

Minirechner 2i

Der Minirechner besteht aus 3 Platinen:

1. einer Logikplatine, die im Wesentlichen ein FPGA (Field Programmable Gate Array) und ein Daten-RAM (Random Access Memory) enthält;
2. einer Display-Platine, auf der sich ca. 300 LEDs befinden;
3. einer Erweiterungsplatine mit 2 Digital-Analog-Wandlern, die aber bei diesem Versuch nicht verwendet wird. Die CPU ist komplett im FPGA realisiert.

Das Daten-RAM ist über den Memory-Bus an die CPU angekoppelt. Dieser Bus besteht aus folgenden Leitungen:

- 8 Datenleitungen von der CPU zum Memory (MEMDO = **M**emory **D**ata **O**ut im Blockschaltbild)
- 8 Datenleitungen vom Memory zur CPU (MEMDI = **M**emory **D**ata **I**n)
- 8 Adressleitungen (MEMA = **M**emory **A**dress)
- 3 Steuerleitungen (CE = **C**hip **E**nable, OE = **O**utput **E**nable, WE = **W**rite **E**nable)

Außer dem Memory hängen an diesem Bus noch 4 Input-Register ("in FC" bis "in FF" auf Adresse FC bis FF), 2 Output-Register ("out FE" und "out FF" auf Adresse FE bzw. FF) und eine serielle Schnittstelle (UART: Universal Asynchronous Receiver and Transmitter, Adressen FA und FB). Die Input-Register können mit den Tastern bitweise beschrieben werden und dienen zum Eingeben von Daten an die CPU; sie können von der CPU nur gelesen werden. Die Output-Register können durch die CPU beschrieben werden und dienen zur Visualisierung von Ergebnissen (werden durch LEDs angezeigt). Über den UART kann die CPU z.B. mit einem PC kommunizieren. Die Input- und Output-Register und der UART befinden sich innerhalb des FPGAs, das Daten-RAM ist ein eigener Baustein. Das Daten-RAM und die Input- und Output-Register sind im Blockschaltbild nicht enthalten, da sie nicht Teil der eigentlichen CPU sind.

Die CPU besteht aus 2 Grundeinheiten, dem Datenpfad und der Steuerung.

1.0.1 Datenpfad

Der Datenpfad kann 8-Bit-breite Daten verarbeiten und besteht aus einem Register-Block aus 8 universellen Registern (zu je 8 Bit) und einer ALU (Arithmetic Logic Unit). Vor den beiden Eingängen A und B der ALU befindet sich jeweils ein 8-Bit-breiter 2-zu-1-Multiplexer, mit dem die für die ALU bestimmten Daten aus je 2 Quellen ausgewählt werden können.

Der Register-Block besitzt einen 8-Bit-breiten Daten-Eingang und zwei 8-Bit-breite Daten-Ausgänge. Mit Hilfe des 3-Bit-breiten Adresseinganges AA (**A**ddress **A**, bestehend aus AA0, AA1 und AA2) kann ausgewählt werden, welches Register an den Ausgang DOA (**D**ata **O**ut **A**) angelegt werden soll. Entsprechendes gilt für AB0 bis AB3 (**A**ddress **B**) und DOB (**D**ata **O**ut **B**). Sollen Daten in eines der Register geschrieben werden, so muss mit dem Eingang

WS (Write Select) ausgewählt werden, welche der beiden Adressen (AA oder AB) für die Auswahl des zu beschreibenden Registers verwendet werden soll. Wird der Eingang WE (Write Enable) aktiviert, so werden die am 8-Bit-breiten Eingang DI (Data In) anliegenden Daten bei der nächsten aktiven Taktflanke in das ausgewählte Register geschrieben.

