut notwendig für beide Busse ist ein Steuersignal, durch das ein Kommunikationspartner dem Prozessorkern signalisieren kann, dass ein Fehler beim Speicherzugriff aufgetreten ist. ARM-Prozessoren zeigen ihren aktuellen Betriebsmodus bei Speicherzugriffen generell an. Es ist auf diese Weise z.B. möglich, den Zugriff auf bestimmte Speicherbereiche davon abhängig zu machen, ob der Prozessor in einem privilegierten Modus arbeitet. Außerdem muss der Betriebsmodus, mit dem ein Datenspeicherzugriff stattfindet, nicht mit dem Modus identisch sein, mit dem zeitgleich eine weitere Instruktion aus dem Instruktionsspeicher gelesen wird. Zum Beispiel kann der Datentransfer mit den Instruktionen `STRT` oder `LDRT` stattfinden, für die der aktuelle Modus ignoriert und stattdessen die Speichersicht des User-Modus verwendet wird. Um beiden Anforderungen gerecht zu werden, müssen Instruktions- und Datenbus jeweils 5 Signalleitungen aufweisen, auf denen der Prozessorkern den Betriebsmodus für Speicherzugriffe anzeigt.

In größeren Prozessorsystemen ist davon auszugehen, dass ein Speicherzugriff nicht immer in einem Takt abgeschlossen wird. Diese Tatsache ist für das Hardwarepraktikum im Prinzip nicht von Bedeutung. Zwecks einfacherer zukünftiger Erweiterbarkeit führen beide Busse aber ein Signal, durch das Peripheriegeräte einen langsamen Speicherzugriff anzeigen können.

Die Adressen auf dem Instruktionsbus werden ausschließlich vom Prozessor erzeugt, er ist der einzige Busmaster[2]. Daten auf dem Instruktionsbus werden immer vom Speicher erzeugt und vom Prozessor gelesen. Es gibt in dieser einfachen Realisierung keine weiteren Geräte am Bus.

Adressen auf dem Datenbus entsprechen gewöhnlich dem Inhalt eines Daten-Adressregisters im Datenpfad des Prozessors, er fungiert auch hier als Busmaster. Wenigstens zur Initialisierung des Speichers muss jedoch ein weiterer Master die Kontrolle über den Bus übernehmen können. Die

[1] Kibi $= 2^{10} = 1024$ statt Kilo $= 10^3 = 1000$

[2] **Busmaster**: Das einzig aktive, d.h. steuernde Gerät an einem Bussystem

Auswahl zwischen den um Buszugriff konkurrierenden Mastern geschieht zentral durch einen *Arbiter*.

Formal existieren mit `SWP` und `SWPB` zwei Instruktionen, für die der Prozessor mehrere Takte in Folge exklusiven Zugriff auf den Datenbus benötigt. Normalerweise müsste dazu ein Bussperrmechanismus existieren. Da der zweite Master aber ausschließlich den Speicher initialisiert und anschließend den Prozessor neu startet, kann darauf verzichtet werden. Neben dem zweiten Busmaster werden Peripheriekomponenten für die Kommunikation des Prozessors mit seiner Umwelt an den Datenbus angeschlossen. Diese sind immer Slaves[3].

Jeder Slave verfügt dabei über einen Steuereingang (ein sogenanntes *Chip-Select*-Signal), durch den er explizit aktiviert werden muss. Die CS-Signale werden zentral in einem *Chip-Select-Generator* auf Basis der Zugriffsadresse erzeugt. Daten auf dem Datenbus können vom Prozessor zur Peripherie als auch umgekehrt transportiert werden (bidirektionale Kommunikation). Entweder werden die Datenleitungen des Datenbusses bidirektional ausgeführt, oder aber separate Datenleitungen für beide Richtungen vorgesehen. Die zweite Variante ist einfacher umsetzbar (die Transportrichtung der Datenleitungen muss nicht umgeschaltet werden) und wurde daher gewählt.

