

# Der Intel 4004: Frühere Skalare Mikroprozessoren

## Seminar: Geschichte der Rechnerarchitektur

Daniel Riedel

Stephan Baller

### 1. EINLEITUNG

Das Wort "Prozessor" wird heutzutage überwiegend mit der Vorstellung von einem Chip assoziiert, der das Herz eines jeden technischen Produkts darstellt. Vor 50 Jahren war das noch anders, denn zu diesem Zeitpunkt waren Prozessoren noch elektronische Schaltungen aus mehreren Komponenten. Fast zeitgleich wurden bis Ende des Jahres 1971 zwei CPUs entwickelt, die das ändern sollten. Der TMX 1795 wurde im Juni von Texas Instruments vorgestellt und galt lange Zeit als der erste Mikroprozessor. Trotz dessen wird oft fälschlicherweise behauptet, der Intel 4004 sei der erste Mikroprozessor, obwohl dieser erst 2 Monate nach dem TMX 1795 und ganze 2 Jahre nach dem tatsächlich ersten Mikroprozessor, dem Mikroprozessor des F14A Kampjets, zum Einsatz kam[4][7]. Grund dafür könnte die Bekanntheit des 4004 gegenüber dem Modell von Texas Instruments sein. Im Gegensatz zum 1802, wurde der 4004 nämlich kommerziell vermarktet. Die Prozessoren von Texas Instrument wurden erst später, durch den Erfolg des TMS-1000 populär[15]. Der Mikroprozessor des F14A wurde im Geheimen für die US Navy entwickelt und fand daher bis zur Veröffentlichung eines Artikels im Wallstreet Journal 1998 keine Aufmerksamkeit.[7]

Die Bekanntheit des 4004 bleibt aber nicht unberechtigt. Er bildet die Grundlage aller darauf folgenden Prozessorgenerationen von Intel und ist der erste Prozessor, der auf die neue SGT-MOS Technologie setzt und damit dramatische Neuerungen auf den Markt bringt. Auf der CPU basiert schließlich die heutzutage meist verbaute x86-64 Architektur. Der 4004 selbst gehört der Architekturklasse der Skalar Prozessoren an, die für das Verarbeiten von einer Information pro Taktzyklus steht.

In den folgenden Kapiteln bringen wir dem Leser Geschichte, Aufbau und Auswirkung des Intel 4004 näher, und gehen dabei insbesondere auf die Eigenschaften ein, die ihn für die damalige Zeit und auch heute noch beachtlich machten.

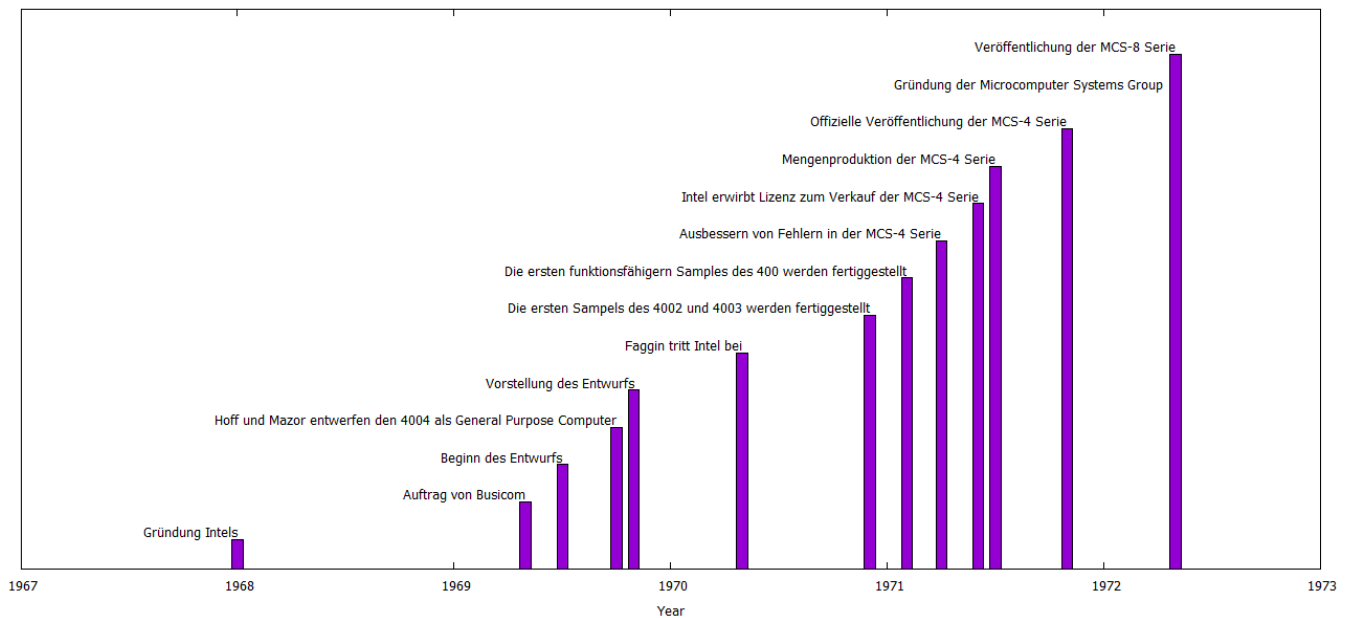
### 2. ENTWICKLUNGSGESCHICHTE

Bob Noyce und Gordon Moore gründeten die Intel Corporation im Jahr 1968 nachdem sie Fairchild Semiconductor verlassen hatten. Ihr Erstreben war es dabei vor allem Mainframe Speicherprodukte mit „Bipolar“ und „MOS“ Prozessen herzustellen. Dabei wollten sie, anstatt vereinzelte Spezialdesigns für Großkunden zu entwerfen, genereller anwendbare Produkte entwerfen. Da diese proprietären Speicherprodukte noch nicht ausreichende Umsätze erreicht hatte waren sie jedoch nicht vollständig von Spezialanfertigungen abgeneigt [10].

Nippon Calculating Machines Company und Electro-Technical Industries beauftragten im April 1969 die Intel Corporation zum Entwurf von mehreren Rechen und Speicher Chips. Diese waren für den Einsatz in Busicom Rechnern gedacht, welche von einfachen druckenden Schreibtisch-Rechnern bis zu Rechnungs-Maschinen reichte. Masatoshi Shima, Masuda und Takayama, Mitarbeiter der japanischen Firmen Nippon Calculating Machines Company und Electro-Technical Industries, waren beauftragt Intel bei dem Entwurf zu unterstützen, was sie im Juni 1969 begannen [10]. Marcian E. Hoff Jr., welcher bis 1968 an der Stanford Universität mit elektronischen neuronalen Netzen arbeitete, wurde von Intel, wo er seit 1968 angestellt war, beauftragt die Vertreter der japanischen Firmen zu unterstützen und Intel zu vertreten. Er hatte bereits während seiner Zeit bei Stanford viel Erfahrung mit dem Entwurf von Hardware Interfaces gesammelt [8].

