ReConf

19. September 2018

Eingereicht von:
Niklas Kühl
inf102861@fh-wedel.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Sollte erklärt worden sein

- Inhalt der Matrizen: Zahlen, Signed Float bzw. Fixed - Inhalt von y train: Zahlen, Klassen der Daten, Unsigned Integer

2 Benutzerhandbuch

2.1 Funktionsumfang

kann evtl weg

Auf dem FPGA ist ein Algorithmus für ein Neuronales Netz implementiert. Die Steuerung des Algorithmus erfolgt von einem PC über eine RS232-Schnittstelle.

2.2 Bedienungsanleitung

2.2.1 Kodierung der Matrizen

Die Elemente der Matrizen sind 8 Bit breit. Diese 8 Bit werden als vorzeichenbehafteter Festkommawert interpretiert. Es werden 4 Bit für das Vorzeichen und den Ganzzahlanteil benutzt. Der Gebrochene Anteil benutzt weitere 4 Bit. Der Wertebereich der Elemente beträgt somit 7.9375 bis -8.0000. Der kleinste von 0.0 verschiedene Wert ist ± 0.0625.

2.2.2 Kommunikation zwischen FPGA und PC

Die serielle Kommunikation zwischen FPGA und PC läuft mit einer Baudrate von 9600, 1 Stoppbit, kein Paritätsbit. Es wird folgendes Protokoll verwendet:

1. Reset des FPGA per SW17
2. FPGA führt Initalisierung durch
3. FPGA sendet Steuerungsbyte an PC: 0xF0
4. PC sendet Steuerungsbyte an FPGA: 0xE0
5. PC sendet 4096 zufällige Werte zur Initialisierung der Gewichte w1
6. PC sendet Steuerungsbyte an FPGA: 0xE1
7. PC sendet 640 zufällige Werte zur Initialisierung der Gewichte w2
8. PC sendet Byte an FPGA zur Auswahl des Modus:
   - Training: 0xE2 (weiter mit 9)
   - Test: 0xA1 (weiter mit 10)

9. Training
    a) PC sendet 4096 Werte zur Initialisierung der Trainingsdaten
    b) PC sendet Steuerungsbyte an FPGA: 0xE3
    c) PC sendet 64 Werte zur Initialisierung der Klassen y_train
    d) FPGA führt Algorithmus zum Training des Neuronalen Netzes aus
    e) FPGA sendet Steuerungsbyte an PC: 0xF1
    f) Zurück zu 8
10. Test
    a) PC sendet 4096 Werte zur Initialisierung der Testdaten
    b) FPGA führt Algorithmus zum Test der Daten aus
    c) FPGA überträgt die Bewertung der Testdaten (640 Bytes)
    d) Zurück zu 8

2.2.3 Bedienung des Altera DE2 Boards

Die LEDs LEDR1 und LEDR2 dienen zur Signalisierung von Fehlern:

- LEDR1: Leuchtet, wenn ein Protokollfehler aufgetreten ist
- LEDR2: Leuchtet, wenn ein Fehler bei der seriellen Kommunikation aufgetreten ist

Der Schalter SW17 dient zum zurücksetzten des Systems.

Der Schalter SW0 dient zum aktivieren des Debug-Modus, über welchen interne Daten des Systems ausgelesen werden können. Das Verlassen des Debug-Modus ist nur mit einem Reset möglich. Eine genaue Beschreibung der Funktionsweise erfolgt in Kapitel .

Referen

Über die Sieben-Segment-Anzeige der Altera DE2 Boards wird der interne Zustand des Systems angezeigt. Die Bedeutung der Anzeige ist in Kapitel beschrieben.

Referen

3 Planung

3.1 Problemanalyse

3.1.1 Analyse des Algorithmus

Für die Implementierung des Algorithmus auf dem FPGA wurden die Möglichkeiten zur Parallelisierung untersucht. Der Algorithmus besteht im Wesentlichen aus zwei Teilen:

1. Berechnung der scores
2. Berechnung der übrigen Matrizen

Die Berechnung der scores (Schritt 1) läuft größtenteils sequentiell ab, weswegen eine parallele Ausführung mehrerer Operationen hier kaum Gewinn bringt.

In Schritt 2 werden die Matrizen w1, w2, b1 und b2 angepasst. Da diese Berechnungen größtenteils unabhängig voneinander ablaufen, ist hier eine Parallelisierung sinnvoll. Die gleichzeitige Ausführung von 2 Befehlen ist hier gut möglich. Die parallele Ausführung von 3 oder mehr Befehlen nur ist aufgrund von Datenabhängigkeiten hingegen nur sinnvoll, wenn die gleiche Operation mehrmals parallel ausgeführt werden kann.

In Tabelle 3.1.1 sind die Schritte des Algorithmus zur Parallelen Ausführung von 2 Operationen dargestellt.

Schritt	Operation 0	Operation 1
0	d = mul(x_train, w1)	-
1	hl = d + b1	-
2	x_train_trans = x_train^T	hl_ReLu = max(0, hl)
3	d2 = mul(hl_ReLu, w2)	-
4	scores = d2 + b2	-
5	w2_trans = $w2^T$	scores = max(scores, 0)
6	-	scores: richtigen Wert um 1 verringern
7	-	scores: Division durch 64
8	dhidden = mul(scores, w2-trans)	-
9	scores2 = Orientierung von scores ändern	dhidden = max(dhidden, 0)
10	dw1 = mul(x-train-trans, dhidden)	w1_reg = w1 * reg
11	dw1 = dw1 + w1_reg	db1 = Spaltenweise Summe von dhidden
12	dw2 = mul(hl-relu, scores2)	dw1 = -step-size * dw1
13	w1 = w1 + dw1	w2_reg = w2 * reg
14	dw2 = dw2 + w2_reg	db = spaltenweise summe scores2
15	-	dw2 = -stepsize * dw2
16	w2 = w2 + dw2	db2 = -stepsize * db2
17	b2 = b2 + db2	db1 = -stepsize * db1
18	b1 = b1 + db1	-

Tabelle 3.1: Parallele Anordnung der Algorithmus-Schritte

Der Algorithmus operiert auf Matrizen, welche gespeichert werden müssen. Es müssen in jedem Fall 4 Speicherplätze (Register) für die statischen Matrizen w1, w2, b1 und b2 vorhanden sein. Zusätzliche werden allerdings Speicherplätze für temporäre Matrizen benötigt.

