

## Technische Informatik im Theoriekurs

Obwohl Themen der technischen Informatik zu den ältesten Inhalten der Schulinformatik gehören, scheint mir auf diesem Gebiet am wenigsten geklärt, ob und wie das Thema an allgemein bildenden Schulen unterrichtet werden soll. Bei der Hardware von Informatiksystemen handelt es sich genauso um „Denkprodukte“ wie bei der Software. Sie manifestieren sich nur in einem anderen Medium. Auch Hardware ist in hohem Maße abstrakt, und sie ist in der Schule ähnlich wie die Softwareentwicklung ein exzellentes Anwendungsfeld, um über abstrakte Entwurfs- und Modellierungsverfahren zu konkreten Ergebnissen zu kommen. In dieser Hinsicht ist sie ein fast schon ideales Anwendungsgebiet für Projektarbeit, die über die Sensorik und Steuerungsvorgänge auch mit anderen Unterrichtsfächern eng verknüpft werden kann.

Wenn es sich bei der digitalen Elektronik um eine Basistechnologie handelt, auf deren Grundlagen weite Bereiche unserer Gesellschaft beruhen, dann gehört dieses Thema in den Unterricht allgemein bildender Schulen, weil Kenntnisse auf diesem Gebiet zum Verständnis unserer Umwelt notwendig sind. Wir haben einen Fall, der direkt den Anwendungen des Elektromagnetismus vergleichbar ist: Auch hier werden die erforderlichen Grundlagenkenntnisse – im Physikunterricht – vermittelt, und zwar allen Schülerinnen und Schülern! Folglich gehört m. E. die Behandlung einfacher Schaltnetze in die Sek. I und ist damit Thema der ITG. Auch vom Anspruchsniveau sind Probleme wie

*WENN die Alarmanlage eingeschaltet worden ist  
UND es später als 22.00 h ist  
UND die Fensterscheibe zerstört wird,  
DANN FOLGT, dass die bösen Buben kommen.*

eher auf dieser Altersstufe angesiedelt und werden dort seit Jahren erfolgreich unterrichtet. Konsequenterweise können diese Themen dann alleine nicht mehr die „technischen“ Inhalte eines Informatik-Grundkurses bilden. Möglicherweise werden sie auftauchen, weil Defizite der Sek. I aufzuarbeiten sind, aber mehr auch nicht. Der Informatikunterricht der Sek. II würde bei Beschränkung auf solche Themen nur ITG-Aufgaben wahrnehmen, die nicht zu seinem eigentlichen Kern gehören.

Sehen wir uns die beim Schaltungsentwurf meist benutzten Methoden der Schaltalgebra etwas genauer an, dann stellen wir fest, dass die eigentliche Schaltungsentwicklung gar nicht Thema dieser Disziplin ist. Schaltfunktionen werden nach unterschiedlichen Kriterien umgeformt, bis sie bestimmten Anforderungen z. B. der Einfachheit oder der Darstellungsform genügen. Jeder dieser Schaltfunktionen entspricht aber direkt eine Schaltung, deren Aufschreiben eine eher triviale Aufgabe darstellt. Haben wir überhaupt eine Schaltfunktion, die ein Problem löst, dann haben wir auch die entsprechende Schaltung. Diese ist vielleicht zu umfangreich, nicht effizient, ..., aber sie löst das Problem; und da Effizienzkriterien m. E. in allgemein bildenden Schulen nur sehr eingeschränkt Sinn haben, sind die mathematischen Methoden der Schaltalgebra zumindest nicht notwendig, um Schaltungsentwurf in der Schulinformatik zu betreiben. Die Methoden der Schaltalgebra sind deshalb ggf. hilfreich, aber zur Begründung der Existenz der Technik innerhalb des Informatikunterrichts ebenfalls nicht geeignet.

Viel wichtiger als mathematisch begründete Umformungen ist die Frage nach der Existenz einer Schaltfunktion für eine gegebene Problemstellung. Im Informatikunterricht sollte deutlich werden, dass einerseits die aussagenlogische Formulierung einer Problemstellung (WENN *dasunddas gegeben ist* DANN FOLGT *diesunddas*) eine Lösung impliziert, andererseits aber nur ein eingeschränkter Satz von Problemklassen auf diese Art behandelbar ist: Gerade bei den „interessanten“ Fragen lässt sich „*dasunddas*“ und „*diesunddas*“ nicht aussagenlogisch scharf fassen. Standardbeispiel für solch einen „unscharfen“ Begriff ist die Größe eines Menschen: Sicherlich ist ein 1,95 m-Mensch „groß“, und ebenso sicher ist ein 1,55 m-Mensch „klein“. Wie aber sind 1,80 m einzuschätzen? Als „ziemlich groß“?<sup>1</sup> Die Existenz anderer als aussagenlogischer Methoden (Fuzzy-Logik, neuronale Netze) zeigt sehr deutlich, dass mithilfe der Aussagenlogik formulierbare Probleme noch nicht einmal die Bereiche vollständig abdecken, in denen schon heute technisch „hart“ gearbeitet (gesteuert, geregelt, ... ) werden soll: Wir brauchen gar nicht die „unscharfen“ Begriffe der Geisteswissenschaften wie „Schönheit“, „Harmonie“, ... zu bemühen; auch die Steuerung eines Brennofens oder einer Videokamera geht oft über die Möglichkeiten der traditionellen Aussagenlogik hinaus<sup>2</sup>. Im Hardwareunterricht können also die Möglichkeiten und Grenzen einer Methode erfahren werden, die zwar für einige der so gern behandelten „klar formulierten“ Probleme wunderbar funktioniert, für andere aus dieser Klasse aber schon nicht mehr; und die Grenzen der Technik sind ein zentrales Thema der Informatik: sei es bei den Algorithmen, sei es an dieser Stelle. Im Gegensatz zu den Berechenbarkeits- und Entscheidbarkeitsproblemen handelt es sich bei der Schaltungsentwicklung aber um sehr handfeste Probleme, so dass die Grenzen der Computer nicht in (für die Schülerinnen und Schüler) esoterische Bereiche verschoben werden.

