Volker Dörsing (Autor)
Adrian Knoth (Autor)
Wolfgang Koch (Autor)
David Neuhäuser (Autor)
Andreas Reinsch (Autor)
Ralf Seidler (Autor)
Daniel Walther (Autor)
Martin Bücker (Herausgeber)

Experimente der Technischen Informatik

Praxisorientierte Trainingseinheiten

FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Kapitel 1

Grundlagen P: DLXJ-RISC-Prozessor

Andreas Reinsch

Inhaltsverzeichnis dieses Kapitels

1.1. Jenaer Version der DLX-Architektur

Die DLX[1]-Prozessorarchitektur wurde von Hennessy und Patterson [3, 4, 5] spezifiziert. Diese beiden kalifornischen Computer-Pioniere erhielten im Jahre 2017 den *ACM A. M. Turing Award*, den man allgemein als Nobel-Preis für Informatik anerkennt. In der Begründung zur Verleihung dieser hohe Auszeichnung heißt es:

> For pioneering a systematic, quantitative approach to the design and evaluation of computer architectures with enduring impact on the microprocessor industry.

Damit werden insbesondere deren persönlichen Verdienste um die Entwicklung von *Reduced Instruction Set Computer* (RISC) honoriert, wie die Zusammenfassung der Rede anlässlich der Preisverleihung beschreibt [6]. Aufgrund seiner Einfachheit ist der DLX-Prozessor gut geeignet, um die Funktionsweise und die wesentlichen Elemente von RISC-Prozessoren leicht verständlich darzustellen, mit VHDL zu modellieren und in Hardware umzusetzen. Zu Lehrzwecken wurde die DLX-Architektur mehrfach mit VHDL synthesefähig spezifiziert [1, 2], jedoch nicht in Hardware realisiert. Die Beschreibung nach [2] verzichtet auf die Implementierung von Gleitkommaarithmetik und vorzeichenlosen arithmetischen Befehlen, sieht aber gegenüber der Spezifikation von Hennessy und Patterson eine Ausnahme- und Unterbrechungsbehandlung vor.

Die *Association for Computing Machinery* verleiht seit 1966 jährlich den nach dem britischen Mathematiker Alan Mathison Turing (1912–1954) benannten Preis, der an dessen wegweisende Arbeiten in unterschiedlichen Bereichen der Informatik erinnert.

Die „DLXJ"-Prozessor-Architektur ist die „Jenaer" Version der DLX-Architektur, die den Buchstaben „J" am Ende der Abkürzung hinzufügt. Sie wurde mit dem Ziel entwickelt, einerseits die wesentlichen Merkmale der DLX-Architektur nach Hennessy und Patterson zu bewahren, andererseits durch konsequente Beschränkung des Befehlssatzes zu einem sehr einfachen und damit leicht verständlichen Prozessor für Lehrzwecke zu gelangen. Alle RISC-typischen Architekturmerkmale wurden beibehalten. Der Befehlssatz wurde auf nur sieben Befehle reduziert. Dabei wurde sichergestellt, dass die drei Befehlsklassen

- Datentransport,
- arithmetisch-logische Operationen und
- Steuerung

vorhanden sind. Gleitkomma-Operationen sowie eine Ausnahme- und Unterbrechungsbehandlung wurden nicht implementiert. Der Befehlssatz enthält Befehle aller drei Befehlsformate der DLX-Architektur.

[1] Die Bezeichnung „DLX" (ausgesprochen „Deluxe") steht für die römische Ziffernfolge DLX und ergibt sich nach [4] als Mittelwert von Nummern aus den Typenbezeichnungen der folgenden 13 experimentellen und kommerziellen Maschinen, die in ihrer Philosophie der DLX sehr ähnlich sind: AMD 29K, DECstation 3100, HP 850, IBM 801, Intel i860, MIPS M/120A, MIPS M/1000, Motorola 88K, RISC I, SGI 4D/60, SPARCstation-1, Sun-4/110, Sun-4/260. Der Mittelwert ist gegeben durch $7280/13 = 560 =$ DLX.