Durch eine Analyse wurde festgestellt, dass der Algorithmus bei der in Tabelle 3.1.1 dargestellten Anordnung der Schritte maximal 10 Matrix-Speicherplätze benötigt. Die Verteilung der 4 statischen Matrizen auf die Speicherplätze ist in Tabelle 3.1.1 dargestellt. Die Verteilung der temporären Matrizen auf die 6 übrigen Matrix-Register ist in Tabelle 3.1.1 dargestellt. In einer Zeile der Tabelle ist dabei der Zustand vor dem jeweiligen Algorithmus-Schritt dargestellt.

Matrix	Register
w1	Reg 0
b1	Reg 1
w2	Reg 2
b2	Reg 3

Tabelle 3.2: Zuordnung der statischen Matrizen zu den Matrixregistern

Schritt	Reg 4	Reg 5	Reg 6	Reg 7	Reg 8	Reg 9
0	x_train	-	-	-	-	-
1	x_train	-	-	d	-	-
2	x_train	-	-	hl	-	-
3	-	x_train^T	-	hl_ReLu	-	-
4	-	x_train^T	-	hl_ReLu	d2	-
5	-	x_train^T	-	hl_ReLu	scores	-
6	$w2^T$	x_train^T	-	hl_ReLu	scores	-
7	$w2^T$	x_train^T	-	hl_ReLu	scores	-
8	$w2^T$	x_train^T	-	hl_ReLu	scores	-
9	-	x_train^T	dhidden	hl_ReLu	scores	-
10	-	x_train^T	dhidden	hl_ReLu	-	scores2
11	dw1	-	dhidden	hl_ReLu	w1_reg	scores2
12	dw1	db1	-	hl_ReLu	-	scores2
13	dw1	db1	dw2	-	w2_reg	scores2
14	dw1	db1	dw2	-	-	scores2
15	dw1	db1	dw2	db2	-	-
16	dw1	db1	dw2	db2	-	-
17	dw1	db1	dw2	db2	-	-
18	dw1	db1	dw2	db2	-	-

Tabelle 3.3: Zuordnung der temporären Matrizen zu den Matrixregistern

3.1.2 Aufteilung der RAM-Blöcke

Da für die Matrizen sehr viel Speicher benötigt wird, müssen die Daten im RAM abgelegt werden. Für die Verwaltung des RAM gibt es zwei verschiedene Ansätze:

1. Benutzung eines großen RAM-Blocks
2. Für jede Matrix wird ein eigener, kleiner RAM-Block benutzt

Die Nutzung eines großen RAM-Blocks besitzt den Vorteil, dass ein größerer Datenbus genutzt werden kann, wodurch eine größere Datenmenge verarbeitet werden kann. Allerdings wird die Menge der Daten, welche parallel verarbeitet werden kann, auch durch andere Faktoren begrenzt (z.B. Anzahl der HW-Multiplizierer). Dadurch ist nur eine geringe Steigerung der verarbeiteten Datenmenge möglich.

Bei Nutzung eines großen RAM-Blocks besteht auch die Möglichkeit, flexibel Speicher zuzuweisen. Dadurch kann der Speicherverschnitt minimiert werden, welcher bei Nutzung von separaten RAM-Blöcken entsteht. Allerdings erfordert dies ein komplexes Speichermanagement, da der Zugriff auf jede Matrix unter Berücksichtigung der Start-Speicheradresse erfolgen muss.

Die Nutzung seperater RAM-Blöcke bietet den Vorteil, dass auf jeden Block parallel zugegriffen werden kann. Dadurch ist eine wesentlich höhere Parallelisierung möglich als bei Verwendung eines großen RAM-Blocks. Bei einem großen RAM-Block ist nur der gleichzeitige Zugriff auf 2 verschiedene Speicheradressen möglich, bei Verwendung von

10 seperaten RAM-Blöcken kann hingegen auf 20 Adressen gleichzeitig zugegriffen werden.

Aufgrund der einfacheren Verwendung und den wesentlich besseren Parallelisierungmöglichkeiten wird daher Variante 2 gewählt.

3.1.3 Struktur der Matrizen

Die Matrizen sollen im RAM abgelegt werden. Der verwendete FPGA besitzt 105 M4K Blöcke (also 420 kB Speicher). Bei 10 zu speicherndes Matrizen bleiben abgerundet auf eine 2er Potenz 32 kB Speicher pro Matrix. Dies erlaubt die Speicherung von 64x64 Matrizen, bei 8 Bit pro Matrixelement.

Bei der Verwendung von 8 M4K-Blöcken (32 kB) ist der Datenbus 256 bit breit. Dies erlaubt die gleichzeitige Verarbeitung von 32 Matrix-Elementen. Damit die Matrixoperationen möglichst schnell ausgeführt werden können, werden die Matrizen daher in Wörtern mit je 32 Elementen gespeichert. Diese Wörter können zeilen- oder spaltenweise gespeichert werden (Orientierung der Matrix).

