Fabian Peddinghaus

Der serielle Binär-Addierer

Back to the basics: Die Grundlagen unserer digitalen Welt

Besondere Lernleistung S2/S3

Hamburg, den 4. März 2016

Gliederung

I. Einleitung
   1) Vorstellung
   2) Motivation und Idee
   3) Warum müssen Computer Rechnen?
      a. Beispiel Waschmaschine
II. Theorie: Wie rechnet der Computer?
   1) Die Arithmetisch Logisch Einheit
   2) Parallele Addition
   3) Serielle Addition
   4) Grundlagen der Digitaltechnik
      a. Binärsystem
      b. AND - Gate
      c. OR - Gate
      d. NOT - Gate
      e. NAND - Gate
      f. XOR - Gate
   5) Half-Adder
   6) Flip-Flop
      a. RS-Flip-Flop
      b. Getaktetes D-Flip-Flop
III. Praxis: Der Bau des Rechners
   1) Verbaute Komponenten
      a. Kabel und Breadboard
      b. Diode
      c. Transistor
      d. Widerstand
   2) Aufbau der Elemente
      a. NOT-Gate
      b. AND-Gate und seine Probleme
      c. NAND-Gate
      d. Bistabile-Kippstufe
   3) Architektur des Rechners
      a. Speicher
      b. Ein- und Ausgabe
      c. Shift-Register
      d. Probleme mit dem Shift-Register
      e. D-MS-FF
      f. Übertrag-Speicher
      g. Taktsignal
      h. Manueller Takt und Taktselekter
      i. Rechenstarter
IV. Namensgebung
V. Vergleich mit existierenden Addierern
VI. Resümee und Ausblick
VII. Anhang
   1) Bildnachweis
   2) Literaturverzeichnis
   3) Schaltpläne
   4) Bilder des Rechners

I. Einleitung

1) Vorstellung

Mein Name ist Fabian Peddinghaus. Gegenwärtig besuche ich die 12. Klasse des Walddörfer-Gymnasiums in Hamburg. Im vergangenen Jahr habe ich mich mit der genauen Funktionsweise von Computern und Mikrocontrollern auseinandergesetzt. Dabei habe ich als Teil dieser Besonderen Lernleistung einen sog. „seriellen Binär-Addierer“ gebaut. Dieser ist in der Lage, zwei Zahlen miteinander zu addieren und so die Funktion eines Computers zu simulieren.

2) Motivation und Idee

Seitdem ich denken kann, bin ich zutiefst von Computern und Mikrocontrollern fasziniert. Es sind unscheinbare, elektrische Maschinen, die überall um uns herum Prozesse steuern und unseren Alltag sicherer und einfacher machen. Gleichzeitig sind sie extrem klein und in der Lage, ohne menschliches Mitwirken Entscheidungen zu treffen. Doch die wenigsten Menschen wissen heutzutage, wie ein solcher Computer funktioniert. Daher ist es auch nicht verwunderlich, dass man auf die Frage, wie diese Rechner eigentlich funktionieren, immer dieselbe ungenaue Antwort erhält: „Sie rechnen“.

Mir erscheint diese Antwort sehr unbefriedigend. Warum müssen Computer rechnen? Wie rechnen sie? Wie hilft uns als Nutzern das Rechnen des Computers dabei, Prozesse im Alltag zu steuern?

Getrieben von diesen Fragen machte ich mich auf den Weg, mehr über diese unscheinbaren, schwarzen Boxen herauszufinden. Dabei interessiere ich mich für die grundlegendste Ebene. Ich wollte den Rechner in seine elementaren Bestandteile zerlegen. Bei weiteren Recherchen zu dem Thema entdeckte ich immer wieder Bastler, die lediglich aus Transistoren und anderen elektronischen Bauteilen kleine, einfache, elektronische Addierer bauen. Diese Addierer haben jedoch alle etwas gemeinsam: Sie können lediglich direkt und unmittelbar auf Änderungen ihrer Eingänge reagieren. Weder sind sie in der Lage, Zahlen und Informationen zwischen zu speichern, noch benötigen sie ein Taktsignal, um ihre Rechenoperationen zeitlich abzustimmen. Somit sind sie nicht in der Lage, richtige Computer und deren Funktionsweise zu simulieren. Eine große Inspiration und ein gutes Beispiel für einen Rechner, der ohne

Taktsignal arbeitet, ist in dem Video von *Simon Inns* auf der Video-Plattform *Youtube* zu sehen. [1]

Damit stand meine Aufgabe fest: Ich wollte einen mit einem Taktsignal gespeisten Binär-Rechner bzw. –Addierer bauen, der lediglich mit Hilfe sog. „diskreter Komponenten" Zahlen addieren kann. Er sollte zwei 4 Bit-Zahlen zu einer 5 Bit-Zahl verrechnen können. „Diskret" bedeutet hier, dass nicht mehrere elektronische Einzelkomponenten vorgefertigt und miteinander in einem fertigen Schaltkreis integriert verbaut werden, sondern dass die Komponenten als isolierte bzw. unverbundene Einzel-"Baustein" zum Einsatz gelangen. Man kann es mit dem Hausbau aus Fertigteilen gegenüber dem Hausbau nach konventioneller Bauweise vergleichen.

3) Warum müssen Computer Rechnen?

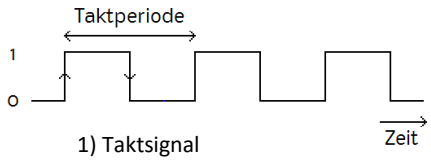
Wir Menschen sind in der Lage, über Dinge und Handlungen nachzudenken und entsprechend zu reagieren. Computer hingegen könne nicht denken. Sie besitzen keine Intelligenz. Man könnte sie im Vergleich zum Menschen als „dumm" bezeichnen. Sie können aber durch komplexes Rechnen und Zählen Denken simulieren bzw. scheinbar denken. Wie genau dieses scheinbare Denken funktioniert, lässt sich gut am Beispiel einer Waschmaschine erklären, die wie fast alle modernen Haushaltsgeräte über Mikrocontroller bzw. Computer verfügt.

Wenn man mit der Waschmaschine Wäsche waschen will, steckt man die Wäsche in die Maschine, gibt Waschmittel hinzu, stellt das gewünschte Programm und die Zeit ein und drückt auf den Startknopf. Die Waschmaschine beginnt, die Wäsche zu waschen. Die Frage ist jetzt: Woher weiß der Computer in der Waschmaschine, dass die Zeit, z.B. 20 Minuten, vorüber und die Wäsche fertig ist? Man ahnt es bereits: Er weiß es durch Rechen und Zählen.

Um systematische Rechenoperationen durchführen zu können, benötigt der Computer ein sog. Taktsignal. Dieses ist vergleichbar mit den visuellen Signalen eines Dirigenten im Orchester. Im Computer wird dieses Signal durch eine Spannungsänderung gegeben. Es ist also elektrischer Natur. Es wird benötigt, um alle Komponenten und Bestandteile des Rechners miteinander zu synchronisieren.[2] Der Computer weiß außerdem, wie lang die Taktperiode

[1] https://www.youtube.com/watch?v=xISG4nGTQYE (29.02.2016)
[2] Meister, Irmtraud/Salzburger, Lukas: AVR-Mikrocontroller-Kochbuch, Franzis, 2013, S.15 ff.

des Taktsignales ist. Aus der gewünschten Länge des Waschganges und der Länge der Taktperiode lässt sich die Anzahl der benötigten Taktperioden berechnen.

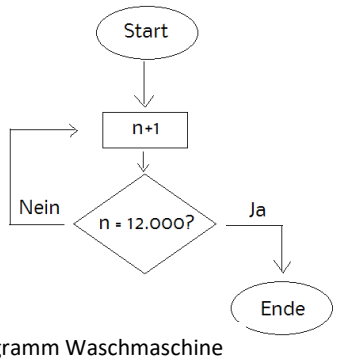


1) Taktsignal

Angenommen, das Taktsignal hätte eine Periodenlänge von 0,1 Sekunden (s) bzw. 100 Millisekunden (ms), dann wäre die Wäsche nach 12.000 Taktperioden fertig.

$$12.000 \times 100ms = 1.200s = 20min$$ 2)

Sobald man den Startknopf drückt, beginnt der Computer mit jeder Taktperiode eine Variable um eins zu „inkrementieren“, d.h. die Variable um den Wert „eins“ zu erhöhen. Mit anderen Worten: Bei jedem Takt addiert der Computer den Wert „eins“ zu der Variablen. Diese Zahl wird im Fachjargon als „Programm Counter“ bezeichnet.[3] Nach jeder Inkrementierung des Programm Counters überprüft der Computer, ob er die gewünschte Zahl, hier 12.000, erreicht hat. Dazu könnte er z.B. 12.000 durch den aktuellen Programm Counter teilen. Ist das Ergebnis dieser Rechnung 1, so stoppt er den Waschvorgang, indem er Signale an die einzelnen Komponenten der Waschmaschine sendet. Ist das Ergebnis jedoch kleiner als eins, so addiert er bei dem nächsten Taktsignal erneut 1 zum Programm Counter. Anschießend überprüft er erneut durch Division, ob der Programm Counter die gewünschte Zahl 12.000 erreicht hat. Diese Schleife wird bis zum Erreichen der 12.000 wiederholt



3) Flussdiagramm Waschmaschine

Das oben beschriebene Verfahren wird deutlich komplexer, wenn man bedenkt, dass der Computer für jede Rechenoperation Zeit bzw. Taktperioden benötigt. So könnte er lediglich

[3] https://en.wikibooks.org/wiki/Microprocessor_Design/Program_Counter (29.02.2016)

nach jeder zehnten Taktperiode den Wert eins addieren und müsste die restlichen neun Überprüfungen und Rechnungen durchführen. Aus diesem Grund benutzen Computer zum Zählen externe Counter. Auf diese soll hier aber nicht weiter eingegangen werden.

Der Computer kann also durch Rechnen und Zählen Prozesse steuern. Indem die Rechenleistung und somit die pro Zeit ausführbaren Rechenoperationen erhöht wird, sind Computer in der Lage, deutlich komplexere Vorgänge zu steuern.

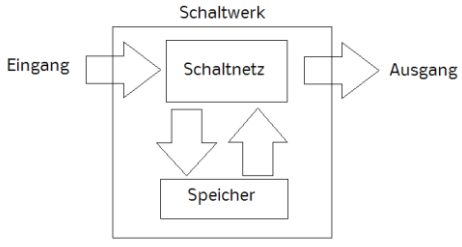
II. Theorie: Wie rechnet der Computer?

1) Die Arithmetisch Logische Einheit

Um rechen zu können, besitzt jeder Computer eine sog. arithmetisch logische Einheit, englisch Arithmetic Logic Unit, abgekürzt ALU. Diese sitzt im Herzen jedes Computers und ist in der Lage, Zahlen in arithmetische Verbindung zu bringen.[4] Wie der Titel bereits erahnen lässt, werden hier ausschließlich ALUs behandelt, die addieren können. Rechenoperationen wie z.B. die Multiplikation lassen sich jedoch aus mehreren Additionen herleiten.

Die ALU zählt zur Gruppe der sog. Schalt*netze*. Das heißt, dass sie unabhängig von ihrem vorherigen Zustand ihre Ausgänge nur in Abhängigkeit ihrer aktuellen Eingänge schaltet.

Die sog. Schalt*werke* hingegen schalten ihre Ausgänge nicht nur in Abhängigkeit ihrer aktuellen Eingänge, sondern auch in Abhängigkeit ihres vorangegangenen Zustandes. Sie verfügen also über die Fähigkeit, ihre Zustände speichern bzw. „sich merken“ zu können. Daher ist ihr Verhalten deutlich komplexer und schwieriger zu beschreiben. Schaltwerke brauchen außerdem ein Taktsignal. Wie bereits erläutert, dient es dazu, alle Bauteile zeitlich abzustimmen bzw. zu synchronisieren. [5]

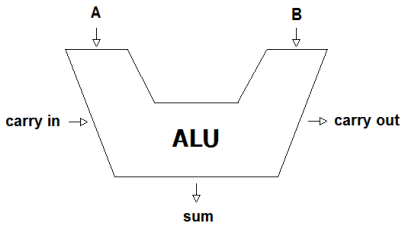


4) Schaltwerk vs. Schaltnetz

[4] Schnabel, Patrick: Computer-Fibel, Selbstverlag, 2014, S.28
[5] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 118

Es gibt unterschiedliche ALU-Typen. Sie unterscheiden sich in Funktionalität und Länge. Die hier behandelten ALUs können lediglich addieren und besitzen eine 1 Bit-Länge. Dass bedeuten, dass sie nur einstellige Zahlen addieren können.

Einstellige ALUs besitzen drei Eingänge und zwei Ausgänge. Durch jeweils einen Eingang kann eine der beiden zu addierenden Zahlen eingespeist werden. Die Belegung ist hier bedingt durch das Assoziativgesetz nicht von Relevanz. An einem der Ausgänge ist man dann in der Lage, das Ergebnis der Rechnung abzugreifen. Die beiden anderen Anschlüsse, Carry-In und Carry-Out genannt, können bei einstelligen Summen und Summanden ignoriert werden.



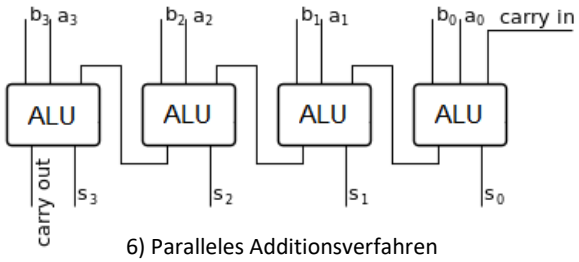
5) ALU

2) Parallele Addition

Werden jedoch zwei Zahlen, deren Summe mehr als eine Stelle benötigt, addiert, so wird der zweite Ausgang benötigt, der den sog. Übertrag ausgibt. Will man außerdem Zahlen mit mehr als einer Stelle addieren, so muss man entsprechend viele einstellige ALUs zusammenschalten. Dabei wird der Carry-Out der ALU, die die kleinste Stelle addiert, in den Carry-In der nächstkleineren ALU gespeist. Der Carry-Out der zweitkleinsten Stelle wird in den Carry-In der dritten ALU gespeist. Dieser Aufbau kann beliebig weitergeführt werden.