Der Entwurf wurde auf der algorithmischen Ebene und auf der Register-Transfer-Ebene in der Hardwarebeschreibungssprache VHDL durchgeführt. Einige Baugruppen mussten den Anforderungen der Schaltkreis-Architektur speziell angepasst werden. Die Spezifikation der Register-Transfer-Ebene ist synthesefähig und gestattet die Implementierung in einen rekonfigurierbaren integrierten Schaltkreis, der als *Field Programmable Gate Array* (FPGA) bezeichnet wird. Zusätzlich zum DLXJ-Prozessorkern wurden auf diesem Schaltkreis, der Teil eines einfachen Experimentalsystems ist, auch ein Arbeitsspeicher, ein Displayinterface sowie ein Interface zu einem Mikrocontroller realisiert. Das Experimentalsystem enthält weiterhin eine Displayeinheit und einen im System programmierbaren Mikrocontroller. Der Mikrocontroller dient als Bindeglied zwischen dem FPGA und einem über eine serielle Schnittstelle angeschlossenen Hostrechner.
Die Programmentwicklung wird durch einen Assembler unterstützt. Die Arbeitsweise des Prozessors und der Ablauf einfacher kleiner Programme kann entweder am Digitalsimulator oder direkt an der Hardware mithilfe eines Debuggers untersucht werden. Die Spezifikation in VHDL ermöglicht auf einfache Weise die Erweiterung der Funktionalität des Prozessors wie etwa die Implementierung weiterer Befehle. Zudem erlaubt sie die Modifikation der ebenfalls auf dem FPGA vorhandenen Prozessorperipherie wie beispielsweise die Erweiterung der Debug-Möglichkeiten.

1.2. Programmiermodell des DLXJ-Prozessors

Die Architektur des DLXJ-Prozessors wurde aus dem von Gumm [2] entwickelten Modell abgeleitet. Der Befehlsumfang wurde stark reduziert und wird wie folgt charakterisiert:

- Die Architektur enthält 32 Universalregister, *General-Purpose Register* (GPR), in denen jeweils 32 Bit gespeichert werden. Der Wert von Register $R0$ ist immer Null.
- Der Speicher ist wortadressiert mit 32-Bit-Adressen und verfügt über einen 4 GByte Adressraum. Es wird im Modus „Big Endian“ gearbeitet. Demnach ist das niedrigstwertige Datenbit in einem DLXJ-Wort der Stelle *Most Significant Bit* (MSB) zugeordnet und das höchstwertige Bit der Stelle *Least Significant Bit* (LSB). Alle Speicherzugriffe müssen ausgerichtet (*aligned*), also bei Wortzugriffen durch 4 teilbar sein.
- Alle Befehle sind 32 Bit lang.

Der Prozessor hat eine einfache Lade/Speicher-Architektur. In solchen Architekturen erfolgen alle arithmetischen und logischen Operationen zwischen den GP-Registern, Speicherzugriffe werden ausschließlich über GP-Register realisiert.

1.2.1. Befehlsklassen

Die Tabelle 1.1 zeigt die Zuordnung der vorhandenen sieben Befehle zu drei Befehlsklassen, den zugehörigen Assemblerbefehl, die Wirkung des Befehls, das Befehlsformat und den Befehlscode.

Tabelle 1.1.: Befehle und Befehlsklassen des DLXJ-Prozessors.

Befehl	Assemblerbefehl		Bedeutung	Typ	Opcode rr_func
			Datentransport		
load word	$LW.I$	$Rd, I(Rs)$	$Rd \leftarrow M(Rs + I)$ [a]	I	000100
store word	$SW.I$	$I(Rs), Rd$	$M(Rs + I) \leftarrow Rd$ [a]	I	001000
			Arithmetisch-logische Operationen		
sub-traction	SUB	$Rd, Rs1, Rs2$	$Rd \leftarrow Rs1 - Rs2$	R	000000 000110
set if lower than	SLT	$Rd, Rs1, Rs2$	$Rs1 < Rs2$: $Rd \leftarrow hex00000001$ else: $Rd \leftarrow hex00000000$	R	000000 010100
no operation	NOP			R	000000 000000
			Steuerung		
branch if equal zero	$BEQZ$	$Rs1, Label$	$Rs1 = 0$: $PC \leftarrow PC + 4 + Label$ [b] $Rs1 \neq 0$: $PC \leftarrow PC + 4$	I	010000
jump	J	$Label$	$PC \leftarrow PC + 4 + Label$ [c]	J	001100

[a] I: vorzeichenerweiterter 16-Bit-Offset,
wortausgerichtete Adresse: letzte zwei Bit = 00
[b] Label: vorzeichenerweiterter 16-Bit-Offset
[c] Label: vorzeichenerweiterter 26-Bit-Offset

Datentransport-Operationen dienen dem Datentransport zwischen den GP-Registern und dem Speicher. Jedes der GPR kann geladen oder gespeichert werden. Das Laden von $R0$ hat keine Wirkung. DLX-Speicherzugriffe können wort-, halbwort- oder byteweise erfolgen. Die DLXJ-Implementierung ist auf zwei wortweise Zugriffe beschränkt.