Die zentrale Aufgabe des Hardwareunterrichts ist es m. E. zu zeigen, wie technische Komponenten Eigenschaften erwerben, die sonst eher dem geistigen Bereich zugeordnet werden – und dafür ein gültiges Modell zu entwickeln: wie also eine Schaltung (in vorgegebener Weise) sinnvoll auf unterschiedliche Situationen reagieren kann, Entscheidungen trifft, sich etwas merkt oder programmiert werden kann.<sup>3</sup> Begreifen wir digitale Systeme als ein System von „Schaltern“, dann reduziert sich unser Problem auf die Frage: „Wer bedient die Schalter?“ Nun kann man schnell zeigen, dass einerseits Schalter von außen bedient werden (durch „Sensoren“, Menschen, ...), andererseits Schaltersysteme auch auf sich selbst zurückwirken können. Wir kommen dadurch zu einer Konkretisierung des Begriffs des **Zustands** des Systems, der z. B. als eine Art Gedächtnis (Speicher, ...) fungieren kann. Die Abfolge solcher Zustände beschreibt dann die Reaktionen des Systems; und wenn unterschiedliche Abfolgen möglich sind, dann kann die Schaltung auch auf unterschiedliche Weise reagieren. Beschreiben wir die Zustände und deren Änderungen auf eine geeignete, systematische Art und zeigen, wie die so gefundene Beschreibung wiederum systematisch in eine Schaltung umgesetzt werden kann, dann haben wir unsere Aufgabe gelöst: Wir können beliebige (auch programmierbare) Schaltungen entwerfen.

Es kommt dabei nicht so sehr auf die Komplexität der Ergebnisse als auf das systematische Vorgehen an: Für die Schule (und die Schülerinnen und Schüler) ist es „normal“, mächtige Verfahren anhand einfacher Beispiele kennen zu lernen und einzuüben. Technische Informatik darf sich damit nicht auf „Kochrezepte“ beschränken,

<sup>1</sup> Die Fragestellung ist direkt auf Sensorenabfragen, wie sie zur Steuerung von Schaltungen benutzt werden, übertragbar.

<sup>2</sup> Was z. B. ist eine „scharfe“ Bremsung?

<sup>3</sup> Ich meine also nicht eine technische Variante des Leib-Seele-Problems, sondern „harte“ Computertechnik.

die zeigen, dass eine vorgegebene Schaltung funktioniert (z. B. ein FlipFlop aus NANDs). Vielmehr müssen Methoden entwickelt werden, die zeigen, wie man zu einer Lösung kommt. Als systematisches Verfahren zum Schaltungsentwurf bieten sich die Notationsformen der Automatentheorie und die dazu passenden Methoden an. Hier liegt auch die enge Verbindung zur theoretischen Informatik, für die der Technikkurs eine Art „projektorientierte Vorstufe“ bilden sollte. Entsprechend betont die technische Informatik *die gleichen fundamentalen Ideen* wie in der Theorie: (hier: konkrete) Automaten mit ihren Zuständen, Berechenbarkeit und Sprachen, die die Automaten steuern, manifestieren sich zwar anders, aber in vergleichbarer Weise<sup>4</sup>. Strukturierte Zerlegung und Algorithmisierung – mit ihren „Unterideen“ – zeigen sich beim Entwurf von Schaltungen sowie im Bereich der Simulation.

Wichtig erscheint mir die Erfahrung, dass quantitative Änderungen der Zahl der Schalter in einer Schaltung qualitative Änderungen bewirkt: Mit einem kann man eine Glühbirne „schalten“, mit zwei oder drei Schaltern eine Alarmanlage bauen, mit weniger als zehn ein Treibhaus steuern, mit einigen Dutzend einfache Rechnungen durchführen, mit Hunderten einen Taschenrechner bauen und mit Tausenden, Millionen, ... ? Zu jeder Größenordnung von Zahlen gehört eine andere Problemklasse, die damit gelöst werden kann. Miniaturisierung ist nicht nur ein technisches Problem.