Bei der Addition von zwei vierstelligen Zahlen, z.B. 5755 und 4338, benötigt man vier einstellige ALUs. Da der Übertrag hier von ALU zu ALU wandert, wird diese Schaltung als Rippel-Carry-Adder bezeichnet.[6] Obwohl es eine kleine zeitliche Verzögerung gibt, die bedingt ist durch die nicht idealen Eigenschaften der elektronischen Komponenten im Inneren der ALU, handelt es sich um ein Schaltnetz. Das Ergebnis der Rechnung wird sich in Abhängigkeit der Eingänge nach wenigen Millisekunden an den Ausgängen einstellen.

[6] https://www.youtube.com/watch?v=xISG4nGTQYE (29.02.2016)

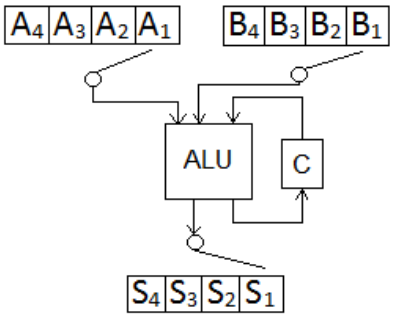


6) Paralleles Additionsverfahren

2) Serielle Addition

Will man nun extrem große Zahlen mit sehr vielen Stellen addieren, so stößt man schnell an die Grenze des Ripple-Carry-Adders. Da Computer extrem große Zahlen miteinander verrechnen müssen, aber eine begrenzte Anzahl an ALUs besitzen, wäre eine Realisierung mit dem Ripple-Carry-Adders nicht praktikabel. Computer greifen deshalb auf ein anders Verfahren zurück.

Computer besitzen eine ALU mit fest definierter Länge. Sie wird durch die Architektur des Mikroprozessors vorgegeben. Hier geht es ausschließlich um einstellige ALUs. Will man mehrstellige Zahlen mit einer einstelligen ALU addieren, so benötigt man Zwischenspeicher. Bei der Berechnung der Zahlen wird damit begonnen, die beiden kleinsten Stellen in die ALU einzuspeisen. Anschießend speichert man sowohl das Ergebnis als auch den Übertrag ab. Dann speist man zusammen mit dem vorher gespeicherten Übertrag die beiden nächstgrößeren Stellen in die ALU ein. Auch hier werden das Ergebnis und der Übertrag wieder ab- bzw. zwischengespeichert. Dieses Verfahren wird wiederholt, bis alle Stellen der beiden Zahlen verrechnet wurden. Anschließend können die Ergebnisse und der zuletzt gespeicherte Übertrag als Summe der Rechnung ausgegeben werden. Das serielle Additionsverfahren ist also in der Lage, die nicht vorhandenen ALUs durch Zwischenspeicher und Taktung zu simulieren.



7) Mein serielles Additionsverfahren

Durch dieses Additionsverfahren unterscheidet sich der von mir gebaute Rechner von den Rechnern, die im Internet verbreiteten und aus diskreten Komponenten zusammengesetzt sind. Zwar gibt es Rechner, die genau dieses Verfahren nutzen. Jedoch tun sie dies ausschließlich mit Hilfe integrierter Schaltkreise.

4) Grundlagen der Digitaltechnik

Um weiter verstehen zu können, wie ein Computer funktioniert und aus welchen Grundbausteinen er zusammengesetzt ist, müssen zunächst einige Grundlagen der Digitaltechnik behandelt werden.

a) Binärsystem

Menschen zählen mit Zahlen von null bis neun. Daraus ergeben sich insgesamt zehn mögliche Zustände, die eine Ziffer bzw. Stelle einnehmen kann. Deshalb bezeichnet man dieses System auch als Dezimalsystem (lat. decem = zehn). Computer hingegen arbeiten mit einem anderen System, dem sog. Binärsystem (von lat. „bini“ = „je zwei“ oder „bina“ = „doppelt“ oder „paarweise“). Es besitzt nur zwei mögliche Ziffern bzw. Zustände: null oder eins. Die Ziffern werden nicht als Stellen, sondern als Bits bezeichnet.[7]

Das Zählen verläuft in beiden Systemen identisch. Ist der höchst mögliche Zustand einer Ziffer erreicht, so wird diese auf null gesetzt und die nächst höhere Ziffer wird um eins inkrementiert.

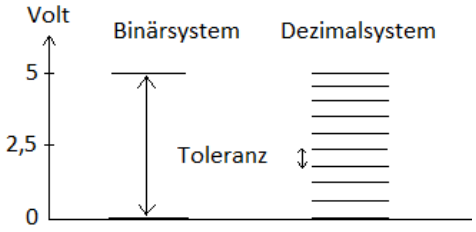
Dezimal	Binär	Dezimal	Binär
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111
8	1000	0	0000

8) Dezimales zu binärem Zählen

Warum benutzen Computer das Binärsystem und nicht das herkömmliche Dezimalsystem? Computer stellen Zahlen oder Zustände in Form von Spannungen dar. Auch hier gibt es unterschiedliche Systeme. Das hier verwendete und wohl bekannteste ist das sog. TTL-System (Transistor-Transistor-Logik). Es stellt den Zustand „null“ bzw. „aus“ bzw. „low“ mit 0 Volt

[7] http://www.elektronik-kompendium.de/sites/dig/0208051.htm (29.02.2016)

(0 V) und den Zustand „eins“ bzw. „an“ bzw. „high“ mit 5 V dar. [8]Der Abstand zwischen den beiden Zuständen ist damit relativ groß. Würde man nun versuchen, 10 Zustände mit 10 möglichen Spannungen zwischen 0 V und 5 V darzustellen, so würde der Abstand zwischen den einzelnen Zuständen deutlich geringer ausfallen. Das wiederum würde zu einer größeren Fehleranfälligkeit führen, da nunmehr eine Abweichung von 0,5 V einen inkorrekten Wert anzeigen würde.



9) Dezimales vs. Binäres System

Hinzu kommt, dass die verwendeten elektronischen Komponenten relativ ungenau sind und gleichzeitig keine Linearität aufweisen. Rechnen mit spannungskritischen Zuständen ist nur mit viel Hardware-Aufwand möglich. Einfache „an“- oder „aus“-Signale hingegen stellen kein Problem für die Elektronik dar.

Im Binärsystem wird mit denselben Regeln wie im Dezimalsystem addiert. Addiert man zwei Nullen, so erhält man eine Null. Ist eine der beiden Zahlen eine Eins, so erhält man als Ergebnis ebenfalls eine Eins. Der einzige Unterschied liegt in der Addition zweier Einsen. Hier wird im Binärsystem, da keine zwei vorhanden ist, auf die nächst höhere Stelle übertragen. Die Summe ist Null und der Übertrag ist Eins.

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

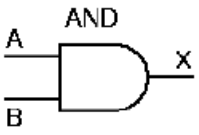
10) Wahrheitstabelle Addition

[8] https://de.wikipedia.org/wiki/Transistor-Transistor-Logik (29.02.2016)

Einfache logische Operationen lassen sich mit sog. Logic-Gates beschreiben. Diese besitzen meist zwei Eingänge und einen Ausgang. Aus diesen einfachen Logic-Gates lassen sich wiederum komplexer Schaltkreise und Logic-Gates zusammensetzen. Mit anderen Worten: Logic-Gates sind die Grundbausteine jedes Computers.

b) AND-Gate

Wenn man die Aussage trifft: „Ich kann Auto fahren, wenn ich ein Auto und Benzin habe“, dann stellt man eine „und“-Verknüpfung auf, das heißt: Auto fahren ist nur möglich, wenn beide Bedingungen eintreten. Nichts Anderes macht das AND-Gate. Nur wenn beide Eingänge eine Eins führen, ist auch der Ausgang „eins“ bzw. „aktiv“. Liegt an mindestens einem der beiden Eingänge eine Null an bzw. ist eine der Bedingungen nicht erfüllt, so ist der Ausgang Null bzw. Auto fahren ist nicht möglich. [9]



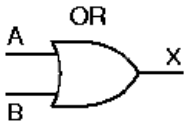
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

11) AND-Gate

c) OR-Gate

Wenn man hingegen behaupte: „Ich kann Brot oder Äpfel essen, um zu überleben“, dann stellt man eine „oder“-Verknüpfung auf. Man kann überleben, egal ob man Brot oder Äpfel oder beides isst. Isst man hingegen nichts, dann kann man nicht überleben. Dasselbe gilt für das OR-Gate. Liegt an mindestens einem der beiden Eingänge eine Eins an, so gibt das OR-Gate eine Eins aus. Liegt an keinem der Eingänge eine Eins an, so liegt am Ausgang eine Null an. [10]

[9] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 21-23
[10] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 23 ff.



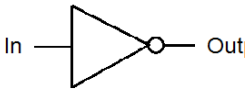
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

12) OR-Gate

d) NOT-Gate

Das NOT-Gate besitzt im Gegensatz zum AND- und zum OR-Gate nur einen Eingang. Liegt an diesem eine Eins bzw. ein High-Pegel an, so gibt das NOT-Gate einen LOW-Pegel aus. Liegt hingegen ein LOW-Pegel an, so gibt es einen HIGH-Pegel aus. Es verkehrt bzw. invertiert das eingehende Signal.

Input	Output
1	0
0	1



13) NOT-Gate

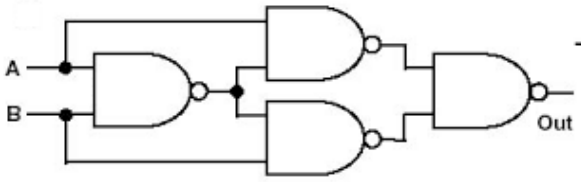
Aus diesen drei Grundbausteinen lassen sich nun weitere Gates konstruieren. Für den Bau eines Rechners sind vor allem das sog. NAND-Gate und das sog. XOR-Gate von Relevanz.

e) NAND-Gate

Das NAND oder NOT-AND-Gate ist ein negiertes AND-Gate. Schließt man ein NOT-Gate hinter ein AND-Gate, so erhält man ein NAND-Gate. [11]

f) XOR-Gate

Das XOR-Gate (auch EXCLUSIVE-OR-Gate) ist mit dem OR-Gate vergleichbar, allerdings mit dem Unterschied, dass es bei gleichen Eingängen, egal ob LOW und LOW oder HIGH und HIGH, einen LOW-Pegel ausgibt. Das XOR-Gate wird aus vier NAND-Gates aufgebaut. [12]



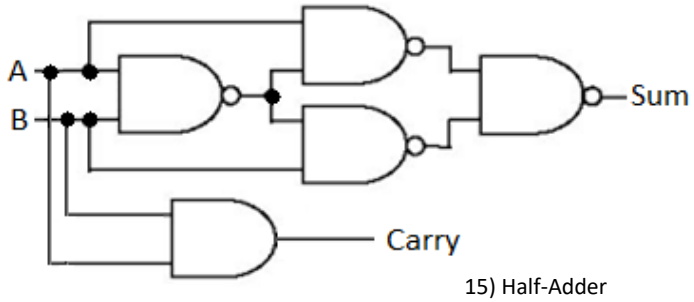
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

14) XOR-Gate aus NAND-Gates

[11] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 26
[12] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 27

5) Half-Adder

Aus einem XOR- und einem AND-Gate lässt sich ein sog. Half-Adder konstruieren. Dieser kann genau zwei 1-Bit-Zahlen addieren. Das AND-Gate ist ausschließlich für den Übertrag zuständig.



15) Half-Adder

Ein Half-Adder ist jedoch noch nicht in der Lage, einen Übertrag in seine Berechnung mit einzubeziehen. Er besitzt einen Carry Out-, aber keinen Carry In-Anschluss. Verbindet man jedoch zwei XOR-Gates und ein NAND-Gate, so erhält man einen Full-Adder. Dieser kann nun als fertige ALU in einem Computer genutzt werden. [13]

6) Flip-Flop

Wie schon erwähnt, ist es ebenfalls wichtig, Informationen zwischen zu speichern. Dazu benötigt man einen Speicher, der aus einzelnen Speicherzellen zusammengesetzt ist. Jede Speicherzelle kann 1 Bit an Information speichern.

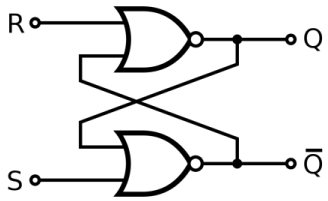
Das Herz jeder Speicherstelle ist das sog. Flip-Flop. Es besteht aus zwei NOR-Gates. Das Flip-Flop besitzt einen Set- und einen Reset-Eingang. Außerdem besitzt es zwei Ausgänge, die als q und q* bezeichnet werden.

Jeweils ein Eingang wird an eines der NOR-Gates angeschlossen. Der andere Eingang des NOR-Gates wird mit dem Ausgang des anderen NOR-Gates versorgt. Gleichzeitig stellen die beiden Ausgänge der NOR-Gates die Ausgänge des Flip-Flops dar. Ein Flip-Flop dieser Anordnung wird als RS-Flip-Flop bezeichnet.

Sind beide Eingänge des Flip-Flops nicht verbunden bzw. LOW, so befindet sich das Flip-Flop im Speicherzustand. Dieser nicht verbundene Zustand wird auch als „Floating“ bezeichnet. Das Flip-Flop behält seinen Zustand bei.

[13] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 72 ff.

Ist eine 0 im Flip-Flop gespeichert, so ist q LOW und q* HIGH. Soll nun eine 1 in dem Flip-Flop gespeichert werden, so wird ein HIGH-Pegel an den Set-Eingang gelegt. Das führt dazu, dass das untere NOR-Gate seinen Ausgang auf LOW schaltet. Das obere NOR-Gate wird deaktiviert und schaltet seinen Ausgang bedingt durch die Negierung auf HIGH. Das Flip-Flop hat eine 1 an q und eine 0 an q* anliegen. Es hat somit den Wert 1 gespeichert.