Arithmetisch-logische Operationen sowie Verschiebe- und Testoperationen bilden eine weitere Befehlsklasse. Der DLX-Befehlssatz enthält neben Operationen zu den Grundrechenarten, die logischen Operationen, logische und arithmetische Schiebeoperationen sowie Vergleichsoperationen.
In den DLXJ-Befehlssatz wurden ein Subtraktionsbefehl, eine Vergleichsoperation und die Nulloperation (*no operation*) aufgenommen.

Die Steuerung des Programmablaufs wird durch Sprung- und Verzweigeoperationen realisiert. Sprünge können beim DLX-Prozessor als einfacher Sprung oder bei Prozeduraufrufen als Sprung mit Rückkehrverbindung ausgeführt werden. Alle Verzweigungen sind bedingt. Sprünge sind an keine Bedingung geknüpft.
Beim DLXJ-Prozessor wird der Programmablauf durch einen Sprungbefehl und einen Verzweigebefehl gesteuert.

Gleitkomma-Operationen beinhaltet der Befehlssatz des DLX-Prozessors für die Grundrechenarten und für Vergleiche sowohl mit einfacher (32 Bit) als auch mit doppelter (64 Bit) Genauigkeit.
Im Befehlssatz des DLXJ-Prozessors sind Gleitkomma-Operationen nicht vorgesehen.

1.2.2. Befehlsformate

Das DLX-Befehlslayout wurde unverändert übernommen. Abbildung 1.1 zeigt die drei verschiedenen Befehlsformate und die Zuordnung der Befehle. Der primäre Operationscode (Opcode) ist bei allen drei Formaten einheitlich durch die Bits $0 \ldots 5$ festgelegt. Die Verwendung der verbleibenden 26 Bit ist von dem Befehlsformat abhängig und wird in den folgenden drei Abschnitten beschrieben.

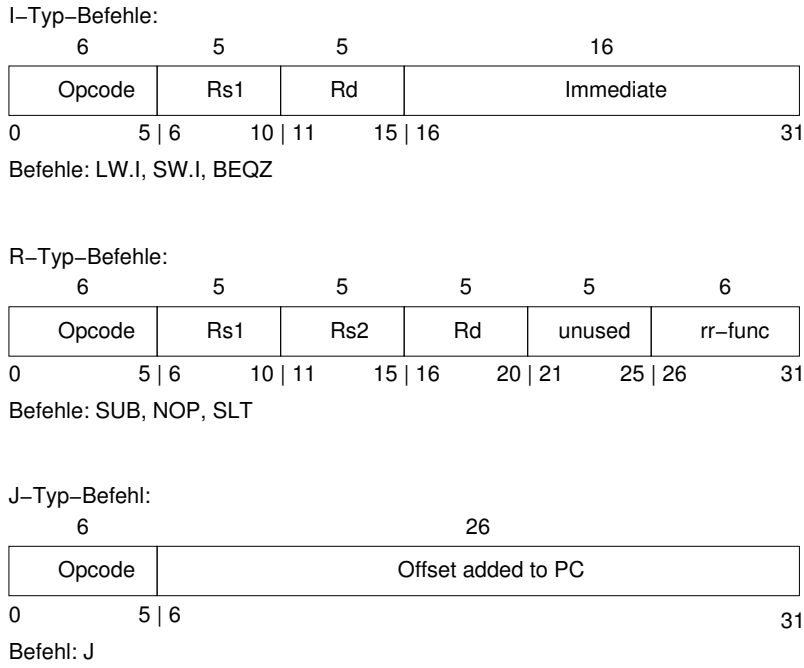

Abbildung 1.1.: Drei Befehlsformate des DLXJ-Prozessors.