Ich habe mit Absicht die sehr „tiefen“ Ebenen des Schaltungsentwurfs zuerst besprochen, obwohl mir natürlich bewusst ist, dass man auf diese Art keinen Computer entwickeln kann. Zur Beschreibung derart komplexer Systeme gehören die Benutzung von Blockschaltbildern, das Denken in Komponenten und die Beschreibungsformen der Rechnerarchitekturen<sup>5</sup>. Legt man eine bestimmte Struktur zugrunde (von-Neumann-Rechner, ...), dann kann in diesem Bild auf der entsprechenden Softwareebene gearbeitet werden: Man programmiert im Assembler. Wohlgemerkt: man kann – aber wozu? In Blockschaltbildern dieser Art erscheint die Struktur von Computern sicherlich differenzierter als im Anfangsunterricht, und es lassen sich natürlich auch unterschiedliche Architekturen unterscheiden. Gehen wir davon aus, dass nicht die Einzelheiten des gerade benutzten Prozessors und seiner Komponenten behandelt werden<sup>6</sup>, sondern prinzipielle Strukturen, dann ist das Blockschaltbild eines von-Neumann-Rechners nicht sehr kompliziert und in kurzer Zeit zu erlernen. Die oben genannten Fragen werden dabei sicherlich präzisiert, aber sie werden nicht beantwortet. Für die Schülerinnen und Schüler bleibt noch offen, wie weit sie von den technischen Grundlagen entfernt sind: Sie „schwimmen“ auf einem Meer von Fragen, dessen Tiefe sie nicht einschätzen können. Die Frage, wie eine Schaltung „etwas entscheidet“, bleibt bestehen. Ich halte deshalb den Umgang mit „echter“ Hardware auf dem Niveau der Schaltalgebra (im Kontext eines Computer-Blockschaltbildes) für unverzichtbar. Sollten aus Zeitgründen nur entweder Rechnerarchitekturen (mit (Pseudo-) Assemblerprogrammierung) oder Schaltungsentwurf betrieben werden können, dann würde ich dem zweiten immer den Vorrang geben. Ohne diese Basis erscheint mir die Assemblerprogrammierung nur als (überflüssiger) Wechsel der Programmiersprache, der kaum neue Erkenntnisse bringt, aber viel Zeit erfordert, die mit anderen Themen m. E. besser genutzt werden könnte. Umgekehrt aber kann bei Kenntnis der Methoden, programmierbare Schaltwerke zu entwerfen, die Arbeit mit einem (Pseudo-)Assembler die Kluft zwischen den (Modell-)Rechnern und den höheren Programmiersprachen schließen. Die algorithmischen Grundstrukturen können

<sup>4</sup> Die Berechenbarkeit tritt hier meist unter dem Aspekt der Durchführbarkeit auf.

<sup>5</sup> Mir ist dieser Begriff für die Schule eigentlich zu hochgestochen, aber er hat sich nun einmal auch hier eingebürgert.

<sup>6</sup> was ich für selbstverständlich halte.

auf Sequenzen maschinennaher Befehle zurückgeführt werden, die einzeln direkt Schaltungskomponenten und -funktionen zuzuordnen sind und deren Komplexität mit den Mitteln der Schule zu bewältigen ist. Wird durch die Methoden des Schaltungs-entwurfs eine solide Basis gelegt, dann kann auf dieses technische Wissen über Computer erworben werden, das keinen Raum mehr für „Geheimnisse“ lässt.<sup>7</sup>

Fassen wir also die Anforderungen an die technische Informatik innerhalb der Kurs-folge noch einmal zusammen:

- 1. Kenntnisse über den Entwurf und die Nutzung von Schaltnetzen gehören zumindest teilweise in den Bereich der ITG und sollten vom Informatikunterricht vorausgesetzt werden können. Fehlen Sie, dann müssen sie nachgeholt werden. In keinem Fall darf sich die Informatik aber auf das Thema beschränken. Zu diesem Bereich gehören die systematische Aufstellung von logischen Schaltfunktionen und deren Umsetzung in Gatterschaltungen. Beispiele finden sich bei Fragen des Steuerns und Regeln und bei einfachen Rechenschaltungen. Die Umformung und Vereinfachung von Schaltfunktionen mit mathematischen Methoden ist weit weniger wichtig als Fragen nach den Möglichkeiten und Grenzen des Verfahrens.*
- 2. Wichtig für das Verständnis sind die Eigenschaften von Schaltnetzen. Welche Art von Schaltern verwendet wird, ist eigentlich egal. Ich glaube nicht, dass der Physikunterricht der Sek. I tragfähige Grundlagen für die Verwendung von Halbleiterschaltern (TTL, ...) liefert. Wir sollten uns deshalb auf ein einfacheres Modell beschränken, z. B. auf elektromechanische Systeme.<sup>8</sup>*
- 3. Wirken Schaltungen auf sich selbst zurück, dann ergeben sich Schaltwerke mit unterschiedlichen Zuständen. Als systematisches Beschreibungsmittel bieten sich endliche Automaten an, die Zugang zu den Verfahren der Automatentheorie öffnen. Auf diesem Wege können alle Komponenten eines Computers (meist in vereinfachter Form) systematisch entwickelt werden: Zähler, Register, Rechenschaltungen, ...*
- 4. Komplexere Systeme können im Automatenmodell nicht mehr übersichtlich beschrieben werden; deshalb bietet sich ab dieser Stufe modulares Arbeiten an. In Blockschaltbildern werden einfache programmierbare Systeme entwickelt, bei denen der Aufbau der wesentlichen Einzelkomponenten bekannt ist. Pseudoassemblerbefehle, die aus den Mikroprogrammschritten dieser Schaltungen abgeleitet werden, bilden den Übergang zu höheren Programmiersprachen.*
- 5. Die Architektur dieser Schaltungen kann leicht zu der bekannter Systeme erweitert werden. Die Stärken und Schwächen unterschiedlicher Strukturen werden aus deren Funktionen abgeleitet. Erst der Vergleich sehr unterschiedlicher Architekturen lässt solche Vergleiche zu.*

---

<sup>7</sup> Wenn auch neuronale Netze nicht gerade im Zentrum des technischen Informatikunterrichts stehen, so halte ich sie als Kontrast zu den systematischen Verfahren des Schaltungsentwurfs für außerordentlich hilfreich. Diese Netze „lernen“ ja (wobei die dabei erfolgenden Vorgänge auf dem Niveau der Sek. II mathematisch gut beschrieben werden können), und sie kommen etwa in der Mustererkennung zu Antworten, die auf anderem Wege nicht oder nur mit wesentlich höherem Aufwand zu erhalten sind. Dabei erkennen sie Muster, die von den gelernten ziemlich stark abweichen können, oft gut wieder. Manchmal aber auch nicht! Und ob sie zu einer Lösung kommen, wie sie zu einer Lösung kommen, ob dieses die beste Lösung ist (oder nur eine lokale Näherung), wie sicher die Antwort ist: All dieses können die Netze nicht beantworten – und wir auch nicht.

<sup>8</sup> Schon um Probleme der Schülerinnen und Schüler mit dem Physikunterricht nicht auf die Informatik zu übertragen!

Die Zahl der im Unterricht bearbeitbaren „Stufen“ hängt natürlich von der zur Verfügung stehenden Zeit ab. Weiterhin soll die Reihenfolge nicht zeitlich verstanden werden: Ich selbst hangele mich meist (sehr schnell) in einer Art Top-Down-Verfahren von einfachen Anweisungen der benutzten Programmiersprache über Pseudoassembler-Befehlssequenzen und die Blockschaltbilder eines Rechners bis zu einzelnen Hardwarekomponenten herab, so dass dann anschließend (ausführlich) entsprechende Schaltungen entworfen und im Bottom-Up-Verfahren zu programmierbaren Systemen zusammengesetzt werden können. Damit lässt sich problemlos ein ganzes Halbjahr verbringen; aber auch wenn wesentlich weniger Zeit zur Verfügung steht, muss auf den Entwurf (einfacher) programmierbarer Schaltwerke nicht verzichtet werden.

Ein Kurs zur technischen Informatik hat damit mehrere sehr unterschiedliche Funktionen:

- Er gestattet schon auf niedrigem Niveau praktisches Arbeiten mit „echter“ Technik<sup>9</sup> – eine Erfahrung, die im Gymnasium sonst kaum geboten wird.
- Er liefert ein solide fundiertes hardwarenahes mentales Computermodell, das durch praktische Arbeiten auf unterschiedlichen Gebieten und sehr unterschiedlichem Niveau gefestigt wird. Damit stützt er das Verständnis z. B. für die Arbeit mit Variablen, Parametern, Rekursionen, Blockstrukturen, ... im Softwarebereich hilfreich ab.
- Er bietet Erfahrungen im Entwurf und der Anwendung endlicher Automaten,
- und er macht allen viel Spaß.

---

<sup>9</sup> Hoffentlich „ausgelagert“ in die Sek. I, denn dort gehört diese Erfahrung hin.