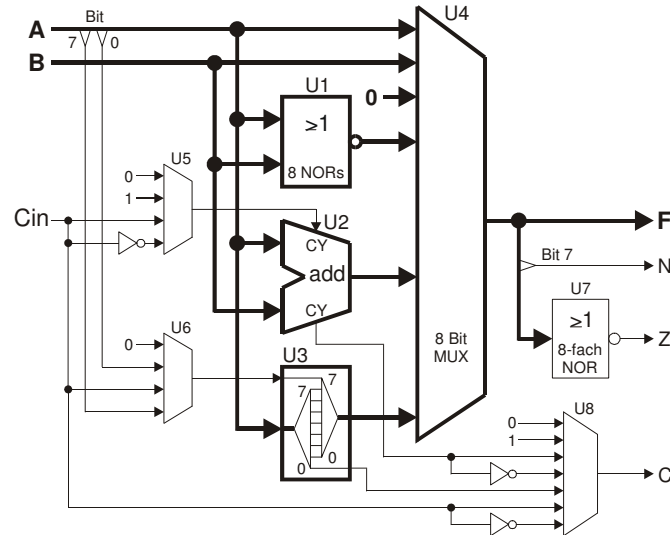


Abbildung 1.1: Blockschaltbild der ALU (dicke Leitungen enthalten 8 Bit)

Als Datenquelle für den Eingang A der ALU kann entweder der Ausgang DOA der Register-File oder der Memory-Datenbus MEMDI (**Memory Data In**) dienen (Auswahl durch Steuereingang MALUIA (**Microprogram-Bit ALU Input A select**) des Multiplexers vor Eingang A). Als Datenquelle für den Eingang B der ALU kann der Ausgang DOB der Register-File oder eine Konstante zwischen -8 und +7 dienen (Auswahl durch MALUIB, die Konstante wird durch die Steuereinheit bestimmt).

Die Adressen für das Datenmemory (Adressbus) kommen immer aus dem Ausgang DOA der Register-File.

Die ALU enthält drei arithmetische bzw. logische Funktionseinheiten:

- das 8-Bit-breite NOR U1 (bestehend aus 8 NORs mit je 2 Eingängen)
- den 8-Bit Volladdierer U2 (mit Carry-Eingang und Carry-Ausgang)
- den Shifter U3, der den Wert um 1 Bit nach rechts versetzt durchreicht

Mit Hilfe des Multiplexers U4 wird der gewünschte Wert ausgewählt und an den Ausgang F gelegt. Der Shifter ist in Wirklichkeit keine eigene Schaltung, sondern es ist lediglich der Eingang A um 1 Bit versetzt an den Multiplexer U4 angeschlossen.

Die gesamte ALU ist ein reines Schaltnetz; deshalb steht der berechnete Wert noch innerhalb des gleichen Taktes am Ausgang zur Verfügung.

Außerdem besitzt die ALU noch 3 Flag-Ausgänge:

- C = Carry = arithmetischer Überlauf oder rausgeschobenes Bit beim Shiften
- Z = Zero = Ergebnis ist 0
- N = Negative = Ergebnis ist eine negative Zahl (= Bit 7 ist gesetzt)

Diese Flag-Ausgänge können in einem 3-Bit-breiten Register ("Flags") zwischengespeichert werden. Dieses Register übernimmt die Werte an den Eingängen D mit steigender

Taktflanke, wenn gleichzeitig am Eingang EN (**enable**) eine 1 anliegt. Solange EN=0 ist, bleiben die Ausgänge Q unverändert. Das zwischengespeicherte Carry-Bit ist an den Carry-Eingang der ALU zurückgeführt und kann für Berechnungen mit mehr als 8 Bit Datenwortbreite benutzt werden. Die Multiplexer in der ALU werden durch die 4 Steuereingänge S0 bis S3 (**select** 0 bis 3) gesteuert.

