Versuch 4

Mikroprogramme

In den vorherigen Versuchen haben Sie den Umgang mit elektronischen Bauteilen kennen gelernt und einfache logische Schaltungen realisiert. Damit kennen Sie nun die grundlegenden Prinzipien, nach denen Computerchips auf Hardwareebene Operationen auf Binärdaten ausführen. Dazu zählen neben logischen Operationen auch bitweises Shiften und arithmetische Operationen. Darauf aufbauend können mithilfe von Schaltwerken komplexere, zustandsabhängige Funktionen realisiert werden. Insbesondere kann so das wichtige theoretische Konzept eines Automaten technisch realisiert werden.

Um mithilfe dieser Prinzipien die umfangreiche und komplexe Programmierbarkeit einer modernen Central Processing Unit (CPU) zu erreichen ist ein effizientes Chipdesign erforderlich. Ein wichtiges Designkonzept ist die Erweiterung der in der Hardware umgesetzten Operationen durch sogenannte Mikroprogramme. Als Beispiel für einen einfachen, mit Mikroprogrammen programmierbaren Rechner lernen Sie in diesem Versuch den Minirechner 2i kennen.

4.1 Von-Neumann-Rechner

Die nach dem US-amerikanischen Mathematiker John von Neumann (1903-1957) benannte Rechnersystem-Architektur bildet die Grundlage für die Arbeitsweise fast aller heute gebräuchlichen Computersysteme. Wie Sie aus der Vorlesung wissen, stellt dieser im wesentlichen ein verallgemeinertes komplexes Schaltwerk dar, das um einen Speicher und eine Ein- und Ausgabe erweitert wird. Abbildung 4.1 verschaulicht, wie die vier zentralen Funktionseinheiten von Datenpfad und und Steuerwerk miteinander verbunden sind.

Der Datenpfad setzt sich aus Rechenwerk, Speicher sowie Ein-/ Ausgabe zusammen. Das Rechenwerk stellt ein universelles Operationswerk dar, dessen Herzstück ein steuerbares Schaltnetz ist, welches als arithmetisch-logische Einheit (engl. arithmetic logic unit, ALU) bezeichnet wird. Die Register des Rechenwerks nehmen die Operanden auf, die die ALU miteinander verknüpfen soll. Das Statusregister dient zur Anzeige besonderer Ergebnisse, wodurch bedingte Verzweigungen in Steueralgorithmen für Maschinenbefehle ausgeführt werden können; die einzelnen Bits des Statusregisters werden als Flags bezeichnet.

Das Steuerwerk eines Von-Neumann-Rechners wird mittels eines so genannten Leitwerks realisiert. Dieses lädt Maschinenbefehle aus dem Speicher ins Befehlsregister (Holephase) und interpretiert diese anschließend (Ausführungsphase). Dazu wird ein Maschinenbefehl in eine Folge von Steuerworten für Rechenwerk, Speicher und Ein-/ Ausgabe umgesetzt. Diese Steuerworte werden durch eine Ablaufsteuerung erzeugt, die entweder festverdrahtet oder als Mikroprogramm-Steuerwerk umgesetzt ist.