## 2.2 Beispiel 2: Rechnermodelle

Fasst man „Programmiertwerden“ als eine Form von „Maschinenlernen“ auf, dann müssen Maschinen über die Fähigkeit verfügen, auf unterschiedliche Situationen unterschiedlich zu reagieren, also Entscheidungen zwischen Alternativen zu fällen. Da es Aufgabe von Unterrichtseinheiten über technische Informatik in ihrer Gesamtheit ist, ein gültiges Maschinenmodell bei den Lernenden entstehen zu lassen, gehört m. E. der Übergang von den elementaren Grundschaltungen zu eben dieser Fähigkeit zu den wichtigen Elementen eines Technikkurses. Darauf aufbauend kann eines der üblichen Rechenwerke, das über mindestens zwei unterschiedliche **Zustände** verfügt (z. B. Addieren und Subtrahieren) gesteuert werden. Die Belegung der Steuerleitungen bildet dann einen Elementarbefehl. Fügt man mehrere Elementarbefehle zu einer **Sequenz** zusammen und speichert sie geeignet, dann hat man schon die rudimentäre Form eines Programms gefunden. Kann der Wert des erforderlichen Programmschrittzählers mithilfe der „Entscheidungsschaltung“ beeinflusst werden, dann können wir **Alternativen** und **Sprünge** und somit **Iterationen** programmieren. Das genügt als Maschinenmodell für ein Grundverständnis programmierbarer Automaten. Wenn man die Baugruppen der Schaltung zu funktionalen Einheiten wie Rechenwerk, Steuerwerk, Speicher, ... zusammenfasst, dann hat man ein schönes Beispiel für **Modularisierung** und **Hierarchisierung** gefunden. Aber auch Erfahrungen mit der zunehmenden Integration der TTL-ICs dienen demselben Zweck. Die angegebenen fundamentalen Ideen treten hier in einem ganz anderen Gewand auf als in den anderen Kursen. Sie verdeutlichen damit deren übergreifenden Charakter.

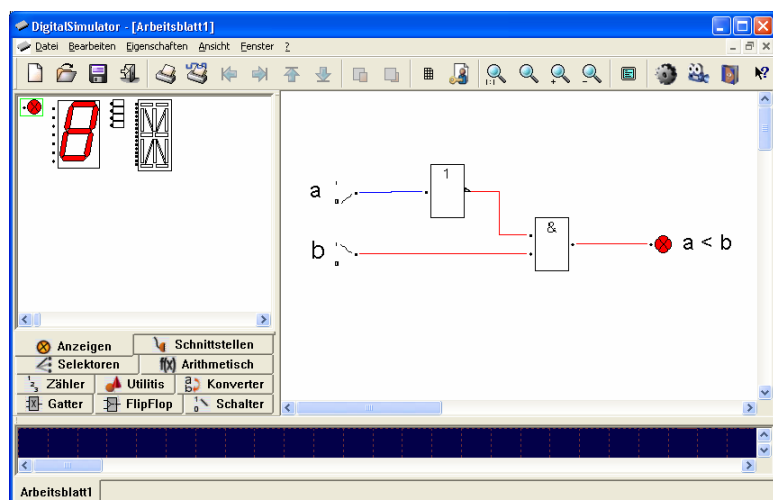
Schülerinnen und Schüler der Sekundarstufe I mögen den Umgang mit echter Hardware (TTL-ICs, ...) – meist nach anfänglichem Sträuben<sup>10</sup> – sehr. Sie stellen der Schulgemeinschaft mit großem Stolz die entwickelten „Rechenwerke“ z. B. in Schaukästen vor. Da die Entwicklung von Schaltungen und die Komposition von einfachen Schaltwerken aus vorhandenen ICs auch vom Anspruch her in diese Altersgruppe gehört, sollten zumindest die Anfänge der technischen Informatik in diesen Bereich verlegt werden. Nicht zu unterschätzen sind die Erfahrungen mit echter Technik, die sonst in der Schule kaum vorkommen.

Beginnen wir mit den Alternativen. Da sich TTL-ICs in einem Buch schlecht unterbringen lassen, benutze ich stattdessen ein frei verfügbares Simulationsprogramm<sup>11</sup>. Reduzieren wir die Fragestellung so weit wie nur möglich, dann lautet die Aufgabe:

„Gesucht ist eine Schaltung, die entscheiden kann, ob eine 1-Bit-Zahl kleiner als eine andere ist.“

Das Problem kann leicht über eine Schaltwerttabelle gelöst werden. Wir erhalten z. B. die folgende Schaltung und ihre Gatterdarstellung:

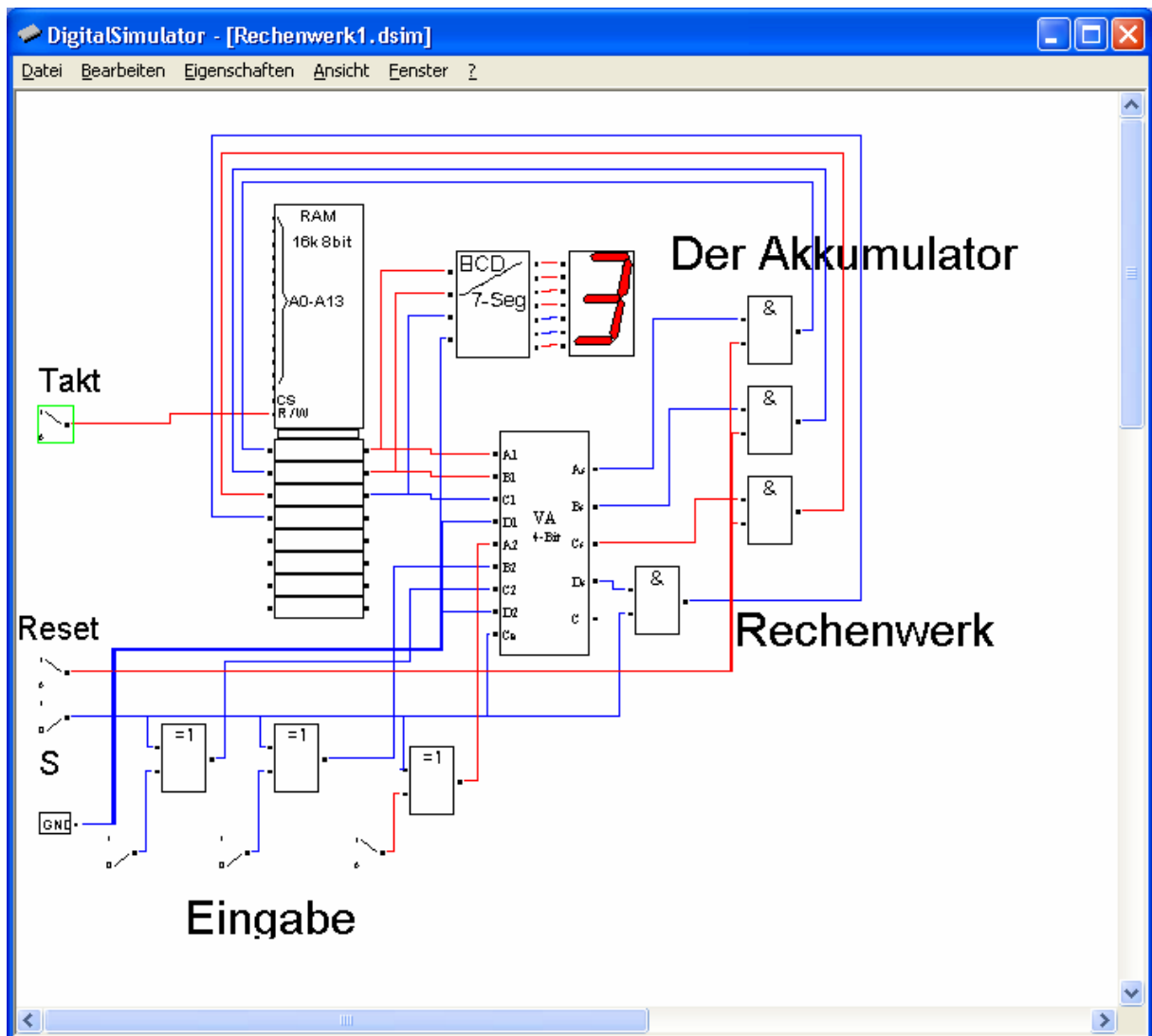
$$a < b = \bar{a} \wedge b$$



<sup>10</sup> Das Thema könnte ja etwas mit Physik zu tun haben, und die ist leider sehr unbeliebt.

<sup>11</sup> Den Digitalsimulator 4.0 von A. Herz





Wollen wir mit der Schaltung eine Rechenaufgabe lösen (deren Ergebnisse allerdings im 3-Bit-Rechenbereich bleiben müssen), dann können wir diese durch eine Folge von Steuerleitungsbelegungen und die erforderlichen „Daten“ (Zahlen) „programmieren“. Wählen wir z. B.

$$3 + 4 - 5 + 1 =$$

dann brauchen wir das folgende Programm:

Reset	S	Eingabe	Kommentar
0	bel.	bel.	Akku löschen
1	0	011	3 laden
1	0	100	4 addieren
1	1	101	5 subtrahieren
1	0	001	1 addieren

