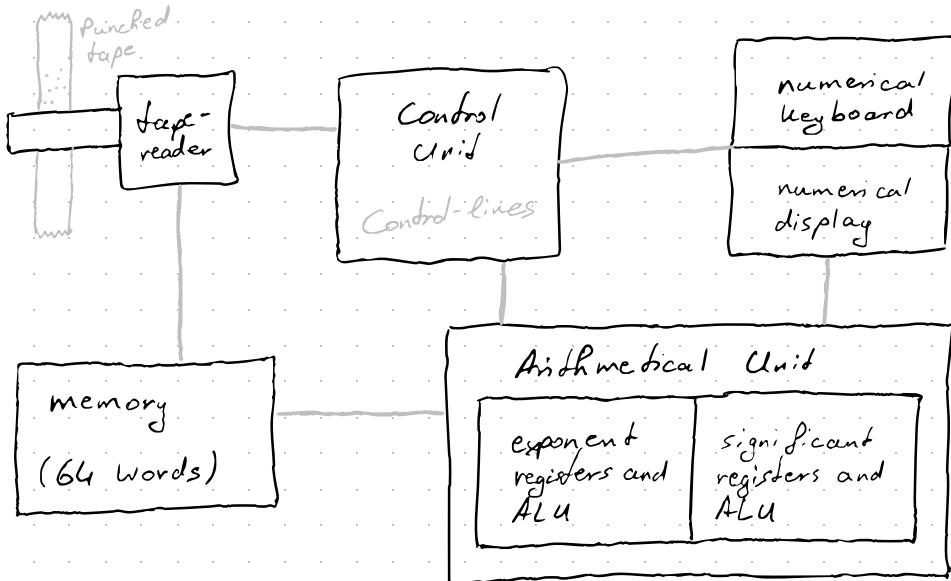


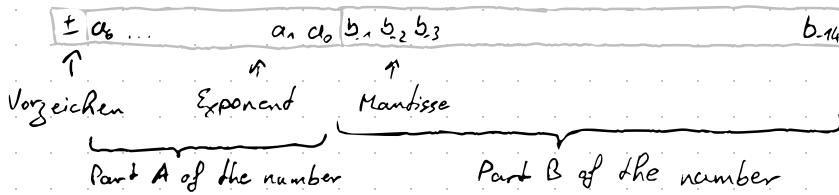
Zuse Z3

Architektur:



- Prozessor und Speicher voneinander getrennt
- Fließkommazahlen können ebenfalls verarbeitet werden. Diese werden in einer sog. „semi logarithmischen“ Repräsentation gespeichert. Diese ist der heutigen float-Repräsentation ähnlich. Der Exponent und die Mantisse werden getrennt verarbeitet.
- Der Tape-Reader liefert den Opcode und den Operanden (= Speicher-Adresse).

Float - Darstellung



Exponent: Zweierkomplement, von -64 bis 63

Mantisse: Normalisierte Darstellung $b_0 = 1$

$$1.b_1 b_2 b_3 \dots b_{14}$$

$2^0 2^1 2^2 2^3 \dots 2^{14}$

↑
Da die erste Ziffer immer eine 1 ist,
muss sie nicht gespeichert werden.

Damit man auch die Ziffer 0 darstellen kann,
wird jede Ziffer mit dem Exponenten -64 als
eine 0 angesehen.

Jede Zahl mit dem Exponenten 63 wird als
unendlich angesehen.

Die Hardware der Z3 behandelt diese zwei
Zustände als Ausnahmefälle und setzt beim
Laden der Ziffern in den Speicher das
Exception flag.

kleinste darstellbare Zahl:

$$2^{-63} = 1,08 \cdot 10^{-19}$$

größte darstellbare Zahl:

$$1,999 \cdot 2^{62} = 9,2 \cdot 10^{18}$$

Für mathematische Operationen konnten die Argumente, Dezimalzahlen mit 4 Nachkommastellen, über ein Tastenfeld eingegeben werden. Der Exponent wurde eingegeben indem die richtigen Knöpfe in einer Reihe, -8, -7, ..., 7, 8, gedrückt wurden. Die originale Z³ konnte nur Eingaben von 10⁻⁸ und 9,999 · 10⁸ entgegen nehmen (der Nachbau im deutschen Museum in München hat mehr Knöpfe)

Die Ausgabe des Rechen-Ergebnisses wurde mit Lampen, sstellvertretend für die Ziffern 0 bis 9, ausgegeben. Die größte ausgegebene Zahl war die 19,999 und die kleinste 00001. Der Exponent wurde separat ausgegeben.

Befehlssatz

Das Programm wird auf Lochstreifen gespeichert und eingelesen: 8 Bits pro Zeile. Die Z3 unterstützt 9 verschiedene Instruktionen:

- I 10 Operationen
- Speicherzugriffe
- Arithmetische Operationen

Opcode: Variable Länge von 2 bis 5 Bits

Speicherzugriffe kodieren den Operanden

(die Adresse des Speicherwortes) in 6 Bits
(max. 64 Wörter, wie der Speicher).

Die Instruktionen LI und LO (Werte über die Tastatur einlesen und auf dem Bildschirm ausgeben) pausieren die Maschine, um für die Eingabe oder das Ablesen eines Wertes genug Zeit bereit zu stellen.

Die Z3 hat keine bedingten Sprünge (Branches). Schleifen lassen sich kondensieren, indem die beiden Enden des Tapes miteinander verbunden werden, aber es gibt keine Möglichkeit Bedingungen zu überprüfen. Damit erfüllt die Z3 nicht die Anforderungen an Turing-Vollständigkeit.

Instructionen

Typ	Befehl	Beschreibung	Opcode
I/O	LU	Tastadateneingabe	01110000
	LO	Bildschirmausgabe	01111000
Speicher	Pr z	Adresse z laden	11z ₆ z ₅ ...z ₁
	Ps z	Adresse z schreiben	10z ₆ z ₅ ...z ₁
arithmetisch	Lm	Multiplication	01001000
	Ld	Division	01010000
	Lw	Wurzel	01011000
	Ls1	Addition	01100000
	Ls2	Subtraktion	01101000

Geschwindigkeit

Die Z3 war eine getaktete Maschine. Jeder Taktzyklus betreibt eine 5-Schläge Pipeline die folgen Schritte durchläuft:

I lesen des Befehls auf dem Tape

II und III Ausführen des Befehls (für den Exponenten & die Mandisse)

IV und V zurückschreiben des Ergebnisses

Die Instruktionen brauchen folgende Anzahl an Ziffern:

Multiplikation 16

Division 18

Wurzel 20

Addition 3

Speichern 0 oder 1

Laden 1

Subtraktion 4 oder 5 abhängig vom Ergebnis

Tastatureingabe 9 bis 41 } abhängig vom Exponenten
Bildschirmausgabe 9 bis 41 }

Land den Dokumenten von Konrad Zuse braucht
eine Multiplikation ca. 3 Sekunden.

Die Anzahl an Taktzyklen für die Ein- und Ausgabe hängt vom Exponenten ab, da die Ziffern vom Dezimalsystem in die zugehörige Binärdarstellung übersetzt werden müssen.

Die Addition und Multiplikation braucht mehr als einen Zyklus, da bei Fließkommazahlen der Exponent beider Argumente auf denselben Wert gesetzt werden muss.

