

Ausgabe: 10.05.2017

Theorie Präsentation: 17.05.2017

Praxis Abgabe: 24.05.2017

#### Arbeitsziel

Den Unterschied Harvard-Architektur vs. von-Neumann-Architektur verstehen, Arbeitsspeicher für den HWPR-Prozessor aufbauen und mit den notwendigen Bussen verbinden.

#### Arbeitsmaterialien

- Dokumentation Anforderungen an Arbeitsspeicher und Speicheranbindung<sup>1</sup>
- Modul `ArmRAMB_4kx8.vhd`
- Modul `ArmRAMB_4kx32.vhd`
- Modul `ArmMemInterface.vhd`
- Testbench `ArmMemInterface_tb.vhd`

#### Theoretische Vorbetrachtungen

Die folgenden Fragen sind zu beantworten:

- Was wird bzgl. der Anbindung von Speicher (Cache, Hauptspeicher etc.) an den Prozessor unter Harvard-Architektur und von-Neumann-Architektur verstanden?
- Was wird unter den Byteordnungen *Little Endian* und *Big Endian* verstanden?
- Was ist im Kontext von Speicherzugriffen die *Ausrichtung (Alignment)* von Daten?
- Welcher Nachteil ergibt sich bei Speicherzugriffen auf nicht ausgerichteten Daten (Misalignment), vorausgesetzt, ein Prozessor gestattet solche Zugriffe?
- Was geschieht beim ARM9TDMI bei Halbwort- und Byte-Schreibzugriffen auf den Speicher mit den formal nicht benötigten Datenleitungen des Datenbus? Warum verhält sich der Prozessor auf diese Weise?
- Was sind Tristate-(Bus)Treiber und wie können Sie in VHDL formuliert werden?

Empfohlene Quellen zur Bearbeitung:

- Moderne Prozessorarchitekturen, [3, Kapitel 1.2 und S. 86ff]
- Mikroprozessortechnik und Rechnerstrukturen, [1, Kapitel 5.2]
- ARM9TDMI Technical Reference Manual, [2, Kapitel 3.6]
- VHDL-Synthese, [4, Kapitel 4.2]

#### Aufgabenbeschreibung

In dieser Aufgabe wird ein universeller 2-Port-Speicher mit Zugriffsbreiten von 8, 16 und 32 Bit implementiert und anschließend an die zwei Speicherbusse des HWPR-Prozessors angepasst.

---

<sup>1</sup>Steht auf ISIS2.0 als Handout zur Verfügung

## Aufgabe 1 (4 Punkte) Bildung eines geeigneten Arbeitsspeichers

Im letzten Aufgabenblatt wurde bereits ein einfacher Speicher implementiert. Dabei handelt es sich um einen 8 Bit breiten und 4096 Adressen „tiefen“ Speicher. Der Speicher verfügt über zwei unabhängige Ports (A und B) mit jeweils vollständigem Satz von Adress-, Daten- und Steuersignalen. Jeder Port verfügt über ein Daten-Ausgangsregister, das nicht umgangen werden kann. Beide Ports können zum Lesen genutzt werden, jedoch kann ein schreibender Zugriff nur über Port B erfolgen.

Der Arbeitsspeicher des ARM-Prozessors enthält je Zeile 32 bit Daten. Zusätzlich erlaubt die ISA Zugriffe auf unterschiedliche Bereiche (Byte, Halbwort, Wort). Die Konfiguration der Ports ist (wie auch schon beim 8-bit Speicher): ein Lese-Port sowie eine Lese-/Schreib-Port.

Nutzen Sie den bereits im letzten Aufgabenblatt implementierten Speicher **ArmRAMB\_4kx8** um den Arbeitsspeicher für den Prozessor zu implementieren. Ziel ist ein Speicher von 32 Bit Breite und 4096 Adressen. Die Schnittstelle des zu implementierenden Moduls (**ArmRAMB\_4kx32**) finden Sie in Tabelle 1.

Sorgen Sie dafür, dass entsprechend des **WEB** Vektor die Schreibzugriffe nur Wirkung auf die gewünschten Bereiche haben.