Das R-Typ Format gestattet die Adressierung von zwei Quellregistern *Rs1, Rs2* (*Source Register*) und einem Zielregister *Rd* (*Destination Register*) mit jeweils 5 Bit. Die auszuführende Register-Register-ALU Operation (RRA) wird durch das Bitfeld rr_func mit 6 Bit bestimmt:

$$Rd \leftarrow Rs1 \text{ rr\_func } Rs2.$$

(Beim DLX stehen für das Bitfeld rr_func dagegen 11 Bit zur Verfügung.)
Die R-Typ-Befehle des DLXJ-Prozessor sind die Subtraktion, eine Vergleichsoperation und die Nulloperation. Bei der Subtraktion wird der Inhalt des Quellregisters $Rs2$ vorzeichenrichtig vom Quellregister $Rs1$ subtrahiert und das Ergebnis im Zielregister Rd abgelegt:

$$Rd \leftarrow Rs1 - Rs2.$$

Die Vergleichsoperation testet die Bedingung $Rs1 < Rs2$. Ist sie erfüllt, wird das Zielregister auf 1, anderenfalls auf 0 gesetzt.

Das I-Typ Format stellt je 5 Bit für die Adressierung eines Quellregisters $Rs1$ und eines Zielregisters Rd zur Verfügung. Die Verwendung der verbleibenden 16-Bit-Immediate ist befehlsabhängig.
Die Datentransport-Operationen und der Verzweigebefehl des DLXJ-Prozessors sind I-Typ-Befehle. Die Speicheradresse der Transportoperationen wird bestimmt, indem der 16-Bit-Immediate-Wert vorzeichenerweitert zum Quellregister addiert wird. Das Zielregister enthält das geladene oder das zu speichernde Datenwort. Der Verzweigebefehl testet das Quellregister und verzweigt, falls es Null ist. Die Verzweigezieladresse wird durch vorzeichenerweiterte Addition des Immediate-Werts zum Befehlszähler, *Program Counter* (PC), berechnet.

Das J-Typ Format wird ausschließlich von Sprungbefehlen genutzt. Die Zieladresse bei Ausführung des Sprungbefehls ergibt sich aus einem zum Befehlszähler PC addierten 26-Bit-vorzeichenerweiterten Offset (*Offset added to PC*) zu der dem Sprungbefehl folgenden Adresse.
Der DLXJ-Prozessor verfügt über einen Sprungbefehl. Prozeduraufrufe mit Rückkehrverbindung sind damit nicht möglich.

1.3. Struktur des DLXJ-Prozessors

Der DLXJ-Prozessor wurde als Einchiprechner realisiert. Auf einem einzigen FPGA-Schaltkreis sind zusätzlich zum Prozessorkern auch der Arbeitsspeicher und zwei Interfaces integriert. Das Blockschaltbild aus Abbildung 1.2 zeigt die vier Baugruppen. Damit enthält das FPGA alle wesentlichen Funktionseinheiten eines Rechners. Der FPGA-Schaltkreis ist Teil eines Experimentalsystems, das für die Implementierung des Prozessors außerdem einen *Personal Computer* als Hostrechner, einen Mikrocontroller, ein LCD-Display mit 2×16 Zeichen und einen Quarz-Taktgenerator nutzt; siehe Abbildung 1.3.
Der Mikrocontroller ist einerseits über eine USB-Verbindung mit dem Hostrechner und andererseits über mehrere 8-Bit-Parallelports mit dem FPGA verbunden. Auf diesem Weg werden