Das Speichern einer Zahl kann direkt bei der Ausführung der arithmetischen Operation erfolgen. Dabei überlappt sich der Speicher-Befehl mit dem Zyklus des vorherigen Befehls.

Programmierung

6.6 Speicherwörter

↑ laden / speichern

2 floating-point Register: R_1 und R_2

Die Ziel-Register sind für die jeweiligen Befehle vorgegeben.

Multiplikation: $R_1 = R_1 \times R_2$ $R_2 = 0$

Division: $R_1 = R_1 / R_2$ -"-

Addition: $R_1 = R_1 + R_2$ -"-

Subtraktion: $R_1 = R_1 - R_2$ -"-

Wurzel: $R_1 = \sqrt{R_1}$ -"-

Speichern und Displayausgabe greifen auf R_1 zu welches danach auf 0 gesetzt wird.

Der erste Lesezugriff speichert das Ergebnis in R_1 , alle darauf folgenden Lesezugriffe speichern die Zahlen in R_2 .

Tastatureingabe lesen: Ergebnis wird in R₁ gespeichert, R₂ wird auf 0 gesetzt da es temporär für die Umrechnung ins Binärsystem gebraucht wird.

Beispiel: Berechnung eines Polynoms

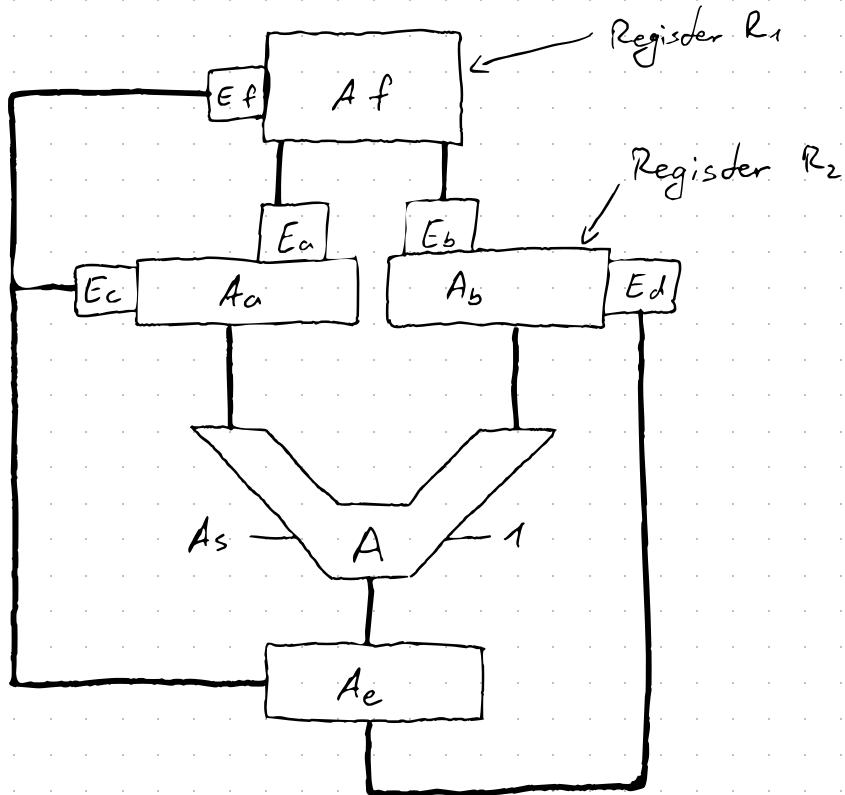
$$x^3 a_3 + x^2 a_2 + x a_1 + a_0 \\ = x(a_1 + x(a_2 + x a_3)) + a_0$$

Annahme: Die Konstanten a₀, a₁, a₂ und a₃ befinden sich im Speicher an den Stellen 0, 1, 2 und 3 und der Wert für x befindet sich an Adresse 5.

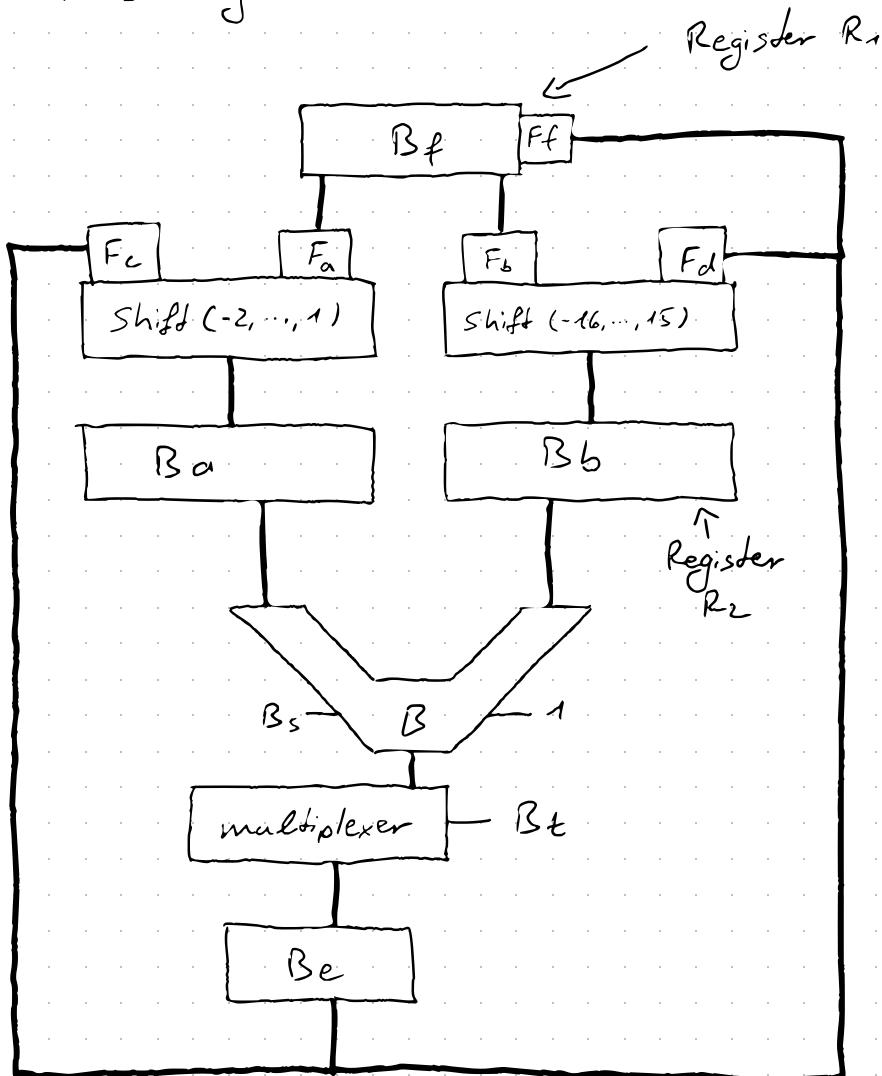
Pr 3	a ₃ in R ₁ laden
Pr 4	x in R ₂ laden
Lm	R ₁ := R ₂ · R ₁ (= a ₃ · x)
Pr 2	a ₂ in R ₂ laden
Ls ₁	R ₁ := R ₁ + R ₂ (= (a ₃ · x) + a ₂)
Pr 4	x in R ₂
Lm	R ₁ = x · (a ₃ · x + a ₂)
Pr 1	:
Ls ₁	
Pr 4	
Lm	
Pr 0	
Ls ₁	
Ld	Ergebnis ausgeben

Block-Diagramm des Prozessors

Verarbeitung des Exponenten:



Verarbeitung der Mantisse



Das Register R_1 setzt sich aus dem Register-Paar $\langle A_f, B_f \rangle$ zusammen und das Register R_2 aus dem Paar $\langle A_b, B_b \rangle$. Das Paar $\langle A_a, B_a \rangle$ ist für den Programmierer unsichtbar und dient als zusätzliches drittes temporäres Register.