Umsetzung

Es ergibt sich für die Speicheranbindung des Prozessors das in Abbildung 1 gezeigte Schema. Port

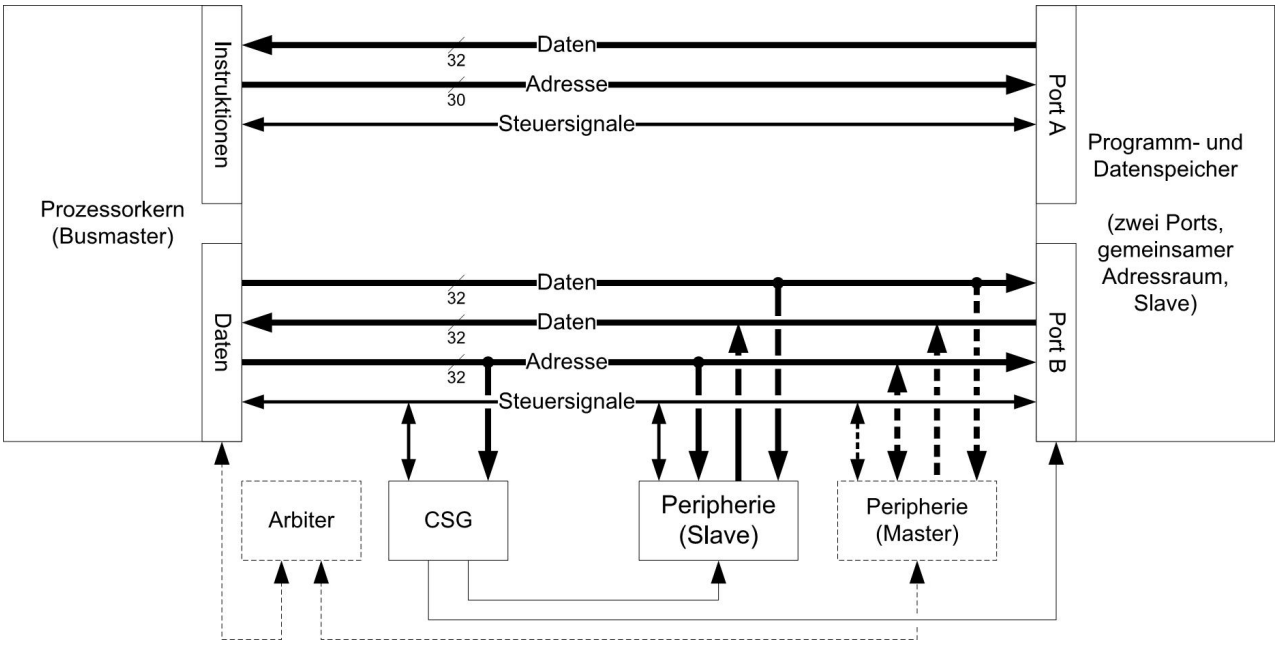

Abbildung 1: Schematischer Aufbau der Arbeitsspeicheranbindung des HWPR-Prozessor

A der Blockram-Bausteine wird für die Kommunikation mit der Instruktionsschnittstelle des Prozessors verwendet. Der Prozessor legt die Zugriffsadresse für einen Speicherzugriff auf die Adressleitungen und die nächste Instruktion wird aus dem Speicher in den Prozessorkern kopiert. Daten- und

[3] **Slave**: Eine passive, d.h. nicht steuernde Komponente am Bussystem

Adressleitungen sind unidirektional. Die individuellen Steuersignale sind ebenfalls unidirektional, allerdings wird das Signal zur Anzeige von Zugriffsfehlern vom Speicher erzeugt.

Port B der Blockram-Bausteine ist mit der Datenspeicherschnittstelle des Prozessors verbunden, die im Wesentlichen aus einigen Registern am Beginn und Ende einer bestimmten Fließbandstufe besteht. Es gibt zwei Bündel von Datenleitungen, damit Daten vom Prozessor zum Speicher und in umgekehrter Richtung transportieren werden können.