Obwohl seine Rolle in dem Entwurf der Chips nur unterstützend und vertretend war studierte er die Entwürfe im Detail. Es war geplant einen ROM-basierten, makroinstruktionsprogrammierbaren dezimal Rechner zu bauen der aus Chips für Programm Steuerung, dezimaler Arithmetik-Einheit, Timing Einheit, ROM, Shift Register, Drucker Steuerung und Ausgabeport Einheit bestand [10]. Dieses Design war bereits bei bisherigen Produkten von Busicom bewährt. Hoff war besorgt über die Umsetzbarkeit dieses Entwurfes, gegeben der Zielpreise, da es viel und komplexe Verknüpfungen erforderte. Bob Noyce ermutigte ihn nach alternativen Lösungen zu suchen [8].

Der Entwurf, der von Busicom vorgeschlagen wurde, sah vor Shift Register zu verwenden. Diese waren gut geeignet um arithmetische Berechnungen, Displays und Drucker zu implementieren. Ein Bit benötigte bei Shift Registers, sowie auch bei statischem RAM 6 Transistoren, jedoch war die



Steuerlogik von Shift Registern aufwändiger und sie waren langsamer bei musterfreiem Zugriff (Random Access) [10]. Dies machte Shift Register unideal zur Anwendung in generelleren Aufgaben. Zur selben Zeit arbeitete Intel auch an dynamischem MOS RAM. Dieser wäre ideal für den Einsatz in den Rechnern, verwendete weniger Siliziumfläche als Shift Register und benötigte nur 3 Transistoren pro Bit [10]. Ein Problem bei DRAM war jedoch, dass es nicht möglich war auf diesen zuzugreifen während dieser sich in einer Refresh Phase befand [15].

Während des Julis und Augusts von 1969 Hoff experimentierte damit statt eines auf Rechner spezialisierten Chipsets einen general-purpose Chip zu entwerfen, der für diese Zwecke programmiert werden konnte. Dafür holte der Arithmetische Chip Instruktionen aus dem ROM, welche daraufhin interpretiert wurden. Daten konnten sowohl in dem von Intel entworfenen DRAM als auch in Scratchpad Registern gespeichert werden. Dieser Vorschlag würde das Problem des DRAM Refreshes lösen, da dieser durchgeführt werden konnte während die Arithmetische Einheit in der Instruktionsholphase auf den ROM Zugriff [11]. Im September 1969 trat Stanley Mazor, der bis zu diesem Zeitpunkt bei Fairchild Semiconductor gearbeitet hatte, Intel bei, wo er mit Hoff den Entwurf dieses programmierbaren general-purpose Computers ausbaute. Da die Möglichkeit dieses Designs Makroinstruktionen auszuführen noch immer ein Besorgnis der Vertreter Busicoms war implementierte Mazor einen Makroinstruktionen-Interpreter in nur 20 Byte [10]. Im Oktober 1969 Intel präsentierte beide Entwürfe dem Management von Busicom. Diese schätzten die Einfachheit und Flexibilität des Vorschlags von Intel und wählten diese über des von den Ingenieuren Busicoms dargebrachten. Shima blieb als einziger Vertreter Busicoms bis Dezember 1969 bei Intel und entwickelte Software für Busicom Rechner anhand der vorgeschlagenen Architektur [10].

Zum großen Enttäuschen Hoff's besaß, obwohl die Architektur vor allem von Intel entwickelt worden war, Busicom noch

immer exklusive Rechte für die Chips. Da weder Mazor noch Hoff Erfahrung mit dem Entwurf von Chips hatten benötigte Intel jemand der den komplexen MOS Entwurf durchführen konnte [10]. Im Dezember 1969 Computer Terminals Corporation of San Antonio, Texas, welches später zu Datapoint umbenannt wurde, ersuchten Intel bezüglich der Modifikation eines bestehenden 64 Bit statischen RAMs von Intel zu einem 4 Bit · 16 Stapelspeichers für ein Terminal das CTC entwickelte. Mazor und Hoff erkannten, als sie den Prozessor, den CTC entwickelte, studierten, dass dieser kaum komplexer war als Intel 4004. Sie entwarfen eine Spezifikation für einen 8-Bit Mikroprozessor und erhielten den Auftrag von CTC diesen für sie zu entwerfen [15].

Federico Faggin, der mit Tom Klein in 1968 am MOS silicon-gate Prozess gearbeitet hatte, wurde im April 1970 bei Intel als „Engineer in charge of the design of the calculator design“ angestellt. Das lange Fehlen einer Anstellung dieses Postens hatte, laut Shima, den Zeitplan irreparabel in Verzug gesetzt. Um diesen Verzug aufzuholen arbeitete Faggin in wildem Tempo oft bis in die frühen Morgenstunden [10]. Dabei entwickelte er seine eigene Entwurfsmethodiken und Prozesse [9].

Obwohl Intel einen zeitgeteilten Mainframe Computer für Critical Circuit Design besaß, wurde Faggin aus Kostengründen davon abgeraten diesen zu verwenden. Daher entwarf Faggin die Intel 4000 Reihe größtenteils mit nur einem Rechenschieber und graphischen Analysen basierend auf gemessenen statischen und dynamischen Transistor Charakteristiken [10]. Dieser vielschrittige Prozess dauerte über 6 Monate bis zur Herstellung eines Musters. November 1970 wurden die ersten Musterexemplare der 4003 und 4002 fertiggestellt und erfolgreich getestet. Die Fertigstellung der funktionsfähigen 4004 Muster dauerte wegen eines Produktionsfehlers bis Januar 1971. Es dauerte weitere 2 Monate, bis März 1971, um die letzten Fehler in diesen zu korrigieren. Mit einem RAM basierten Emulator des 4001, den Shima entwickelt hatte, konnten die ROM Patterns für den Busi-

com Calculator fertiggestellt werden [10].

Bei ausführlichem Testen entdeckte Faggin Fehlerhaftes Verhalten des 4004 bei höheren Temperaturen. Trotz fehlender Diagnose- und Debug-Werkzeuge fand Faggin nach wenigen Tagen, dass Daten im DRAM bei höheren Temperaturen korumpiert werden konnten. Nach mehr intensivem Testen konnte er dies auf Probleme des RAM Dekodierers zurückführen. Da der 4002 einen ähnlichen Dekodierer verwendete entwarf Faggin spezielle Testsequenzen um diesen auf eventuelle zu testen. Das auffinden derselben Fehler bestätigte ihn in seiner Annahme, dass diese Fehler durch den Dekodierer verursacht wurden, und er konnte den 4004 und 4002 genügend modifizieren um diese Fehler zu beheben. Ab Juni 1971 konnte die Intel 4000 Serie in Masse produziert werden und stellten ab August desselben Jahres einen großen Teil Intels Einkommens dar [10].