Die zwei Prozessor-Hälften besitzen beide eine ALU (A und B) die für die Ausführung der Additions oder Subtraktion der Exponenten und Mantissen zuständig ist. Der Ergebnis-Exponent landet im Register A_e . Für das Mantissen-Equivalent B_e kann über das Flag B_t ausgewählt werden, ob in B_e das Ergebnis der ALU oder der unveränderte Inhalt des Registers B_a landen soll (falls $B_t = 0$ dann gilt $B_e := B_a$). Das Paar $\langle A_e, B_e \rangle$ kann ebenfalls als ein internes unsichtbares Register angesehen werden.

Die Kästchen $E_a, E_b, \dots, E_F, F_a, F_b, \dots, F_d$ sind Schalter, die den Daten-Bus öffnen oder schließen. Um den Inhalt des Registers A_f nach A_a zu kopieren, muss der Schalter E_a geöffnet sein und es erfolgt $A_a := A_f$.

Der Abschnitt für die Verarbeitung der Mantisse hat im Gegensatz zum Bereich für die Verarbeitung des Exponenten zwei Shifter verbaud. Der Shifter zwischen B_f und B_a kann die Mantisse um bis zu zwei Stellen nach rechts und eine Stelle nach links verschieben, was einer Division durch 4 oder einer Multiplikation mit 2 entspricht. Der zweite Shifter, zwischen B_f und B_b , kann die Mantisse um 16 Positionen nach rechts oder 15 Positionen nach links verschieben. Diese Verschiebungen werden für die arithmetischen Operationen benötigt. Multiplikationen oder Divisionen mit Potenzen der Zahl 2 können damit erledigt werden, während die Operanden für

den nächsten Befehl geholt werden

Die Register haben folgende Größen:

Af 7 bits

Bf 17 bits

Aa 8

Ba 19

Ae 8

Bb 18

Ab 8

Be 18

Die zusätzlichen Speicherstellen dienen dem Speichern von Überträgen oder um die Genauigkeit der ALU zu erhöhen oder weil Zwischenergebnisse mancher Befehle ($\sqrt{ }$) diese benötigen.

Die Schalter As und Bs weisen an, dass das Argument aus R2 negiert werden muss.

Darüber wird zwischen einer Addition und einer Subtraktion unterschieden. Die eingebaute 1-Konstante wird benötigt um das Zweierkomplement zu bilden.

Beispiel: Addition zweier Zahlen mit gleichem Exponenten.

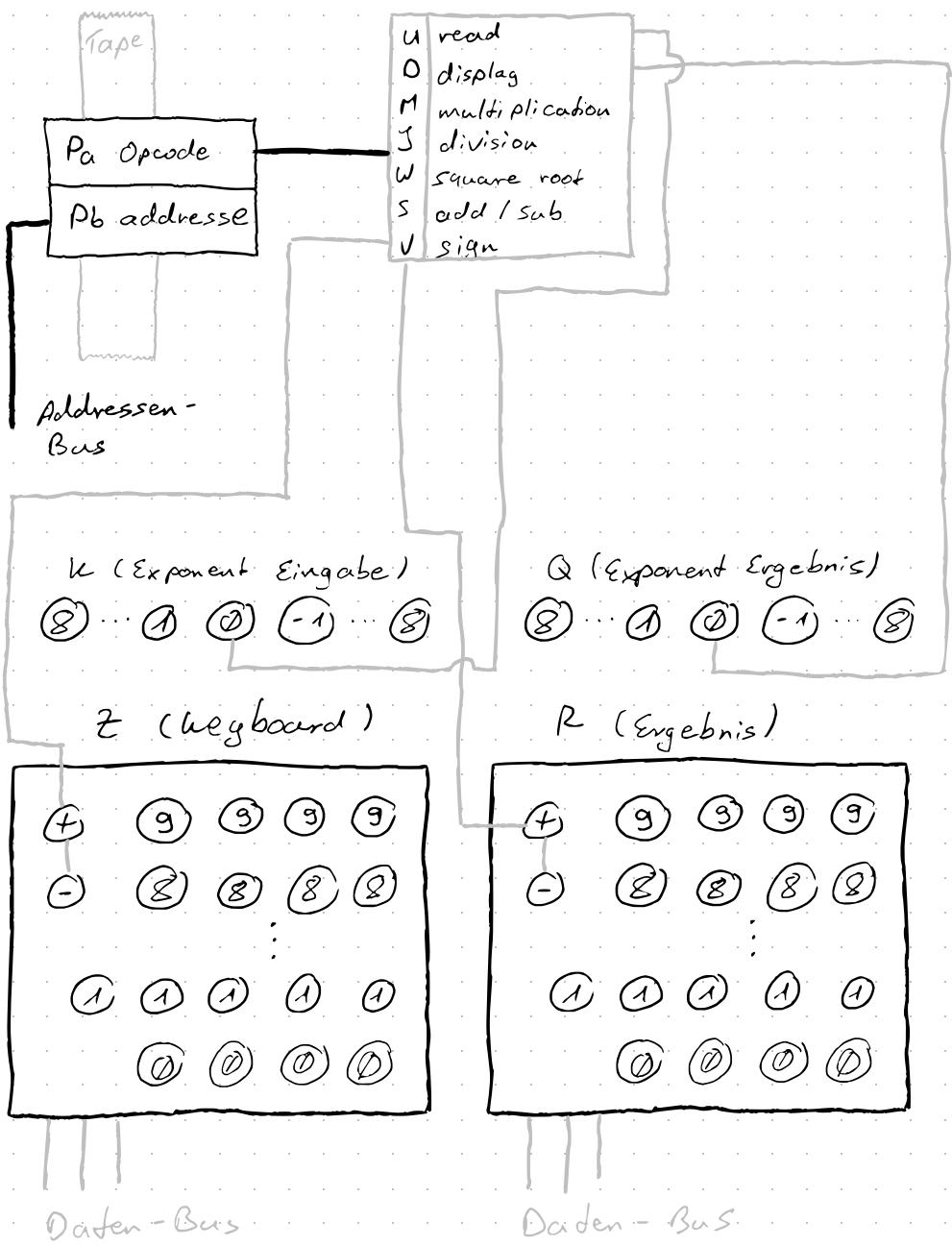
Zuerst werden die 2 Exponenten in die Register Af und Ab geladen. Da sie gleich sind, muss hier nichts weiteres passieren.

Die Mantissen werden in die Register Bf und Bb geladen. Das Register Ba wird mit dem Wert von Bf geladen indem der Schalter Fa eingeschaltet wird. Das Ergebnis der Addition wird über das Relais Bt in das Register Be gespeichert. Die Relais in Ff weisen an, dass das Ergebnis in Bf abgelegt werden soll.

Damit die richtigen Schalter für die jeweilige Operation gesetzt sind, wird die Maschine von einem Mikroprogramm gesteuert.