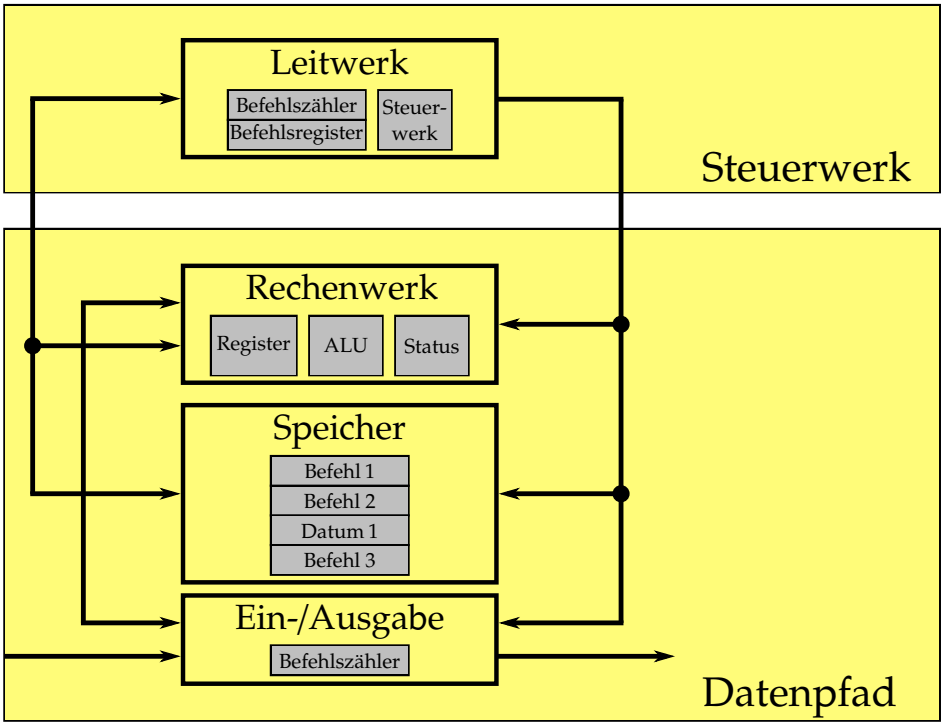

Abbildung 4.1: Blockschaltbild eines Von-Neumann-Rechners.

4.2 Befehlssatz und Mikroprogrammierung

Der Befehlssatz eines Prozessors und die Fähigkeiten des darin enthaltenen Rechenwerks sind in der Regel nicht identisch. Während beispielsweise in einer ALU meist nur einfache Befehle wie Addition oder Shift-Operationen umgesetzt sind, erlauben alle gängigen Prozessorbefehlssätze auch Multiplikation und Division. Verknüpft werden Prozessor-Befehlssatz und Rechenwerk-Fähigkeiten durch so genannte Mikroprogramme, die durch das Steuerwerk des Rechners ausgeführt werden. Zu jedem Maschinenbefehl gibt es einen Bereich im Speicher, der eine Menge entsprechend definierter Steuerworte enthält und den man als Mikroprogramm bezeichnet.

Der Befehlssatz eines Prozessors wird somit durch die Menge sämtlicher Mikroprogramme definiert. Ein Mikroprogramm-Steuerwerk kann also auch als ein Hardware-Interpreter für den Befehlssatz des Prozessors betrachtet werden. Mikroprogrammierbare Rechner verfügen über einen RAM-Speicher zur Aufnahme der Mikroprogramme und können aufgrund dieser Flexibilität leichter entwickelt und gewartet werden als festverdrahtete Steuerwerke. Die Entwicklung von CISC-Prozessoren etwa, die über sehr umfangreiche Befehlssätze verfügen, ist ohne Mikroprogrammierung nicht denkbar. In der Praxis haben die meisten Prozessoren einen festen Befehlssatz, d.h. der Anwender kann die Mikroprogramme bzw. Firmware nicht verändern.

Eine Instruktion eines Mikroprogramms setzt sich im wesentlichen aus einem Steuerwort zur Auswahl der Operationen im Rechenwerk und einem Adressauswahlwort, mit dem die Adresse der nächsten Instruktion ausgewählt wird, zusammen. Das Steuerwort ist dabei in mehrere voneinander unabhängige Felder unterteilt. Jedes Feld kodiert eine Mikrooperation, die gleichzeitig zu den Mikrooperationen anderer Felder ausgeführt werden kann. Je mehr Mikroperationen gleichzeitig ausgeführt werden, umso weniger Instruktionen werden benötigt. Andererseits erhöht die Zahl der parallelen Mikrooperationen auch die Zahl der Steuerbits bzw. den Hardwareaufwand des Rechenwerks.

4.3 Der Minirechner 2i

Der Minirechner 2i ist eine Modellarchitektur für einen Mikroprozessor, welcher eine deutlich geringere Komplexität aufweist als viele in der Praxis eingesetzten Prozessoren. Im Gegensatz zu modernen 32- oder 64-Bit-Architekturen handelt es sich um einen 8-Bit-Rechner. Außerdem sind deutlich weniger arithmetische und logische Funktionen implementiert. Der Minirechner ist dadurch vergleichsweise einfach mit binären Befehlen programmierbar.

Für die Durchführung der Versuche soll der hierfür entwickelten Emulator verwenden[1] werden. Hinweise zur Installation und Nutzung des Emulators finden Sie in der Bedienungsanleitung des Minirechners 2i.

4.3.1 Datenpfad

Die Komponenten des Datenpfads sind jeweils über 8-Bit-Leitungen miteinander verknüpft. Insbesondere verarbeitet das Rechenwerk jeweils 8-Bit-breite Daten.

Rechenwerk. Der Register-Block besteht aus 8 Registern zu je 8 Bit und besitzt entsprechende Daten-Eingang und zwei Daten-Ausgänge mit je 8 Bit. Mit Hilfe des 3-Bit-breiten Adresseinganges AA kann ausgewählt werden, welches Register an den Ausgang angelegt werden soll. Sollen Daten in eines der Register geschrieben werden, so wird mit dem Eingang WS festgelegt, welche der beiden Adressen (AA oder AB) zur Auswahl des zu beschreibenden Registers verwendet wird.