3.1.4 Matrixoperationen

Für den Algorithmus müssen folgende Matrixoperationen implementiert werden:

- Matrixmultiplikation
- Matrixaddition
- Zeilenweise Addition von einem Vektor mit einer Matrix
- Matrix transponieren
- Elementweise Multiplikation mit Skalaren Wert
- Elementweise Division durch Konstante
- Elementweises Maximum mit Konstante
- Berechnung der spaltenweisen Summe einer Matrix
- In jeder Zeile einer Matrix von einem bestimmten Element einen konstanten Wert subtrahieren

Die Matrixmultiplikation und die skalare Multiplikation benutzten die HW-Multiplizierer des FPGA. Es sind jeweils 32 8-Bit-Multiplizierer notwendig, da in einem Takt 32 Elemente verarbeitet werden. Da der verwendete FPGA 70 9-Bit-Multiplizierer besitzt, sind genug Ressourcen vorhanden.

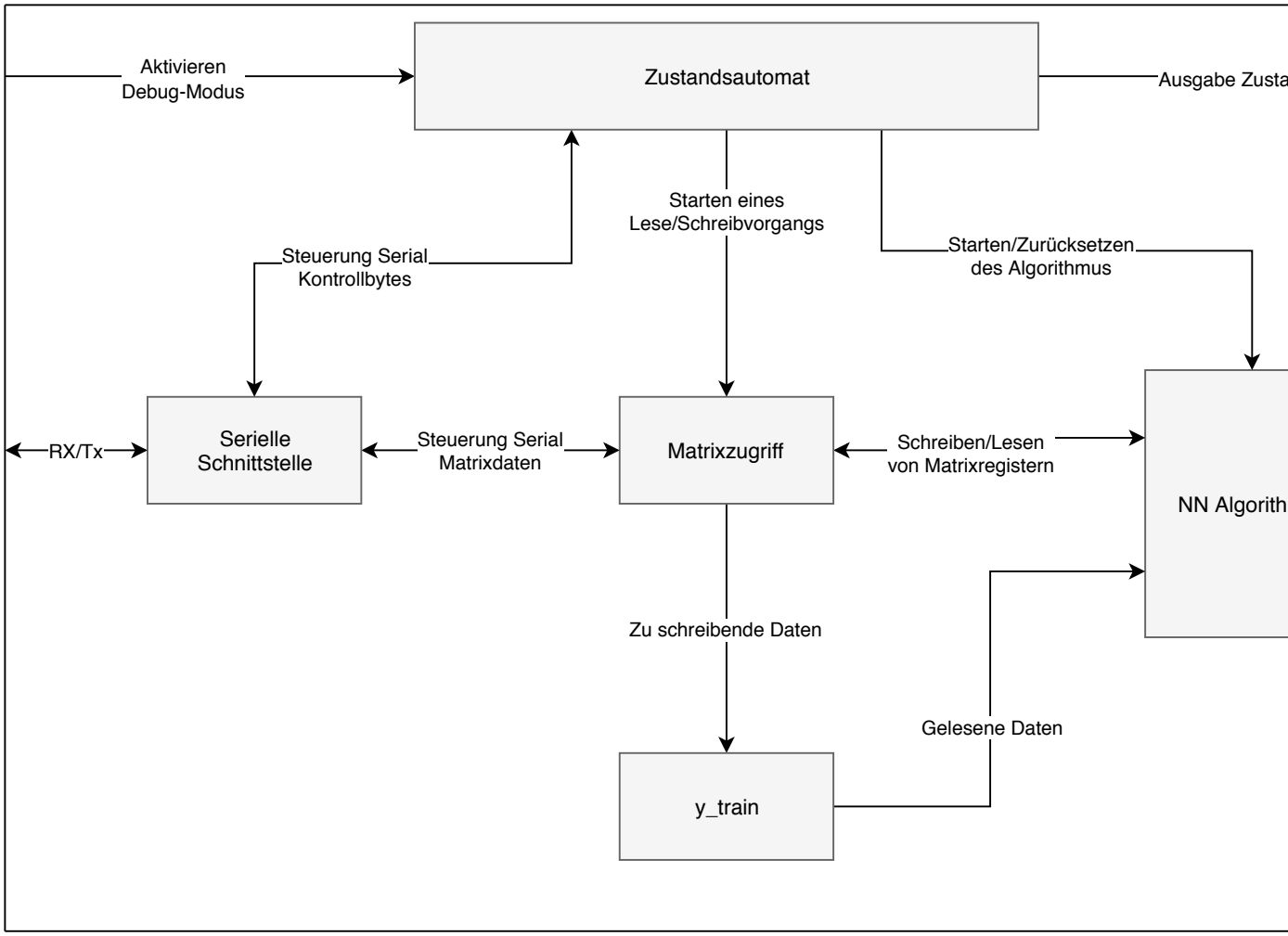


Abbildung 3.1: Struktur der Top-Level-Entity

3.2 Systemdesign

3.2.1 Top-Level-Entity

Die grundlegende Struktur des Gesamtsytems ist in Abbildung 3.1 dargestellt. Die Kommunikation zwischen PC und FPGA erfolgt über eine serielle Schnittstelle. Sowohl der Zustandsautomat als auch der Matrixzugriff benutzen diese Schnittstelle. Der Zustandsautomat legt dabei fest, wann der Matrixzugriff auf die serielle Schnittstelle zugreifen darf.

Der Matrixzugriff liest bzw. schreibt einzelne Bytes über die serielle Schnittstelle und liest bzw. schreibt Matrix-Wörter in die Matrixregister des NN Algorithmus. Außerdem schreibt er Daten in den RAM-Block y_train. Dieser wird nicht als Matrix gespeichert, da die Elemente einen anderen Datentyp besitzen.

Im Block „NN Algorithmus“ ist der Hauptteil des Systems realisiert. Hier werden die Matrizen gespeichert und die Operationen des Algorithmus ausgeführt.