Steuerwerk

Kontroll-Einheit



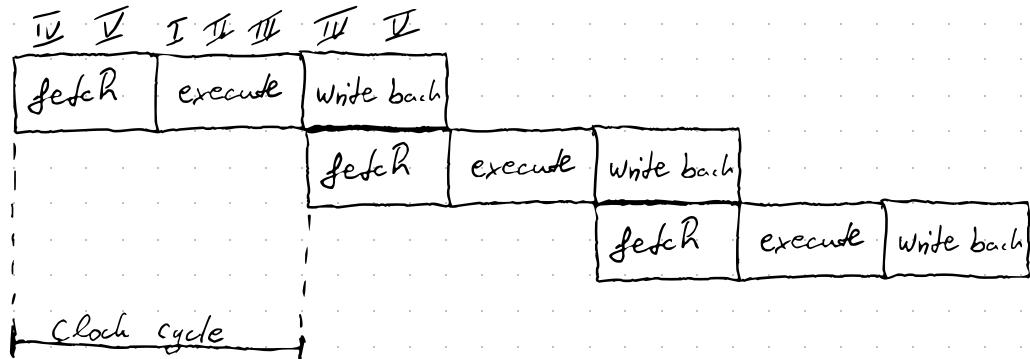
Der Schaltkreis P_A dekodiert den Opcode vom Tape. Falls es sich dabei um einen Speicherzugriff handelt setzt der Schaltkreis P_B die Adresse auf den Bus für Speicheradressen. Die Kontrollleinheit ermittelt das richtige Mikroprogramm für den jeweiligen Befehl.

Die Tasdataereingabe wird von der Schaltung 2 entgegengenommen, wobei nur ein Knopf pro Zeile gedrückt sein konnte. Das Ausgabe-Panel wurde mit Lampen realisiert. Nachdem eine Zahl eingegeben wurde, wurde über eine Datenleitung die Eingabe in Register R_A abgelegt und das Gerät begann damit, die Eingabe in die Binärdarstellung umzurechnen. Wenn der Exponent 0 war, konnte die Umrechnung in 9 Taktzyklen erfolgen. Ansonsten brauchte sie 4 mal den Betrag des Exponenten längere ($= 9 + 4 \cdot (\text{Exponent} + 1)$).

Der Mikrosequenzer

Das Herzstück des Steuerwerks bildet der Mikrosequenzer.

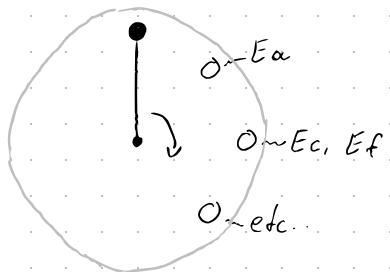
Bei der Z3 ist jeder Zyklus in 5 verschiedene Abschnitte entlang der internen Pipeline unterteilt:



In den Zuständen IV und V werden Daten innerhalb des Gerätes transportiert. Die Ausführung eines Befehls geschieht in den Phasen I, II und III.

Die Z3 verwendet indem eine 3-schläfige von-Neumann-Pipeline um die Ausführung des Programms zu beschleunigen.

Die Mikrosequenzen werden von speziellen Kontrollrädern ausgeführt. Für jede Mikrosequenz existiert ein eigenes Kontroll-Rad: Multiplikation, Addition, ... Mit jedem Taktzyklus wandert der Kontroll-Arm eine Position wieder und schließt damit einen elektrischen Kontakt, mit dem gezielt Relais oder Relais-Boxen ($E_a, \dots, E_f, F_a, \dots, F_d$) aktiviert werden. Damit wird der Datenfluss zwischen den Registern gesetzen. Diese Kontrollleitungen an den Mikrosequenzen waren fest verdrahtet.



Dieser Aufbau vereinfachte den Entwurf der Z3, da sie damit in einzelne Module zerlegt werden konnte. Beim Entwurf der Mikrosequenzen müssen viele Details beachtet werden, da versehentliche Kurzschlüsse das Gerät beschädigen könnten.

Der Addierer

Der Addierer der Z3, welcher für Additionen und Subtraktionen benötigt wird, wurde so gebaut, dass der Additions-Prozess möglichst schnell durchgeführt werden kann. Dafür wird ein Verfahren namens "Carry lookahead" verwendet.

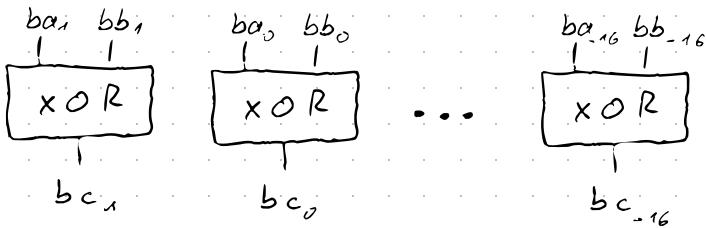
Wenn die Addition auf direktem Wege durchgeführt wird, müssen die Überträge von einem Bit ans nächste weitergegeben werden. Dies kann ziemlich lange dauern (bis zu 16 Taktzyklen).

Für eine Subtraktion wird zuerst das 2er Komplement gebildet indem die Zahl binär invertiert und dann um 1 erhöht wird.

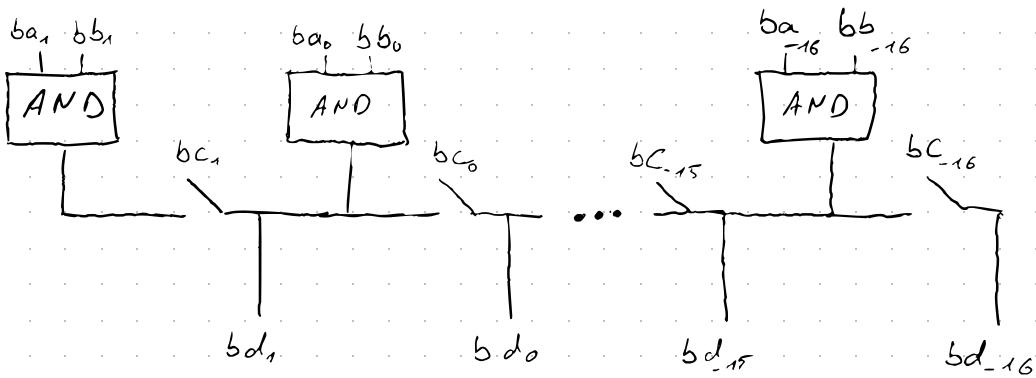
Zuerst wird ein binäres XOR der beiden Zahlen gebildet und temporär gespeichert.

Damit gilt für die Mantissen in $B_f(B_a)$ und B_b :

$$B_c := B_a \text{ XOR } B_b$$

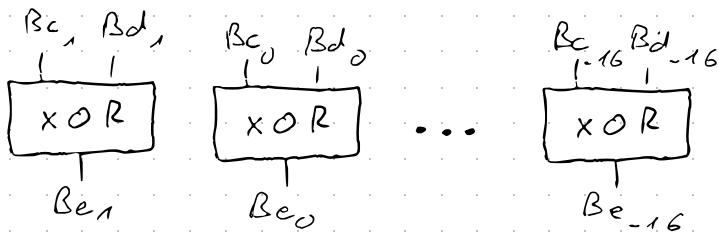


Im nächsten Schritt wird das binäre Und dazu verwendet, die Positionen zu ermitteln, an denen ein Übertrag benötigt wird. Die Zwischenergebnisse $B_d[1:-16]$ werden nach folgendem Schaltkreis gebildet:



Wenn ein Bit auf 1 gesetzt ist, steht die Leitung unter Strom. Andernfalls ist die Leitung nicht angeschlossen (Tri-State). Wenn ein Bit in B_c 1 ist, wird der zugehörige Schalter geschlossen.