Die ALU enthält drei Funktionseinheiten (Abbildung 4.2), mit denen man binäre und unäre Operationen durchführen kann (siehe auch Tabelle 4.1): ein 8-Bit-breites NOR, ein 8-Bit-Volladierer und einen 8-Bit-breiten Shifter. Als Eingang A dient entweder der Ausgang DOA des Registerblocks oder der Memory-Datenbus MEMDI (Memory Data In). Als Datenquelle für den Eingang B der ALU kann der Ausgang DOB des Registerblocks oder eine Konstante

Tabelle 4.1: Funktionen der ALU

Befehl allg.	Befehl bei $B = A$	Funktion	C	N	Z	Bemerkung
ADDH	LSLH	$F = A + B$	OR	*	*	**add** and **h**old carry: $C = Cin \vee C$
A	-	$F = A$	0	*	*	Eingang **A** durchreichen
NOR	COM	$F = A$ NOR B	0	*	*	bei $B = A$: **com**plement
0	-	$F = 0$	0	0	1	Ergebnis immer 0
ADD	LSL	$F = A + B$	Ca	*	*	bei $B = A$: **l**ogical **s**hift **l**eft
ADDS	(SL1)	$F = A + B + 1$	$\overline{Ca}$	*	*	**add** for **s**ubtraction bei $B = A$: **s**hift **l**eft, rechts **1** einschieben
ADC	RLC	$F = A + B + Cin$	Ca	*	*	**ad**d with **c**arry bei $B = A$: **r**otate **l**eft through **c**arry
ADCS	-	$F = A + B + \overline{Cin}$	$\overline{Ca}$	*	*	**ad**d with **c**arry for **s**ubtraction
LSR	-	$F(n) = A(n+1), F(7) = 0$	$A(0)$	*	*	**l**ogical **s**hift **r**ight, links 0 einschieben
RR	-	$F(n) = A(n+1), F(7) = A(0)$	$A(0)$	*	*	**r**otate **r**ight
RRC	-	$F(n) = A(n+1), F(7) =$Cin	$A(0)$	*	*	**r**otate **r**ight through **c**arry
ASR	-	$F(n) = A(n+1), F(7) = A(7)$	$A(0)$	*	*	**a**rithmetic **s**hift **r**ight
B CLC	-	$F = B$	0	*	*	Eingang **B** durchreichen **cl**ear **c**arry flag
SETC	-	$F = B$	1	*	*	**set c**arry flag
BH	-	$F = B$	Cin	*	*	**B** and **h**old carry flag
INVC	-	$F = B$	$\overline{Cin}$	*	*	**inv**ert carry flag

A, B = Dateneingänge, F = Ergebnis, C = carry out, N = negative out, Z = zero out, * = entsprechend dem Ergebnis F; Cin = Carry input in ALU, Ca = Carry aus Addierer, $\overline{xy}$ = Signal xy invertiert

[1] `https://git.informatik.uni-leipzig.de/ti/hwprak/2i-emulator`

zwischen -8 und +7 dienen. Welche Datenquelle auf welchem Eingang liegt, wird mittels eines 8-Bit-breiter 2-zu-1-Multiplexer (Abbildung 4.2) festgelegt. Die Adressen für das Datenmemory (Adressbus) kommen immer aus dem Register-Ausgang DOA.

Außerdem besitzt das Rechenwerk noch 3 Flag-Register: „Carry“ für arithmetischen Überlauf oder rausgeschobenes Bit beim Shiften, „Zero“ für Ergebnisse mit dem Wert 0 und „Negative“ für Ergebnisse aus negativen Zahlen. Das zwischengespeicherte Carry-Bit ist an den Carry-Eingang der ALU zurückgeführt und kann für Berechnungen mit mehr als 8 Bit Datenwortbreite benutzt werden.

Speicher. Das Daten-RAM ist über den Memory-Bus an das Rechenwerk angekoppelt. Dieser Bus besteht aus 8 Datenleitungen vom Rechenwerk zum Speicher, 8 Datenleitungen vom Speicher zum Rechenwerk, 8 Adressleitungen und 3 Steuerleitungen. Außer dem Speicher hängen an diesem Bus noch 4 Input-Register, 2 Output-Register und eine serielle Schnittstelle (Universal Asynchronous Receiver and Transmitter, UART).