Faggin nutzte die Flexibilität des 4004 indem er diesen als Controller eines Produktions Testers, der beim Testen des 4004 in der Produktion verwendet wurde, verwendete. Das Programm hierfür konnte er auf einem EPROM (Electrical Programmable Read Only Memory), eine Technologie die von Dov Frohmann-Bentchkovsky bei Intel erfunden wurde, speichern anstatt einen RAM programmierbaren Programmspeicher entwerfen zu müssen. Des Weiteren konnte Faggin damit dem Management Intels die generelle Anwendbarkeit der 4000 Serie demonstrieren. So nutzte auch Hoff den 4004 um das EPROM zu programmieren. Faggin, der sehr von der generellen Anwendbarkeit des 4004 überzeugt war, erfuhr in einem Telefonat mit Shima, dass Busicom finanzielle Probleme hatte und einen Preisnachlass der 4000 Serie ersuchte. Im Mai 1971 erwarb Intel die Lizenz die Intel 4000 Serie zu verkaufen mit Ausnahme von Schreibtisch Rechenmaschinen. Dennoch war Intels Management und Marketing zurückhaltend, da sie befürchteten der komplexe 4004 wäre zu kompliziert für Intels Support, was ihrem guten Ruf schaden könnte. Des Weiteren war der 4004 nicht so leistungsfähig wie Minicomputer wie der PDP-11. Der ab Sommer 1971 neue Vizepräsident von Marketing bei Intel ernannte Hank Smith zum ersten Manager für Mikrocomputer Marketing. Beide waren mehr zuversichtlich bezüglich der Marketbarkeit des 4004, welcher daraufhin im November 1971 offiziell vorgestellt wurde [10].

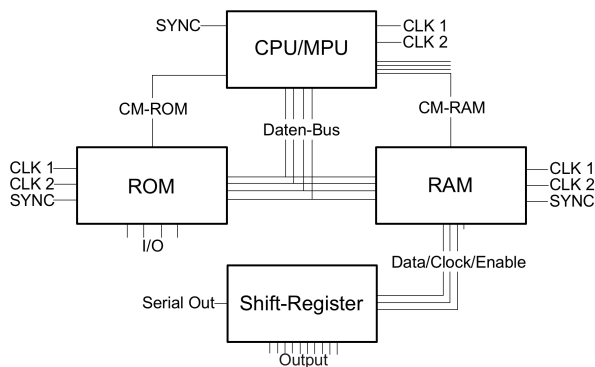
Im April 1972 formte Hank Smith die erste Mikrocomputer Systems Group, enthaltend Hardware Ingenieure, Software Entwickler, Produktion und Marketing. In diesem Rahmen entwarf Intel mehrere Produkte die Intels Mikroprozessoren einfacher zu verwenden machen sollten. Da Kunden die Intel Mikroprozessoren verwendeten voraussichtlich eine Menge an Software Investment kreieren würden und dieses dann nur mit erheblichen Kosten auf ein anderes System übertragen konnten, war es dem Mikroprozessor Marketing Intels vor allem wichtig ihre Kunden zu diesem ersten Schritt zu bringen und sie damit an Intel Mikroprozessoren zu binden. Diese Produkte enthielten extensive Dokumentationen und Anleitungen ihrer Mikroprozessoren, Simulatoren die zur Entwicklung, Preproduktion und Kleinproduktionen verwendet werden konnten, die von Gary Kildall entwickelte PL/M high-level language und die Intellec Entwicklungssysteme, vollständige und erweiterbare Systeme mit CPU, Speicher, Ein-/Ausgabe, Clock, TTY Interface und allen weiteren zur

Entwicklung von Software nötigen Modulen inklusive der dafür verwendeten Standardsoftware [10].

Intel änderte den Namen der 4000 Serie zu MCS-4, was für Micro Computer System 4-bit steht. Die Annahme des Mikroprozessors war äußerst vielversprechend, insbesondere aufgrund der extensiven Anleitungen und Dokumentationen. Im April 1972 stellte Intel den 8-bit fähigen 8008, den Mikroprozessor der MCS-8 Serie, vor. Die MCS-8 Serie unterschied sich von der MCS-4 Serie durch einen höheren Preispunkt, zusätzliche notwendige TTL ICs, größeren Adressraum (bis zu 16 Kilobyte) aus RAM und ROM und die Fähigkeit 8-bit Quanten zu verarbeiten [11].

**Table 1: Die MCS-4 Familie**

Komponente	Name	Funktion
4001	ROM	Programmspeicher, 256 x 8Bit pro ROM
4002	RAM	Arbeitsspeicher, 80 x 4Bit pro RAM
4003	Schieberegister	Erweiterung der Ausgabe-Ports von RAM und ROM
4004	CPU/MPU	Mikroprozessor/CPU
4008/4009	Kompatibilitäts-Register	Zur Nutzung mit anderen Intel-Speicherchips



**Figure 1: Überblick über die Interaktionsweisen der einzelnen MCS-4 Komponenten**

### 3. AUFBAU UND FUNKTIONSWEISE

#### 3.1 Die MCS-4 Familie

Die CPU (oder auch Micro-Processing-Unit MPU) alleine ist nur ein Teil der Micro-Computer-Set-4 Familie von Intel (kurz: MCS-4). Sie umfasst noch 6 weitere Mitglieder, die mit 4001 bis 4004 und 4008 bis 4009 durchnummeriert sind. Zusammen bildet die CPU, der i4004, mit den anderen Komponenten eine Recheneinheit. Der Anfang der Namen steht dabei für die 4-Bit Architektur, die schon ein Jahr später von der 8-Bit Architektur abgelöst wurde.

Um eine funktionsfähige MCS-Recheneinheit zu bilden ist außer der MPU, noch ein 2-Phasen-Taktgeber und der Intel 4001 ROM (Read-Only-Memory) als Hauptspeicher notwendig. Gemeinsam bilden sie die Mindestkonfiguration und können durch die weiteren MCS-4 Komponenten erweitert werden. ROM, RAM und MPU sind dabei durch einen gemeinsamen Datenbus mit einer Breite von 4 Bit miteinander verbunden (siehe Abb. 3.1).

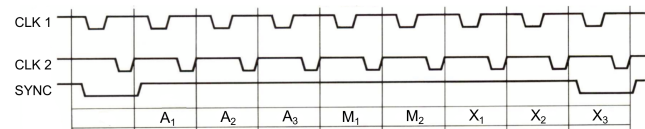
Zur Unterscheidung, ob ROM oder RAM gerade angesprochen wird, sendet der 4004 "Memory-Control-Signale an die entsprechende Komponente.

Mit bis zu 16 ROM, und 16 RAM Chips kann der Hauptspeicher auf bis zu 4kB, und der Arbeitsspeicher auf bis zu 512B Daten + 128B Status erweitert werden. Durch Hinzufügen einiger Schaltkreise soll eine MCS-4 Einheit laut Bedienungsanleitung auch bis zu 48 ROM und RAM Chips umfassen können. Tabelle 3.1 gibt eine allgemeine Über-

sicht über Bestandteile und Funktionen der einzelnen Mitglieder.[6]

Die konkreten Funktionsweisen der einzelnen Bausteine werden in den nächsten Abschnitten in chronologischer Reihenfolge behandelt.