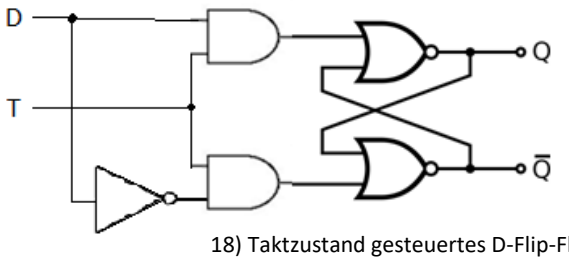
16) Flip-Flop

S	R	Q	$\overline{Q}$
0	0	no change	
0	1	0	1
1	0	1	0
1	1	undefined	

17) Wahrheitstabelle: Flip-Flop

Um Speicherbausteine und Schaltwerke nutzen zu können, müssen sie getaktet werden. Das RS-Flip-Flop ist dabei keine Ausnahme. Diese Taktung wird durch zwei AND-Gates realisiert. Diese AND-Gates werden vor den Set- und den Reset-Eingang geschaltet. Set und Reset werden mit jeweils einem AND-Gate verbunden. Die beiden anderen Eingänge der AND-Gates werden zusammengeführt und mit dem Taktsignal gespeist. Es liegt ein sog. „Taktzustand gesteuertes RS-Flip-Flop“ vor. Änderungen können bedingt durch die AND-Verknüpfung der Eingänge mit dem Taktsignal nur getätigt werden, solange das Taktsignal HIGH ist.

Um Daten einfacher eingeben zu können, werden Set und Reset zu einem Eingang zusammengeführt. Dieser wird als Dateneingang bezeichnet und mit Hilfe eines NOT-Gates adäquat realisiert. Sind sowohl Takt- als auch Datensignal HIGH, so wird das Flip-Flop gesetzt und eine 1 gespeichert. Liegt hingegen eine 0 an dem Datensignal an und ist das Taktsignal HIGH, so wird eine 0 im Flip-Flop gespeichert. Diese Konfiguration des Flip-Flops nennt man „Taktzustand gesteuertes D-Flip-Flop“.



18) Taktzustand gesteuertes D-Flip-Flop

III. Praxis: Der Bau des Rechners

Bei dem Bau meines Rechners war es mir wichtig, keine integrierten Schaltkreise zu verwenden. Weil sie, wie der Name vermuten lässt, fertige Schaltkreise enthalten, die beinahe ideale Eigenschaften besitzen, vereinfachen sie den Bau eines Rechners um ein Vielfaches. Ich wollte jedoch transparent und direkt zeigen, wie ein Computer funktioniert. Daher habe ich bewusst auf die Verwendung solcher schwarzen unscheinbaren Boxen verzichtet und lediglich diskrete Komponenten verwendet. Ihre Funktionen lassen sich genau und nachvollziehbar erklären.

1) Verbaute Komponenten

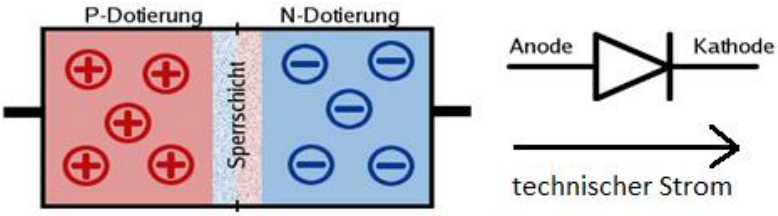
Mein Rechner setzt sich nahezu ausschließlich aus den folgenden fünf Komponenten zusammen:

a) Kabel und Breadboard

Um einzelne elektronische Bauteile miteinander verbinden zu können, benötigt man Kabel. Da man diese nicht einfach aneinanderstecken kann und um die Übersicht zu behalten, verwendet man sog. Breadboards. Sie besitzen kleine integrierte Metallklammern, die in Reihen miteinander verbunden sind. Dies ermöglicht das einfache Ein- und Ausstecken der einzelnen Komponenten.

b) Diode

Eine Diode besitzt die Eigenschaft, Strom nur in eine Richtung durchfließen zu lassen. Diese Eigenschaft verdankt sie einer sog. Sperrschicht, die sich in ihrem Inneren zwischen einer N- und einer P-dotierten Siliziumstruktur ausbildet. Sie ist hilfreich, um z.B. Flip-Flops anzusteuern oder um Interferenzen zwischen Leitungen und Schaltkreisen zu minimieren. Ein großer Nachteil ist jedoch der ebenfalls durch die Sperrschicht bedingte Spannungsabfall. Diesen findet man in allen Diodentypen. Bei den hier verwendeten N4001-Silizium-Dioden beträgt dieser Spannungsabfall ca. 0,7 V. Somit ist es unbedingt notwendig, das Signal nach Verlassen der Diode zu verstärken.



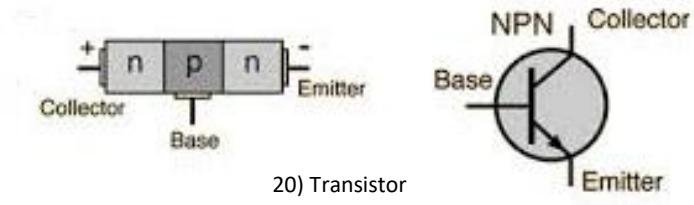
19) Diode

c) Der Transistor

Das Herz eines jeden modernen Computers ist der sog. Transistor. Er verhält sich wie ein Schalter, allerdings mit dem Unterschied, dass er nicht mechanisch, sondern durch elektrischen Strom gesteuert wird. Man kann somit Transistoren mit Hilfe anderer Transistoren schalten.

Transistoren können in zwei Kategorien eingeteilt werden: unipolar und bipolar. Ich habe ausschließlich bipolar Transistoren verwendet, weil sie einfacher und günstiger in den benötigten Mengen zu beziehen sind. Heutzutage wurden aber in elektronischen Anwendungen nahezu alle bipolaren Transistoren von unipolaren Transistoren abgelöst, weil die unipolaren Transistoren einen deutlich geringeren Stromverbrauch und bessere elektrische Eigenschaften aufweisen.

Bipolare Transistoren besitzen drei Anschlüsse: den Kollektor, den Emitter und den Basisanschluss. Der Kollektor ist der Anschluss, an dem der zu schaltende Strom anliegt. Der Emitter ist der Ausgang des Transistors und wird meist mit Masse verbunden. Der Basisanschluss schließlich steuert mit verhältnismäßig niedrigem Basisstrom, ob Kollektorstrom fließen soll oder nicht. Dabei wird der Strom durch die Basis um ein Hundertfaches verstärkt. Bei einem kleinen Basisstrom fließt also ein sehr großer Kollektorstrom.



20) Transistor

Bipolare Transistoren lassen sich weiter in zwei Untergruppen aufteilen: die PNP- und die NPN-Transistoren. Bei dem hier verwendeten Typen handelt es sich um einen NPN-Transistor. Er besitzt zwei Sperrschichten, die ohne angeschlossene, im Verhältnis zum Emitter positive Spannung den Transistor ähnlich wie eine Diode sperren. Wird nun jedoch eine positive Spannung an die Basis angelegt, so werden die Sperrschichten durch den kleinen Basisstrom aufgehoben und ein größerer Kollektorstrom kann fließen. Der Transistor leitet. Wird die positive Spannung an der Basis entfernt, so fließt kein Strom mehr durch die Basis, die Sperrschichten bilden sie erneut aus und sperren den Transistor.

d) Widerstand

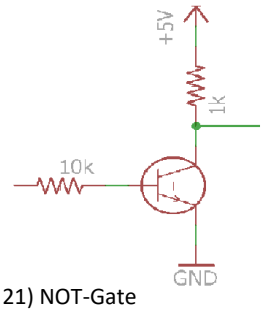
Um den Strom durch die Transistoren zu limitieren und so Logic-Gates konstruieren zu können, benötigt man Widerstände. Die in meinem Rechner verbauten Widerstände besitzen im Wesentlichen zwei unterschiedliche Werte: 1.000 Ω und 10.000 Ω. Diese Werte sind jedoch nicht besonders kritisch. So konnte ich die Widerstände teilweise auch durch Widerstände mit kleineren bzw. größeren Werten ersetzen. Dies erleichterte den Bau des Rechners um ein Vielfaches. Das Schaltsymbol eines Widerstandes sind einer Feder ähnelnde zickzackartige Linien.

2) Aufbau der Elemente

Computer werden aus einzelnen Grundbausteinen bzw. Elementen zusammengesetzt. Damit sind im wesentlichen Logic-Gates und Speicherzellen unterschiedlicher Ausführung gemeint. Wie diese aus den oben behandelten Komponenten zusammengesetzt sind, werde ich im folgendem erläutern.

a) NOT-Gate

Mit zwei Widerständen und einem Transistor lässt sich bereits ein einfaches NOT-Gate Konstruieren. Der 1K Ω-Widerstand wird dabei an den Kollektoreingang des Transistors gehängt und gleichzeitig mit einer Spannung von 5 V versorgt. Diese Anordnung entspricht der eines Spannungsteilers. Ist der Widerstand des Transistors im Mega-Ω-Bereich (d.h., es liegen 0 V an der Basis des Transistors an), so liegt eine Spannung von beinahe 5 V am Ausgang des Gates. Ist der Transistor jedoch geöffnet (d.h., es liegen 5 V an der Basis des Transistors an), so sinkt der Widerstand in den Milli-Ω-Bereich. Es liegen beinahe 0 V am Ausgang an.

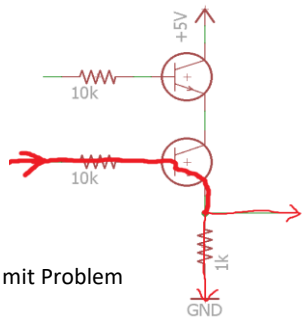


21) NOT-Gate

Der 10K Ω-Widerstand wird zwischen das Schaltsignal und den Basis-Eingang „gehängt“. Er limitiert den Strom durch den Basis-Anschluss des Transistors und verhindert so eine Überlastung. Weiter ist der Wert dieses Widerstandes kritisch, um eine zu große Wechselwirkung mit dem vorangehenden Logic-Gate zu verhindern. Ist der Wert zu klein gewählt, so sinkt die Spannung am Ausgang des vorangegangenen Logic-Gates zu stark ab. Wählt man diesen Wert zu groß, so ist der durch die Basis des Transistors fließende Strom zu klein und die einwandfreie Funktion des Transistors ist nicht mehr gewährleistet.

b) Das And-Gate und seine Probleme

Das AND-Gate ist ähnlich wie das NOT-Gate aufgebaut. Es besteht jedoch bedingt durch zwei vorhandene Eingänge aus zwei in Reihe geschalteten Transistoren. Weiter sind der Ausgang und der damit verbundene Widerstand unterhalb der beiden Transistoren angeordnet. Sind beide Transistoren geöffnet, so liegen am Ausgang nahezu 5 V an. Ist einer der beiden Transistoren jedoch geschlossen, so ist der Widerstand der Transistoranordnung so groß, dass der Widerstand den Ausgang nahezu auf Null zieht.



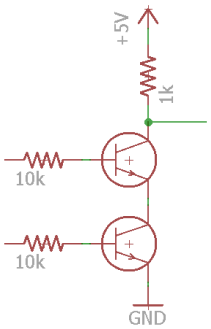
22) AND-Gate mit Problem

Probleme können auftreten, wenn der untere Transistor geöffnet und der obere geschlossen ist. Es sollte in diesem Fall (siehe AND-Gate Tabelle) eigentlich keine Spannung an dem Ausgang des AND-Gates anliegen. Da der untere Transistor jedoch aktiv ist, fließt ein kleiner Basisstrom durch die Basis und zwangsläufig durch den Emitter-Ausgang und über den Wider-

stand zu Masse. Hervorgerufen durch diesen Strom und den damit einhergehenden Spannungsabfall über den Widerstand stellt sich eine kleine Spannung am Ausgang des Gates ein. Diese kleine Spannung liegt ebenfalls am Eingang des nächsten Gates an und kann bedingt durch den Stromverstärkungsfaktor des Transistors diesen und damit das nachfolgende Gate aktivieren. Das AND-Gate, das aus zwei bipolaren Transistoren und einem Widerstand zusammengesetzt ist, bietet somit keine verlässliche Implementierung. Dieser Aufbau wird jedoch vor allem im Internet als mögliche Lösung angepriesen. Dass ein AND-Gate eine häufige Fehlerquelle in meinem Rechner war, konnte ich erst nach langwieriger Fehleranalyse, unter anderem unter der Zuhilfenahme meines Oszilloskops, eindeutig feststellen.

c) NAND-Gate

Das dargestellte Problem kann durch eine Veränderung der Gate-Struktur behoben werden. Man verschiebt den Widerstand über die zwei Transistoren und ordnet den Ausgang des Gates so an, dass er wieder zwischen Widerstand und Transistoren zu finden ist. Die Funktion des so konstruierten Gates ist nun identisch mit der eines NAND-Gates. Der Ausgang wird so lange auf einem positivem Potenzial gehalten, bis beide Transistoren in den leitenden Zustand gehen und die Spannung am Ausgang des Gates auf fast 0 V absinkt. Wird nun der obere, näher am Ausgang des Gates positionierte Transistor aktiviert, so hat das am Basisanschluss liegende Potenzial keine Einfluss auf den Ausgang, da dieses bereits auf einem 5 V-Potenzial ist.