RAM_CLK	in	Taktsignal für den Speicher.
ADDRA (11:0)	in	Zugriffsadresse für Port A des Gesamtspeichers.
DOA (31:0)	out	Ausgangsregister von Port A. Wird gebildet aus 4 Datenausgängen gerade adressierter <b>ArmRAMB_4kx8</b> -Komponenten. Der Wert des letzten Lesezugriffs muss für ENA = 0 <b>nicht</b> gehalten werden (DOA benötigt kein zusätzliches Latch oder Register!).
ENA	in	Mit ENA = 0 sind alle Zugriffe auf den Blockram an Port A wirkungslos. Änderungen des Datenausgangs, beispielsweise durch Anlegen einer anderen Adresse, sind aber zulässig.
ADDRB (11:0)	in	Zugriffsadresse für Port B.
DIB (31:0)	in	Daten, die bei einem Schreibzugriff auf Port B an die Adresse <b>ADDRB</b> geschrieben werden.
DOB (31:0)	out	Ausgangsregister von Port B. Wird gebildet aus 4 Datenausgängen gerade adressierter <b>ArmRAMB_4kx8</b> -Komponenten.
ENB	in	Mit ENB = 0 sind alle Zugriffe auf den Blockram an Port B wirkungslos.
WEB (3:0)	in	Mit ENB = 1 und WEB = 0 wird ein Lesezugriff an <b>ADDRB</b> durchgeführt und das gelesene Datum an <b>DOB</b> zur Verfügung gestellt. Mit ENB = 1 und WEB = 1111 wird ein Schreibzugriff mit dem Datum an <b>DIB</b> an Adresse <b>ADDRB</b> durchgeführt. Jedes Bit von WEB ist einem Byte zugeordnet, mit WEB = 0011 würde beispielsweise nur das untere Halbwort von <b>DIB</b> in den Speicher geschrieben.

Tabelle 1: Schnittstelle der 32 Bit Speichers

## Aufgabe 2 (8 Punkte) Anbindung des Speichers an Instruktions- und Datenbus

Ergänzen Sie die vorgegebene Komponente **ArmMemInterface.vhd** mit der Schnittstelle aus Tabelle 2, sodass sie eine Instanz von **ArmRAMB\_4kx32** sinnvoll mit Instruktions- und Datenbus des HWPR-Prozessors verbindet. Verknüpfen Sie PORT A mit den zum Instruktionsbus gehörenden Si-

gnalen von **ArmMemInterface** und PORT B mit den zum Datenbus gehörenden Signalen von **ArmMemInterface**. Neben der Verknüpfung von Speicher- und Bussignalen hat die Speicherschnittstelle noch prüfende Funktionen.

RAM_CLK	in	Speicher-Taktsignal. Der Speicher schreibt neue Inhalte mit der steigenden Taktflanke von RAM_CLK und aktualisiert seine Ausgangsregister ebenso mit der steigenden Flanke. Dieses Verhalten wird bereits durch <b>ArmRAMB_4kx32</b> sichergestellt.
IDE	in	Enable-Signal des Instruktionsbus. Für IDE = 0 wird ID in den Tristate (hochohmig) geschaltet (Hinweis: Wenn sich ID im Tristate befindet, darf DOA von ArmRAMB16_4kx32_Wrapper einen beliebigen Wert annehmen).
IA(31:2)	in	Adressleitungen des Instruktionsbus.
ID(31:0)	out	Instruktionsseitiger Datenausgang, entspricht dem Datenausgang von PORT A, wenn IDE = 1 ist und ist sonst hochohmig.
IABORT	out	Hochohmig für IDE = 0. Für IDE = 1 ist IABORT = 1, wenn IA einen Wert hat, der außerhalb des durch die Konstanten INST_LOW_ADDR und INST_HIGH_ADDR im Package <b>ArmConfiguration.vhd</b> vorgegebenen Adressbereichs liegt.
DDE	in	Enable-Signal der Schnittstelle zum Datenbus. Alle Ausgänge von ArmMemInterface, die eine Verbindung mit dem Datenbus herstellen, sind für DDE = 0 hochohmig. In diesem Fall darf DOB einen beliebigen Wert annehmen.
DA(31:0)	in	Adresssignale des Datenbus.
DDIN(31:0)	in	Datenbus-Datensignale vom Prozessor zum Speicher.
DDOUT(31:0)	out	Datenbus-Datensignale vom Speicher zum Prozessor. Entspricht dem Datenausgang von PORT B für DDE = 1 und DnRW = 0. In allen anderen Fällen ist der Ausgang hochohmig.
DMAS(1:0)	in	Zugriffsart (Wort, Halbwort, Byte) bei Schreib- und Lesezugriffen. Die Codierungen entsprechen den Konstanten von DMAS_TYPE im Package <b>ArmTypes</b> . Die Adresse DA muss einen Wert haben, der zum Typ von DMAS passt. Beispiel: DA(0) kann nur bei Bytezugriffen 1 sein, da Halbwortzugriffe immer an Halbwortadressen und Wortzugriffe immer an Wortadressen erfolgen. Eine Diskrepanz zwischen DA und DMAS wird durch DABORT = 1 angezeigt. Gleiches gilt für die verbotene Kombination DMAS = 11. Lesezugriffe erfolgen unabhängig von DMAS immer wortweise (DA(1:0) wird ignoriert). Der Prozessor passt später das gelesene Wort an den Operationstyp an. Unausgerichtete Lesezugriffe werden durch DABORT = 1 angezeigt und dennoch durchgeführt, unausgerichtete Schreibzugriffe werden angezeigt und dürfen sich nicht auf den Speicherinhalt auswirken.
DnRW	in	Bestimmt die Zugriffsart auf den Datenspeicher. DnRW = 0: lesend; DnRW = 1: schreibend