#### 3.2 Ablauf eines Befehlszyklus[6]



**Figure 2: Ablauf eines Befehlszyklus**

Bevor man die Komponenten im Detail betrachtet, lohnt es sich den Befehlszyklus genauer anzusehen, den der Prozessor während seiner Laufzeit dauerhaft wiederholt. Bei einem Durchlauf wird eine Instruktion aus dem Speicher geholt und ausgeführt. Das dauert genau 8 Takte zu insgesamt  $10.8\mu s$  und lässt sich damit in einer Sekunde bei standardmäßiger Taktfrequenz von bis zu 740kHz ca. 10.000 Mal wiederholen (vgl. heute: Über 4.000.000kHz).

Zunächst lässt sich der Befehlszyklus in 3 Phasen unterteilen (Abb. 2):

- $A_1-A_3$ : CPU sendet 3x4Bits der Adresse an den ROM (4 kleinsten Bits werden zuerst gesendet)
- $M_1 \& M_2$ : ROM sendet 2x4Bits der Instruktion an die CPU: (4 größten Bits werden zuerst gesendet)
- $X_1-X_3$ : CPU führt Instruktion aus (Ist die Instruktion 2 Wörter lang, wird der Befehlszyklus wiederholt, um die nächsten 8 Bit des Befehls zu holen)

Die höherwertigen Bits geben bei der Adressierung die ROM-Nummer an. Die niederwertigen, die Adresse innerhalb des entsprechenden ROMs. Zur zeitlichen Abstimmung wird ein Taktgeber mit zwei Ausgängen benötigt, der unabhängig von den MCS-4 Komponenten funktioniert (CLK 1 und CLK 2 aus Abb. 2). Zusätzlich geht von der CPU noch ein SYNC-Signal aus, um einen Befehlszyklus an die anderen Komponenten zu signalisieren.[6]

#### 3.3 Funktionsweisen der Komponenten im Überblick

##### 3.3.1 i4001: ROM

Wichtigster Bestandteil neben der MPU ist der ROM, welcher als Erstes Mitglied der MCS-4 Familie entwickelt wurde. Durch ihn kann die Recheneinheit programmiert und mit ihr interagiert werden. Er stellt damit den Hauptspeicher und eine Ein-/Ausgabe-Schnittstelle für die MCS-4 Einheit dar. Da der ROM im Gegensatz zum Hauptspeicher in neueren Rechenmaschinen nicht beschreibbar ist, muss der Ablauf schon bei der Herstellung feststehen.

Die Größe einer ROM-Einheit beträgt  $256 \times 8$  Bit, wobei jeder Befehl bis zu 2 Zellen belegen kann.



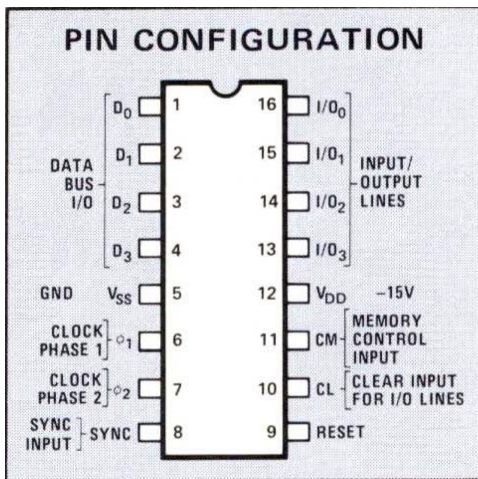


Figure 3: Diagramm vom Aufbau des i4001[6]

Die Belegung der Ports aus Abb. 3 lautet wie folgt[6]:

- *SYNC*: Sync-Eingang
- $\varphi_1$  &  $\varphi_2$ : Clock-Eingangssignale
- $D_0 - D_3$ : Datenbus-Ports
- *CM*: Memory-Control-Eingang
- $I/O_0 - I/O_3$ : Ein-/Ausgabe-Ports
- *CL*: Clear-Input-Eingang (für I/O Ports)
- *RESET*: Reset-Eingang
- $V_{DD}$ : Strom-Anschluss(-15V)
- *GND*: Strom-Anschluss(Ground)

Zu Beginn eines Befehlszyklus ( $A_1$  und  $A_2$ ) werden die Adress-Bits durch den Datenbus empfangen, die eine 16x16 Speicherzellen adressieren. In dem darauf folgenden Takt  $A_3$  wird die Chip-Nummer des ROMs empfangen. Diese wird bei der Herstellung für jeden ROM dauerhaft festgelegt und entscheidet darüber, ob die Komponente bei der Adressierung angesprochen wird, oder nicht. Der Inhalt der 8-Bit Zelle wird in den Takten  $M_1$  und  $M_2$  auf den Datenbus gelegt, die niederen 4 Bits als Erstes.

Vier weitere Ports dienen zur Ein- und Ausgabe von Daten. Dazu wird mit dem SRC Assembler-Befehl ein ROM (oder RAM) ausgewählt und mit einer weiteren E/A-Instruktion die Ports gelesen, oder auf ihnen geschrieben. Wird der SRC-Befehl ausgeführt, so ist im  $X_2$ -Takt das CM-ROM-Bit gesetzt und die Daten auf dem Bus stehen wiederholt für die Chip-Nummer. Die Daten, die dabei im  $X_3$ -Takt auf dem Bus liegen, haben nur bei der Auswahl eines RAMs Bedeutung. Nun kann im nächsten Befehlszyklus eine E/A Instruktion für den entsprechenden ROM ausgeführt werden, sodass Daten von den Ports gelesen oder geschrieben werden können.

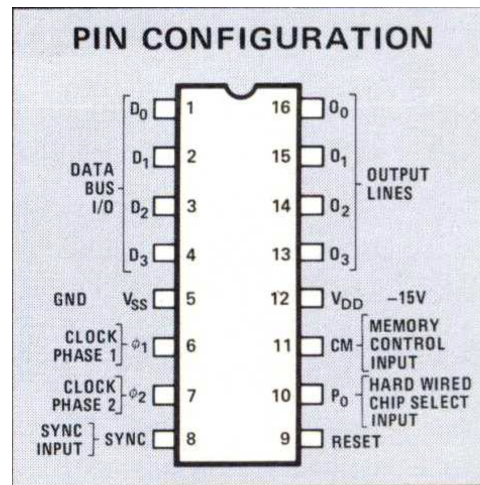


Figure 4: Diagramm vom Aufbau des i4002[6]

Ob ein einzelner Port für Ein- oder Ausgabe verantwortlich ist, wird bei der Herstellung festgelegt und kann danach nicht weiter konfiguriert werden. Mit dem Clear-Input-Signal können die Bits auf den E/A-Ports gelöscht werden. Das Reset-Signal setzt alle internen Flip-Flops (Bit-Speicher) inklusive dem Output-Register zurück. Clock- und Sync-Signale dienen lediglich der zeitlichen Abstimmung.[6]