Das Endergebnis wird gebildet indem die beiden temporären Werte B_c und B_d miteinander mit einer XOR Operation verbunden werden: $B_e = B_c \text{ XOR } B_d$.



Dadurch, dass alle Relais die einen übertrag weitergeben müssen (die B_c -Schalter) gleichzeitig geschlossen sind, wandert diese Information direkt an das zugehörige Bit und muss nicht erst eine Kette von Relais aktivieren / deaktivieren.

Das ganze Verfahren funktioniert folgendermaßen:

Die xor Operation setzt alle Stellen an denen ein Übertrag ankommt auf 1 und alle an denen einer entsteht auf 0. Durch die and Operation geben alle binären Stellen ihren Übertrag aus, falls einer dabei entsteht.

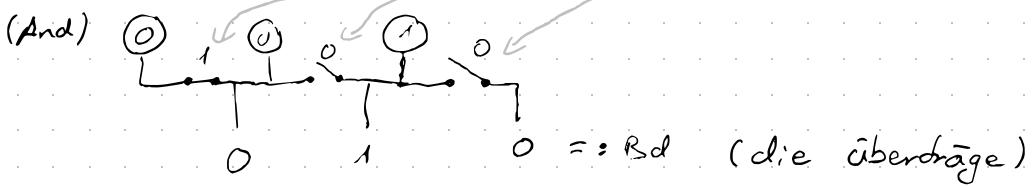
Die Schalter sind mit dem Ergebnis aus der xor Operation so gesetzt, dass der Übertrag nach links wandert bis zu der Stelle an der er eingesetzt werden muss. Die letzte xor Operation verbindet die Überträge mit dem Rechenergebnis ohne Überträge.

► Beispiel mit nur 3 bit:

$$B_a := 101$$

$$B_b := 001$$

$$B_c := B_a \text{ xor } B_b = 100$$



$$B_d \text{ xor } B_c = 110 \quad (\text{Ergebnis})$$

Numerische Algorithmen

Die in der Z3 implementierten numerischen Algorithmen sind identisch zu denen die sich in jedem modernen Prozessor befinden.

Floating-point-Exceptions

Bei der Float-Darstellung müssen einige Spezialfälle berücksichtigt werden, z.B. eine Ausnahmehandlung für die Zahl 0, Über- oder Unterläufe (overflow, underflow). Dafür werden die Exponenten der Zahlen beim Laden aus dem Speicher oder nach einer arithmetischen Operation überprüft. Ein spezieller Schaltkreis analysiert das Register Ae: Wenn der Exponent den Wert -63 annimt, wird die Zahl eine Markierung gesetzt, um sie als 0 zu vermerken. Ein Relais namens N1 wird gesetzt, wenn die 0 in Register $\langle Af, Bf \rangle$ gespeichert wird und das Relais N2 verhält sich identisch für das Registerpaar $\langle Ab, Bb \rangle$.

Damit kann für eine arithmetische Operation geprüft werden, ob eines der Argumente oder sogar beide 0 sind.

Wenn der Exponent den Wert 63 annimmt (unendlich), werden dies 2 Relais N1 und N2 verwendet.

Falls nun eine Operation mit einer dieser beiden besonderen Zahlen ausgeführt wird, wird der Exponent des Ergebnisses am Ende auf -64 gesetzt. Die Operation für die Mandisse wird regulär ausgeführt, da es am Ende egal ist, welche Zahl sich darin befindet, der Exponent gibt die Zahl 0 an. Invaliden mathematische Operationen werden von dem Gerät abgefangen. Eine zugehörige Kontrolllampe leuchtet auf und die Z3 hält ihre Arbeit an. Dazu gehören folgende Operationen:

$$\frac{0}{0}, \infty - \infty, \frac{\infty}{\infty}, 0 \times \infty$$

Ein weiterer Schaltkreis überprüft das Ergebnis des Addierers. Falls der Exponent den Wert 63 annimmt, hat es einen Überlauf (overflow) gegeben und das Ergebnis wird auf unendlich gesetzt. Falls der Exponent den Wert -64 bekommt, gab es einen Unterlauf (underflow) und das Ergebnis wird auf 0 gesetzt. Dies wird erreicht indem das Relais M_{11} oder M_{10} gesetzt wird.

Addition und Subtraktion

Um zwei Fließkommazahlen zu addieren oder voneinander zu subtrahieren, müssen diese zuerst auf denselben Exponenten gebracht werden. Danach müssen nur noch die Mantissen addiert oder subtrahiert werden. Dazu wird die Mantisse der kleineren Zahl um so viele Stellen wie nötig nach rechts verschoben während der Exponent um dieselbe Anzahl erhöht wird. Es kann vorkommen, dass eine Zahl nach 17 Verschiebungen zur 0 wird.

Die Vorzeichen der beiden Argumente werden vor der Operation miteinander verglichen. Falls die Vorzeichen nicht identisch sein sollen, wird eine Addition zu einer Subtraktion umgewandelt und umgekehrt. Das Vorzeichen des Ergebnisses wird aus den Vorzeichen der Argumente und der Zwischenergebnisse bestimmt.

Die Addition und Subtraktion wird über eine Kette an Relais gesteuert, da die Anzahl an Schritten niedrig ist und sich auch ohne Kontrollrad als Mikrosequenzer realisieren lässt.

Zu Beginn werden die beiden Argumente in den Registernpaaren $\langle A_f, B_f \rangle$ und $\langle A_b, B_b \rangle$ abgelegt. Im ersten Schritt werden die Exponenten voneinander subtrahiert. Daraufhin wird die Mantisse mit dem größeren Exponenten in Register B_a und die Mantisse mit dem kleineren Exponenten in Register B_b geladen.

Der Shifter vor Register Bb wird verwendet um den Wert des Registers um die Differenz der Exponenten zu verschieben. Falls der Wert dabei zu weit nach rechts geschoben wird und zu einer 0 wird, wird dieser Wert abgefangen.

Danach werden die Mantissen addiert und zum Schluss prüft der Prozessor, ob der Wert größer als 2 ist. Falls das der Fall ist, muss die Mantisse des Ergebnisses um eine Position nach rechts verschoben und der Exponent um 1 erhöht werden.

Die folgende Tabelle stellt eine Übersicht über alle Schritte dar:

Tuht	Stage	Exponent	Mandisse
1	<u>IV, V</u>	$Aa := Af$	-
	<u>I, II, III</u>	$Ae := Aa - Ab$	$Be := 0 + Bd$
2	<u>IV, V</u>	$\text{if } (Ae > 0) \text{ then}$ $Ab := 0, Aa := Af$ $\text{else } Aa := 0$	$\text{if } (Ae > 0) \text{ then}$ $Ba := Bf, Bd := Be \text{ (shifted)}$ else $Ba := Be, Bd := Bf \text{ (shifted)}$ $(Ba \text{ oder } Bf \text{ werden } Ac \text{ Plätze nach rechts verschoben})$
	<u>I, II, III</u>	$\text{if } (Be > 2) \text{ then}$ $Ae := Aa + Ab + 1$ $\text{else } Ae := Aa + Ab$	$Be := Ba + Bd$
3	<u>IV, V</u>	$Af := Ae$	$\text{if } (Be > 2) \text{ then}$ $Bf := Be / 2$ $\text{else } Bf := Be$

Für eine Subtraktion werden 5 Takte gebraucht.
Die ersten 2 Takte sind mit der Addition
identisch. Danach werden die Mantissen
subtrahiert. Dieser Schritt wird nur ausgeführt,
wenn die Differenz der beiden Mantissen negativ
ist, damit diese wieder positiv wird.

