

Ausgabe: 07.06.2017

Theorie Präsentation: 14.06.2017

Praxis Abgabe: 21.06.2017

**Arbeitsziel** Verständnis für die Funktionsweise eines Barrelshifters. Grundlegende Kenntnisse über die Möglichkeiten, Shift- und Rotationsoperationen auf Operanden in ARM-Instruktionen zu codieren.

Implementierung von generischen Komponenten.

### Arbeitsmaterialien

- Barrelshifter im Hamburger Design System (Hades) [4]
- Dokumentation ARM Architecture Reference Manual [1]
- Übersicht über die die ARM-Befehlssatzarchitektur, Abschnitt 1.8.3
- Dokumentation Anforderungen an den Shifter
- Modul `ArmBarrelShifter.vhd`
- Modul `ArmBarrelShifter_HILEVEL.vhd` (vollständige Barrelshifterreferenz)
- Modul `ArmShifter.vhd`
- Test Bench `ArmBarrelShifter_tb.vhd`
- Test Bench `ArmShifter_tb.vhd`
- Testvektordatei `BARRELSHIFTER_TESTDATA`
- Testvektordatei `SHIFTER_TESTDATA`

### Theoretische Vorbetrachtungen

Die folgenden Fragen sind von der eingeteilten Gruppe in einem kurzen Vortrag zu beantworten. Alle Fragen bezüglich der ARM-Architektur beziehen sich immer auf ISA ARMv4:

- Beschreiben Sie den grundlegenden Aufbau und die Funktionsweise eines multiplexerbasierten Barrelshifters.
- Wie verhält sich die Zahl der Stufen eines multiplexerbasierten Barrelshifters zur maximalen Schiebeweite?
- Zeichnen Sie ein Strukturbild eines 4-Bit-Barrelshifters. Der Shifter soll die folgenden Operationen ermöglichen:
  - kein Shift
  - Linksshift
  - Rechtsshift
  - Rechtsrotation
- Wie funktioniert die iterative Instantiierung in VHDL?
- Welche Besonderheit existiert in der ARMv4 beim Schieben von Operanden? (*Hinweis: 2 Operand, es gibt keine expliziten Schiebeoperation*)

Empfohlene Quellen zur Bearbeitung:

- Beschreibung des Barrelshifters bei Hades [4]
- ARM9TDMI Technical Reference Manual [2][Kapitel 1]
- Dokumentation der Anforderungen an den Shifter des HWPR-Prozessors (Handout)
- Übersicht über den ARM-Befehlssatzarchitektur [3, Kapitel 1.8.3]

### Aufgabenbeschreibung

Die folgenden Aufgaben dienen der Realisierung des Shifters eines ARM-Prozessors. Der Shifter besteht aus zwei Komponenten: einem multifunktionalen, nicht auf diesen Prozessor beschränkten Barrelshifter und einem Wrapper (einer Abstraktionsschicht) zur Anpassung an Besonderheiten der ARM-Architektur.

### Aufgabe 1 (4 Punkte) Implementierung eines allgemeinen 4-Bit-Barrelshifters

Vervollständigen Sie nun das vorgegebene Modul **ArmBarrelShifter.vhd**, dessen Schnittstelle der Tabelle 1 entnommen werden kann. Berücksichtigen Sie, dass Sie ihre Lösung in Aufgabenteil 2 zu einem 32-Bit-Shifter erweitert werden soll. Sie müssen daher eine Lösung anstreben, die durch *Generics* an andere Operandengrößen angepasst werden kann (konkret: 32 statt 4 Bit). Achten Sie darauf einen Barrelshifter zu implementieren und keinen einfachen Shifter. Verwenden Sie nicht die in VHDL vorhandenen Shiftoperatoren! Bei der Synthese entsteht damit eine Lösung, die zwar kleine Signallaufzeiten aufweist, jedoch mehr FPGA-Ressourcen verbraucht. Informationen zur Implementierung, insbesondere zu den Carry-Bits, können auch in der “Übersicht über die ARM-Befehlssatzarchitektur” auf den S. 19-21 gefunden werden.