Die Signale und Signalnamen für beide Speicherbusse orientieren sich am ARM9TDMI, wurden jedoch auf das absolut notwendige Maß reduziert. Die Signalrichtung in den Tabellen 1 und 2 ist auf den Prozessorkern bezogen. Für Speicher und Peripheriekomponenten kehren sich die Richtungen gerade um. Außerdem wird nicht jede Steuerleitung für jeden Peripheriebaustein benötigt, das `WAIT`-Signal des Instruktionsbus wird beispielsweise gar nicht verwendet und dient nur zukünftigen Erweiterungen. Alle Steuersignale sind, bezogen auf ihren Namen, highaktiv (z.B. ABORT = 1: Fehler, ABORT = 0: kein Fehler).

Signalname	Richtung	Beschreibung
`IA(31:2)`	`out`	*Instruction Address*: Speicheradresse einer Instruktion, formal 32 Bit breit, wobei Bits(1:0) immer 00 sind (Ausrichtung von Instruktionen an Wortgrenzen) und deshalb nicht übertragen werden.
`ID(31:0)`	`in`	*Instruction Data*: Datensignalbündel zur Übertragung von Instruktionen zum Prozessor.
`IABORT`	`in`	*Instruction Abort*: Zeigt an, dass der Speicherzugriff mit der aktuellen Adresse fehlgeschlagen ist, beispielsweise weil der adressierte Speicherbereich physisch nicht existiert. Das Signal wird vom Arbeitsspeicher bzw. der aktuell aktivierten Peripheriebaustein erzeugt.
`IBE`	`in`	*Instruction Bus Output Enable*: Zeigt einem Busmaster an, dass der Instruktionsbus für einen Zugriff zur Verfügung steht.
`IEN`	`out`	*Instruction Bus Enable*: Zeigt den Versuch eines Speicherzugriffs im gleichen Takt durch den Busmaster an. Effektiv ist `IEN` ein Bus-Request-Signal. Ein erfolgreicher Buszugriff ist nur möglich, wenn der Master den Zugriff mit `IEN` beantragt, und der Arbiter mit `IBE` bestätigt. Im Hardwarepraktikum existiert kein zweiter Master am Instruktionsbus, weshalb der `IBE`-Eingang des Prozessorkerns direkt mit dem `IEN`-Ausgang verbunden werden kann. Aus dem gleichen Grund kann `IEN` unmittelbar als Chip-Select-Signal des Instruktionsspeichers verwendet werden.
`MODE(4:0)`		Der aktuell gültige Betriebsmodus des Prozessorkerns für das Holen der nächsten Instruktion, wird im Hardwarepraktikum nicht verwendet.
`WAIT`		Zeigt an, dass der Instruktionsspeicherzugriff im aktuellen Takt nicht beendet werden kann und der Prozessor einen Wartetakt einlegen muss. Wird im HWPR nicht benötigt.

Tabelle 1: Instruktionsbus

Signalname	Richtung	Beschreibung
DA(31:0)	out	*Data Address*: Speicherzugriffsadresse.
DDIN(31:0)	in	*Data Input Bus*: Daten von der Peripherie zum Prozessor.
DDOUT(31:0)	out	*Data Output Bus*: Daten vom Prozessor zur Peripherie.
DMAS(1:0)	out	*Data Memory Access Size*: Zeigt die Art des Speicherzugriffs an: 00: Byte 01: Halbwort 10: Wort 11: Tritt formal nicht auf
DnRW	out	*Data not Read, Write*: Richtung des Speicherzugriffs durch den Prozessor: 0: Lesezugriff 1: Schreibzugriff 4
DABORT	in	*Data Abort*: Zeigt an, dass ein Speicherzugriff auf dem Datenbus fehlgeschlagen ist. Neben dem Zugriff auf physisch nicht vorhandene Adressen kann dies auch auftreten, wenn die durch DMAS angezeigte Zugriffsart nicht zur angelegten Adresse passt.
DEN	out	*Data Bus Output Enable*: Jeder Busmaster verfügt über ein exklusives DEN-Signal, mit dem er einen Buszugriff im gleichen Takt beantragt. Tatsächlich auf Speicher oder Peripherie wirksam sind die vom Master erzeugten Steuersignale jedoch nur, wenn der Zugriffswunsch durch DBE vom Arbiter bestätigt wird.
DBE	in	*Data Bus Enable*: Zeigt dem Master an, dass der Datenbus für einen Zugriff zur Verfügung steht. Es existiert je ein DBE-Signal für jeden Master, sie alle werden von der zentralen Arbitrierungsschaltung gesteuert.
MODE(4:0)		Zeigt die Speichersicht an, mit der ein Master einen Transfer auf dem Datenbus durchführen möchte. Normalerweise, aber nicht zwingend, entspricht dies dem aktuellen Betriebsmodus des Prozessors. Im Hardwarepraktikum wird diese Information vom Chip-SelectGenerator ausgewertet und der Zugriff auf einen bestimmten Peripheriebaustein im User-Modus unterbunden.
WAIT		Peripheriebausteine zeigen mit diesem Signal an, dass ein Zugriff nicht im aktuellen Takt beendet werden kann, im HWPR nicht benötigt.