Im 4. Takt wird die Mantisse normalisiert.
Da nach einer Subtraktion die ersten Bits mit
einer 0 belegt sein können, wird der Wert in
B6 so weit wie nötig durch den Shifter vor
B5 nach links geschoben. Die Anzahl an
Verschiebe-Operationen wird vom Exponenten
subtrahiert und das Ergebnis wird im ledzten
Takt in das Registerpaar $\langle Af, Bf \rangle$
abgelegt.

Dieses Verfahren lässt sich ebenfalls
tabellarisch darstellen:

Takt	Stage	Exponent	Mandisse
1	IV, V	$A_a := A_f$	-
	I, II, III	$A_e := A_a - A_b$	$B_e := 0 + B_d$
2	IV, V	if ($A_e > 0$) then $A_b := 0, A_a := A_f$ else $A_a := 0$	if ($A_e > 0$) then $B_a := B_f, B_d := B_e$ (shifted) else $B_a := B_e, B_d := B_f$ (shifted) (B_a oder B_f werden $ A_e $ Plätze nach rechts verschoben)
	I, II, III	if ($B_e > 2$) then $A_e := A_a + A_b + 1$ else $A_e := A_a + A_b$	$B_e := B_a + B_d$
3	IV, V	$A_a := A_e$ $A_b := 0$	$B_a := 0$ $B_b := B_e$
	I, II, III	$A_e := A_a + A_b$	$B_e := B_a - B_b$
4	IV, V	$A_a := A_e$ $A_b := \text{numShifts}$	$B_b := B_e$ (shifted) (normalisierung: B_e shift links)
	I, II, III	$A_e := A_a - A_b$	$B_e := 0 + B_b$
5	IV, V	$A_f := A_e$	$B_f := B_e$

Multiplikation

Die Multiplikation wird durch häufiges Hintereinanderausführen von Additionen umgesetzt. Die Argumente werden in die Registerpaare $\langle A_f, B_f \rangle$ und $\langle A_b, B_b \rangle$ abgelegt und das temporäre Register $\langle A_a, B_a \rangle$ wird auf 0 gesetzt. Danach werden die Zahlen in einer Schleife aufsummiert.

Die Prozedur braucht insgesamt 16 Takte. Währenddessen wird das Zwischenprodukt in Register B_a gespeichert. Die Schleife führt immer folgende Operation aus:

$$B_a := \frac{B_e}{2} \quad (\text{Shift nach Rechts})$$

$$B_e := B_a + B_b \cdot B_f[i] \quad \text{\scriptsize i-th Bit von } B_f$$

Dabei werden von B_f nur die Bits von $i = -14, \dots, 0$ verwendet. Die Division durch 2 wird über den Shifter gemacht (1 Bit nach rechts). Die Addition der Exponenten wird einmal zu Beginn durchgeführt.

Takt	Stage	Exponent	Mantisse
1	IV, I	$Aa := Af$	-
	II, III	$Ae := Aa + Ab$	if ($Bf[-16] = 1$) then $Be := Ba + Bb$ else $Be := Ba$
2	IV, V	$Aa := Ae, Af := 0,$ $Ab := 0$	$Ba := Be / 2$
	II, III	$Ae := Aa + Ab$	if ($Bf[-13] = 1$) then $Be := Ba + Bb$ else $Be := Ba$
3	IV, VI	$Aa := Ae, Af := 0,$ $Ab := 0$	$Ba := Be / 2$
	II, III	$Ae := Aa + Ab$	if ($Bf[-12] = 1$) then $Be := Ba + Bb$ else $Be := Ba$
		⋮	⋮

i IV, V $Aa := Ae, Af := 0, Ba := Be / 2$
 $Ab := 0$

I, II, III $Ae := Aa + Ab$ if ($Bf[0] = 1$) then
 $Be := Ba + Bb$
 $else Be := Ba$

⋮
⋮
⋮

15 IV, V $Aa := Ae, Af := 0, Ba := Be / 2$
 $Ab := 0$

I, II, III if ($Be > 2$) if ($Bf[0] = 1$) then
then $Ae := Aa + 1$ $Be := Ba + Bb$
 $else Be := Ba$

16 IV, V $Af := Ae$ if ($Be > 2$) then
 $Bf := Bc / 2$
 $else Bf := Be$
 $Bb := 0$

Das Ergebnis der Multiplikation ist eine Zahl r im Bereich $1 \leq r \leq 4$.

Im letzten Schritt der Operation wird das Ergebnis, falls es größer als 2 ist, um eine Stelle nach rechts verschoben und der Exponent des Ergebnisses inkrementiert.

Division

Der Algorithmus zur Division ist ähnlich mit dem zur Multiplikation, bis auf den Unterschied, dass nicht addiert sondern subtrahiert wird.

Die Registerpaare $\langle A_f, B_f \rangle$ und $\langle A_b, B_b \rangle$ werden mit den entsprechenden Werten belegt, $\langle A_a, B_a \rangle$ wird auf 0 gesetzt.

Der Exponent des Ergebnisses entspricht der Differenz der beiden Exponenten.

Der Algorithmus zur Berechnung von x^y arbeitet wie folgt: Die erste Ziffer des Ergebnisses ist 1, wenn $x \geq y$ und 0 wenn $x < y$.

Falls das erste Bit 1 ist, wird der Rest $x - y$ berechnet und rekursiv durch y dividiert. Falls das erste Bit 0 ist, ist der Rest x und die Rekursion wird ebenfalls ausgeführt. Dazu wird der Rest um eine Position nach links geschoben und das Ergebnis dieses Vergleichs wird an Position -1 gespeichert.

Die Schleife dieses Algorithmus hat folgende Form:

$$(i = 0, \dots, -14)$$

$$B_a := 2x \quad B_e \quad (\text{+ shift nach links})$$

if ($B_a - B_b \geq 0$) then

$$B_e := B_a - B_b$$

$$B_f[i] := 1$$

else

$$B_e := B_a$$

$$B_f[i] = 0$$

Das Ergebnis der Division ist eine Zahl r :

$\frac{1}{2} < r < 2$. Wenn r kleiner als 1 ist wird der Exponent dekrementiert und die Mantisse um eine Stelle nach links geschoben.