1.0.2 Steuerung

Die Steuerung besteht hauptsächlich aus dem Mikroprogramm-RAM. In diesem RAM (im FPGA integriert) können 32 Worte zu je 25 Bit abgelegt werden. Mit diesen 25 Bits (= Signalen) werden alle Einheiten des Minirechners gesteuert; z.B. sind 4 dieser Bits mit den 4 Steuereingängen der ALU verbunden. Bei jedem Takt wird ein neues Wort aus dem Mikroprogramm-RAM ausgelesen und z.B. an die Steuereingänge des Datenpfades angelegt. Die Reihenfolge des Auslesens wird dabei nicht von einem Zähler vorgegeben, sondern vom Steuerwort selbst bestimmt: 7 Bits des Steuerwortes dienen dazu, die nächste auszulesende Adresse festzulegen. Mit den 5 Bits NA0 bis NA4 (**N**ext **A**ddress) wird die nächste auszulesende Adresse angegeben. Das unterste Adressbit (NA0) kann dabei durch einen der Flag-Ausgänge der ALU, durch das zwischengespeicherte Carry-Flag oder durch eine Oder-Verknüpfung von 3 externen Signalen ersetzt werden. Welches Signal als Adressbit 0 dient, wird durch den 8-zu-1-Multiplexer vor dem Adresseingang A0 des Mikroprogramm-RAMs ausgewählt. Dieser Multiplexer wird durch die Mikroprogramm-Bits MAC1, MAC0 (**M**icroprogram **A**ddress **C**ontrol) und NA0 angesteuert. Die 3-stelligen Binärzahlen an den Eingängen des Multiplexers geben an, bei welchem Code an den Steuereingängen 2,1,0 der jeweilige Dateneingang an den Ausgang durchgeschaltet wird. Auf diese Weise sind bedingte Verzweigungen im Mikroprogramm möglich.

In der Steuerung befindet sich noch der Memory-Controller. Dieser erzeugt aus den Mikroprogramm-Bits BUSEN (**B**us **E**nable) und BUSWR (**B**us **W**rite) die Steuersignale CE (**C**hip **E**nable), OE (**O**utput **E**nable) und WE (**W**rite **E**nable) für das Daten-Memory. Damit das Daten-Memory richtig angesteuert wird, benötigt jeder Zugriff auf den Memory-Bus 2 Takte. Deshalb generiert der Memory-Controller noch das Signal "Wait", das alle Abläufe in der Steuerung und dem Datenpfad einfriert, so lange es aktiv ist (es wird jeweils für 1 Takt aktiviert).

Blaue Signalnamen im Blockschaltbild sind Mikroprogramm-Bits, grüne kommen aus dem Datenpfad, rote vom Memory-Controller und schwarze von außen.

Alle Signale im Datenpfad, die aus dem Mikroprogramm kommen, beginnen mit dem Buchstaben 'M'.

Tab. 1: Funktionen der ALU

Die ALU verknüpft die beiden 8-Bit-breiten Eingänge A und B auf die in der Tabelle angegebene Weise und gibt das Ergebnis am Ausgang F aus.

Steuer- eingänge S... 32 10	Befehl allg.	Befehl speziell bei $B = A$	Funktion	C	N	Z	Bemerkung
00 00	ADDH	LSLH	$F = A + B$	OR	*	*	add and hold carry: $C = Cin \vee C$
00 01	A	-	$F = A$	0	*	*	Eingang A durchreichen
00 10	NOR	COM	$F = A \text{ NOR } B$	0	*	*	bei $B = A$: complement
00 11	0	-	$F = 0$	0	0	1	Ergebnis immer 0
01 00	ADD	LSL	$F = A + B$	Ca	*	*	bei $B = A$: logical shift left
01 01	ADDS	(SL1)	$F = A + B + 1$	Ca	*	*	add for subtraction bei $B = A$: shift left, rechts 1 einschieben
01 10	ADC	RLC	$F = A + B + Cin$	Ca	*	*	add with carry bei $B = A$: rotate left through carry
01 11	ADCS	-	$F = A + B + \overline{Cin}$	\overline{Ca}	*	*	add with carry for subtraction
10 00	LSR	-	$F(n) = A(n+1), F(7) = 0$	$A(0)$	*	*	logical shift right, links 0 einschieben
10 01	RR	-	$F(n) = A(n+1), F(7) = A(0)$	$A(0)$	*	*	rotate right
10 10	RRC	-	$F(n) = A(n+1), F(7) = Cin$	$A(0)$	*	*	rotate right through carry
10 11	ASR	-	$F(n) = A(n+1), F(7) = A(7)$	$A(0)$	*	*	arithmetic shift right
11 00	B CLC	-	$F = B$	0	*	*	Eingang B durchreichen clear carry flag
11 01	SETC	-	$F = B$	1	*	*	set carry flag
11 10	BH	-	$F = B$	Cin	*	*	B and hold carry flag
11 11	INVC	-	$F = B$	\overline{Cin}	*	*	invert 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