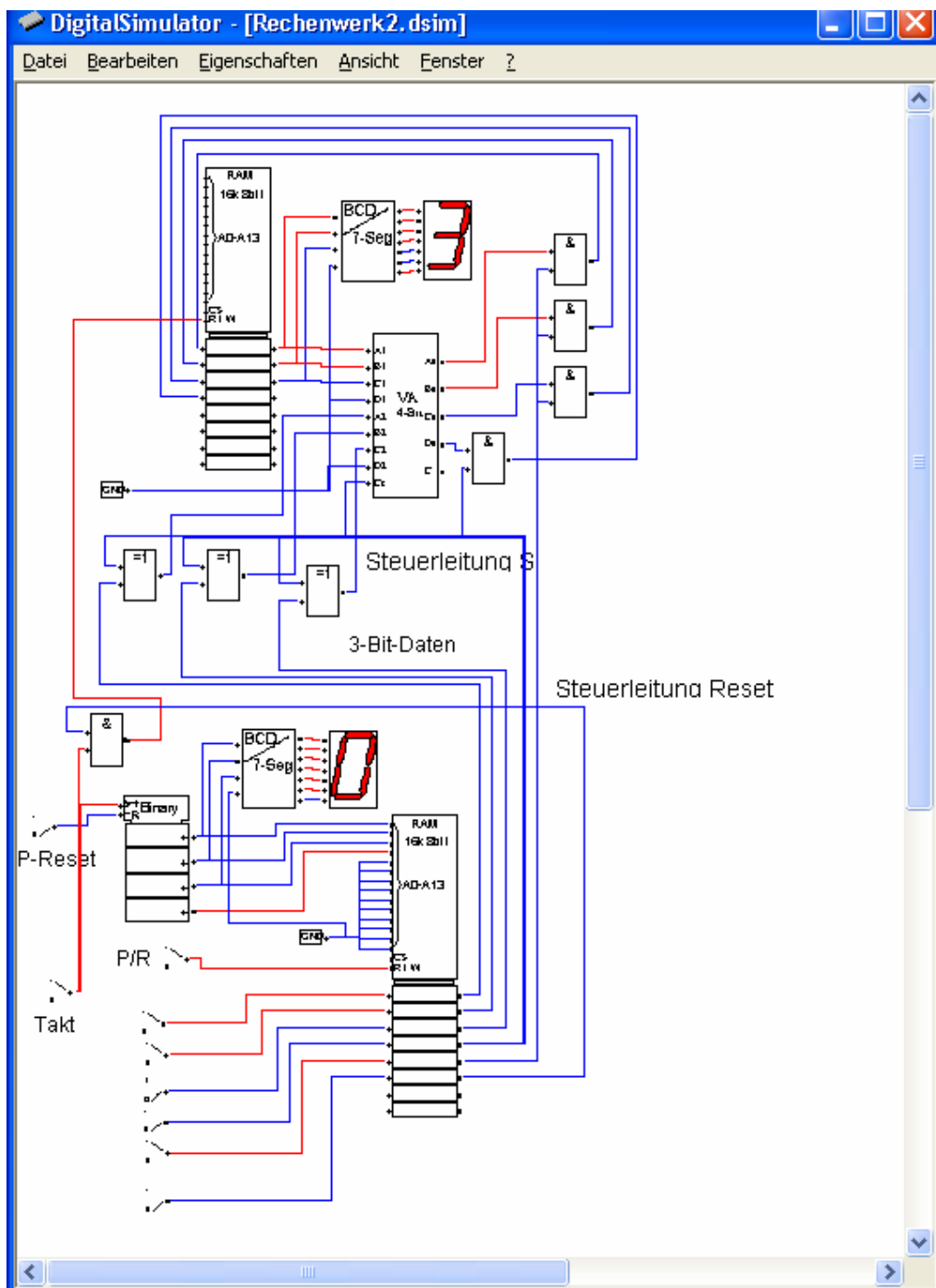
Wir können die Bitfolge  
als „Maschinenprogramm“  
auffassen. Zwischen den  
Befehlen folgen „Takte“.

00000  
10011  
10100  
11101  
10001

Bis zu dieser Ebene ist auch in der Sek. I mühelos vorzudringen. Die Schülerinnen und Schüler können mit der Schaltung experimentieren, lernen vergleichend mit Simulationsprogrammen und echter Hardware umzugehen, können die Schaltung abwandeln

- durch andere Bauteile (hier: „richtige“ Register benutzen, Anzeigen, ...),
- durch andere Steuerleitungen (z. B. um den Akkumulators zu laden),
- durch weitere Register (z. B. als Speicher für Zwischenergebnisse)
- oder durch Erweiterung des Zahlenbereichs.

Kurz: Sie machen sowohl auf einer relativ abstrakten wie auf einer konkreten Ebene Erfahrungen im Umgang mit echter Technik, lernen abzuschätzen, ob ihnen diese Art des Vorgehens liegt.



Betrachten wir das oben abgebildete Rechenwerk als ein Modul, in das drei Datenleitungen, zwei Steuerleitungen und eine Taktleitung hereinführen, dann können wir dieses von außen steuern, indem wir die erforderlichen Befehle in einem Speicher ablegen. Dieser selbst muss natürlich auch geordnet arbeiten, also „gesteuert“ werden. Dazu benötigen wir einen Binärzähler als *Program-Counter*, der den aktuellen Befehl angibt, und eine Möglichkeit, die Programminhalte einzugeben. Ich realisiere das hier über geeignete Schalter im unteren Teil des Bildes.

Leider kann unser Binärzähler (im Gegensatz zu vielen „echten“<sup>15</sup>) keine Werte laden. Deshalb ist in dieser Schaltung nur ein „Sprung zur Adresse 0“ möglich – über Reset. (Ich habe hier stattdessen einfach die Taktleitung zum Rechenwerk unterbrochen, um einen Stopp zu realisieren.) Trotzdem tauchen schon die ersten Elemente eines echten Prozessors auf. Das System ist – bei einem geeigneten Zähler – leicht zu erweitern. Es können von den Schülerinnen und Schülern Maschinen erfunden werden, die natürlich nur immer einige Ausschnitte eines richtigen Befehlssatzes „verstehen“, trotzdem aber z. B. in der Lage sind, rekursive Programme auszuführen.