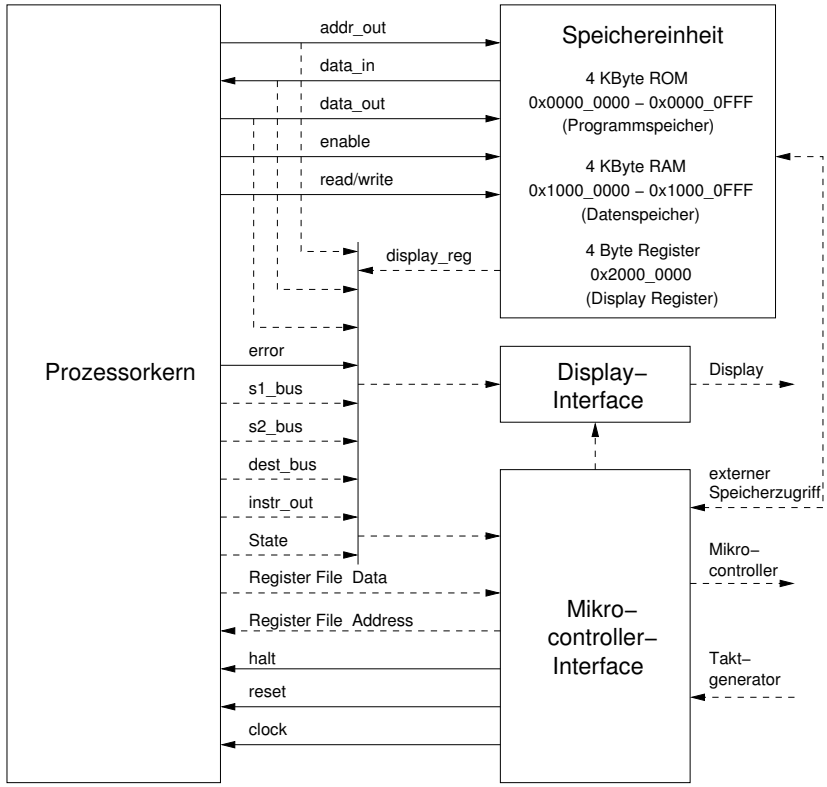

Abbildung 1.2.: Grundlegende Struktur des DLXJ-Prozessors.

- Programme zum DLXJ-Prozessor übertragen,
- Daten zwischen dem DLXJ-Prozessor und dem Hostrechner ausgetauscht und
- Steuersignale an den DLXJ-Prozessor gesendet.

Mit einer zusätzlichen USB-Verbindung zwischen dem FPGA-Board und dem Host-PC wird das FPGA konfiguriert. Die Konfigurationsdaten des FPGA bleiben nur solange erhalten, solange das Experimentalsystem mit Spannung versorgt wird. Nach jeder Betriebsunterbrechung muss das FPGA neu konfiguriert werden, um so die Funktionalität des DLXJ-Prozessors wieder herzustellen.
Weiterhin besteht eine Netzwerkverbindung zwischen dem Host-PC und einer Workstation. Alle für den Hardwareentwurf benötigten Programme werden von der Workstation zur Verfügung gestellt.

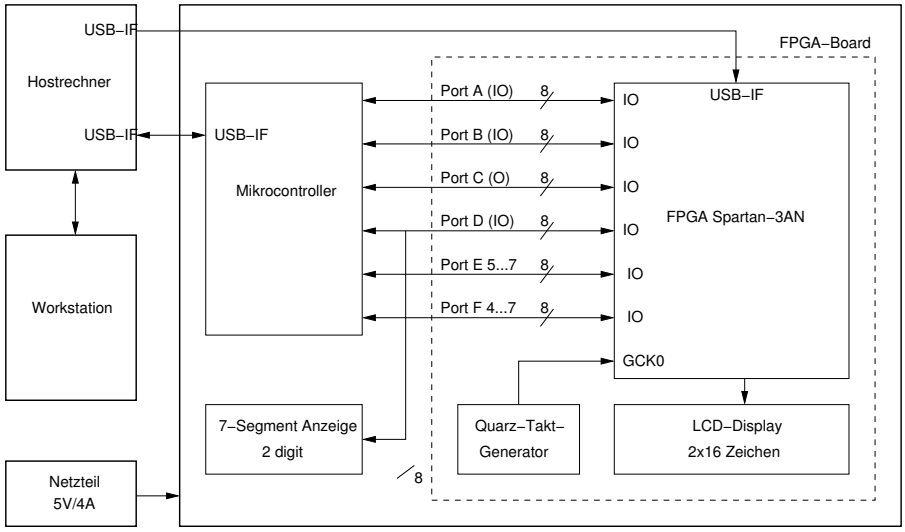

Abbildung 1.3.: Verwendetes Experimentalsystem und seine Anbindung.