Tab. 2: Mikroprogramm-Bits

Name	Anzahl Bits	Bit Nr.	Bedeutung
Datenpfad:			
MCHFLG	1	0	Change Flags: 1 = Ausgänge C,Z,N der ALU in Register übernehmen
MALUS3...MALUS0	4	4...1	Funktion der ALU
MALUIB	1	5	Auswahl Eingang B der ALU: 0 = Register Ausgang B (DOB) 1 = Konstante
MALUIA	1	6	Auswahl Eingang A der ALU: 0 = Register Ausgang A (DOA) 1 = Datenbus (MEMDI: Memory Data In, Input-Register usw.)
MRGWE	1	7	Register Write Enable
MRGWS	1	8	Register Write Select: 0 = Write-Adresse ist AA2...AA0 1 = Write-Adresse ist AB2...AB0
MRGAB3...MRGAB0	4	12...9	Register Adresse Port B und Konstante für Eingang B der ALU xnnn: Register-Adresse = nnn Konstante: 1000 ... 0111 = -8 ... +7
MRGAA2...MRGAA0	3	15...13	Register Adresse Port A
Summe Datenpfad:	16		
Steuerung:			
BUSEN	1	16	Bus Enable: Datenbus wird angesprochen (read oder write)
BUSWR	1	17	Bus Write: Datenrichtung: 0 = lesen, 1 = schreiben
NA4...NA0	5	22...18	Next Address: nächste Mikroprogramm-Adresse (siehe nächste Tab.)
MAC1...MAC0	2	24...23	Microprogram Address Control: siehe Tabelle 3
Summe Steuerung:	9		
gesamt:	25		

Tab. 3: Nächste Adresse im Mikroprogramm

Zeile	MAC1...0	NA0	nächste Adresse				
			Bit 4	3	2	1	0
1	0 0	x	NA4	NA3	NA2	NA1	NA0
2	0 1	0	NA4	NA3	NA2	NA1	(INTA)
3	0 1	1	NA4	NA3	NA2	NA1	CF
4	1 0	0	NA4	NA3	NA2	NA1	CO
5	1 0	1	NA4	NA3	NA2	NA1	ZO
6	1 1	0	NA4	NA3	NA2	NA1	NO
7	1 1	1	NA4	NA3	NA2	NA1	(INTB)

CF = Carry Flag vom Ausgang des Flag-Registers
CO = Carry Out direkt von der ALU
ZO = Zero Out direkt von der ALU
NO = Negative Out direkt von der ALU

Aufbau eines Mikroprogramm-Wortes:

Gruppe:	Steuerung				Datenpfad							
	microprogram control		bus control		register control				ALU control			flag register control
	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

Aufbau der Programmtabelle:

	Adr	Befehl	Steuerung		Bus	Register			ALU		Flags	Adresse	Steuerung		Bus		Register				ALU			Flags	
			adr. control	next adr.		func	adr A	adr B	write	in A			in B	func.	load	MAC	NA	BUS WR	BUS EN	MRG AA	MRG AB	MRG WS	MRG WE		MALU IA
													24...23	22...18	17	16	15...13	12...9	8	7	6	5	4...1	0	
1	0	LD R3, FC	-	1	-	3	FC	A	-	C	B	-	00000	00	00001	0	0	011	1100	0	1	0	1	1100	0
2	1	ADD R1, (R3) / JCO 3	CO	2	R	3	1	B	M	R	ADD	X	00001	10	00010	0	1	011	0001	1	1	1	0	0100	1
3																									

In der Spalte "Adr." steht die Adresse im Mikroprogramm-Speicher, an der der jeweilige Befehl abgelegt werden soll. Unter "Befehl" steht eine (hoffentlich einigermaßen...) lesbare Abkürzung (mnemonic) des Befehls. Danach bis zum senkrechten Doppelstrich wird eine stark abgekürzte textuelle Beschreibung der einzelnen Funktionen eingetragen. Rechts vom Doppelstrich trägt man den entsprechenden Binärcode ein; dieser kann dann in genau der eingetragenen Reihenfolge (von links nach rechts) mit den Tastern eingegeben werden (man muss deshalb in den einzelnen Feldern auch die führenden Nullen eintragen!).