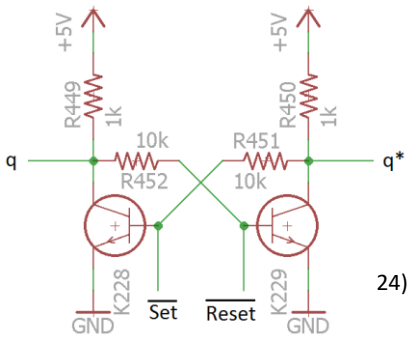
23) NAND-Gate

Will man nun weiterhin die Funktion eines AND-Gates nutzen, so muss ein NOT-Gate hinter das NAND-Gate geschaltet werden. Dieses hebt die Invertierung des Signales auf und führt zu einem Verhalten, das mit dem eines AND-Gates identisch ist. Da für die Realisierung eines AND-Gates jedoch zwei weitere Widerstände und ein weiterer Transistor erforderlich sind,

wird meist versucht, Logikschaltungen, die bipolare Transistoren enthalten, ausschließlich aus NAND-Gates zu konstruieren.

d) Bistabile-Kippstufe

Wie bereits beschrieben, wird ein Flip-Flop eigentlich aus zwei NOR-Gates konstruiert. Durch das Bestreben, die Anzahl der verbauten Komponenten möglichst gering zu halten, hat sich jedoch eine alternative Konstruktionsmöglichkeit des Flip-Flops ergeben. Die Konstruktion benötigt lediglich zwei Transistoren und vier Widerstände und wird als Kippstufe bezeichnet.[14] Dabei fungieren der Kollektor und der Basiseingang des Transistors gemeinsam als OR-Gate. Ist der linke Transistor aktiv, so wird der Basisanschluss des rechten Transistors auf nahezu 0 V gezogen. Dadurch sperrt der rechte Transistor den Strom und hält den Kollektoranschluss somit auf einem 5 V-Potenzial. Somit fließt ein Strom durch den 10K Ω-Widerstand und die Basis des linken Transistors. Dieser beleibt aktiviert und es stellt sich ein stabiler Zustand ein. Will man diesen Zustand ändern, so muss lediglich der Basisanschluss des linken Transistors auf ein Nullpotenzial gezogen werden. Der Transistor wird „hochohmig“, die Spannung an seinem Kollektoreigang steigt auf nahezu 5 V, der rechte Transistor wird aktiviert und hält seinen Kollektor und somit den Basisanschluss des linken Transistors auf 0 V. Da sich zwei stabile Zustände, „gesetzt“ (Set) und „nicht gesetzt“ (Reset) einstellen können, spricht man von einer „Bistabilen-Kippstufe“.



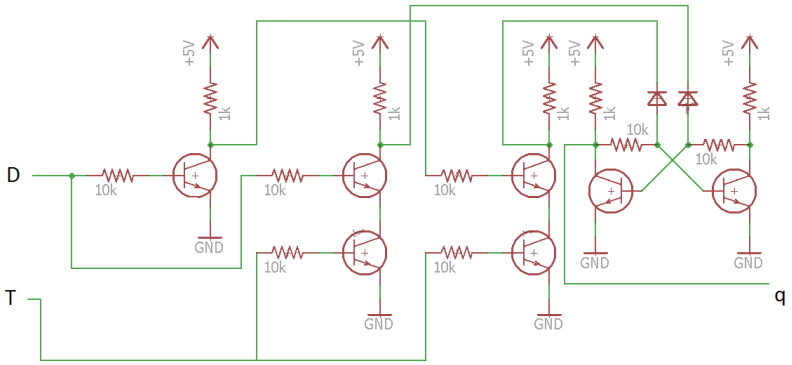
24) Bistabile Kippstufe

Weil die Basisanschlüsse lediglich auf ein Nullpotenzial gezogen werden sollen, muss bei der Verwendung in Kombination mit Logic-Gates bzw. anderen Schaltkreisen auf die sperrende Funktion der Diode zurückgegriffen werden. Diese wird, wie in Abbildung 25) gut sichtbar, zwischen den Basen der Transistoren und den Set- und Reset-Anschlüssen geschaltet. Dadurch, dass Strom nun lediglich *aus* dem Flip-Flop *heraus*, jedoch nie *in* das Flip-Flop *hin-*

[14] http://www.homofaciens.de/technics-base-circuits-multivibrator_ge_navion.htm (29.02.2016)

ein fließen kann, wird eine ungewollte Wechselwirkung zwischen den 5 V-Potenzialen und den Flip-Flops verhindert.

Die getaktete Variante des Flip-Flops sieht dementsprechend wie folgt aus:



25) Getaktetes D-Flip-Flop

3) Architektur des Rechners

Dies führt zur Frage nach der eigentlichen Architektur des Addierers. Dieser soll in der Lage sein, zwei Zahlen mit jeweils 4 Bit miteinander zu addieren. Das heißt, dass er Zahlen von 0 bis 15 miteinander addieren können soll. Da die Summe dieser Addition höchstens 30 sein kann, benötigen wir 5 Bits (2^5 = 32 mögliche Zustände) zur Darstellung.

a) Speicher

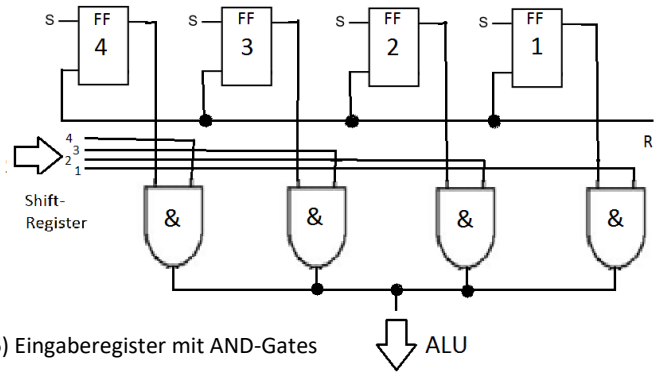
Es kann also festhalten werden, dass 8 Flip-Flops zum Speichern der Summanden und 5 Flip-Flops zum Speichern der Summe benötigt werden. Damit diese nicht nach jeder Rechnung mühsam einzeln zurückgesetzt werden müssen, werden alle Reset-Eingäne der Flip-Flops zu einem allgemeinen „Master-Reset-Knopf“ zusammengeführt. Dieser zieht bei Betätigung die Basisanschlüsse der 13 Flip-Flops auf 0 V und löscht so die gespeicherten Daten. Gleichzeitig müssen die 2 x 4 Eingangs-Flip-Flops gesetzt werden können. Dazu schließt man je einen „Set-Knopf“ pro Flip-Flop an den Basisanschluss des jeweils anderen Transistors an. Wird dieser gedrückt, so zieht er wie der Master-Reset-Knopf die Basis des Transistors zu 0 V. Das Flip-Flop wird so gesetzt bzw. es wird eine 1 gespeichert.

Um den Status jedes Flip-Flops auslesen zu können, benötigt man außerdem eine Vorrichtung, diesen optisch sichtbar zu machen. Das kann z.B. unter der Zuhilfenahme von 5mm

LEDs geschehen. Diese werden zwischen Widerstand und Kollektoreingang des Transistors, dessen Basis mit dem Set-Knopf verbunden ist, auf der einen Seite und Masse auf der anderen Seite geschaltet. Zu beachten ist jedoch, dass die LEDs zusammen mit dem 1K Ω-Widerstand einen Spannungsteiler formen. Somit sind im gesetzten Zustand des Flip-Flops keine vollen 5 V mehr abzulesen. Werden LEDs mit einem zu kleinen Widerstand oder ein alternative Anzeigevorrichtung genutzt, so kann eine einfache Verstärkerstufe zwischen Flip-Flop und Anzeigegerät geschaltet werden. Dies wird hier jedoch nicht weiter erläutert.

b) Ein und Ausgabe

Wie weiter oben bereits behandelt, müssen nun die einzelnen Bits nacheinander eingegeben und es muß ihre Summen abgespeichert werden. Das bedeutet, dass die Flip-Flops, beginnend mit dem sog. LSB (Least Significant Bit) und endend mit dem MSB (Most Significant Bit), an die Eingänge bzw. den Ausgang der ALU angeschlossen werden müssen. Dazu werden in meinem Rechner einfache AND-Gates zwischen Flip-Flop und Daten-Eingang bzw. -Ausgang der ALU geschaltet. Wird nun nacheinander ein HIGH-Pegel an die jeweiligen NAND-Gates gelegt, so können die Bits sequentiell ein- bzw. ausgelesen werden. So werden pro HIGH-Pegel jeweils drei AND-Gates aktiviert. Lediglich bei dem fünften HIGH-Pegel wird nur ein AND-Gate aktiviert.



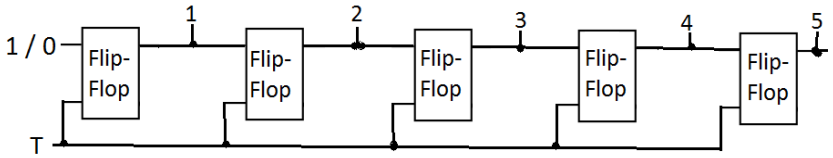
26) Eingaberegister mit AND-Gates

Diese Methode erfordert jedoch fünf nacheinander und an unterschiedlichen Kabeln liegende HIGH-Pegel. Es wird ein Schaltkreis benötigt, der ein „normales“ Taktsignal aufteilt und nacheinander auf fünf unterschiedlichen Kabeln ausgibt. Diese können dann an die AND-Gates geschlossen werden und auf diese Weise die einzelnen Bits ein- und ausgeben.

Dieses Verfahren zu Einspeisung von Daten wurde von mir entwickelt. Wie ich später genauer erläutern werde, besitzt es im Vergleich zu anderen Verfahren eindeutige Vorteile.

c) Shift-Register

Es gibt mehrere Möglichkeiten, den dargestellten Schaltkreis zu konstruieren. Ich habe mich für die Verwendung eines sog. Shift-Registers entschieden.[15] Es ist in der Lage, Bits durch mehrere Instanzen, hier Flip-Flops, laufen zu lassen. Schaltet man fünf dieser Instanzen hintereinander und „shiftet“ eine 1 in die erste, so erhält man mit jedem Taktimpuls einen HIGH-Pegel an dem entsprechenden Ausgang der jeweiligen Instanz. Diese können dann mit den entsprechenden Eingängen der AND-Gates verbunden werden.



27) Shift-Register

Da dieser Schaltkreis in der Lage ist, auf dasselbe wiederkehrende Signal mit unterschiedlichem Verhalten zu reagieren, handelt es sich zwangsläufig um ein Schaltwerk. Somit ist auch die Verwendung von Speicherbausteinen bzw. Flip-Flops offensichtlich. Dabei werden die bereits beschriebenen getakteten D-Flip-Flops verwendet. Schaltet man diese so in eine Reihe zusammen, dass der Datenausgang des ersten in den Dateneingang des zweiten Flip-Flops und der Ausgang des zweiten Flip-Flops in den Eingang des dritten und so weiter führt, so erhält man ein einfaches Shift-Register. Führt man nun alle Takteingänge zusammen und legt ein Taktsignal an, so kann man Bits von einem zum andern Flip-Flop verschieben. Mit jedem HIGH-Pegel sollte so das Bit eine Instanz weiter „geshiftet“ werden.

d) Probleme mit dem Shift-Register

Dieser in vielen Sachbüchern und häufig auch im Internet (etwa bei Wikipedia) zu findende Aufbau würde in der Praxis jedoch nicht funktionieren, wie sogleich erklärt werden wird. Da eine Alternative zu dieser Beschreibung von Shift-Registern praktisch nicht zu finden ist, war die Konstruktion dieses Bauteils besonders aufwendig.

Angenommen, man wollte einen HIGH-Pegel oder eine 1 durch das Shift-Register shiften, dann würden man zunächst eine 1 an den Eingang des ersten Flip-Flops anlegen. Gleichzeitig würde man einen HIGH-Pegel auf die Taktleitung geben. Dadurch würde das erste Flip-Flop

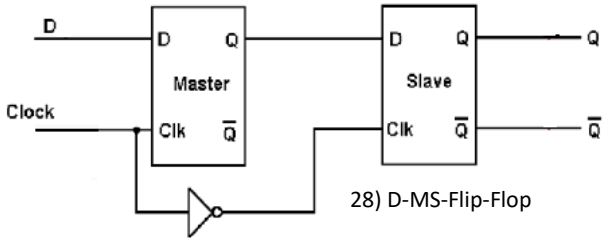
[15] Schnabel, Patrick: Elektronik-Fibel, Selbstverlag, 2014, S. 314 ff.

gesetzt werden. Es würde den Status 1 in sich aufnehmen und unmittelbar danach diesen an seinen Ausgängen wieder ausgeben. Gleichzeitig mit dem ersten Flip-Flop würden jedoch auch alle anderen Flip-Flops aktiviert werden. So würde das zweite Flip-Flop in dem Moment, in dem das erste Flip-Flop den HIGH-Pegel an seinem Ausgang einstellt, beginnen, diesen ebenfalls in sich aufzunehmen. Unmittelbar danach wäre auch das zweite Flip-Flop gesetzt und würde nun ebenfalls den HIGH-Pegel an seinem Datenausgang ausgeben. Dieser Prozess würde sich an allen nachfolgenden Flip-Flops wiederholen. So hätten alle Flip-Flops in einem Bruchteil einer Sekunde den Status des Eingangs übernommen. Theoretisch läuft der HIGH-Pegel zwar von Flip-Flop zu Flip-Flop, dabei ist er jedoch extrem schnell und vollkommen unkontrolliert.

Das Shift-Register muss also aus Instanzen bzw. Flip-Flops aufgebaut werden, die ihren Status erst bei erneutem Einkehren eines LOW-Pegels auf der Taktleitung ausgeben. So kann verhindert werden, dass die nachfolgenden Flip-Flops den Status zu früh übernehmen und so ein unkontrolliertes und schnelles Durchschalten verursachen.

e) Das Daten-Master-Slave-Flip-Flop

Ein Flip-Flop, das diesen Anforderungen genügt, ist das sog. D-Master-Slave-Flip-Flop oder kurz D-MS-FF.[16] Es besteht aus zwei getakteten D-Flip-Flops und einem NOT-Gate. Die Anordnung der beiden Flip-Flops entspricht der Anordnung in einem Shift-Register. Der Takteingang des zweiten Flip-Flops ist jedoch bedingt durch das NOT-Gate invertiert. Liegt nun eine 1 an Takt- und Datensignal des ersten Flip-Flops an, so ist dieses aktiv und übernimmt den Status. Da es durch das invertierte Taktsignal deaktiviert ist, übernimmt das zweite Flip-Flop den Status des ersten Flip-Flops vorerst nicht. Erst wenn das Taktsignal wieder auf LOW sinkt, wird der Status in das zweite Flip-Flop übernommen und so am Ausgang des D-MS-Flip-Flop ausgegeben. Ein „Durchfliegen“ der Bits durch das Shift-Register wird verhindert.