### 3.3.2 i4002: RAM

Der RAM ist der Arbeitsspeicher einer MCS-4 Einheit und kann im Gegensatz zum ROM auch beschrieben werden. Die Pin-Belegung der beiden Komponenten sind sehr ähnlich, mit dem Chip-Select-Eingang als einziger Unterschied[6]:

- *SYNC*: Sync-Eingang
- $\varphi_1$  &  $\varphi_2$ : Clock-Eingangssignale
- $D_0 - D_3$ : Datenbus-Ports
- $V_{DD}$ : Strom-Anschluss(-15V)
- *GND*: Strom-Anschluss(Ground)
- $O_0 - O_3$ : Ausgabe-Ports
- *RESET*: Reset-Eingang
- *CM*: Memory-Control-Eingang
- $P_0$ : Chip-Select-Eingang

Beim RAM können eine von bis zu 16 Chips adressiert werden. Dazu hat die MPU 4 CM-ROM Ausgänge, die jeweils eine von 4 RAM-Bänken ansteuern. Zur Auswahl einer RAM-Bank gibt es den DCL-Befehl, die weitere Adressierung erfolgt dann mit SRC. Dabei wird in den ersten 2 Bits in  $X_2$  die Chip-Nummer innerhalb einer Bank spezifiziert, mit den letzten 2 Bits in  $X_2$  eines von 4 Registern innerhalb eines Chips und schließlich, mit den 4 Bits in  $X_3$ , eine von 16 Speicherzellen. Zusätzlich besitzt jeder Chip noch 4 Statusregister, die mit jeweiligen Befehlen gelesen und beschrieben werden können. Mit einer Zellengröße von 4 Bit, kommt man auf eine Kapazität von 320 Bit pro RAM.

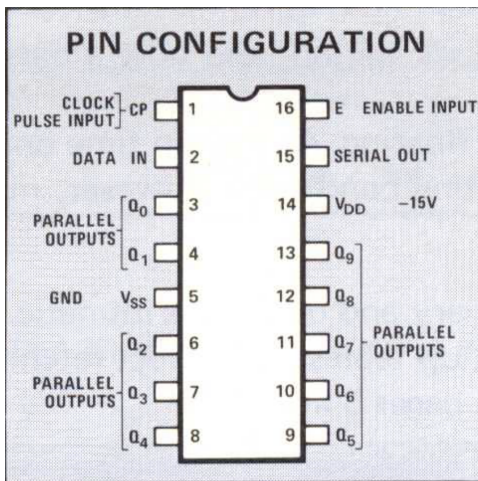


Figure 5: Diagramm vom Aufbau des i4003[6]

Die Ausgabe erfolgt, nach voriger Auswahl des Chips mit DCL und SRC, durch eine entsprechende Instruktion.[6]

### 3.3.3 i4003: Das Schieberegister

Zur primären Ausgabe von Daten an Tastaturen, Displays, Drucker, etc. kommt das Schieberegister zum Einsatz[6]:

- $V_{DD}$ : Strom-Anschluss(-15)
- $GND$ : Strom-Anschluss(Ground)
- $SERIALOUT$ : Serielle Ausgabe von Daten
- $Q_0 - Q_9$ : 10 Ausgabe-Ports
- $E$ : Enable-Input-Port zur Kontrolle, ob gespeicherte Daten an  $Q_0 - Q_9$  ausgegeben werden sollen
- $DATAIN$ : Serielle Eingang zur Eingabe von Daten
- $CP$ : Clock-Eingang

Gesteuert wird das Schieberegister durch Anbindung des CP-, SERIAL IN- und E-Eingabeports an die Ein-/Ausgabeports einer ROM- oder RAM-Einheit. Das Schieberegister besitzt einen internen Speicher von 10 Bits, die durch Setzen des Clock-Eingangs-Bits um eine Stelle weiter geschoben werden können. Dabei wird das erste Bit von DATA-IN überschrieben, und das letzte Bit auf SERIAL-OUT geschoben. Ist der Enable-Input-Port auf "low", wird der interne Speicher an die Ausgabe-Ports  $Q_0 - Q_9$  weitergeleitet. Der Serial-Out-Port ermöglicht eine Hintereinanderschaltung beliebig vieler 4003-Einheiten. Die parallele Ausgabe von Daten ist damit nur zeitlich beschränkt. Will man beispielsweise 400 Bits gleichzeitig ausgeben, erfordert dies mindestens 1600 Takte ( $800 \times LDM, WRR$ ).[6]

### 3.3.4 i4004: Die MPU

Nun zum Highlight der MCS-4 Familie: Die Micro-Processing-Unit. Die Belegung der Pins erklärt sich schon durch die vorigen Abschnitte:

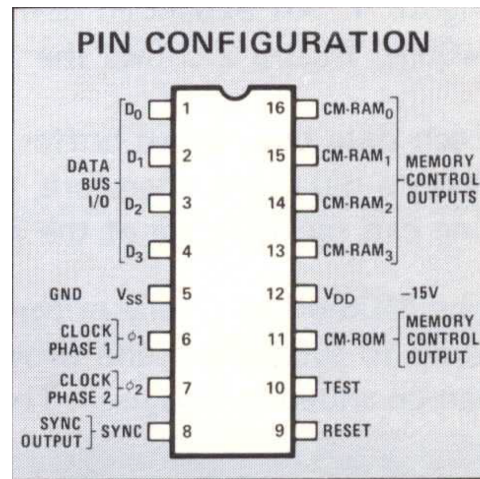


Figure 6: Diagramm vom Aufbau des i4004[6]

- $CM - ROM$ : Memory-Control-Ausgang zur Aktivierung des ROMs
- $CM-RAM_0 - CM-RAM_3$ : Memory-Control-Ausgang zur Auswahl eines RAMs
- $\varphi_1$  &  $\varphi_2$ : Clock-Eingangssignale
- $SYNC$ : Sync-Ausgang
- $RESET$ : Reset-Eingang zum Zurücksetzen der Register
- $TEST$ : Eingang, der bei bedingten Sprungbefehlen einbezogen werden kann
- $D_0 - D_3$ : Datenbus-Ports
- $GND$ : Strom-Anschluss(Ground)
- $V_{DD}$ : Strom-Anschluss(-15V)

Der 4004 ist in der Lage, im ROM vorprogrammierte, 8 bzw. 16 Bit breite Assemblerbefehle auszuführen und bestimmt durch das SYNC-Signal teilweise auch den zeitlichen Ablauf der anderen Komponenten. Wie die Register genutzt werden, erklärt sich bei näherer Betrachtung des inneren Aufbaus.

Die internen Register der CPU setzen sich wie in Abb. 7 zusammen: Sechzehn 4 Bit breite Register zum Zwischenspeichern von Daten, ein 8 Bit breiter Daten-Pointer zum Speichern des aktuellen Befehls, ein 12 Bit breiter Code-Pointer, der den nächsten Befehl im ROM adressiert, ein 4 Bit breiter Akkumulator mit Carry-Bit und ein Return-Stack.