DABORT	out	Zeigt Fehler beim Speicherzugriff auf dem Datenbus an. DABORT ist hochohmig für DDE = 0, DABORT ist 1 für DDE = 1 wenn ein Fehler beim Speicherzugriff auftritt und sonst 0. DABORT wird innerhalb von <b>ArmMemInterface</b> <b>nicht</b> verwendet, um einen versuchten Zugriff außerhalb des tatsächlich vorhandenen Adressbereichs anzuzeigen.
--------	-----	--

Tabelle 2: Schnittstelle des Speicher-Interfaces

Testen Sie ihre Implementierung mit der Testbench **ArmMemInterface\_tb**.

---

### HINWEIS

Beachten Sie unbedingt, dass die vier Ausgänge IABORT, ID, DABORT und DDOUT in Abhängigkeit von IDE bzw. DDE in den Tristate geschaltet werden.

Testen Sie für alle Speicherzugriffe auf dem Datenbus, ob die Zugriffsadresse (DA) zum angezeigten Zugriffstyp (DMAS) passt oder eine fehlerhafte Datenausrichtung (Misalignment) vorliegt. Nur die Bits (1 : 0) von DA werden für diese Prüfung benötigt. Zeigen Sie Fehler durch DABORT = 1 an.

Testen Sie für alle Zugriffe auf dem Instruktionsbus, ob die Zugriffsadresse im physisch vorhandenen Adressbereich liegt. Signalisieren Sie Fehler mit IABORT = 1.

Beachten Sie, dass die Write-Enable-Signale des Datenspeichers nicht unmittelbar der Codierung von DMAS entsprechen, sondern aus DMAS und der Zugriffsadresse abgeleitet werden müssen.

Offensichtlich muss ein Teil der je 32 Datenadressleitungen und 30 Instruktionsadressleitungen mit den je 12 Bit breiten Adresseingängen beider Ports von **ArmRAMB\_4kx32** verbunden werden. Denken Sie gründlich darüber nach, welcher Teil der Adressbusse (Adresssignale von Instruktions- bzw. Datenbus) benutzt werden muss, wenn insgesamt 16KiByte (4Ki Worte) adressierbar sein sollen.

---

### Mündliche Rücksprache - 4 Punkte

### Literatur

- [1] FLIK, THOMAS, H. LIEBIG und M. MENGE: *Mikroprozessortechnik: CISC, RISC, Systemaufbau, Assembler und C*. Springer Berlin, 6., neu bearb. Aufl. Auflage, 2001. ISBN 3540420428.
- [2] LIMITED, ARM: *ARM9TDMI - TechnicalReference Manual*, 2000. URL <http://infocenter.arm.com/help/topic/com.arm.doc.ddi0180a/DDI0180.pdf>.
- [3] MENGE, MATTHIAS: *Moderne Prozessorarchitekturen. Prinzipien und ihre Realisierungen*. Springer, Berlin, 1 Auflage, März 2005. ISBN 3540243909.
- [4] REICHARDT, JÜRGEN und BERND SCHWARZ: *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*. Oldenbourg, 4., überarbeitete Auflage. Auflage, Oktober 2007. ISBN 3486581929.