1.3.1. Prozessorkern

Der Prozessorkern ist über einen Adressbus ($addr\_out$, 32 Bit), zwei Datenbusse ($data\_out$, $data\_in$, je 32 Bit) und zwei Steuersignale ($enable$, $read/write$) mit der Speichereinheit verbunden.
Das Signal $reset$ versetzt den Prozessor in einen definierten Anfangszustand, d.h., dass der Programmzähler, das Speicheradressregister und das Speicherdatenregister zurückgesetzt werden. Folglich wird die Programmausführung an der Adresse 0 fortgesetzt, vorausgesetzt das Signal $halt$ wurde zurückgesetzt.
Das Signal $halt$ löst eine Unterbrechung der Programmabarbeitung aus. Der gerade ausgeführte Befehl wird beendet und die Programmabarbeitung wird mit dem folgenden Befehl erst dann fortgesetzt, wenn das $halt$-Signal nicht mehr aktiv ist.
Das Signal $error$ zeigt an, dass der Prozessor einen nicht decodierbaren Befehl erhalten hat und die Programmabarbeitung abgebrochen wurde oder dass Implementierungsfehler im Schaltwerk, welches in Abschnitt 1.4.4 beschrieben wird, vorliegen.
Der Zweiphasentakt $clock$ besteht aus zwei periodischen Signalen, deren Impulsdauer und Phasenlage so zueinander angeordnet sind, dass niemals beide Signale gleichzeitig 1-Pegel annehmen. Gegenüber einem Einphasentakt kann der Prozessor so einen insgesamt höheren Datendurchsatz erreichen.
Im Abschnitt 1.4 wird die Funktionsweise des Prozessorkerns ausführlich beschrieben.

Anmerkung: Alle gestrichelt gezeichneten Signale der Abbildungen 1.3, 1.4 und 1.6 sind für die Funktion des Prozessors nicht erforderlich. Sie werden benötigt, um prozessorinterne Abläufe experimentellen Untersuchungen zugänglich zu machen.

1.3.2. Speichereinheit

Die Speichereinheit erhält vom Prozessorkern ihre Adressinformationen ($addr\_out$). Die Daten gelangen über je einen Bus vom Prozessor zum Speicher ($data\_out$) und vom Speicher zum Prozessor ($data\_in$), was in Abschnitt 1.4.1 genauer beschrieben wird. Die Speicherfreigabe geschieht durch das Signal $enable$. Das Signal $read/write$ entscheidet darüber, ob Daten vom Speicher gelesen oder in den Speicher geschrieben werden; vgl. Abschnitt 1.4.2.
Die Speichereinheit beinhaltet einen Adressdecoder und drei verschiedene Speicherblöcke: einen Datenspeicher, einen Programmspeicher und ein Register. Der Adressdecoder ordnet den drei Speichereinheiten jeweils einen Adressbereich zu. Die Adressbereiche können der Abbildung 1.2 entnommen werden.
Der Programmspeicher und der Datenspeicher mit je 4 KByte Speicherkapazität (d.h. je 1K Worte) sind Zweitor-Speicher mit wahlfreiem Zugriff (*Dual Port RAM DP-RAM*). Ein Port jedes Speichers ist mit dem Prozessorkern verbunden. Der Programmspeicher wird prozessorseitig als *Read Only Memory* (ROM) betrieben, kann also vom Prozessorkern nur gelesen werden. Damit wird verhindert, dass die Programmdaten durch ein fehlerhaftes Programm überschrieben werden können. Der Datenspeicher wird prozessorseitig als *Random Access Memory* (RAM) betrieben. Hier bestimmt der Zustand des Signals $read/write$, ob Daten gelesen oder geschrieben werden. Das zweite Port jedes *Dual Port RAM* ist mit dem Mikrocontroller-Interface verbunden. Das Interface hat Schreib- und Lesezugriff auf beide Speicherblöcke, so dass über diese Verbindung Programme und Daten mit dem Hostrechner ausgetauscht werden können.
Das Register *Display Register* dient der Ergebnisanzeige. Es ist eine einzelne Speicherzelle, die aus ebenfalls 32 Bit besteht, im Adressraum des Prozessors, auf die nur geschrieben werden kann. Der Wert dieses Registers kann über das Display-Interface auf dem LCD-Display angezeigt werden.

1.3.3. Display-Interface

Das Display-Interface wird für die direkte Anzeige von Daten und von Steuerinformationen des Prozessors verwendet. Mit einem Multiplexer (8×32 auf 1×32) können 8 verschiedene 32-Bit-Datenquellen ($addr\_out$, $data\_in$, $data\_out$, $display\_reg$, $s1\_bus$, $s2\_bus$, $dest\_bus$, $instr\_out$) auf das LCD-Display geschaltet werden; vgl. Abschnitt 1.4.1. Die Auswahl des Multiplexer-Eingangs, der zur Anzeige gelangen soll, wird über das Mikrokontroller-Interface vom Hostrechner aus festgelegt. Der momentane Zustand der Steuerung (*State*) und das Signal $error$ des Prozessorkerns werden ebenfalls auf dem LCD-Display angezeigt.