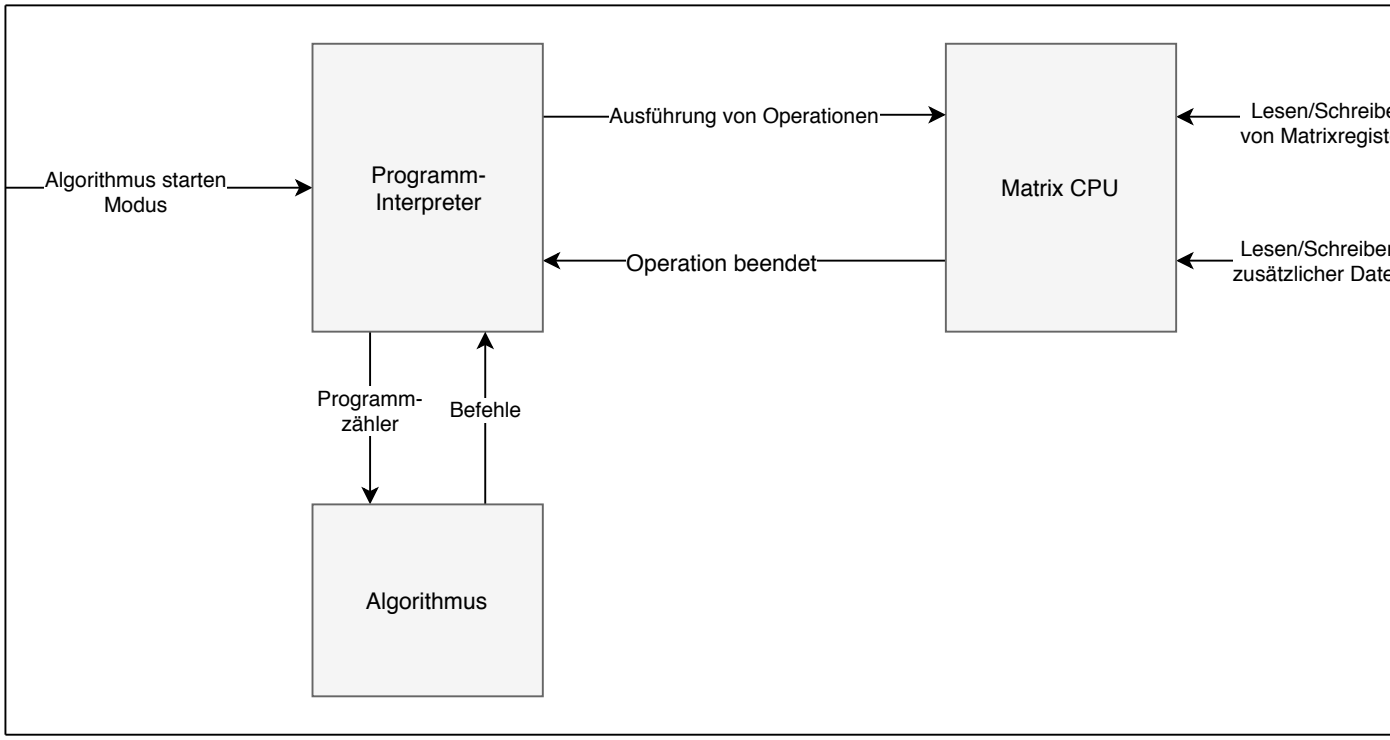


Abbildung 3.2: Struktur der Entity für den Algorithmus

3.2.2 Algorithmus

In Abbildung 3.2 ist die Struktur des Algorithmus dargestellt. DIe Matrix-CPU stellt die Matrixregister zur Verfügung und kann Operationen auf diesen Registern ausführen. Der Programminterpreter liest die Operationen des Algorithmus und setzt die entsprechenden Eingangsports der CPU.