Takt	Stage	Exponent	Mantisse
1	IV, I	$A_a := A_f$	$B_a := B_f$
	I, II, III	$A_e := A_a - A_b$	if ($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t = 1$ else $B_e := B_a$, $B_t = 0$
2	IV, V	$A_a := A_e$ $A_b := 0$	$B_f := 0$ if $B_t = 1$ then $B_f[0] = 1$ $B_a := 2 \times B_e$ (shift links)
	I, II, III	$A_e := A_a + A_b$	if ($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t = 1$ else $B_e := B_a$, $B_t = 0$
3	IV, V	$A_a := A_e$	if $B_t = 1$ then $B_f[-1] = 1$ $B_a := 2 \times B_e$ (shift links)
	I, II, III	$A_e := A_a + A_b$	if ($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t = 1$ else $B_e := B_a$, $B_t = 0$
		•	•
		•	•
		•	•

i	IV, V	$Aa := Ae$	if $Bt = 1$ then $Bf[2-i] = 1$ $Ba := 2 \times Be$ (shift links)
	I, II, III	$Ae := Aa + Ab$	if $(Ba - Bb \geq 0)$ then $Be := Ba - Bb$, $Bt = 1$ else $Be := Ba$, $Bt = 0$
		⋮	⋮
16	IV, V	$Aa := Ae$	if $Bt = 1$ then $Bf[-14] = 1$ $Ba := 2 \times Be$ (shift links)
	I, II, III	$Ae := Aa + Ab$	if $(Ba - Bb \geq 0)$ then $Be := Ba - Bb$, $Bt = 1$ else $Be := Ba$, $Bt = 0$
17	IV, V	if $(Bf[0] = 0)$ then $Ab := 1$	$Ba := Bf$ $Bb := 0$
	I, II, III	$Ae := Aa + Ab$	$Be := Ba + Bb$
18		$Af := Ae$	if $(Bf[0] = 0)$ then $Bf = 2 \times Be$ (shift links) else $Bf := Be$

Wurzelberechnung

Die Z3 kann in 20 Zeichen die Wurzel einer Zahl im Register $\langle Af, Bf \rangle$ berechnen. Die Berechnung funktioniert aber nur mit geraden Exponenten. Falls der Exponent ungerade sein sollte wird die Mantisse um eine Stelle nach links geschoben und der Exponent decrementeiert. Der Exponent des Ergebnisses entspricht der Hälfte des eingegebenen Exponenten.

Wenn wir von einer Zahl x die Wurzel $\sqrt{x} = Q$ berechnen dann suchen wir eine Zahl Q für die gilt: $Q^2 = x \Leftrightarrow Q = \frac{x}{Q}$. Dazu können sequentiell die Bits $Q[i]$ gesetzt werden bis diese Bedingung erfüllt ist.

Angenommen, wir haben die Bits 0 bis $1-i$ des Ergebnisses schon berechnet:

$$Q_{1-i} = Bf[0] \cdot 2^0 + Bf[-1] \cdot 2^{-1} + \dots + Bf[1-i] \cdot 2^{1-i}$$

Dann wird das Bit q_{-i} hinzugefügt, wonach folgende Bedingung erfüllt sein muss:

$$x \geq Q_{-i}^2 = (Q_{1-i} + q_{-i} \cdot 2^{-i})^2$$

Wenn diese Bedingung erfüllt ist, gilt auch:

$$x - Q_{-i}^2 = (x - Q_{1-i}^2) - 2^{-i} q_{-i} (2 \cdot Q_{1-i} + 2^{-i} q_{-i}) \geq 0$$

Wir definieren diesen Ausdruck als

$$\begin{aligned} 2^{-i} t_{-i} &= x - Q_{-i}^2 \\ &= (x - Q_{1-i}^2) - 2^{-i} q_{-i} (2 \cdot Q_{1-i} + 2^{-i} q_{-i}) \end{aligned}$$

und schreiben den Ausdruck um zu

$$2^{-i} t_{-i} = t_{1-i} 2^{1-i} - 2^{-i} q_{-i} (2 \cdot Q_{1-i} + 2^{-i} q_{-i})$$

und bekommen daraus die rekursive Form

$$2^{1-i} t_{1-i} = x - Q_{1-i}^2$$

Dieser Ausdruck lässt sich vereinfachen:

$$t_{-i} = 2 \cdot t_{1-i} - q_{-i} (2 \cdot Q_{1-i} + 2^{-i} q_{-i})$$

Falls t_{-i} für ein gesetz des q_{-i} positiv ist, wird das Bit $-i$ des Ergebnisses gesetzt ($Bf[-i] := 1$) ansonsten wird $Bf[-i] := 0$ gesetzt. Die rekursive Berechnung startet bei $t_0 = x$. Die Zwischenergebnisse Q_{1-i} werden in den Registern gehalten. Bit $-i$ wird auf 1 gesetzt und das Vorzeichen des Ergebnisses wird untersucht.

Die Programm-Schleife sieht wie folgt aus:

$$Ba := 2 \times Be$$

$$Bb := 2 \times Bf$$

$$Bb[-i] = 1$$

if ($Ba - Bb > 0$) then

$$Be := Ba - Bb, Bf[-i] := 1$$

else

$$Be := Ba, Bf[-i] := 0$$

Takt Stage Mandisse (beim Exponent passiert noch nichts)

- 1 IV. II if ($\Delta f[0] = 1$) then $B_a := 2 \cdot B_f$
 else $B_a := \Delta f$
 $B_b[0] := 1$
- I,II,III :f($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t := 1$
 else $B_e := B_a$, $B_t := 0$
- 2 IV. II $B_f := 0$
 if ($B_t = 1$) then $B_f[0] := 1$
 $B_a := 2 \cdot B_e$, $B_b := 2 \cdot B_f$, $B_b[-1] := 1$
- I,II,III :f($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t := 1$
 else $B_e := B_a$, $B_t := 0$
- 3 IV. II if ($B_t = 1$) then $B_f[-1] := 1$
 $B_a := 2 \cdot B_e$, $B_b := 2 \cdot B_f$, $B_b[-2] := 1$
- I,II,III :f($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t := 1$
 else $B_e := B_a$, $B_t := 0$
- : : :
- i IV. II if ($B_t = 1$) then $B_f[2-i] := 1$
 $B_a := 2 \cdot B_e$, $B_b := 2 \cdot B_f$, $B_b[1-i] := 1$
- I,II,III :f($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t := 1$
 else $B_e := B_a$, $B_t := 0$
- : : :
- 18 IV. II if ($B_t = 1$) then $B_f[-16] := 1$
 $B_a := 2 \cdot B_e$, $B_b := 2 \cdot B_f$
- I,II,III :f($B_a - B_b \geq 0$) then $B_e := B_a - B_b$, $B_t := 1$
 else $B_e := B_a$, $B_t := 0$

Takt Stufe Exponent Mandisse

19 IV, V $Aa := Af / 2$ $Ba := Bf$
 $Bb := 0$

I, II, III $Ae := Ae + 0$ $Be := Ba + Bb$

20 IV, V $Af := Ae$ $Bf := Be$

Ein- und Ausgabe

Die zwei komplizierdesten Instruktionen sind für die Ein- und Ausgabe verantwortlich. Eine 6-stellige Dezimalziffer die über das Tastenfeld eingegeben wurde, wird zuerst in eine Binärzahl umgewandelt. Dazu werden der Reihe nach die Ziffern in ihre Binärdarstellung umgewandelt, in den Bits $Ba[-10:-13]$ gespeichert und das Register Ba wird anschließend mit der Zahl 10 multipliziert. Währenddessen wird der Exponent e wird dabei durch Verschiebungen mit den Multiplikationen angepasst. Wenn der eingegebene Exponent e positiv ist, muss die Mandisse mit 10^e multipliziert werden, wenn e negativ ist dann mit $0,1^{10|e|}$.

Eine Multiplikation mit 10 lässt sich einfach realisieren: durch eine Verschiebung um 2 Stellen nach links und 8 Stellen nach links lassen sich die Produkte $B_a := 2 \times Be$ und $B_b := 8 \times Be$ bilden, deren Summe $10 \times Be$ entspricht.