Das am meisten genutzte Register ist der Akkumulator (Mit Carry-Bit)[1]. Durch ihn findet nicht nur die gesamte Kommunikation zwischen den 16 internen Registern und ROM bzw. RAM statt, es ist auch das einzige Register, auf das arithmetische Operationen angewandt werden können.

Eine Instruktion kann eine Breite von bis zu 16Bit haben, nimmt dabei allerdings 2 Befehlszyklen in Anspruch. Im Code-Pointer befindet sich die Adresse der nächsten Instruktion (im ROM). Nach jedem Befehlszyklus wird der Pointer inkrementiert, außer es wird eine Jump-Instruktion ausgeführt, die den Wert verändert. Nach dem Holen des Befehls nach Schritt  $A_1-M_2$  im Befehlszyklus, befindet sich der

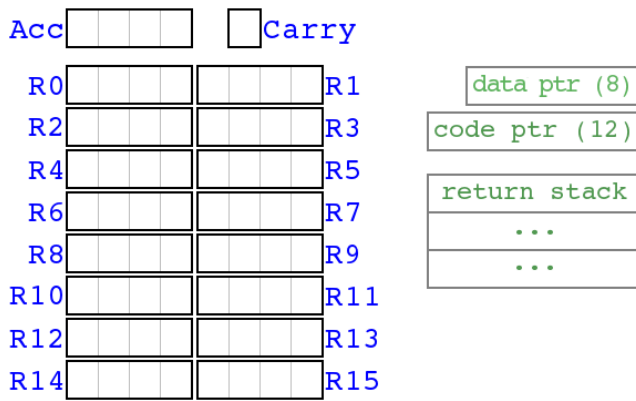


Figure 7: Interne Register des i4004[1]

adressierte Befehl schließlich im Data-Pointer Register. Von dort aus verarbeitet die CPU den Befehl dann weiter. Der return-Stack ermöglicht das Zwischenspeichern der anstehenden Befehlsadresse durch einen speziellen Jump-Befehl (Vergleichbar mit "call" bei x86), der Branch-Befehl dazu das Zurückspringen an die zuvor gespeicherte Adresse (Vergleichbar mit "ret" bei x86).

### 3.3.5 i4008 & i4009: Eine Brücke zu anderen Intel-Speicherchips

Bevor bei Intel die MCS-4 Familie in Entwicklung war, hat die Firma überwiegend Speicherchips hergestellt. Um schon vorhandene und zukünftige Speicherchips mit dem 4004 kompatibel zu machen, wurde deshalb später noch der i4008 und i4009 zu der Familie hinzugefügt. Der 4008 kann wie ein ROM behandelt werden und übersetzt die einzelnen 4 Bit Adressteile aus  $A_1$ - $A_3$  in eine 12Bit Adresse, die dann parallel ausgegeben wird. Die Verarbeitung der Adresse findet dann durch externe Schaltungen und Speicherchips statt. Ausgegeben werden die Daten durch den 4009, der den Adressinhalt zu entsprechender  $M_1$  und  $M_2$  Zeit in 4Bit-Signale zurückübersetzt und auf den Datenbus legt. Die beiden Komponenten ermöglichen außerdem das Auslesen von Instruktionen aus einem RAM, also einem beschreib- und lesbaren Speicher.[5]

## 3.4 Programmieren mit Assembler

Zum Programmieren des 4004 stehen einem 45 Assembler-Instruktionen zur Verfügung. Diese sind eingeteilt in 16 Maschinenbefehle, 15 Ein-/Ausgabe- & RAM-Befehle und 14 Akkumulator-Gruppen-Befehle. Die folgenden Assembler-Code Beispiele sollen die möglichen Anwendungen einiger Befehle etwas näher bringen:

### 3.4.1 Beispiel 1: 8Bit-Addition und Ein-/Ausgabe

Beim Programmieren eines 6-Bit Prozessors stellt sich zunächst die Frage, wie man mit Zahlen rechnet, die mehr als 4 Bit belegen. Folgendes Codebeispiel zeigt die Implementation eines minimalen 8-Bit Addierers, der 2 Zahlen durch den ersten ROM-Chip empfängt und das Ergebnis durch den ersten RAM-Chip wieder ausgibt. Hierbei wird davon ausge-

gangen, dass die 4-Bit großen Zahlteile jeweils zu den Takten 6, 8, 10 und 12 an den ROM-Ein-/Ausgabe-Ports anliegen (Niederwertigere Bits zuerst):

```

00 LDM 0 //Ersten ROM-Chip auswählen
01 XCH r0
02 LDM 0
03 XCH r1
04 SRC rp0
05 RDR //Erste Zahl einlesen
06 XCH r3
07 RDR
08 XCH r2
09 RDR //Zweite Zahl einlesen
0A XCH r5
0B RDR
0C XCH r4
0D XCH r3 //Niederwertige Bits addieren
0E ADD r5
0F XCH r3
10 XCH r2 //Höherwertige Bits addieren
11 ADD r4
12 XCH r2
13 LDM 0 //Erste RAM-Bank auswählen
14 DCL
15 XCH r3 //Ergebnis ausgeben
16 WMP
17 XCH r4
18 WMP
19 TCC
1A WMP

```

Als erstes muss dafür der RAM-Chip mit SRC ausgewählt werden. SRC verlangt ein Registerpaar als Parameter, in dem die Chipnummer stehen soll. Sei 0 die Chipnummer und R0 und R1 das Registerpaar. Zum Laden einer Zahl in ein Register wird zuerst der LDM-Befehl verwendet, mit dem eine beliebige 4-Bit Zahl in den Akkumulator geladen wird (Niederwertigere Bits zuerst). Infolge darauf kann dann der XCH-Befehl angewandt werden, mit dem der Inhalt eines Registers mit dem des Akkumulators getauscht wird. In diesem Fall also LDM 0, XCH 0, LDM 0, XCH 1 zum Laden der Zahl 0 in die Register 0 und 1. Schließlich SRC 0 und RDR, um mit dem Inhalt der Register die Chipnummer des zu lesenden ROMs zu wählen und in den Akkumulator zu lesen. Zu beachten ist, dass hierbei auch gleichzeitig der erste RAM-Chip der derzeit Aktiven RAM-Bank adressiert wird und somit später nicht mehr mit SRC ausgewählt werden muss.

Zum Speichern des niederen Teils der Zahl in ein Register, kann wieder XCH verwendet werden. Analog dazu werden nun die jeweiligen Teile der Zahlen in die Register R2:R3 und R4:R5 geladen.