3.2.3 Matrix-CPU

Die Matrix-CPU

3.2.4 Matrix-ALU

3.2.5 Matrixoperationen

3.2.6 Hilfsmodule

3.3 Schnittstellenbeschreibung

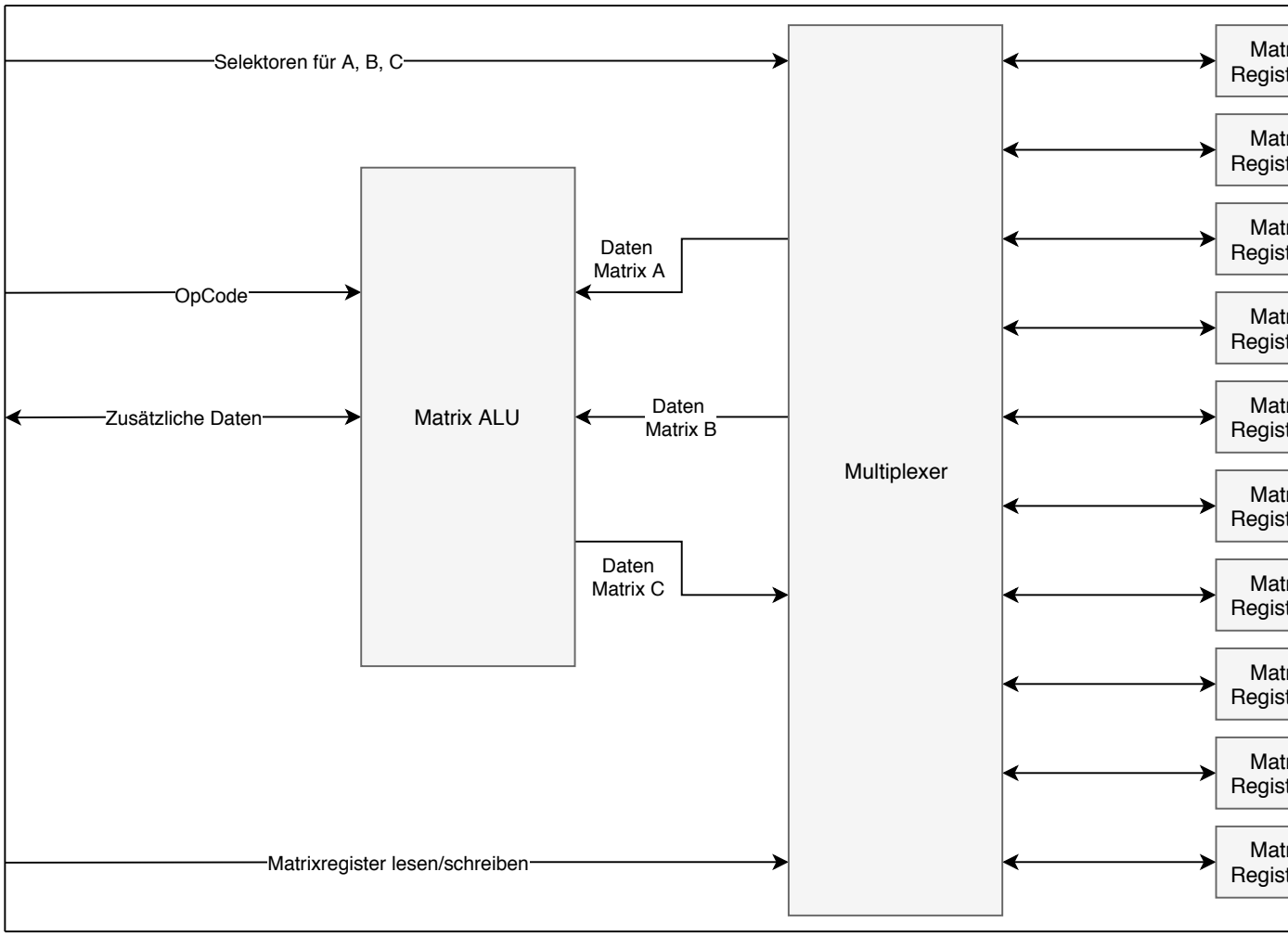


Abbildung 3.3: Struktur der Matrix-CPU

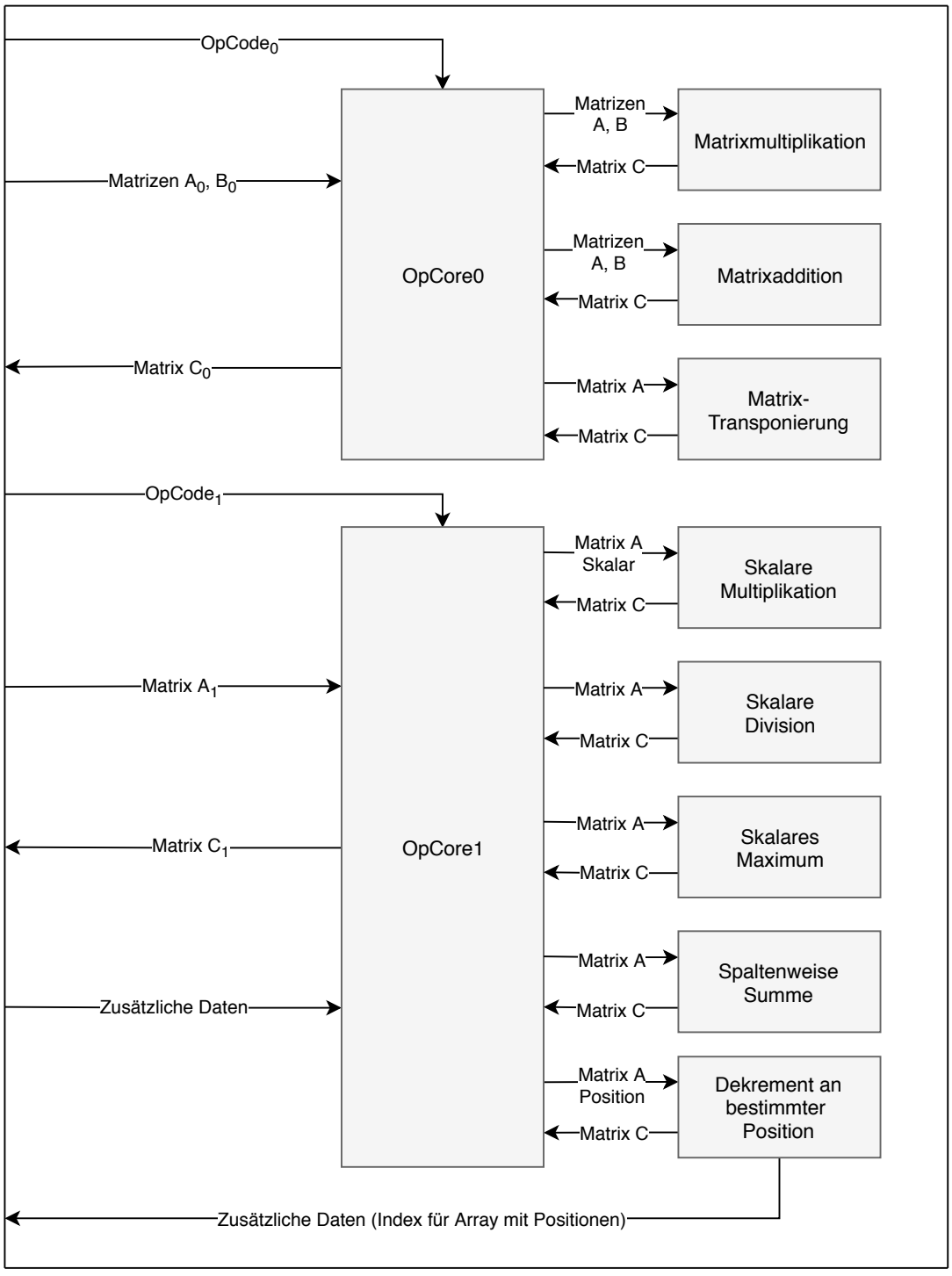


Abbildung 3.4: Struktur der Matrix-ALU

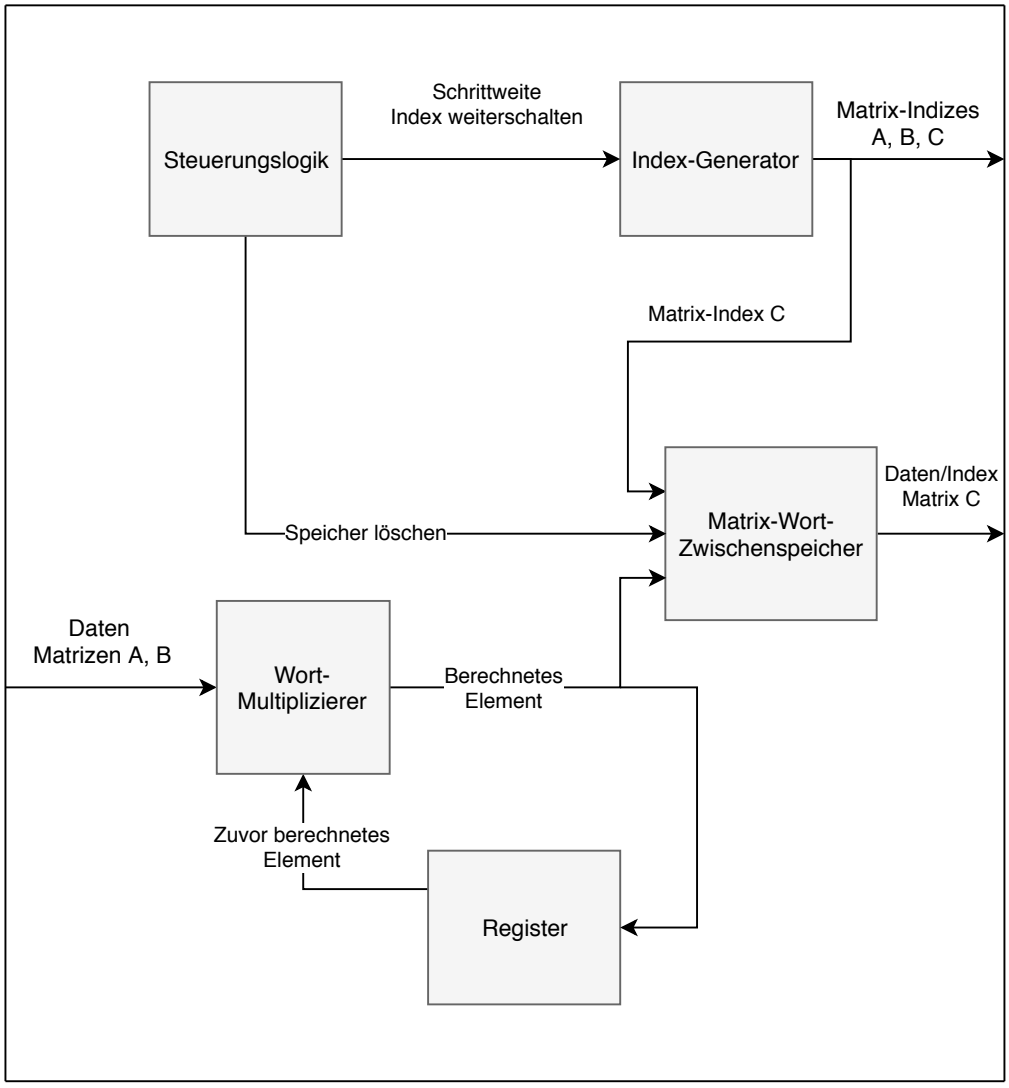


Abbildung 3.5: Struktur der Einheit zur Matrixmultiplikation

