**Rechnerarchitekturgroßpraktikum**

Entwicklung eines RISC-V-Prozessors

# Dokumentation

Dominik Fuchsgruber
Charlie Groh
Franz Rieger
Jan Schuchardt

30. Januar 2017

# Inhaltsverzeichnis

# Kapitel 1

# Das Leitwerk

Das Leitwerk ist die zentrale Steuereinheit des Prozessors. Es interpretiert die Befehle und überwacht ihre Ausführung durch ALU und MMU. Dazu verwaltet das Leitwerk den Program-Counter (PC) und das Instruction-Register (IR).

## 1.1 Überblick

Im Prozessor wurden die durch das *RV32I Base Integer Instruction Set* und durch die *RV32M Standard Extension for Integer Multiplication and Division* definierten Befehle implementiert. Eine Ausnahme bilden dabei alle Befehle, die Multitasking ermöglichen sollen, also **SCALL** , **SBREAK**, **FENCE** und **FENCE.I**. Besondere Aufmerksamkeit wurde dabei mehr auf Robustheit und weniger auf maximale Geschwindigkeit gelegt.
Das Leitwerk besitzt für jeden Befehl eine eigene Zustandsmaschine. Welche ausgeführt wird hängt allein vom Inhalt des Instruction-Register ab. Daher muss jeder Befehl als letztes das IR mit dem folgenden Befehl neu laden. Wann dieser neue Befehl geladen wird kann nun in jedem Befehl einzeln optimiert werden.
Um den Implementierungsaufwand bei Änderungen von Befehlen zu minimieren wurde ein Compiler-Skript erstellt, das mehrere Makros bereitstellt, aus denen dann die Befehle zusammengebaut werden können. Das Skript kompiliert dann eine Eingabe aus diesen Makros in VHDL-Code.

### 1.1.1 Legende

Da jeder Befehl eine eigene Zustandsmaschine besitzt wird hier für jeden Befehl ein eigenes Zustandsübergangsdiagramm gezeigt. Die Präfixe “MMU:” und “ALU:” werden genutzt um anzuzeigen, dass eine Aktion von der jeweiligen Einheit ausgeführt wird und das Leitwerk lediglich eine Anweisung gibt.

## 1.2 Integer Rechenbefehle

Der Prozessor wurde auf die in RV32I und RV32M definierten Rechenbefehle optimiert. Dadurch können diese RISC-typischen Befehle mit drei Takten sehr schnell ausgeführt werden.

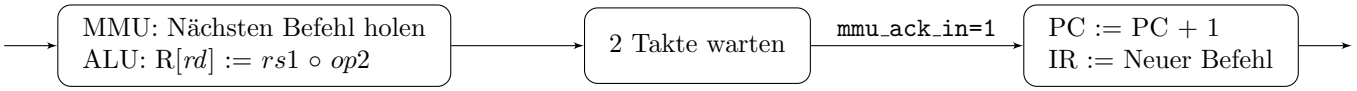


Abbildung 1.1: Zustandsübergangsdiagramm der Befehle **ADD**[**I**], **SUB**, **SLT**[**I**][**U**], **AND**[**I**], **OR**[**I**], **XOR**[**I**], **SLL**[**I**], **SRL**[**I**], **SRA**[**I**], **MUL**[**W**] und **MULH**[[**S**]**U**]. *op*2 ist entweder das Register, das mit *rs2* angegeben ist, oder eine Immediate (*imm*). "∘" repräsentiert die jeweilige Operation (+, −, ...).

### 1.2.1 Division

Da bei der Division unmöglich zu garantieren ist, dass diese immer nach drei Takten beendet ist, muss das Leitwerk hier auf eine Bestätigung der ALU warten. Diese sieht vor, dass das Rechenwerk die Leitungen `alu_data_in` auf 0 setzt.

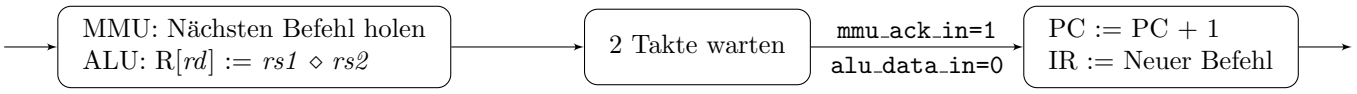


Abbildung 1.2: Zustandsübergangsdiagramm der Befehle **DIV**[**U**][**W**] und **REM**[**U**][**W**]. "⋄" repräsentiert die jeweilige Operation (/, mod, ...).

## 1.3 LUI und AUPIC

## 1.4 Bedingte Sprünge

## 1.5 Unbedingte Sprünge

## 1.6 Speicherzugriffsbefehle

## 1.7 Timer und Counter

Wie in der RISC-V-ISA gefordert gibt es einen Counter, der die Anzahl der bisher ausgeführten Befehle speichert. Außerdem gibt es einen Timer, der die Anzahl der vergangenen Takte speichert. Da das FPGA keine Echtzeituhr bereitstellt, wurde auch hierfür der Taktzähler verwendet.
Ausgelesen werden können diese Counter durch die Befehle **RDINSTRET**[**H**], **RDCYCLE**[**H**] und **RDTIME**[**H**].

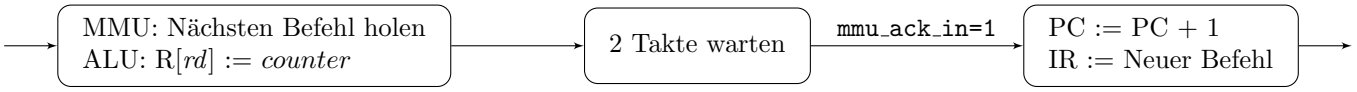


Abbildung 1.3: Zustandsübergangsdiagramm der Befehle **RDINSTRET**[**H**], **RDCYCLE**[**H**] und **RDTIME**[**H**]. *counter* sind die oberen bzw. unteren 32 Bit des jeweiligen 64 Bit Zählers.