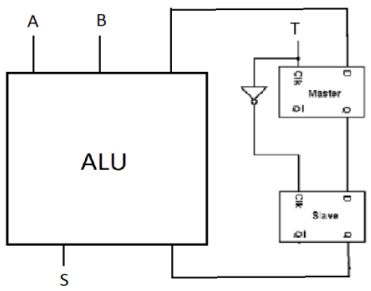
28) D-MS-Flip-Flop

[16] Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 98

Das Shift-Register benötigt jedoch noch eine weitere Modifikation, um plangemäß zu funktionieren. Da der HIGH-Pegel an den Ausgängen der einzelnen Instanzen unmittelbar nacheinander von einer Instanz zur nächsten springt, die ALU jedoch Zeit benötigt, um ihre Ausgänge den Eingängen anzugleichen, würden auf diesem Weg die Ausgangs-Flip-Flops den Status annehmen, der eigentlich für das vorherige Flip-Flop bestimmt ist. Das würde zu einem vollkommen falschen Ergebnis führen. Man benötigt also eine Art Puffer, welcher der ALU die nötige Zeit gibt, ihre Ausgänge den Eingängen anzugleichen. Diesen Puffer stellt der von mir so gennannte „Takt-AND“ dar. Er wurde von mir genau für diese Anwendung entwickelt. Der Takt-AND nimmt die Ausgänge des Shift-Registers und bringt sie in eine AND-Verknüpfung mit dem Taktsignal. So sind die AND-Gates an den Ein- und Ausgabe-Flip-Flops nur dann aktiv, wenn das Taktsignal einen HIGH-Pegel trägt. Ist das Taktsignal LOW, so sind alle Anschlüsse an die ALU deaktiviert.

f) Übertrag-Speicher

Um Zahlen richtig addieren zu können, ist es schließlich erforderlich, den Übertrag einer Rechnung zwischenspeichern zu können. Der Zwischenstand muss zunächst am Carry-Out Ausgang der ALU abgegriffen werden, um anschließend über den Carry-In Eingang wieder in die ALU eigegeben werden zu können. Ein einfaches, getaktetes D-Flip-Flop würde hier zu Fehlern führen. Wie bereits zuvor im Zusammenhang mit dem Shift-Register dargestellt, gleicht es seine Ausgänge beinahe direkt den Eingängen an. Man benötigt also einen alternativen Speicherbaustein. Auch hier lässt sich das D-MS-Flip-Flop verwenden.



29) ALU mit Übertrag

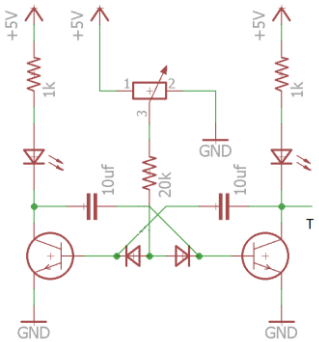
Wird z.B. zweimal eine 1 in die ALU eingegeben, so liegt am Übertrag der ALU ebenfalls eine 1 an. Die Summe der ALU gibt entsprechend 0 aus. Der Übertrag wird anschließend in dem Übertrag-Flip-Flop zwischengespeichert. Wäre ein herkömmliches Flip-Flop verbaut, so würde dieses den Übertrag sofort wieder in die ALU eingeben, zusätzlich zu dem Übertrag die

Summe auf 1 setzten und so zu einem falschen Ergebnis führen. Das D-MS-Flip-Flop hingegen hält seinen Ausgang zunächst auf LOW und gibt die gespeicherte 1 erst bei einem LOW-Pegel des Taktsignales aus. Auf diese Weise wird eine „doppelte Berechnung“ verhindert.

g) Das Taktsignal

Das Taktsignal meines Rechners wird mit Hilfe einer sog. astabilen Kippstufe generiert.[17] Ihr Aufbau ist beinahe identisch mit der der bereits zuvor dargestellten bistabilen Kippstufe. Sie besitzt jedoch anstelle der zwei Basiswiderstände zwei Kondensatoren. Die Basiswiderstände werden in diesem Aufbau verschoben und sind hier jeweils mit 5 V verbunden. Schließt man die Schaltung an eine Spannungsquelle (5 V) an, so schaltet ein Transistor bedingt durch kleine Produktionsunterschiede und den daraus resultierenden Basisstrom seinen Kollektoreingang gegen Masse. Dabei wird der an dem Kollektoreingang liegende Kondensator geladen. Erreicht dieser eine gewisse Schwellspannung (meist 0,7 V), so schaltet der Transistor, dessen Basis mit dem Kondensator verbunden ist, seinen an dem Kollektoreingang liegenden Kondensator gegen Masse und stoppt somit den Basisstrom durch den anderen Transistor. Gleichzeitig wir der Kondensator, der an Masse liegt, geladen, bis er ebenfalls die Schwellspannung erreicht und somit ein erneutes „Kippen“ verursacht. Dabei werden abwechselnd beide Kondensatoren geladen und entladen und generieren so ein Taktsignal an den Kollektoreingängen der Transistoren.

Um die Periodenlänge der Kippstufe und damit die Frequenz des Taktsignales einstellen zu können, besitzt die von mir verwendete Kippstufe einen variablen Widerstand in Form eines Potentiometers. Mit zwei 10 Mikrofarad (µF) Kondensatoren und einem 20K Ω-Potentiometer lassen sich so Frequenzen von 1,3 bis 4,5 Hz einstellen.

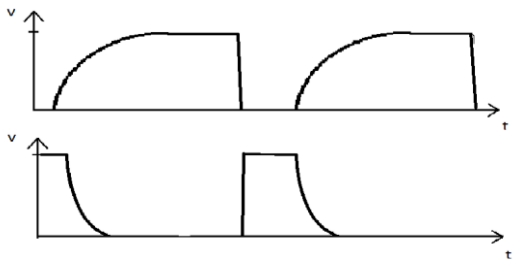


30) Astabile Kippstufe mit Potentiometer

[17] http://www.homofaciens.de/technics-base-circuits-multivibrator_ge_navion.htm (29.02.2016)

Da beide Basisanschlüsse der Transistoren zu einem gemeinsamen Widerstand zusammengeführt werden, benötigt man zwei Dioden, die ungewollte Wechselwirkungen zwischen den Transistoren unterbinden.

Idealerweise sollten sich der Wechsel zwischen 0 V und 5 V bzw. 5 V und 0 V ohne Verzögerung vollziehen. Da jedoch vor allem die Ladekurve der Kondensatoren ein begrenztes Wachstum beschreibt, wäre eine saubere Flanke nicht gegeben. Darum habe ich nach langer Fehlersuche entschieden, das Taktsignal durch ein NOT-Gate zu invertieren. So ist die Entladekurve, die wegen des extrem kleinen Widerstands des Transistors vergleichsweise steil ausfällt, die steigende Flanke des Taktsignales. Da die fallende Flanke weniger kritisch ist, ruft die exponentielle Abnahme als fallende Flanke deutlich weniger Fehler hervor. Zusätzlich dazu wird das Taktsignal mit Hilfe zweier NOT-Gates nochmals geglättet, und die Kanten werden steiler gemacht. Diese Glättung lässt sich auf den enormen Stromvertärkungsfaktor der Transistoren zurückführen.



31) Vergleich: Taktsignal vor und nach Filterstufe

h) Manueller Takt und Taktselekter

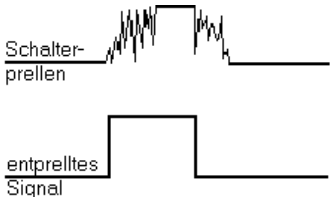
Um den Rechner leichter „debuggen“ und um seine Funktionsweise einfacher erklären bzw. vorführen zu können, besitzt er einen manuellen Taktknopf. Pro Knopfdruck wird eine Taktperiode simuliert, und es werden alle entsprechenden Funktionen ausgeführt. Wird z.B. ein Teil des Rechners erklärt, so kann der Rechenvorgang an beliebige Stellen angehalten werden.

Wird der Taktknopf gedrückt bzw. losgelassen, so kommt es bedingt durch suboptimale Eigenschaften des Knopfes und durch leichtes Zittern der drückenden Person zum sog. „bouncen“. Dabei springt der Kopf einige Male zwischen einem geöffneten und geschlossenen Zustand hin und her. Um diesen unkontrollierten Wechsel zwischen HIGH und LOW am

Taktsignal und die dadurch hervorgerufenen Fehler zu unterbinden, muss das aus dem Knopf kommende Taktsignal gefiltert bzw. „debouncet“ werden.[18]

Da es sich um hochfrequente Schwingungen handelt, kann ein Tiefpass-RC-Filter an das Taktsignal angelegt werden.[19] Dieser setzt sich in meinem Rechner aus dem Innenwiderstand der Kabel und einem 100 Nanofarad Kondensator zusammen. Der Wert des Kondensators darf bedingt durch die unsaubere Ladung bzw. Entladung nicht zu groß gewählt werden. Trotzdem ist diese Anordnung in der Lage, Spannungsspitzen aus dem Taktsignal herauszufiltern.

Damit die Taktflanken weiterhin steil verlaufen, wird das Taktsignal durch ein D-Flip-Flop ohne Takteingang geführt. Das Flip-Flop verhält sich wie ein Verstärker. Wegen der Trägheit der Transistoren kommt es zu einer weiteren Filterung bzw. Glättung des Taktsignales.



32) Debouncing von Taktsignalen

Damit zwischen automatischem und manuellem Takt umgeschaltet werden kann, besitzt der Rechner zwei Knöpfe. Jeweils einer ist mit einem der Eingänge eines weiteren RS-Flip-Flop verbunden. Die Ausgänge sind jeweils mit einem AND-Gate verbunden, in das außerdem entweder der manuelle oder der automatische Takt führt. Anschließend werden die beiden Ausgänge über Dioden zusammengeführt und dem Shift-Register bzw. dem Übertrags-Speicher zur Verfügung gestellt. Je nachdem, ob das Flip-Flop gesetzt ist, wird entweder der manuelle oder der automatische Takt in den Rechner gegeben.

Der Aufbau des Rechners ist nun fast komplett. Damit die Berechnung zweier Zahlen gestartet werden kann, müssen diese zuerst mit Hilfe der Knöpfe, natürlich in Binärkodierung, in die beiden Eingangsregister eingegeben werden. Anschließend wird eine 1 bzw. ein HIGH-Pegel an den Eingang des Shift-Registers gelegt. Sobald das Taktsignal einen HIGH-Pegel führt, wird die 1 in das Shift-Register übernommen. Fällt das Taktsignal anschießend wieder auf LOW, so gibt die erste Instanz des Shift-Registers eine 1 aus und aktiviert somit die ers-

[18] http://www.ganssle.com/debouncing-pt2.htm (29.02.2016)
[19] Schnabel, Patrick: Elektronik-Fibel, Selbstverlag, 2014, S.215

ten 3 AND-Gates. Die LSBs werden berechnet und es werden das Ergebnis und der Übertrag ab- bzw. zwischengespeichert. Erfährt das Taktsignal einen weiteren LOW-HIGH- und HIGH-LOW- Übergang, wird das nächst kleinere Bit berechnet.

i) Rechenstarter

Hat das Taktsignal bedingt durch den automatischen Taktgenerator eine relativ hohe Frequenz, so ist es schwierig, den richtigen Moment für das „Einshiften“ einer 1 in das Shift-Register abzupassen. Außerdem würde mit großer Wahrscheinlichkeit mehr als nur eine 1 in das Shift-Register eingegeben und so ein falsches Ergebnis generiert werden.

Um das Einshiften zu vereinfachen, setzt man beim Aktivieren meines Rechners ein speziell dafür angelegtes RS-Flip-Flop. Da es die Berechnung der Zahlen beginnt, wurde es von mir als „Rechenstarter“ bezeichnet. Sein q-Ausgang ist mit dem Eingang des Shift-Registers verbunden. Außerdem ist der Reset-Eingang des Flip-Flops mit dem Ausgang der ersten Instanz des Shift-Registers verbunden.

Wird das Flip-Flop gesetzt und dadurch eine 1 in die erste Instanz des Shift-Regsiters übernommen und anschießend wieder ausgegeben, so wird der Rechenstarter bedingt durch die Verbindung mit dem Ausgang der ersten Instanz des Shift-Registers zurückgesetzt. Das Einshiften von mehr als nur einer 1 kann so verhindert werden.

Der Rechner ist nun komplett und einsatzbereit.

IV. Namensgebung

Die Bezeichnung „Serieller 4-Bit Binär-Addierer“ setzt sich aus mehreren Teilen zusammen.

Der Zusatz „Seriell“ bezieht sich auf die im Vergleich zum Ripple-Carry-Adder zeitlich verzögerte und somit serielle Abarbeitung der einzelnen Stellen. Der Ripple-Carry-Adder hingegen stellt die Ergebnisse seiner Rechnungen nahezu direkt an seinen Ausgängen dar.

Es handelt sich um einen 4-Bit Binär-Addierer, weil er zwei 4-Bit Binärzahlen miteinander addieren kann.

V. Vergleich mit existierenden Binärrechnern

Nach weiteren Online-Recherchen fand ich auf Wikipedia unter diesem Namen einen ähnlich konfigurierten Rechner.[20] Dieser Rechner nutzt ebenfalls eine 1 Bit-ALU, um Berechnungen von mehrstelligen Zahlen durchzuführen. Diese werden jedoch aus Shift-Registern generiert, die aus D-Flip-Flops konstruiert werden sollen. Auch einen Übertragsspeicher in Form eines (vermutlich) taktflankengesteuertem D-Flip-Flops ist bei diesem Rechner vorhanden. Wie bereits dargelegt, ist eine Realisierung mit solchen Komponenten weder sinnvoll noch praktisch umsetzbar. Auch unter der Verwendung geeigneter Bauteile ist der Hardwareaufwand dieser Architektur im Vergleich zu meiner beinahe doppelt so hoch.