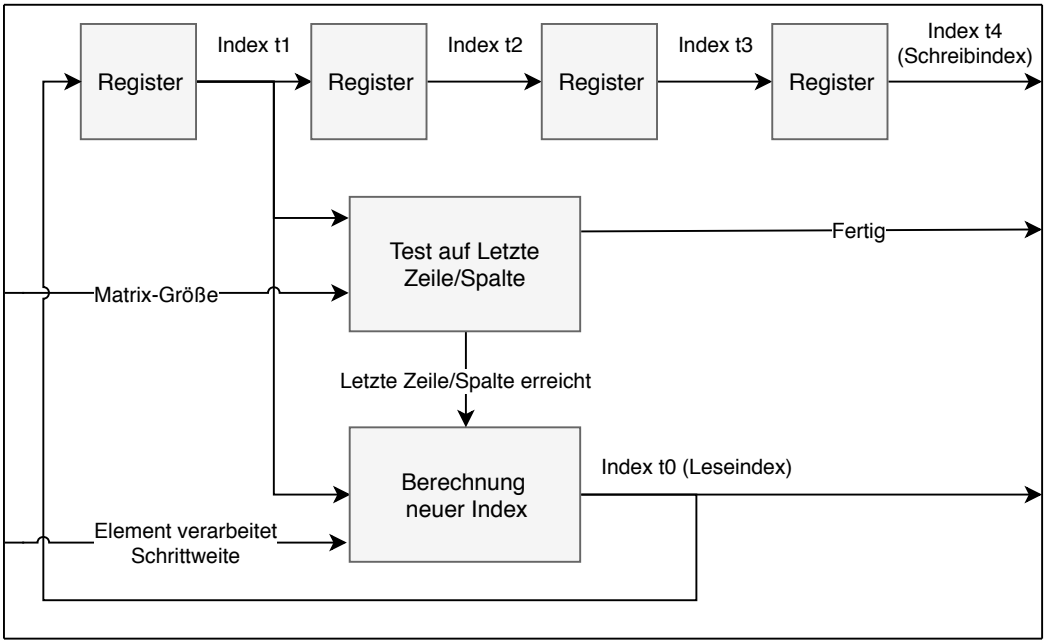


Abbildung 3.6: Struktur des Indexgenerators

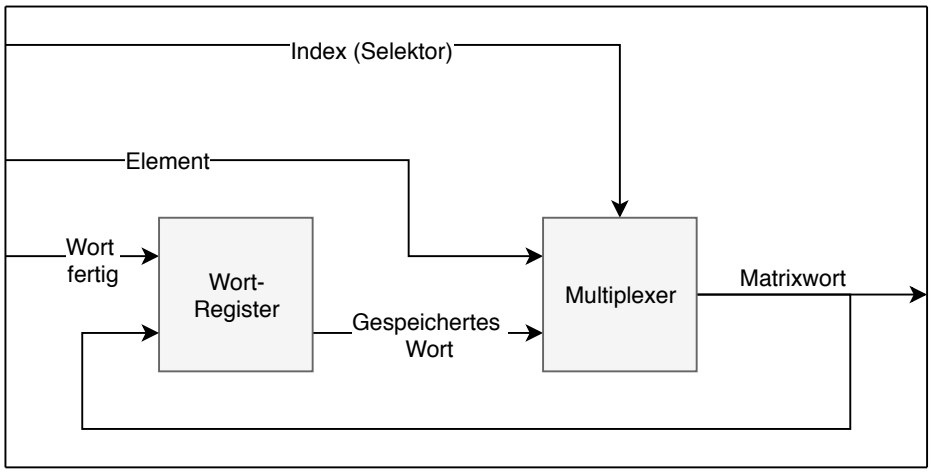


Abbildung 3.7: Struktur der einheilt zum Elementweisen Schreiben einer Matrix

4 Implementierung

4.1 Erklärung wesentlicher Design-Teile

```vhdl
proc_change_state : PROCESS(p_clk_i, p_rst_i)
BEGIN
  IF p_rst_i = '1' THEN
    s_cur_state <= st_init;