Ein-/ Ausgabe. Die Input-Register können bitweise beschrieben werden und dienen zum Eingeben von Daten in den Minirechner; sie können vom Rechenwerk nur gelesen werden. Die Output-Register können durch das Rechenwerk beschrieben werden und dienen zur Visualisierung von Ergebnissen durch LEDs. Über den UART kann der Minirechner z.B. mit einem PC kommunizieren. Die Input- und Output-Register und der UART befinden sich innerhalb des FPGAs.

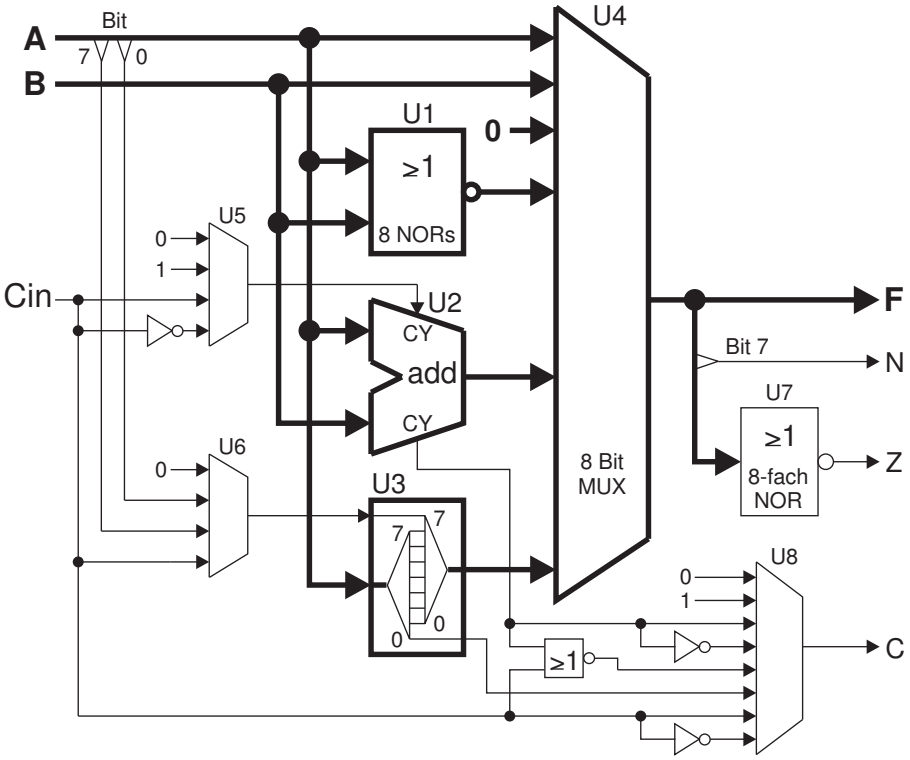

Abbildung 4.2: Blockschaltbild der ALU (dicke Leitungen enthalten 8 Bit).

4.3.2 Steuerwerk

Im Mikroprogramm-RAM des Steuerwerks können 32 Befehlsworte abgelegt werden. Ein Befehlswort setzt sich aus 25 Bit zusammen (Tab. 4.1) und steuert alle Einheiten des Minirechners; vier dieser Bits sind z.B. mit der ALU verbunden. Bei jedem Takt wird ein neues Wort aus dem Mikroprogramm-RAM ausgelesen und an die entsprechenden Steuerleitungen angelegt. Ein vollständige Darstellung, welche Einheit mit welchem Befehlsbit gesteuert wird, kann der Bedienungsanleitung des Minirechners 2i entnommen werden.

Die Reihenfolge des Auslesens wird dabei nicht von einem Zähler vorgegeben, sondern vom Befehlswort selbst bestimmt. Mit den fünf Bits im Feld `next address` wird die nächste auszulesende Adresse angegeben. Da das unterste Adressbit (NA0) durch den 8-zu-1-Multiplexer vor dem Adresseingang A0 des Mikroprogramm-RAMs ausgewählt wird, können dadurch bedingte Verzweigungen im Mikroprogramm realisiert werden. Die dreistelligen Binärzahlen an den Eingängen des Multiplexers geben an, bei welchem Code an den Steuereingängen 2,1,0 der jeweilige Dateneingang an den Ausgang geschaltet wird.

Der Memory-Controller erzeugt aus den Steuerbits „bus write“ bzw. „bus enable“ Steuersignale für den Daten-RAM. Da zur korrekten Ansteuerung des Daten-RAM jeder Zugriff auf den Speicher-Bus zwei Takte benötigt, generiert der Controller noch ein Wartesignal; dieses friert alle Abläufe in Steuerwerk und Datenpfad für je einen Takt ein.

