Rechnerarchitekturgroßpraktikum

Entwicklung eines RISC-V-Prozessors

Dokumentation

Dominik Fuchsgruber
Charlie Groh
Franz Rieger
Jan Schuchardt

5. Februar 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 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 belegen. Wann dieser 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.
In der ISA wird regelmäßig verlangt, dass Operanden sign-extended werden. Dies wird in den Zustandsübergangsdiagrammen der Übersicht halber weggelassen. Im Prozessor ist es selbstverständlich wie in der ISA beschrieben implementiert.

1.2 Integer Rechenbefehle

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

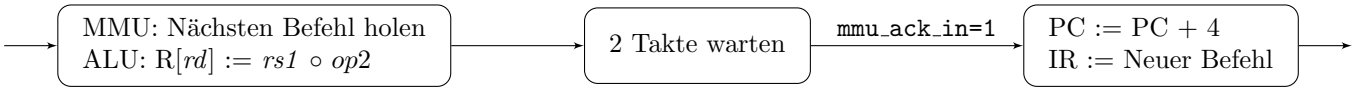

Abbildung 1.1: Zustandsübergangsdiagramm der Befehle **ADD[I]**, **SUB**, **SLT[I][U]**, **AND[I]**, **OR[I]**, **XOR[I]**, **SLL[I]**, **SRL[I]**, **SRA[I]**, **MUL[W]** und **MULH[[S]U]**. *op*2 ist entweder das Register, das mit *rs2* angegeben ist, oder eine Immediate (*imm*). "∘" repräsentiert die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

1.2.1 Division und Modulo

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.

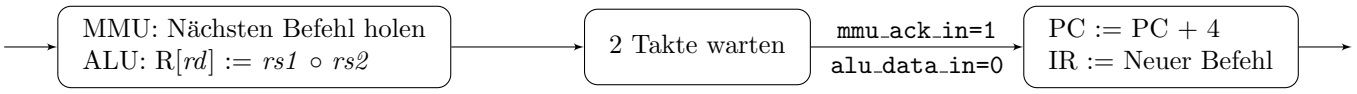

Abbildung 1.2: Zustandsübergangsdiagramm der Befehle **DIV[U][W]** und **REM[U][W]**. "∘" repräsentiert die jeweilige Operation (/, mod, ...).

1.3 LUI und AUPIC

Da durch die Integer Rechenbefehle keine 32-Bit Immediates direkt geladen werden können, definiert die RISC-V-ISA die Befehle **LUI** und **AUPIC**, die diesen Mangel beheben. Auch diese Befehle wurden auf Geschwindigkeit optimiert.

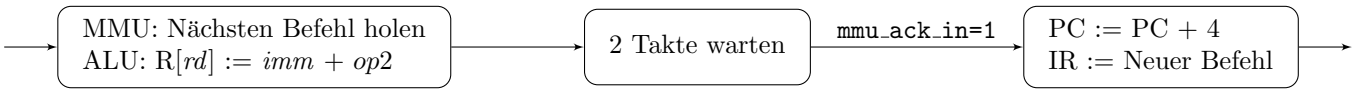

Abbildung 1.3: Zustandsübergangsdiagramm der Befehle **LUI** und **AUPIC**. *op*2 ist entweder 0 (bei **LUI**) oder der aktuelle Program-Counter (bei **AUPIC**.

1.4 Bedingte Sprünge

Da bei dem DDR2-Speicher des benutzten Boards nicht garantiert werden konnte, dass ein Speicherzugriff ohne Zeit- und Datenverlust abgebrochen werden kann, wurde auf eine einfache Branch-Prediction gänzlich verzichtet. Dadurch sind die bedingten Sprünge unter den teuersten Befehlen des Prozessors.