  ELSIF rising_edge(p_clk_i) THEN
    s_cur_state <= s_next_state;
  END IF;
END PROCESS proc_change_state;
```

Listing 4.1: VHDL Test Listing

4.2 Beschreibung der Zustandsautomaten

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie

vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

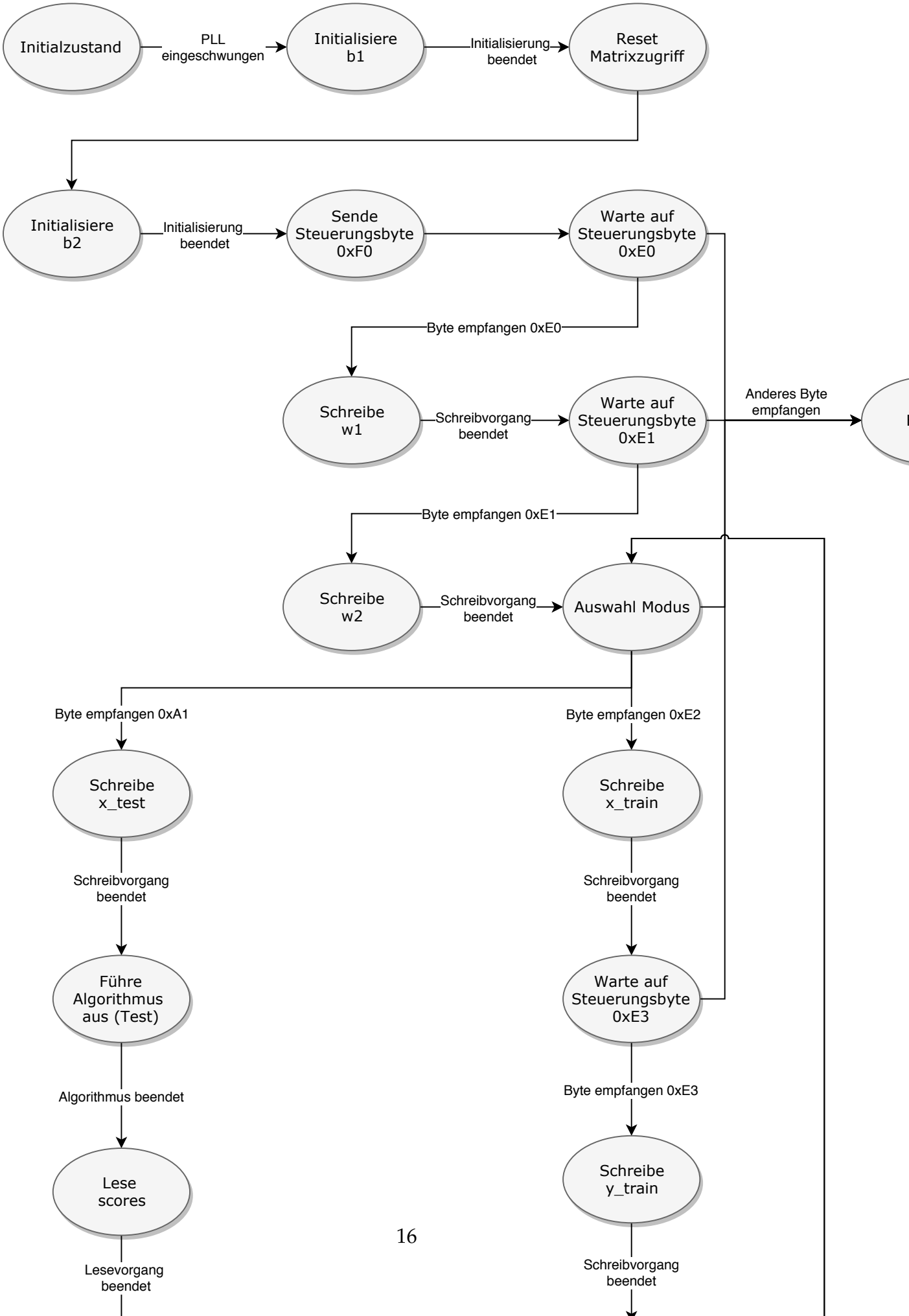
Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