Die programmierbaren Schaltwerke erfordern einiges an analytischem Denken und sorgfältige Abstimmung der Schaltungsmodule. Sie sind von der Komplexität her in der Sek. II anzusiedeln, können da aber durchaus auch im Grundkurs entwickelt werden. Sie liefern ein stark reduziertes, aber im Prinzip richtiges Hardwaremodell des Von-Neumann-Computers, das für das Verständnis der Abläufe bei Unterprogrammaufrufen und deren Parameterübergabe, Rekursionen und Referenzen m. E. unerlässlich ist. Und sie liefern auch in dieser Altersstufe Erfahrungen in technik-orientierter Arbeit.

Haben wir als Lerngruppe z. B. einen Grundkurs der Stufe 13, dann sollte dieser einerseits solide Kenntnisse über Entwurfstechniken und Programmierung besitzen, andererseits an Gruppen- und selbstständige Einzelarbeit gewohnt sein. Weil die Simulation von TTL-Bauteilen in der Schule eines der besten Beispiele für den Einsatz von OOP-Techniken ist, kann kursbegleitend ein eigener Hardwaresimulator als Hausarbeit entwickelt werden, wobei der im Unterricht neben der echten Hardware eingesetzte Simulator als Vorlage dient. Die dabei auftauchenden Probleme lassen sich sowohl auf einem elementaren Niveau als auch außerordentlich anspruchsvoll lösen: Es bietet sich ein weites Feld für Binnendifferenzierung.

Ziel des Unterrichts ist es neben der Anwendung fachlicher Grundlagen, bei den Schülerinnen und Schülern ein valides Rechnermodell entstehen zu lassen.

Unter diesen Voraussetzungen wird der Entwurf eines Rechnermodells im zweiten Kursteil liegen, weil anfangs die *logischen Grundschaltungen*, *Rechenschaltungen*, *Speicher* und *Rechenwerke* kennen gelernt werden müssen<sup>16</sup>. Zusammen mit Teilproblemen der Hardwaresimulation (Anwendung von Listen, OOP-Techniken, ...) braucht das seine Zeit. Wählen wir jetzt den oben skizzierten Weg, über stufenweise Erweiterungen von einem umschaltbaren Rechenwerk zu einem programmierbaren System zu kommen, dann erfordern die einzelnen Schritte von Stufe zu Stufe nur relativ wenig Zeit für gemeinsame Unterrichtsgespräche, die bei Bedarf eingestreut werden können. Der überwiegende Teil der Unterrichtszeit wird für Einzel- und Gruppenarbeit sowie intensive Einzelgespräche benötigt, in denen auftauchende Probleme geklärt werden - insbesondere, wenn mit echter Hardware gearbeitet wird.

<sup>15</sup> z. B. IC 74191

<sup>16</sup> Auf entsprechende Vorkenntnisse aus der Sek. I wird man (noch) kaum zurückgreifen können.

Mit diesen Informationen lässt sich die Unterrichtseinheit klassifizieren:

Thema: Rechnermodelle	
Zeitbedarf: ab 10 WStd. (je nach Ziel)	
Voraussetzungen: logische Grundsaltungen, einfache Rechenschaltungen, einfache Schaltwerke (Zähler, Speicher, ...)	
Die Einheit dient der Verdeutlichung	
der kulturellen Bedeutung des Themas	zu 10 %
gesellschaftlicher Auswirkungen des Themas	zu 20 %
rein fachlicher Aspekte	zu 70 %
Die folgenden Unterrichtsmethoden erfordern an Unterrichtszeit ca.	
Lehrervortrag:	
Unterrichtsgespräch:	20 %
Partnerarbeit:	40 %
Einzelarbeit:	
Projektarbeit:	40 %
Das Thema verdeutlicht die folgenden fundamentalen Ideen:	
1. Algorithmisierung	
1.1 Entwurfsparadigmen (Branch and Bound, Backtracking, ...)	
1.2 Programmierkonzepte (Alternative, Iteration, Rekursion, ...)	zu 20 %
1.3 Ablauf (Prozess, Nebenläufigkeit, ...):	
1.4 Evaluation (Verifikation, Komplexität, ...):	
2. strukturierte Zerlegung	
2.1 Modularisierung (Methoden, Hilfsmittel, ...)	zu 20 %
2.2 Hierarchisierung (Darstellung, Realisierung, ...)	zu 20 %
2.3 Orthogonalisierung (Emulation, ...)	
3. Formalisierung	
3.1 formale Sprache (Syntax, Semantik, ...)	
3.2 Automat (Zustand, Übergang, Vernetzung, ...)	zu 20%
3.3 Berechenbarkeit (Grenzen, Durchführbarkeit, ...)	zu 20 %