Addiert wird mit dem ADD-Befehl, der den Wert eines Registers und des Carry-Bits auf den Akkumulator addiert. Gibt es dabei einen Übertrag, wird das Carry-Bit gesetzt und bei der nächsten Operation mit eingerechnet (Außer es wird davor gelöscht). Dadurch wird garantiert, dass bei der Addition der höherwertigen Bits der mögliche Übertrag bei den niederwertigeren mit einbezogen wird. Ausgegeben werden die Daten durch den ersten RAM Chip mit dem WMP Befehl. Um sicher zu gehen, dass die erste RAM-Bank ausgewählt ist, wählen wir diese mit LDM 0 und DCL aus. TCC lädt das Carry-Bit in den Akkumulator und sorgt im letzten



Schritt dafür, dass auch der mögliche Übertrag mit ausgegeben wird.

Nimmt man zu den Registern noch den RAM als Zwischenspeicher, lassen sich so nahezu beliebig große Zahlen addieren. Berechnungen mit Fließkommazahlen werden vom 4004 nicht unterstützt.

### 3.4.2 Beispiel 2: Sprung-Befehle

Das nächste Beispiel behandelt Konditionen und Sprungbefehle:

```

00 LDM 0          //Register 1 und 2 loeschen
01 XCH r1
02 LDM 0
03 XCH r2
04 start:        //Beginn der Schleife
05 LD r0          //Ueberpruefen, ob r0=r1
06 SUB r1
07 CLC            //Carry-Bit loeschen
08 JCN zero       //Wenn ja, beenden
09             end
0A INC r1
0B ADD r2         //n-r1 auf r2 addieren
0C XCH r2
0D JUN start      //Zum Anfang springen
0E             start
0F end:

```

Angenommen man hat in r0 bereits eine Zahl zwischen 1 und 5 stehen, berechnet der Code die Gaußsche Summenformel für die Zahl n, also die Summe von 1 bis n. Das Ergebnis steht dann in Register 2. Um den Code kurzzuhalten, rechnen wir hier nur mit 4 Bit Zahlen, deshalb ist das Ergebnis auch nur bei  $0 \leq n \leq 5$  korrekt. Umgesetzt ist die Summe hier mit einer Schleife, die von 0 bis n hoch zählt und dabei alle Zahlen kleiner gleich n auf das Ergebnis aufaddiert. Weiter betrachten wir r1 als Zähler, und r2 als Zwischenergebnis während der Berechnung. Diese werden vor Beginn der Schleife auf 0 gesetzt.

Der erste Befehl, der dazu kommt, ist der LD-Befehl, der den Inhalt eines Registers in den Akkumulator liest. In diesem Fall wird der Zähler in den Akkumulator geladen, um ihn mit n zu vergleichen. Ist der Zähler gleich n, wird die Schleife verlassen. Für bedingte Sprünge bietet der 4004 den zwei Byte langen JCN-Befehl mit zwei Parametern. Der erste Parameter ist 4 Bit lang, wobei mit jedem Bit die Kondition spezifiziert werden kann, unter der der Sprung ausgeführt werden soll. Der zweite Parameter gibt die Sprungadresse an. Diese ist auf 8 Bit begrenzt und muss sich daher innerhalb der ROM-Einheit befinden, da die ROM-Nummer noch weitere 4 Bit in Anspruch nehmen würde. In unserem Beispiel subtrahieren wir den Zähler von n und springen, falls der Akkumulator gleich 0 ist. Um dabei sicherzustellen, dass sich die Subtraktion nicht auf die folgenden Rechenoperationen auswirkt, löscht CLC das Carry-Bit. Falls der Zähler ungleich n ist, wird das Zähler-Register mit INC inkrementiert und der Inhalt vom Akkumulator auf das Zwischenergebnis addiert. Mit JUN springen wir wieder zum Anfang der Schleife. Im Gegensatz zu JCN, kann der ebenfalls 2 Byte lange JUN Befehl mit seinem 12 Bit langem Parameter auch auf Speicheradressen zugreifen, die sich auf anderen ROM-Chips befinden.

### 3.4.3 Weitere Befehle

Mit den zwei Beispielen haben wir nun 13 von 45 Befehlen behandelt. Besonders beim i4004 ist noch der DAA-Befehl. Dieser passt den Akkumulator nach einer Berechnung an eine Dezimalzahl an. Steht z.B. 2 im Akkumulator mit Carry Bit 1, wurde hier die Zahl 18 berechnet ( $16 + 2$ ). In diesem Fall würde im Akkumulator die Zahl 2 in 8 umgewandelt werden, das Carry Bit steht nun nicht mehr für den Übertrag von 16, sondern 10 ( $10 + 8$ ).

Neben Sprüngen ist beim Intel 4004 auch eine Art Unterprogramm-Aufruf möglich. Bei Ausführung des JMS-Befehls wird der Code-Pointer im return-Stack gespeichert und wie beim JUN-Befehl zu der Adresse gesprungen, die als Parameter angegeben ist. BBL springt zur alten Adresse zurück und lädt den Parameter in den Akkumulator.

Abschließend lässt sich sagen, dass der 4004 zwar einen sehr mächtigen Befehlssatz hat, jedoch durch seine minimale Registergröße stark eingeschränkt ist. Bis auf ein paar wenige Ausnahmen, wird jeder Befehl in der festgelegten Zeit von einem Befehlszyklus abgearbeitet. Der Prozessor lässt sich daher den RISC-Prozessoren zuordnen.

## 3.5 Der Intellec 4: Eine Entwicklungsumgebung für den i4004



Figure 8: Der Intellec 4[2]

Um das Programmieren für Entwickler einfacher zu gestalten, hat Intel den Intellec für unter anderem den i4004 und i8008 entwickelt. Dieser beinhaltete neben 1 bis 4kB PROM-Speicher (Programmable ROM), 320 bis 2560x4 Bit RAM-Speicher, 4 bis 16x4 Bit Input -und 8 bis 48x4 Bit Output-Ports noch eine 750kHz Clock, sowie interne Netzteile und Kontrolleinheiten, um die MCS-4 Einheit nutzbar zu machen. Der PROM kann im Gegensatz zum normalen ROM nach der Herstellung noch modifiziert werden. Dies ermöglichte jedem Benutzer eine individuelle Programmierung. Geliefert wurde der Intellec noch mit zusätzlicher System-Monitor und Resident-Assembler Software. Zudem standen noch Simulations-Software und in der Programmiersprache FORTAN IV geschriebenen Pakete zur Verfügung, die zusätzlich erworben werden konnten.[3]

Das Interface des Intellec ist aus einer Reihe an LEDs und Knöpfen aufgebaut, durch die man mit dem System interagieren kann (Abb. 8):

#### 4. EINE NEUE TRANSISTORTECHNOLOGIE: DAS RENNEN UM DEN BAU DES ERSTEN MIKROPROZESSORS

Ohne Frage ist der Intel 4004 in der Welt der Prozessoren hoch angesehen und wird oft als Wegbereiter einer neuen "Ära" dargestellt, so wie Intel es selbst betitelt[15]. Bleibt jedoch die Frage, was das Unternehmen tatsächlich für Neuerungen und Innovationen mit seinem Produkt auf den Markt brachte.