Zur Behandlung von Ausnahmesituationen enthält das Steuerwerk außerdem eine Interrupt-Logik, mit deren Hilfe man den Programmablauf manuell beeinflussen kann. Dies kann z.B. verwendet werden, um ein Mikroprogramm per Tastendruck aus einer Warteschleife herauszuholen, um eine (größere) Programmschleife genau einmal abzuarbeiten.

	Steuerwerk				Datenpfad							
Gruppe:	microprogram control		bus control		register control				ALU control			flag register control
Bedeutung:	microprog address control	next address	bus write	bus enable	register address port A	register address port B	register write port select	register write enable	ALU input A select	ALU input B select	ALU function select	change flags
Signal:	MAC 1 - 0	NA 4 - 0	BUSWR	BUSEN	MRGAA 2 - 0	MRGAB 3 - 0	MRGWS	MRGWE	MALUIA	MALUIB	MALUS 3 - 0	MCHFLG
Bit-Nr.:	24-23	22-18	17	16	15-13	12-9	8	7	6	5	4-1	0

Tabelle 4.1: Aufbau eines Steuerwortes für den Minirechner 2i.

4.4 Weiterführende Literatur

Bei diesem Kapitel handelt es sich um eine Zusammenfassung, die Sie in den Themenkomplex einführen soll. Zusätzlich werden folgende Materialien zum Selbststudium empfohlen:

- Wolfram Schiffmann. Technische Informatik 2 - Grundlagen der Computertechnik. Springer, Berlin, 5. Auflage, 2005. (Kapitel 2)
- Helmut Bähring. Mikrorechner-Technik 1, Band 1 (Mikroprozessoren und Digitale Signalprozessoren). Springer, Berlin, 3. Auflage, 2002. (Kapitel 2)
- Max Braungardt, Thomas Schmid. Der Minirechner 2i - Handbuch und Bedienungsanleitung Version 1.2. Leipzig, 2020.[2]

[2]Dieses Dokument steht Ihnen als PDF im Moodle-Kurs zum Download zur Verfügung.

4.5 Versuche

Hinweis: Die folgenden Aufgaben sollen mit dem Emulator zum Minirechner 2i umgesetzt werden.
Hinweis: Wenn Ihr Quellcode vollständig kommentiert ist, genügt als Ausarbeitung die Abgabe der beantworteten Fragen und die (digitalen) Quellcodedateien.

1. **Speicherzugriff**
   (a) Rechnen Sie die Hexadezimal-Konstanten `0x02` und `0xAA` in die Binärwerte um. Laden Sie beide Konstanten hintereinander konkateniert (also `0x02AA`) als Mikroprogramm an die Adresse 0. Versuchen Sie diese Zeile als Befehl in einer Dauerschleife auszuführen und erklären Sie kurz, was passiert.
   Hinweis: Füllen Sie nicht verwendete höherwertige Bits mit Nullen auf.
   (b) Schreiben Sie die Hexadezimalwerte `0x2` in das Input-Register FE und `0xA` in das Input-Register FF. Führen Sie die Programmzeile aus 1.(a) erneut als Schleife aus. Was hat sich geändert?

2. **Mikroprogramme**
   (a) Schreiben Sie für die nachfolgenden Aufgaben Mikroprogramme, die die gegebenen Operationen mit den Eingaben A und B durchführen. A und B seien zwei beliebige 8-Bit-Zweierkomplement-Zahlen (Wertebereich von -128 bis 127). Das Ergebnis soll in einem Output-Register abgelegt sein und das Programm an den Anfang zurückspringen. Die Werte 2 bzw. 4 sind Konstanten.
   (b) Führen Sie folgende Rechenaufgabe im Emulator aus: `0xFF`$+1$. Was fällt Ihnen bezüglich der Flags auf? Erklären Sie diese Auffälligkeit.
   (c) Führen Sie folgende Rechenaufgabe im Emulator aus: $(A + 4B) \div 2$
   (d) Führen Sie folgende Rechenaufgabe im Emulator aus: $A \veebar B$ (bitweises XOR)
   (e) Führen Sie folgende Rechenaufgabe im Emulator aus: $A \cdot B$. Verwenden Sie den Algorithmus für die schriftliche Multiplikation: *Shift-and-Add*.
   (f) Führen Sie folgende Rechenaufgabe im Emulator aus: $A \div B$. Gehen Sie bei dieser Aufgabe davon aus, dass A und B zwei positive Zahlen (Wertebereich: 0 bis 255) sind. Geben Sie das Ergebnis und den Rest aus. Beachten Sie die Fälle, dass Divisor oder Dividend den Wert 0 annehmen können.