1.3.4. Mikrocontroller-Interface

Das Mikrocontroller-Interface verbindet den DLXJ-Prozessor mit dem Mikrocontroller des Experimentalsystems. Hierzu werden die Parallelports des Controllers genutzt. So erfolgt der Datenaustausch mit der Speichereinheit, die Steuerung des Prozessorkerns, die Erzeugung von Einzeltakt-Impulsen und die Übertragung von Debuginformationen (Registerinhalte, Speicherinhalte, alle prozessorinternen Buszustände und der aktuelle Zustand der Steuerung) über drei 8-Bit-Ports, die in Abbildung 1.3 als A, B und C gekennzeichnet sind.

Im Wesentlichen besteht das Mikrocontroller-Interface aus mehreren Multiplexern und Registern. Die Multiplexer schalten die jeweils zu übertragende Datenquelle oder Datensenke auf die Ports des Controllers. In den Registern werden die Steuerinformationen für die Multiplexer und den Prozessorkern gespeichert.

1.4. Interner Aufbau des Prozessorkerns

Der Prozessorkern besteht aus Datenpfad und Steuerung und ist in Abbildung 1.4 skizziert. Der Datenpfad enthält mehrere Register. Das Befehlsregister, *Instruction Register* (IR), ist

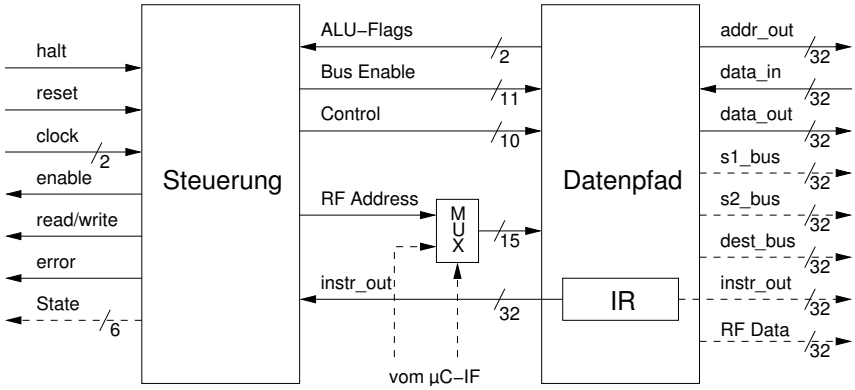

Abbildung 1.4.: Aufbau des DLXJ-Prozessorkerns.

außerdem mit der Steuerung verbunden. Es wird zu Beginn eines jeden Befehls mit dem aktuellen Befehlswort geladen. Weitere wesentliche Bestandteile des Datenpfads sind die *Arithmetic Logic Unit* (ALU) und ein Bussystem. Der Datenpfad führt alle Operationen an den Daten aus, liefert der Steuerung Statusinformationen (ALU-Flags) und das aktuelle Befehlswort.
Die Steuerung besteht neben dem Befehlsregister aus drei Decodern und einem Schaltwerk. Das Schaltwerk wird auch als binärer Automat oder als *Finite State Machine* (FSM) bezeichnet. Aus dem Befehlswort und den Statusinformationen werden Signale für die Ansteuerung des Datenpfads ($Bus\ Enable$, $Control$, $RF\ Address$), für Zugriffe auf die Speichereinheit ($enable$, $read/write$) und ein Signal für die Fehleranzeige ($error$) erzeugt. Die Steuerung wird mit einem Zweiphasentakt ($clock$) betrieben.

1.4.1. Datenpfad

Der Datenpfad ist in der Abbildung 1.5 dargestellt. Die Universalregister $GPR0, \ldots, GPR31$ werden zu einem Block zusammengefasst und als Registerdatei, *Register File* (RF), bezeichnet. Der Datenpfad hat drei interne Busse: die beiden Quellbusse S1, S2 ($s1\_bus$, $s2\_bus$) und den Zielbus Dest ($dest\_bus$). Die grundlegende Operation des Datenpfades ist das Lesen von Operanden aus dem RF über die Busse S1 und S2, deren Verarbeitung in der ALU