Bedeutung der einzelnen Spalten der Programmtabelle

next adr.: Adresse des im Programmablauf folgenden Befehls (ggf. wird diese noch modifiziert). Bitte beachten: bei einer bedingten Verzweigung wird der nächste Befehl immer von einer geraden Adresse gelesen, wenn das als Bedingung ausgewählte Signal 0 ist, andernfalls von der folgenden ungeraden Adresse. Trotzdem kann es notwendig sein, als Binär-code für die nächste Adresse (NA) eine ungerade Zahl anzugeben, da Bit 0 (NA0) Teil der Auswahl-Adresse des Multiplexers ist.

Bus func: read (R) vom Bus, write (W) zum Bus oder Memory-Bus nicht ansprechen (-).

Register adr A: Wert am Adresseingang A des Register-Blocks.

Register adr B: Wert am Adresseingang B des Register-Blocks oder Konstante.

Register write: Adresseingang, der zum Schreiben in ein Register benutzt werden soll (A, B oder - = nicht schreiben).

ALU in A: was soll an Eingang A der ALU angelegt werden: R = Register DOA, M = Memory MEMDI.

ALU in B: entsprechend Eingang B: R = Register DOB, C = Konstante.

ALU funct.: Funktion der ALU.

Flags load: sollen die Flag-Ausgänge der ALU in das Flag-Register übernommen werden? (X = ja, - = nein)

Beispiel-Einträge in der Programmtabelle

Zeile 1: Lade (LD = load) Register R3 mit der Konstanten FC (hexadezimal).

Steuerung: keine Modifikation der nächsten Adresse, nächste Befehlsadresse soll 1 sein.

Bus: wird nicht angesprochen.

Register: Die Konstante muss an MRGAB3...0 angelegt werden, sie steht deshalb in der Spalte "adr. B" (zu beachten ist, daß die höherwertigen 5 Bits miteinander verbunden sind; daher ergibt der 4-Bit-Binärkode 1100 an MRGAB3...0 die Konstante 1111 1100 = FC hexadezimal). Damit muss das zu beschreibende Register über den Adresseingang A ausgewählt werden (write = A), an Adresseingang A wird daher 3 angelegt.

ALU: Eingang A wird nicht benutzt, der Multiplexer vor Eingang B muss auf die Konstante geschaltet werden ("C"), Eingang B wird auf den Ausgang durchgeschaltet (F=B).

Flags: der Zustand der Flag-Ausgänge der ALU soll nicht gespeichert werden (-), also MCHFLG = 0.

Zeile 2: addiere den Inhalt der Memory-Adresse, auf die R3 zeigt, zu R1 / weiter an Adr. 3, wenn CO = 1 (Carry Out von ALU), sonst an Adr. 2 (JCO = Jump if Carry Out set).

Steuerung: address control mit CO, nächste unmodifizierte Adresse soll 2 sein. Aus Tabelle 4, Zeile 4 ergibt sich, daß MAC1...0 = 10 und NA0 = 0 sein muss, also NA = 00010.

Bus: R = es soll vom Bus gelesen werden.

Register: an Ausgang DOA muss die Memory-Adresse anliegen, also Adresseingang A = 3 (für Register R3). Register R1 muss deshalb über den Adresseingang B ausgewählt werden (adr. B = 1), und damit liegt auch die Register-Adresse zum Schreiben an Adresseingang B an (write = B).

ALU: Input A kommt vom Memory (M), Input B vom Register-Block (R), Funktion ist ADD = addieren.

Flags: der Zustand der Flag-Ausgänge der ALU soll gespeichert werden (x), also MCHFLG = 1.

© 2000 – 2017 Universität Tübingen, Universität Leipzig

Die Neufassung des Hardwarepraktikums wurde ausgehend von seinem Vorgänger im Frühjahr 2000 von Werner Dreher an der Universität Tübingen ausgearbeitet und später an der Universität Leipzig von Jörn Hoffmann, Klemens Schölhorn, Hannes Thalheim und Max Braungardt weiterentwickelt.

Kopieren, Verbreiten und Modifizieren der vorliegenden Anleitung ist unter Beachtung des entsprechenden Copyright/Copyleft nur für die Lehre erlaubt. Jegliche kommerzielle Nutzung ist untersagt.

Für Verbesserungsvorschläge oder Hinweisen zu Fehlern sind wir immer dankbar!