Eine Multiplikation benötigt 4 Takte und muss in der Anzahl des Betrags des Exponenten ausgeführt werden. Das Einlesen der Mantisse benötigt 9 Takte, womit sich die Eingabe im Bereich von 9 bis 61 Takten aufhält. Eine Multiplikation mit 0,1 ist viel schwieriger zu realisieren, da die 0,1 im Binärsystem eine periodische Zahl ist und kann in der Radientanmeldung für die z3 nachgelesen werden.

Die Funktion zur Ausgabe funktioniert genau so nur in umgekehrte Richtung indem die Zahl mit 10 oder 0,1 multipliziert wird und die letzten 4 Bits ausgegeben werden.

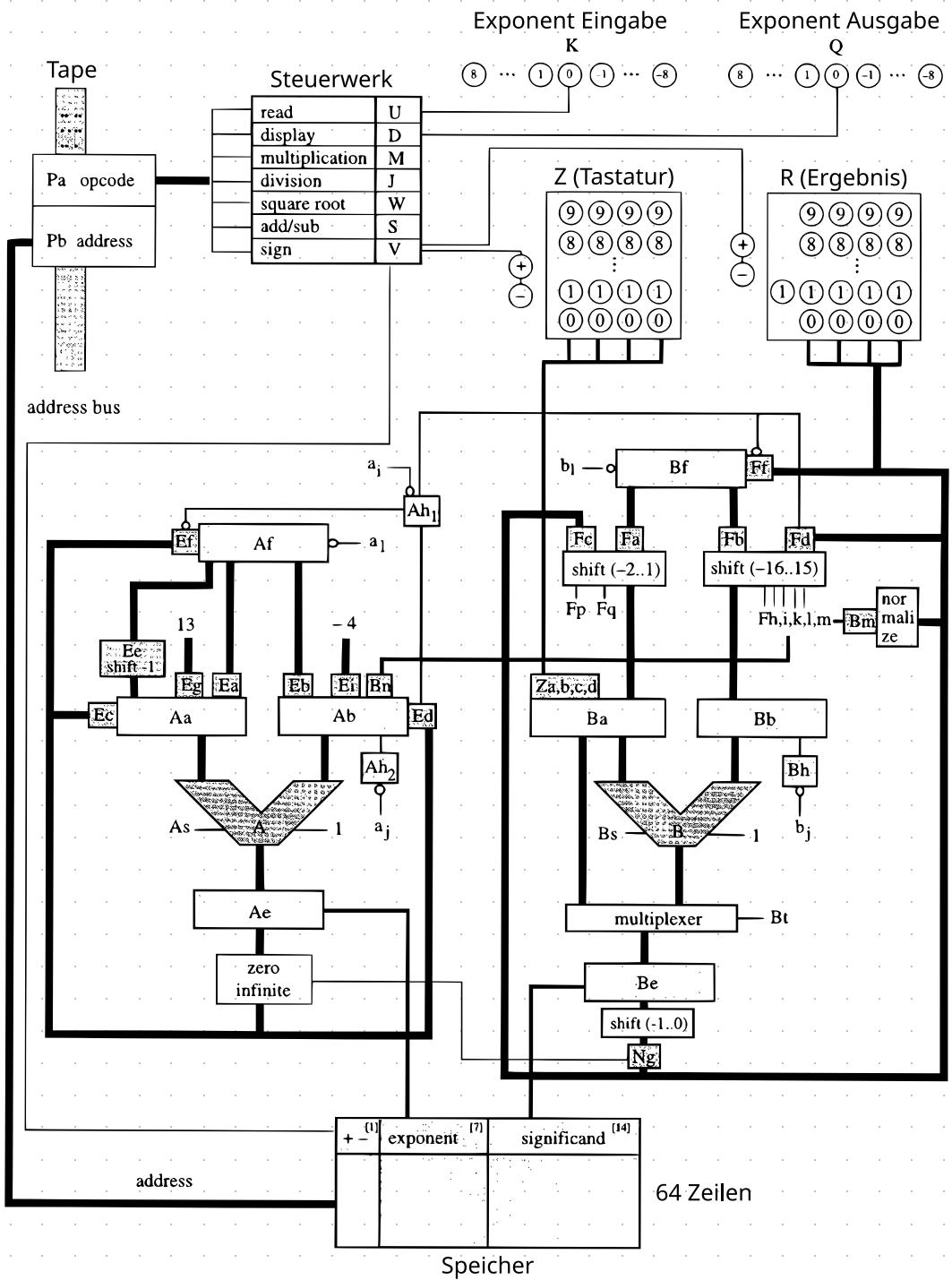
Das Vorzeichen der auszugebenden Zahl wird ebenfalls auf dem Panel ausgegeben.

Gesamtübersicht

Der Transfer von dem Eingabe-Panel in das Register Pa wird über die Relais-Boxen Za, zB, zc und zd gesteuert, die nacheinander aktiviert werden.

Die Relais-Boxen Eg und Ei werden benötigt um die zwei Konstanten +13 in Register Aa und -4 in Register Ab zu laden. Weiters befindet sich noch ein Shifter Ee zwischen Register Af und Aa. Diese Funktionen werden für die Wurzelberechnung benötigt.

Ah arbeitet als Flipflop. Wenn es auf 0 gesetzt ist, landen die Ergebnisse von Lade-Operationen im Registerpaar $\langle A_f, B_f \rangle$, ansonsten landen sie im Registerpaar $\langle A_b, B_b \rangle$. Dieses Relais wird über die Steuerleitung a; auf 0 gesetzt.



Die Steuerleidungen a_j, a_e, b_j, b_e werden verwendet um die Register Af, Ab, Bf und Bb auf 0 zu setzen falls das benötigt wird.

Die Schaltung mit der Beschriftung „Zero, infinite“ ist für die Behandlung von floating point - Exceptions da. Der shifter nach Be wird verwendet um die Mantisse ein Bit nach rechts zu verschieben um Be im Fall von $Be > 2$ zu normalisieren.

Fp und Fq sind relais die die Anzahl an Shifts im Shifter zwischen Bf und Ba steuern. Selbiges machen Fl, Fi, Fu und Fe für den Shifter zwischen Bf und Bd. Die Anzahl an Shifts wird über die Zahl in Fh bis Fn festgelegt, welche über die Relais-Box Bn an das Register Ab übergeben wird um den Exponenten zu korrigieren. Bei einem Shift um 10 stellen nach links (+10) wird vom Exponenten die 10

subtrahiert. Diese Shifts werden zum Normalisieren des Ergebnisses gebraucht.

Der Prozessor der Z3 bestand aus 600 Relais, der Speicher benötigte 3 mal so viele.

Vergleich mit ähnlichen Maschinen aus dieser Zeit

Die Z3 konnte keine bedingten Sprünge durchführen wodurch sie kein universeller Computer nach dem Modell einer Turing Maschine war.

Dennoch war sie die erste Rechenmaschine die mit Fließkommazahlen umgehen konnte und einen Aufbau nach der von-Neumann Architektur besaß. Außerdem fehlte ihr die Fähigkeit, indirekte Zugriffe auf den Speicher auszuführen oder in den Programmspeicher zu schreiben. Anders als die Z1 war sie komplett elektrisch aufgebaut, sowohl der Prozessor als auch der Speicher.