Zudem kann bei meinem Rechner, wenn man größere Zahlen addieren möchte, anstelle einer 1 Bit- eine 2 Bit-ALU verbaut werden. Die Länge des Shift-Registers und die für die Berechnung benötigte Zeit bleiben auf diese Weise trotz doppelter Anzahl der zu berechnenden Stellen konstant. Eine vergleichbar „hardwareschonende“ Erweiterung des online dargestellten Rechners wäre nicht möglich.

VI. Resümee und Ausblick

Der Bau meines Rechners war ein äußerst langwieriger Prozess. Von den ersten Ideen bis zu einem fertigen und voll funktionsfähigen Rechner vergingen gut 13 Monate. Während dieser Zeit war ich mehr als nur ein Mal komplett frustriert und hatte eigentlich mit dem Projekt abschlossen. Vor allem die großen Probleme, z.B. rund um das Shift-Register, bereiteten mir viel Ärger und Verzweiflung. Besonders die Tatsache, dass Theorie und Praxis nicht immer hundertprozentig übereinstimmen, war eine große Hürde.

Häufig gibt es genaue Ausarbeitungen der theoretischen Zusammenhänge und Konzepte. Gleichzeitig werden einfache praktische Schaltkreise sehr ausführlich beschrieben. Der Übergang zwischen theoretischen Konzepten und praktischer Umsetzung für komplexere Schaltkreise wird jedoch in den seltensten Fällen erläutert. Werden größere Schaltkreise konstruiert, so machen sich die nicht idealen Eigenschaften der elektronischen Komponenten bemerkbar. Diese waren bei einfach ausgeführten Schaltkreisen zuvor nicht von Rele-

[20] https://de.wikipedia.org/wiki/Addierwerk (29.02.2016)

vanz. Es ist somit eine große Herausforderung, komplexe Schaltkreise aus einfachen Bauteilen zu konstruieren.

Doch wie sich gezeigt hat, lohnt es sich auch in harten Phasen eines Projektes, nicht aufzugeben. So lag auch der Rechner bei mir nie länger als drei Wochen unberührt herum. In dieser Zeit schwebten die Gedanken rund um das aktuelle Problem in meinem Kopf herum und ließen mich nicht los. So war unter anderem die Dusche ein äußerst „spiritueller“ Ort, an dem mir immer wieder neue Ansätze und Lösungsmöglichkeiten einfielen. Durch neu erlangte Motivation und die Bereitschaft, neue Ideen zu testen, gelang es mir, Problem für Problem und Bauteil für Bauteil einen voll funktionsfähigen Rechner zu konstruieren. Das Gefühl, das einen bei erfolgreicher Umsetzung eines Lösungsansatzes überkommt, ein Gefühl der grenzenlosen Möglichkeiten, ist diesen Aufwand allemal wert.

Abgesehen von dem rein praktischen und technischen Wissen habe bei dem Bau des Rechners vieles gelernt. Mir ist klar geworden, dass man mit der richtigen Motivation und Unterstützung auch große und zunächst scheinbar unlösbare Problem lösen kann. Außerdem habe ich, unter anderem durch das Verfassen dieser Lernleistung, begonnen, wissenschaftliche Arbeitstechniken anzuwenden. Dies wird mir in meiner weiteren Ausbildung und bei weiteren Projekten helfen.

Und so steht auch schon das nächste größere Projekt in den Startlöchern: Ich möchte einen frei programmierbaren 8-Bit Rechner bauen. Dieser soll einen sog. „Hauptbus“ besitzen und in der Lage seien, einfache Programme auszuführen. Er wird separate Speicherbausteine, einen „Instruction Decoder“ sowie eine multifunktionale ALU besitzen. Doch dazu später mehr …

Ich möchte mich an dieser Stelle bei meinen Tutoren, Herrn Fritz und Herrn Plotkin, für ihre fachliche und menschliche Unterstützung bedanken. Auch möchte ich meinen Eltern für die moralische wie finanzielle Unterstützung danken.

VII. Anhang

1) Literaturverzeichnis

Dembowski, Klaus: Mikrocontroller, dpunkt, 2014

Dohlus, Rainer : Physik mit einer Prise Mathe, Springer, 2014

Göbel, Holger: Einführung in die Halbleiter-Schaltungstechnik, Springer, 2014

Meister, Irmtraud/Salzburger, Lukas: AVR-Mikrocontroller-Kochbuch, Franzis, 2013

Schnabel, Patrick: Computer-Fibel, Selbstverlag, 2014

Schnabel, Patrick: Elektronik-Fibel, Selbstverlag, 2014

Wöstenkühler, Gerd: Grundlagen der Digitaltechnik, Carl Hanser, 2012

http://www.elektronik-kompendium.de/ (29.02.2016)

https://de.wikibooks.org/wiki/Digitale_Schaltungstechnik (29.02.2016)

https://de.wikipedia.org/wiki/Addierwerk (29.02.2016)

https://en.wikibooks.org/wiki/Microprocessor_Design (29.02.2016)

https://www.youtube.com/watch?v=xISG4nGTQYE (29.02.2016)

2) Bildnachweis

Nr.1:http://de.f-alpha.net/elektronik/digitale-elektronik/flip-flop/los-gehts/experiment-8-master-slave/ (20.02.2016)

Nr.2: Von mir erstellt mit MS-Word

Nr.3: Von mir erstellt mit MS-Paint

Nr.4: Gerd Wöstenkühler: Grundlagen der Digitaltechnik, Carl Hanser, 2012, S. 118

Nr.5: Von mir erstellt mit MS-Paint

Nr.6: https://de.wikipedia.org/wiki/Carry-Ripple-Addierer (20.02.2016)

Nr.7: Von mir erstellt mit MS-Paint

Nr.8:http://www.airnet.de/basic_internetworking1/de/html/Math_learningObject9.xml (20.02.2016)

Nr.9: Von mir erstellt mit MS-Paint

Nr.10: http://www.electronicshub.org/half-adder-and-full-adder-circuits/ (20.02.2016)

Nr.11:https://hhapcomputerscience.files.wordpress.com/2015/09/andortruthtable.gif (21.02.2016)

Nr.12:https://hhapcomputerscience.files.wordpress.com/2015/09/andortruthtable.gif (21.02.2016)

Nr.13:https://hhapcomputerscience.files.wordpress.com/2015/09/not-gate1.png (21.02.2016)

Nr.14: http://cpuville.com/logic_gates.htm (20.02.2016)

Nr.15: http://cpuville.com/logic_gates.htm (20.02.2016)

Nr.16:https://upload.wikimedia.org/wikipedia/commons/5/53/RS_Flip-flop_%28NOR%29.svg (20.02.2016)

Nr.17: http://worldclassprogramme.com/RS-FlipFlop.php (20.02.2016)

Nr.18: Von mir erstellt mit MS-Paint

Nr.19: http://entladung.net/ep004/ (20.02.2016)

Nr.20: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/trans.html (20.02.2016)

Nr.21: Von mir erstellt mit Eagle Lite

Nr.22: Von mir erstellt mit Eagle Lite und MS-Paint

Nr.23: Von mir erstellt mit Eagle Lite

Nr.24: Von mir erstellt mit Eagle Lite

Nr.25: Von mir erstellt mit Eagle Lite

Nr.26: Von mir erstellt mit MS-Paint

Nr.27: Von mir erstellt mit MS-Paint

Nr.28: https://students.cs.byu.edu/~cs224ta/labs/L02-fsm/HowToUseMasterSlave.php

Nr.29: Von mir erstellt mit MS-Paint

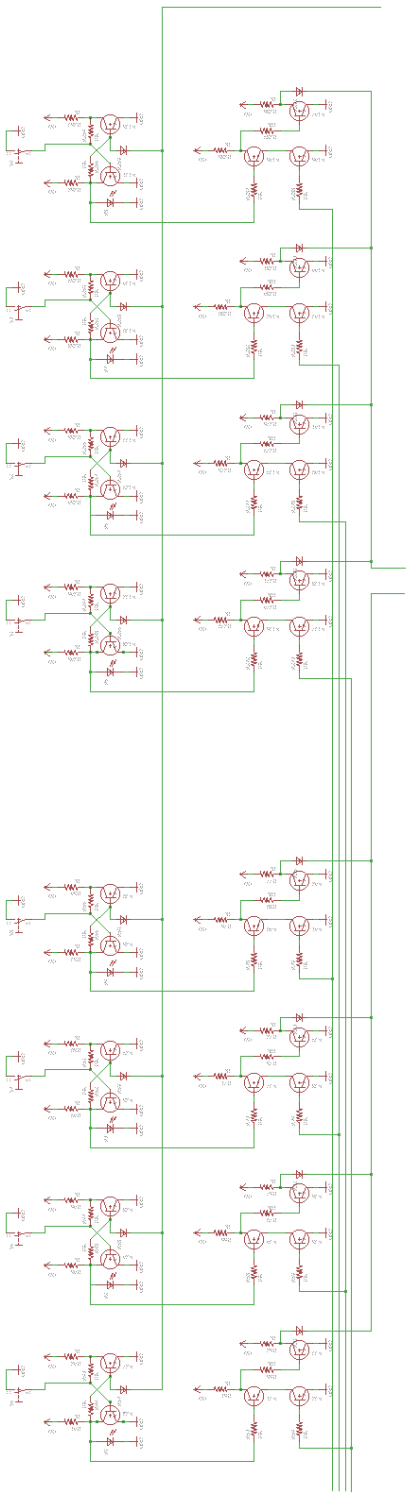
Nr.30: Von mir erstellt mit Eagle Lite

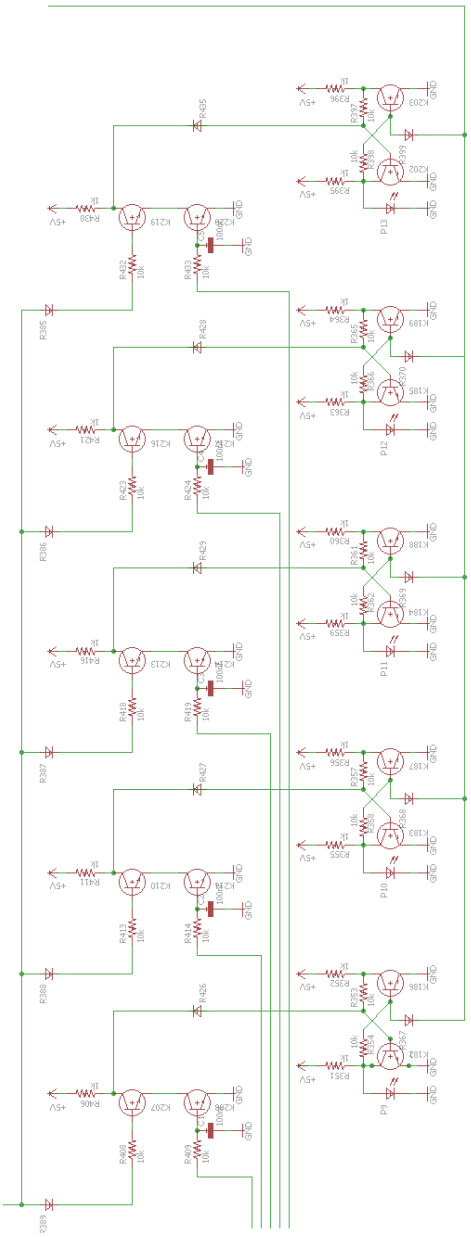
Nr.31: Von mir erstellt mit MS-Paint

Nr.32: http://www.netzmafia.de/skripten/digitaltechnik/flipflop.html

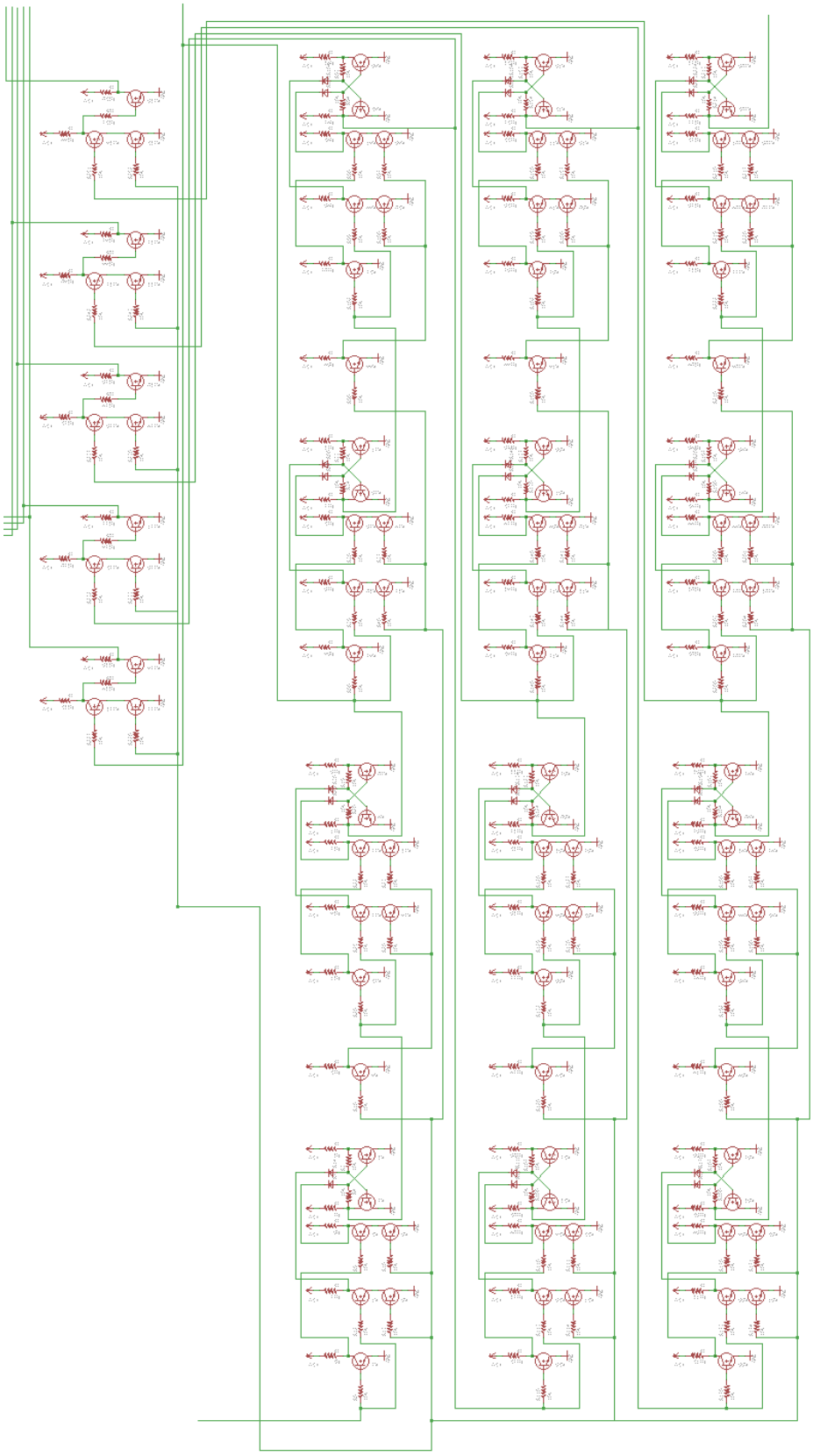
3) Schaltpläne

Input-Register und Ouput-Register mit AND-Gates:

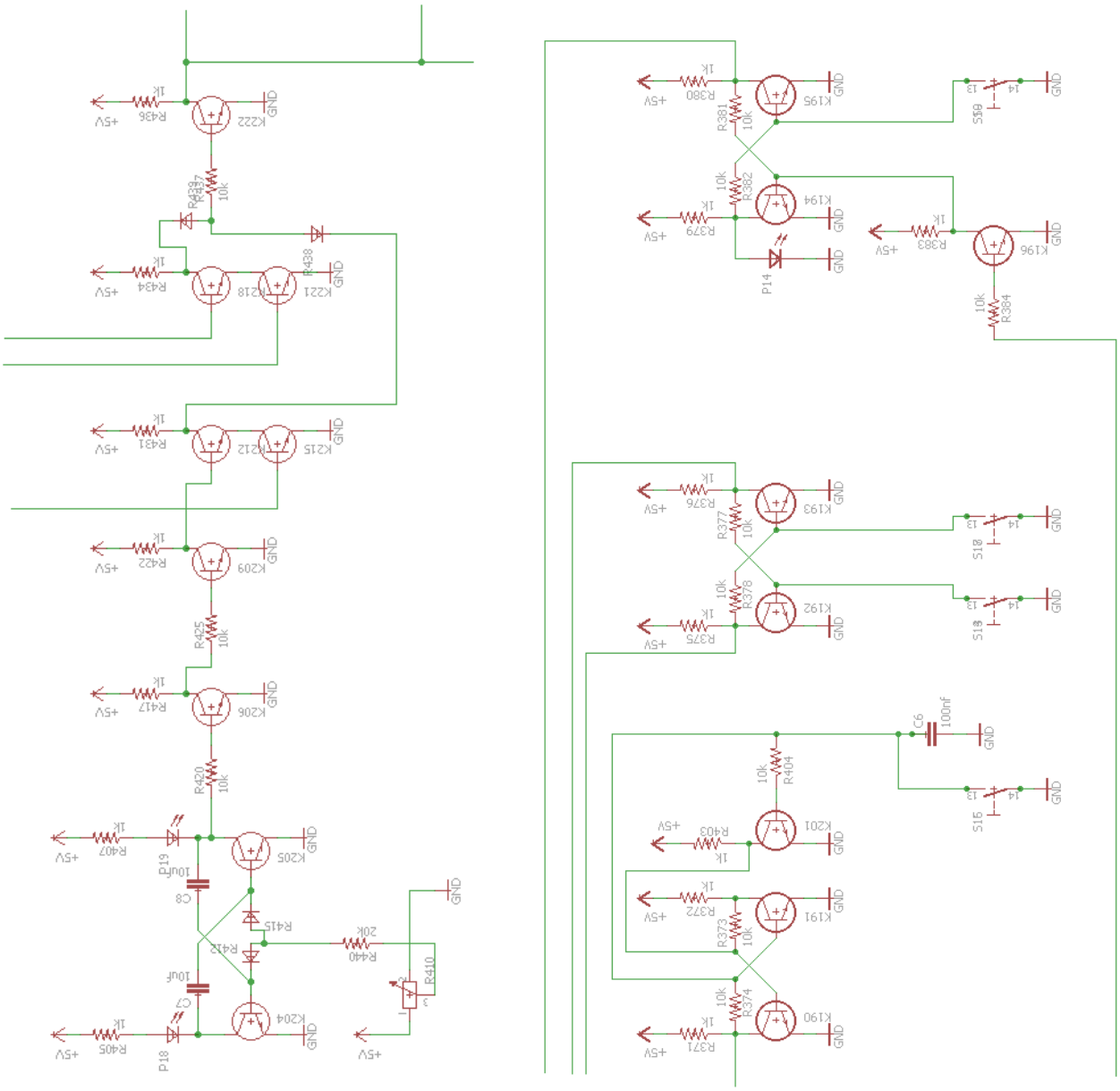




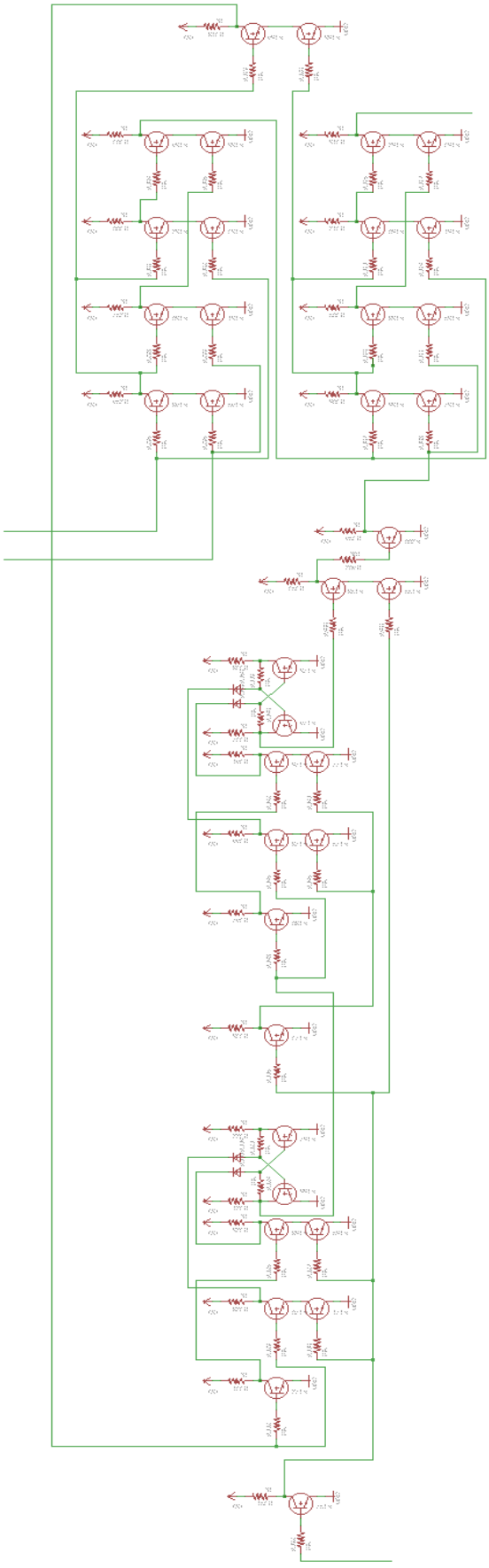
Shift-Register mit Takt-AND:



Taktgenerator und Kontrolleinheit:



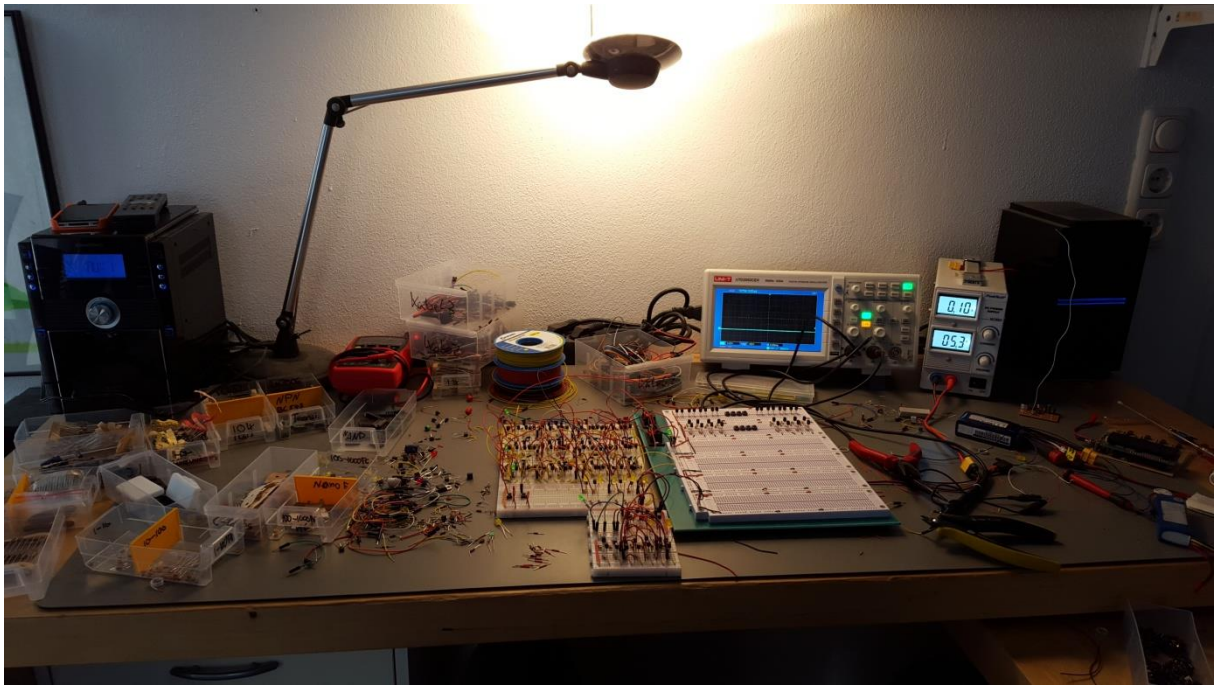
ALU mit Übertragsspeicher:



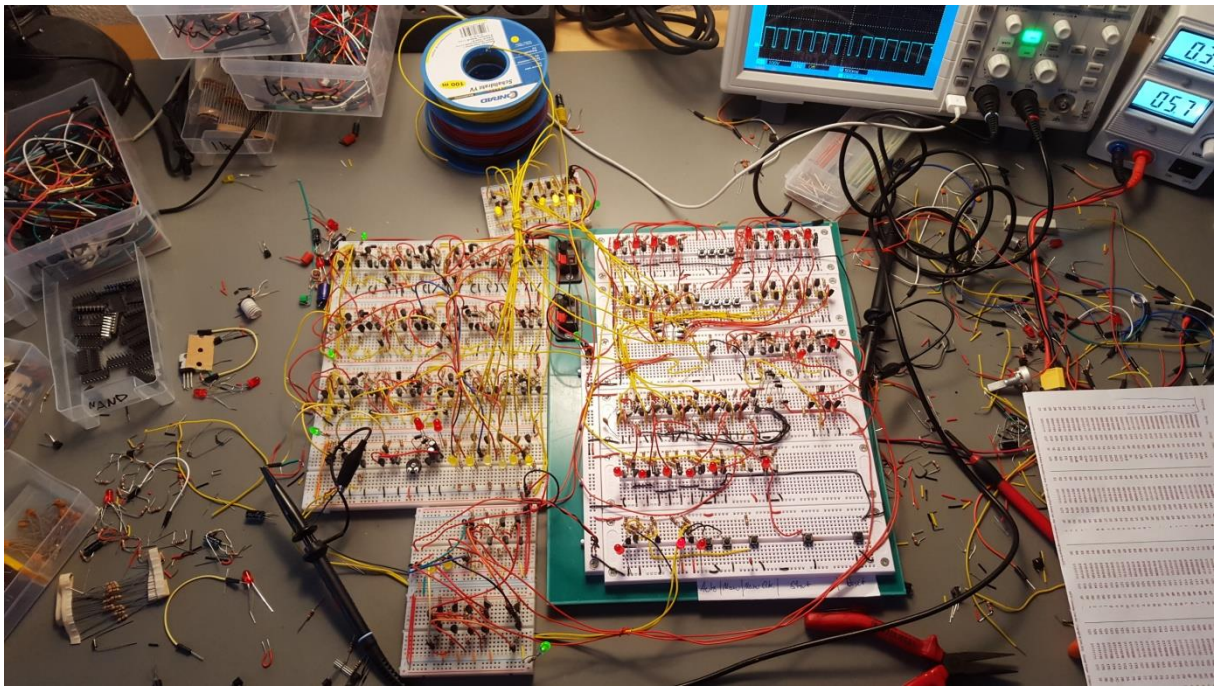
4) Fotos des Rechners

Da der größte Teil der Arbeit in dem Bau des Rechners und der Suche nach damit verbunden Fehlern bestand, finden sich hier ein paar Ausschnitte aus dem Entstehungsprozess und dem fertigen Rechner in Form von Fotos:

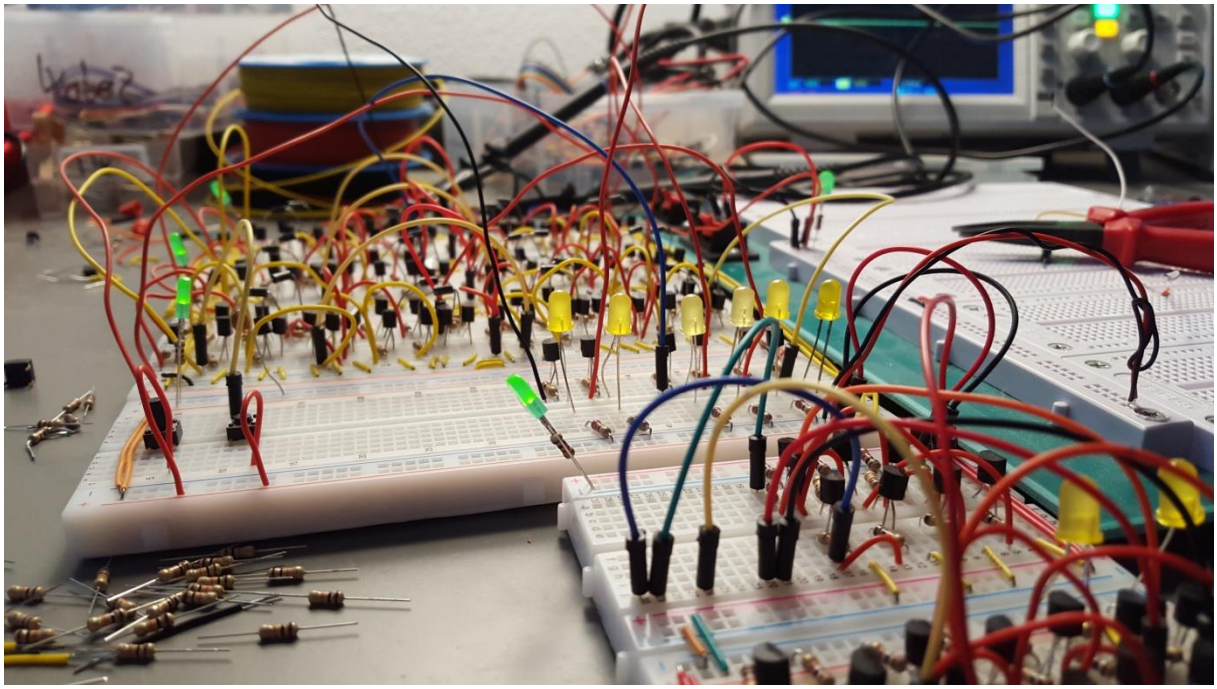
Der Arbeitsplatz mit halbfertigem Rechner:



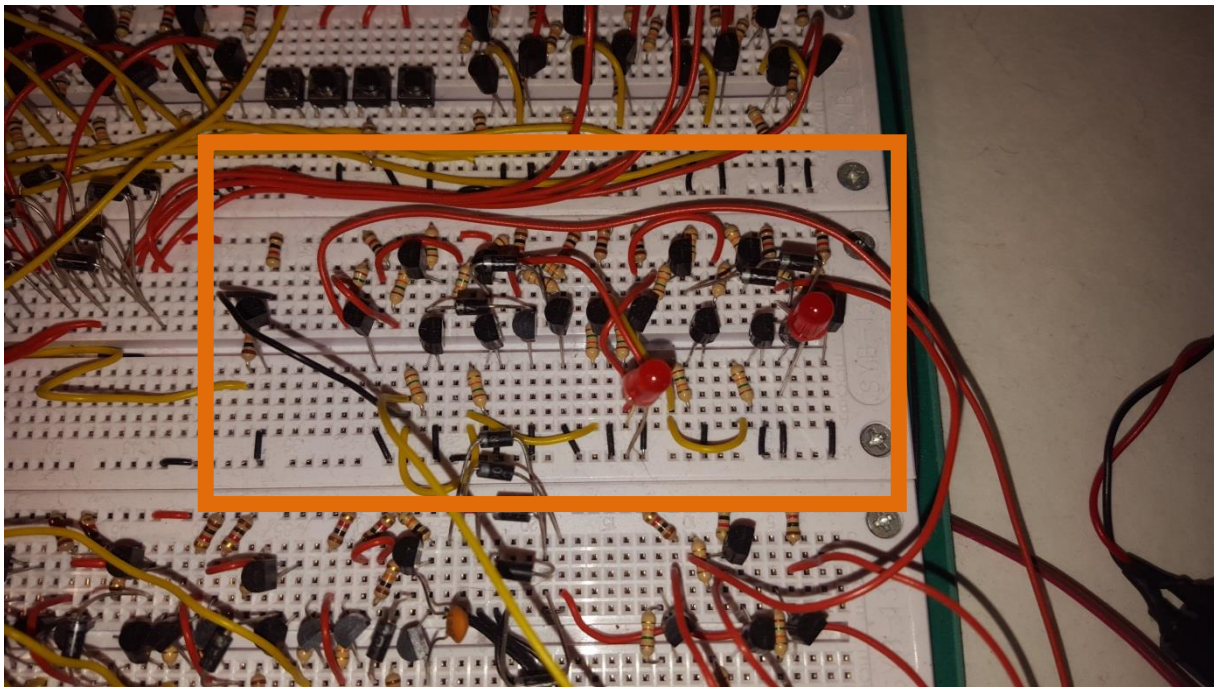
Der Rechner neben Oszilloskop und Spannungsquelle:



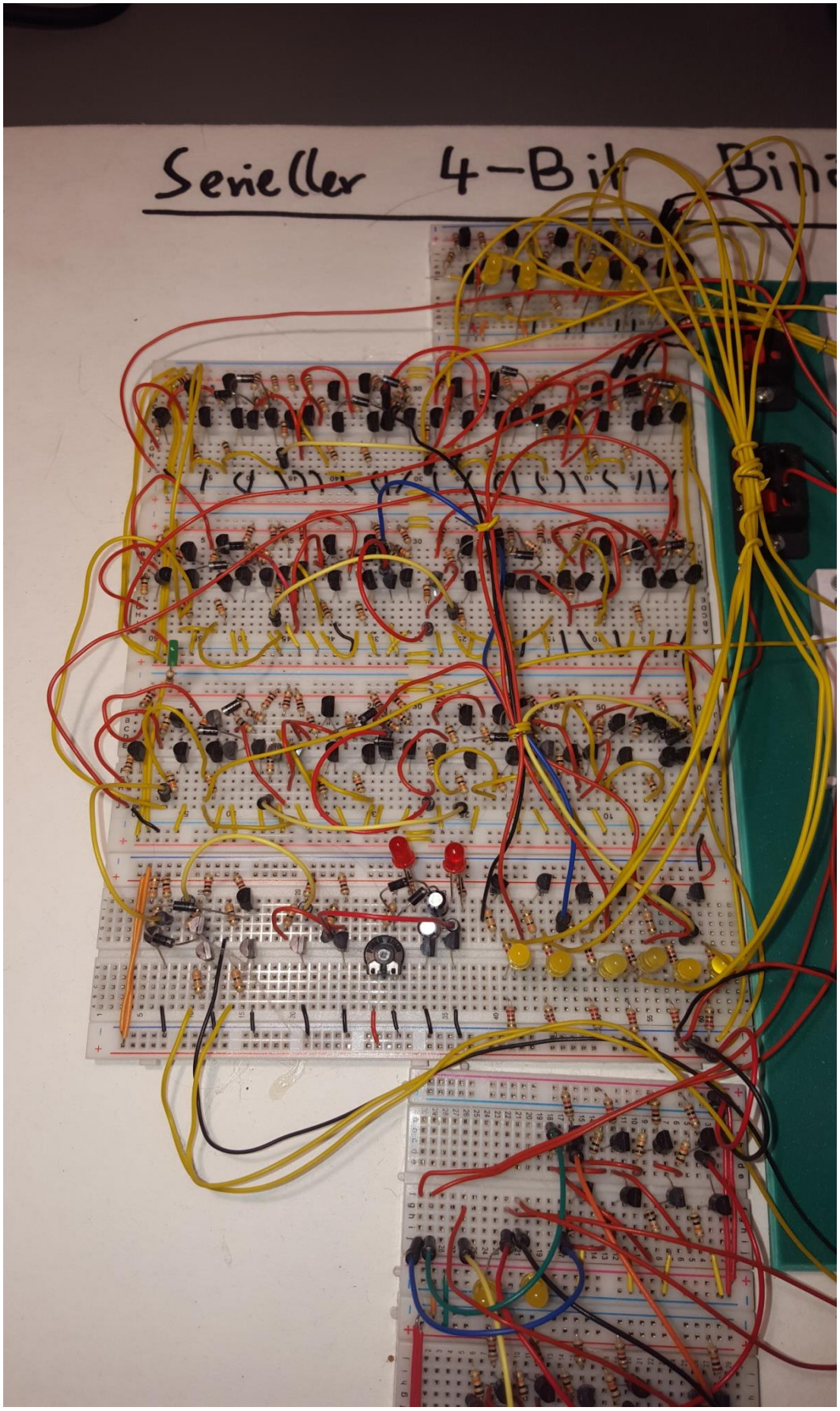
Nahaufnahme der einzelnen Komponenten:



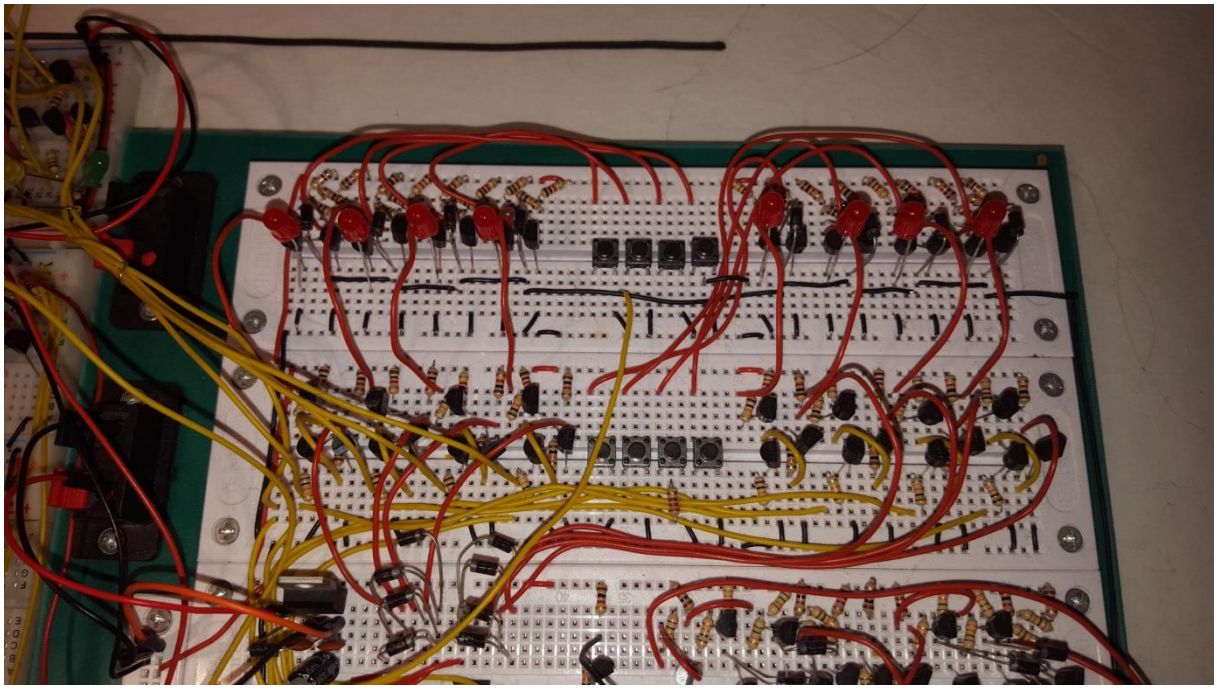
Übertragsspeicher:



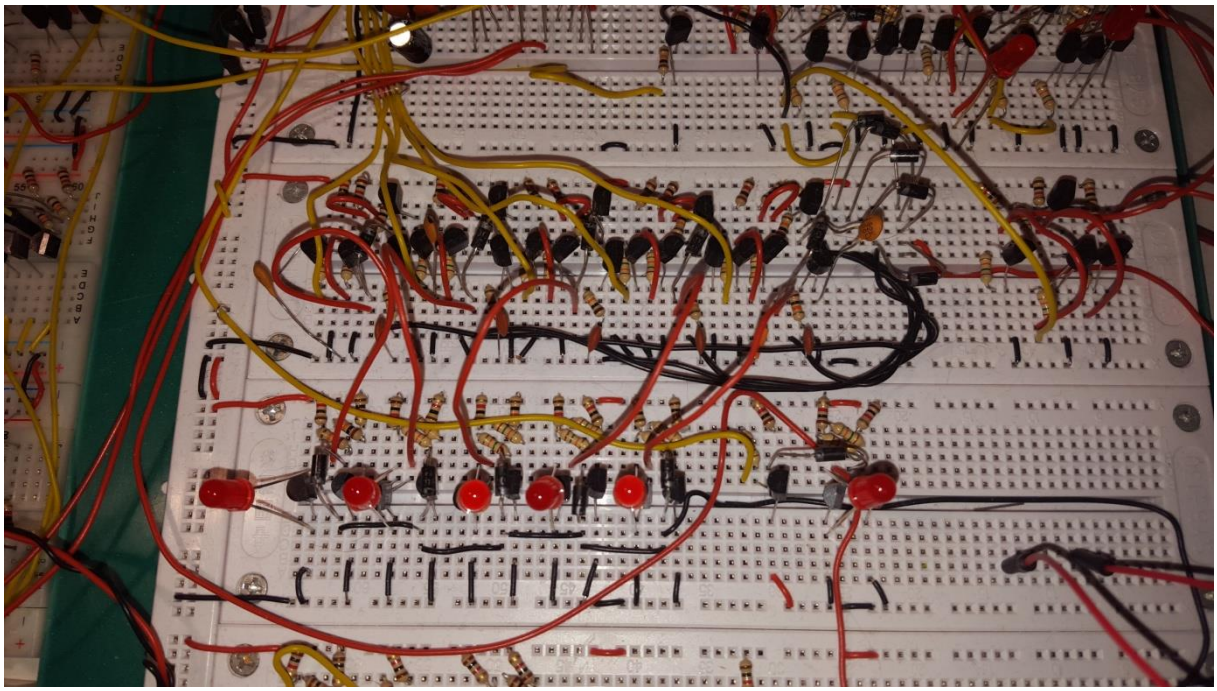
Shift-Register mit Takt-AND oben und ALU unten im Bild:



Input-Register:



Output-Register:



Ich versichere, dass die Präsentation von mir selbstständig erarbeitet wurde und ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Diejenigen Teile der Präsentation, die anderen Werken im Wortlaut oder dem Sinn nach entnommen wurden, sind als solche kenntlich gemacht.

Hamburg, den 01.03.2016

_____

Fabian Peddinghaus