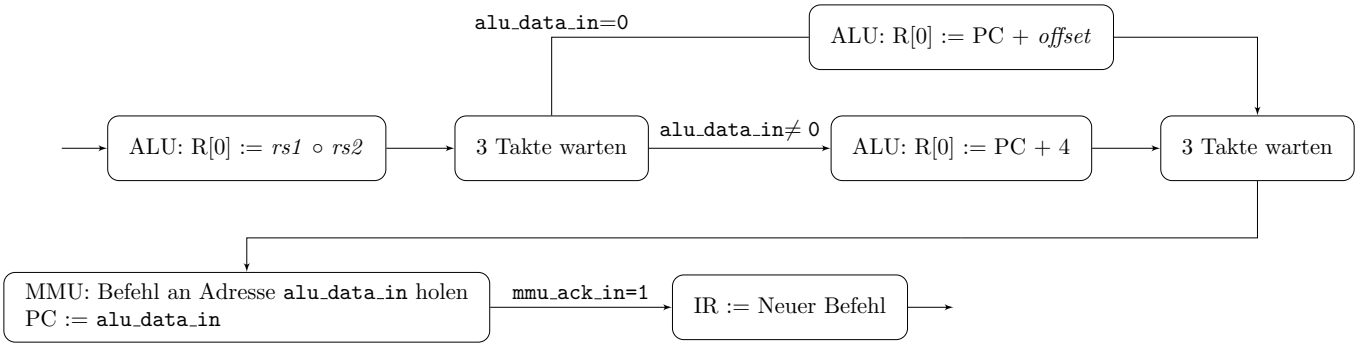

Abbildung 1.4: Zustandsübergangsdiagramm der Befehle **BEQ** und **BGE[U]**. "∘" ist bei **BEQ** "−", bei **BGE** die SLT-Operation und bei **BGEU** die SLTU-Operation.

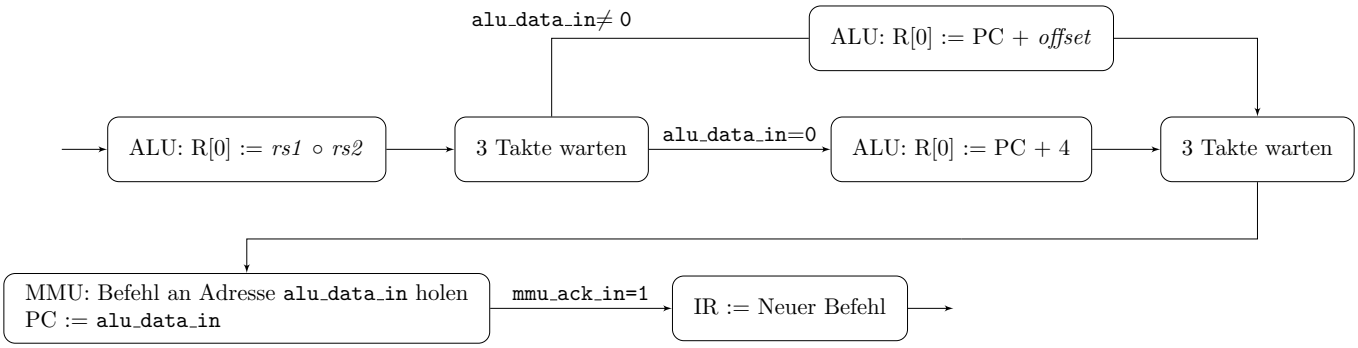

Abbildung 1.5: Zustandsübergangsdiagramm der Befehle **BNE** und **BLT[U]**. "∘" ist bei **BNE** "−", bei **BLT** die SLT-Operation und bei **BLTU** die SLTU-Operation.

1.5 Unbedingte Sprünge

Anders als bei bedingten Sprüngen, kann bei unbedingten Sprüngen das Sprungzeil immer vorhergesagt werden, wodurch das schreiben der Return-Adresse und das holen des nächsten Befehls parallelisiert werden kann, was zu einer merklichen Geschwindigkeitssteigerung führt.

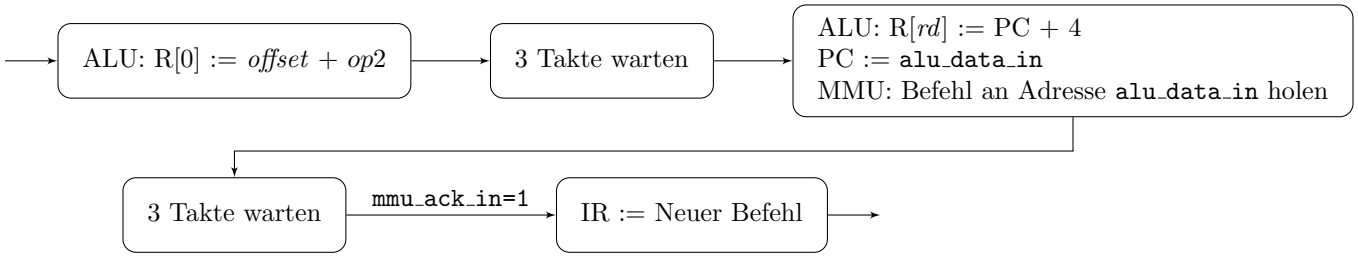

Abbildung 1.6: Zustandsübergangsdiagramm der Befehle **JAL** und **JALR**.

1.6 LOAD

Der LOAD-Befehl lädt aus dem Speicher immer einen 32-Bit Wert, den das Leitwerk dann zuschneidet. Dies sollte ursprünglich die Implementierung der MMU vereinfachen und aligned-Speicherzugriffe beschleunigen, es hat sich allerdings herausgestellt, dass diese Entscheidung derzeit nur Nachteile mit sich bringt. Die Ausführung des Befehls ist durchschnittlich und wird in der Praxis hauptsächlich von der Speichergeschwindigkeit bestimmt.