1

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor


Initialzustand
PLL eingeschwungen
Initialisiere b1
Initialisierung beendet
Reset Matrixzugriff
Initialisiere b2
Initialisierung beendet
Sende Steuerungsbyte 0xF0
Warte auf Steuerungsbyte 0xE0
Byte empfangen 0xE0
Schreibe w1
Schreibvorgang beendet
Warte auf Steuerungsbyte 0xE1
Anderes Byte empfangen
Byte empfangen 0xE1
Schreibe w2
Schreibvorgang beendet
Auswahl Modus
Byte empfangen 0xA1
Byte empfangen 0xE2
Schreibe x_test
Schreibe x_train
Schreibvorgang beendet
Schreibvorgang beendet
Führe Algorithmus aus (Test)
Warte auf Steuerungsbyte 0xE3
Algorithmus beendet
Byte empfangen 0xE3
Lese scores
Schreibe y_train
Lesevorgang beendet
Schreibvorgang beendet

semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies

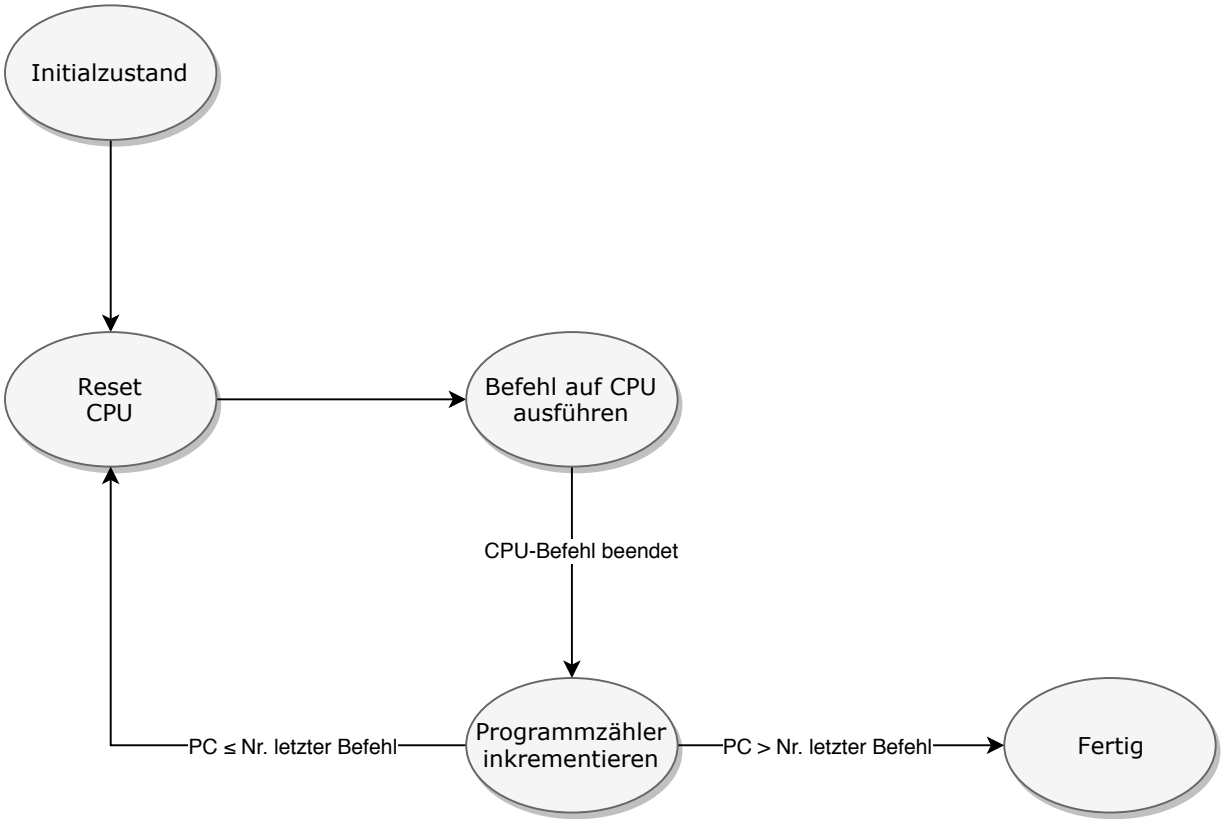


Abbildung 4.2: Zustandsautomat zur Ausführung des Algorithmus

tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

1

5 Simulation

5.1 Beschreibung der Testfälle

5.2 Beschreibung der Testergebnisse

6 Fazit

- Eventuell werden die temporären ergebnisse w1-reg und w2-reg nicht benötigt:

Statt:

$w1_{reg} = w1 * reg$
$dw1' = dw1 + w1_{reg}$
$dw1'' = dw1' * stepsize$
$w1' = w1 + dw1''$

Das entspricht ja: $w1' = w1 + (dw1 + (w1 * reg))' * stepsize$
$w1' = dw1 * stepsize + w1 * (1 * reg * stepsize)$

Deshalb besser:

$w1_{tmp} = w1 * (1 + stepsize * reg)$
$dw1' = dw1 * stepsize$
$w1' = w1_{tmp} + dw1'$

das spart 1 Operation und 1 Matrix Register