Tabelle 2: Datenbus

Mehrere (mehr als zwei) Geräte an einem Bus

An den Datenbus werden neben dem Speicher weitere Peripheriekomponenten angebunden. Bei Transaktionen auf dem Datenbus muss aber eine Komponente gezielt angesprochen werden. Da die Kommunikation mit der Peripherie memory-mapped stattfindet, genügt die Datenbus-Adresse zur Auswahl des betreffenden Geräts.

Der Chip-Select-Generator wertet permanent die Adresse auf dem Bus und die DEN-Signale der Master aus und erzeugt maximal ein *Chip-Select-Signal*, mit dem die gewünschte Komponente aktiviert wird. Inaktive Komponenten dürfen weder Daten übernehmen (speichern), noch Bus-Signale treiben. Es ist daher eine echte Abkopplung der durch eine Peripheriekomponente potenziell getriebenen Signalleitungen notwendig, wenn die Peripheriekomponente inaktiv ist (eine 0 an ihrem Chip-Select-Eingang anliegt). Die Abkopplung von Signalen vom Bus erfolgt durch Tristate-Treiber. Bei Schreibzugriffen auf Peripheriekomponenten werden nur die ggf. vorhandenen ABORT-Signale durch die Komponente getrieben. Bei Lesezugriffen treiben Peripheriekomponenten zusätzlich die Datenleitungen zum Prozessor.

Bustreiber : Verstärkerschaltungen, mit denen Komponenten an den Bus gekoppelt werden. Die Verstärker werden benötigt, weil häufig zahlreiche kapazitive Lasten (Signaleingänge der Komponenten am Bus) durch einen sendenden Baustein versorgt werden müssen.

Tristate-Treiber : Treiberbausteine, die neben den logischen Zuständen 0 und 1 einen hochohmigen Zustand zur Signalabkopplung annehmen können. Dies kann z.B. durch gesperrte Feldeffekttransistoren (hoher Kanalwiderstand) geschehen.

Zweiter Busmaster

Der Arbiter stellt sicher, dass niemals zwei Busmaster gleichzeitig Bussignale treiben, indem er die Busmaster untereinander priorisiert. Nur der Master mit der höchsten Priorität, der im aktuellen Takt die Kontrolle über den Bus beantragt, kommt zum Zug. Alle übrigen Master koppeln sich durch Tristate-Treiber vom Bus ab.

Neben dem Prozessor existiert nur ein weiterer Busmaster, der *System Controller*. Er wird primär dazu verwendet, den Speicher mit einem Programm zu initialisieren. Der Controller ist gegenüber dem Prozessor immer priorisiert. Er verfügt wie der Prozessorkern über ein DEN-Signal, dem jedoch immer sofort durch den Arbiter stattgegeben wird.

Der Controller treibt die Bussignale genau dann, wenn er durch sein DEN die Übernahme des Busses anzeigt. Der Prozessor treibt den Datenbus permanent, solange der Arbiter ihm DBE = 1 signalisiert. DBE zum Prozessorkern ist genau dann 1, wenn das DEN-Signal des System Controllers gleich 0 ist. Es gibt deshalb immer einen aktiven Treiber auf allen Datenbus-Signalen, die durch Master erzeugt werden.