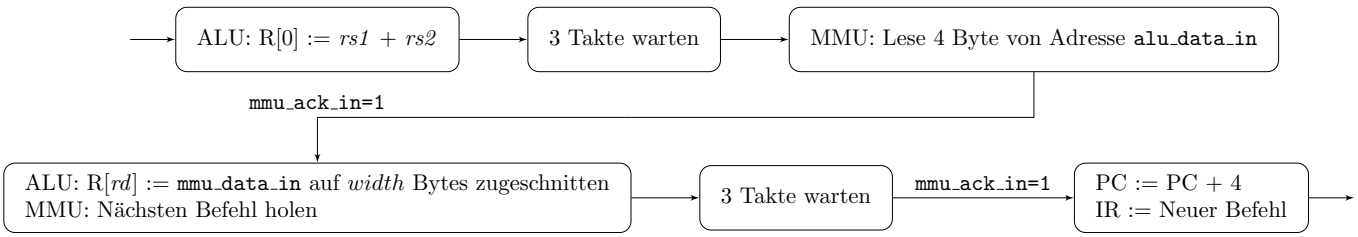

Abbildung 1.7: Zustandsübergangsdiagramm der Befehle **LB[U]**, **LH[U]** und **LW**. *width* ist je nach Befehl 1, 2 oder 4.

1.7 STORE

Der Store-Befehl ist mit nur einer ALU und nur einer MMU nicht zu parallelisieren, was dazu führt, dass er der langsamste Befehl des Prozessors ist. Da dieser Befehl jedoch generell auf RISC-Architekturen sehr langsam ausgeführt wird, versuchen Compiler und Programmierer ohnehin schreibende Speicherzugriffe zu vermeiden.

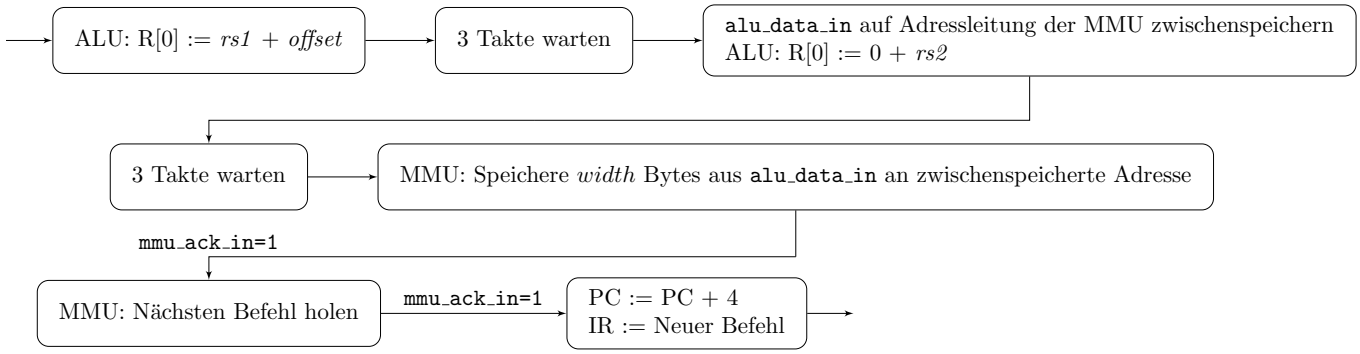

Abbildung 1.8: Zustandsübergangsdiagramm des Befehle **SB**, **SH** und **SW**. *width* ist je nach Befehl 1, 2 oder 4.

1.8 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**].

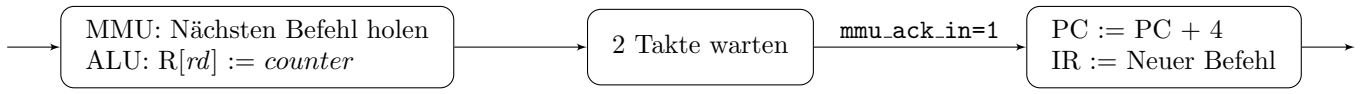

Abbildung 1.9: Zustandsübergangsdiagramm der Befehle **RDINSTRET**[**H**], **RDCYCLE**[**H**] und **RDTIME**[**H**]. *counter* sind die oberen bzw. unteren 32 Bit des jeweiligen 64 Bit Zählers.

1.9 Minimale Taktanzahl

Die hier aufgeführten Taktzahlen, die der Prozessor zur Ausführung eines Befehls benötigt, sind Mindestangaben, dass der Speicherzugriff auf den nächsten Befehl hinreichend schnell erfolgt. (siehe dazu auch MMU)