OPERAND (3:0)	in	Zu schiebender/rotierender Operand, 4 Bit breit
AMOUNT (1:0)	in	Die Zahl der Stellen (codiert im Dualcode), um die der OPERAND geschoben oder rotiert wird
MUX_CTRL (1:0)	in	Bestimmt die Art der Operation, die durch die Multiplexer auf den Operanden angewendet wird: 00 kein Shift 01 Linksshift 10 Rechtsshift 11 Rechtsrotation
ARITH_SHIFT	in	Ist das Signal gesetzt während MUX_CTRL einen Rechtsshift anzeigt, so wird ein arithmetischer Rechtsshift durchgeführt, sonst ein logischer Rechtsshift
C_IN	in	Carry-Eingangssignal, das beim Schieben berücksichtigt wird
DATA_OUT (3:0)	out	Datenausgang des Shifters
C_OUT	out	Durch den Shifter erzeugtes Carry-Signal

Tabelle 1: Schnittstelle des 4-Bit-Barrelshifters

Testen Sie ihre Implementierung selbstständig mit einer eigenen Testbench, welche qualitativ dem Signalverlauf aus Abbildung 1 entspricht. Halten Sie sich aber an die Reihenfolge der vorgegebenen Testfälle. Es ist ihnen freigestellt, weitere Signalkombinationen neben den vorgegebenen zu testen.



SHIFT_OPERAND (31:0)	in	32 Bit Datum, auf das eine von 5 möglichen Operationen angewendet wird.
SHIFT_AMOUNT (7:0)	in	Gibt die Zahl der Stellen an, um die das Datum geschoben/rotiert wird. Beachten Sie, dass Werte zwischen 0 und 255 möglich sind.
SHIFT_TYPE_IN (1:0)	in	Gibt die Operation an, die auf das Datum angewendet wird. Die Codierung entspricht den Konstanten des Typs SHIFT_TYPE aus dem Package <b>ArmTypes.vhd</b> . 00 SH_SHL Linksshift 01 SH_LSR log. Rechtsshift 10 SH_ASR arith. Rechtsshift 11 SH_ROR Rechtsrotation
SHIFT_RRX	in	Ist das Steuersignal gesetzt, werden SHIFT_TYPE_IN und SHIFT_AMOUNT ignoriert und stattdessen eine RRX-Operation durchgeführt.
SHIFT_C_IN	in	Übertragseingang des Shifters
SHIFT_RESULT (31:0)	out	Datenausgang des Shifters
SHIFT_C_OUT	out	Übertragsausgang des Shifters

Tabelle 2: Schnittstelle des ARM-spezifischen Shifters

### **HINWEIS**

Beachten Sie, dass SHIFT\_TYPE\_IN nicht unmittelbar auf MUX\_CTRL des Barrelshifters abgebildet werden kann. MUX\_CTRL enthält u.a. eine Codierung, für die die Multiplexer den Operanden unverändert durchleiten. SHIFT\_TYPE\_IN steht immer für eine Veränderung des Operanden, die allerdings nur dann ausgeführt wird, wenn SHIFT\_AMOUNT > 0 ist.

---

### **Mündliche Rücksprache - 4 Punkte**

## **Literatur**

- [1] *ARM Architecture Reference Manual*, 2005.
- [2] LIMITED, ARM: *ARM9TDMI - TechnicalReference Manual*, 2000. URL <http://infocenter.arm.com/help/topic/com.arm.doc.ddi0180a/DDI0180.pdf>.
- [3] RECHNERTECHNOLOGIE, TU BERLIN FG: *Hardwarepraktikum Technische Informatik Übersicht über die ARM-Befehlssatzarchitektur*, Oktober 2010.
- [4] (TAMS), UNIVERSITÄT HAMBURG: *Barrelshifter*. <http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/10-gates/60-barrelshifter8.html>. zuletzt abgerufen am 09.11.2011.