Frederico Faggin schreibt in einem selbstverfassten Artikel: SZu dieser Zeit war der Pfad zum Mikroprozessor ein Rennen zwischen den besten im Business, keine revolutionäre Idee[...]"[9]. Er beschreibt die Entwicklung also als ein Rennen, welche Firma zuerst die nötige Transistortechnologie entdeckt, um sie ohne weitere Leistungs- und Kosteneinbußen in einem Chip unterzubringen. Faggin sieht die von ihm selbst ins Leben gerufene Silicon-Gate(SGT) MOS Technologie als eben diesen notwendigen Schritt, zur Single-Chip-CPU. Diese basierte auf der MOSFET(metal-oxide-semiconductor filed-effective transistor) Technologie, die zuvor noch nicht ausgereift genug war, um die derzeit etablierte bipolar Technologie zu ersetzen. Durch die Weiterentwicklung von Faggin und Tom Klein zur Silicon-Gate MOS Technologie(kurz: SGT MOS) im Jahr 1968 bei Fairchild Semiconductor, übertrumpfte sie jedoch die Vorteile der konkurrierenden Technologie und setzte sich auf lange Zeit durch. Die Silicon-Gate Technologie war im Vergleich zur bipolar Technologie energieeffizienter und nahm in einer Schaltung halb so viel Platz weg. Auch was die Aluminium-MOS Technologie betrifft, war SGT zuverlässiger, fünfmal so schnell und hat auf ca. die Hälfte der Fläche gepasst.

"Mit SGT ist eine neue Kategorie von Geräten möglich, wie Arbeitsspeicher (RAMs)[...], Bildsensoren, und Mikroprozessoren Frederico Faggin[9]. Silicon-Gate brachte außerdem den bootstrap load mit sich, der den Bau von Registern und RAM-Einheiten vereinfachte. Kurzgesagt konnte eine SGT-Schaltung, im Vergleich zu bipolar-Schaltungen, mit weniger Transistoren realisiert werden. Ein Flip-Flop, also ein Bit-Speicher, brauchte beispielsweise nun nur noch 10 Transistoren, anstatt von 15(oder mehr).

Resultat war der sogenannte dynamic-random-access-memory (kurz: DRAM). Ursprünglich gar nicht im Entwurf von Busicom enthalten, überarbeitete Ted Hoff die Pläne, und ersetzte den shift-read/write-memory(SRM) und den static RAM (SRAM, innerhalb der CPU) durch den neuen DRAM[9].

Der Fortschritt des Transistors unterstützte nebenbei auch noch die Behauptung von Gordon Moore, der von 1968 bis 1987 in der Geschäftsführung von Intel arbeitete. Dieser stellte im Jahr 1965 fest, dass die Anzahl an Prozessoren in einer Integrierten Schaltung sich seit 1962 jedes Jahr verdoppelte. Beim Intel 4004 belief sich die Zahl der Transistoren auf ca. 2300. 1975 änderte er seine Vorhersage auf eine Verdopplung der Transistoranzahl im Intervall von 2 Jahren. Die Behauptung ist auch noch bis heute gültig und ist als Moore's Law bekannt. Fest steht aber, dass der Zeitpunkt kommt, an dem die Größe eines Transistors sein physikalisches Limit erreicht und Moore's Law nicht mehr zutreffen wird.

Abb. 9 zeigt den handgezeichneten inneren Aufbau des 4004

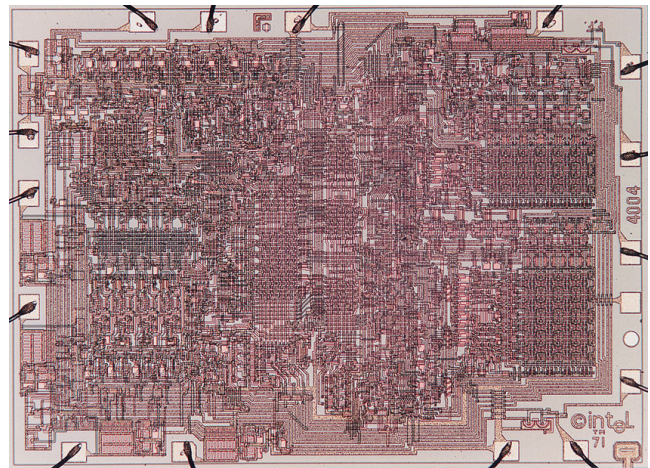


Figure 9: Der innere Aufbau des 4004[10]

mit all seinen Schaltungen.[9] Was damals noch unumgänglich war, um einen Prozessor zu entwickeln, ist mit der heutigen Anzahl an Schaltungen in einer CPU gar nicht mehr möglich.

Trotz dessen scheint die von Texas Instruments angewendete, noch auf Aluminium basierende MOS-Technologie ausgereicht zu haben, um Ziel noch vor Intel zu erreichen. Laut Faggin gab es jedoch zwei Dinge zu tun, um die Entwicklung entscheidend voranzutreiben: Den CPU auf einen Chip zu bringen - **und** ihn dem allgemeinen Markt zugänglich zu machen[9]. So hat Texas Instruments zwar den ersten Schritt, zur Ein-Chip-CPU zuvor erreicht, der zweite Schritt zur kommerziellen Vermarktung bleibt jedoch Intel zuzuschreiben.

Beide Unternehmen beanspruchten den Titel des ersten Mikroprozessors für sich, ohne zu wissen, dass sie schon längst aus dem Rennen ausgeschieden waren. Im Juni 1968 startete die Entwicklung des tatsächlich ersten Mikroprozessors, dem MP944 als Teil des Central Air Data Computer (CADC), und wurde 1970 vollendet. Design wurde der Chip für den F14A TomCat Kampffjet von einem Team, zu dem Mr. Steve Geller und Mr. Ray Holt gehörten. Geller und Holt arbeiteten dabei für die Garret AirResearch Corp unter Vertrag von Grumman Aircraft, dem primären Auftragnehmern der US Navy. Da das Projekt im Geheimen stattfand, wurde die Wahrheit erst mit einem Zeitungsartikel aus dem Wallstreet-Journal aus dem Jahr 1998 bekannt.

Der CADS sollte verschiedene motorisierte Teile des Flugzeugs steuern und Informationen für den Piloten berechnen, die durch Displays im Cockpit ausgegeben werden konnten. Dazu war er an verschiedene Sensoren, Motoren und Anzeigergeräte angebunden, durch die er Informationen lesen und ausgeben konnte. Die Architektur des F14A basiert, so wie der Texas Instruments Prozessor, auf der MOS Transistor Technologie mit Aluminium.

Jegliche mögliche Auswirkungen neuer Technologien des CPUs auf die Industrie, blieben dem F14A verwehrt. Mr. Ray Holt hat zwar ein Dokument über das Design des Chips verfasst, die Veröffentlichung wurde jedoch von der U.S. Navy aus nationalen Sicherheitsgründen nicht erlaubt.[7]

## 5. DIE NACHFOLGE DES INTEL 4004

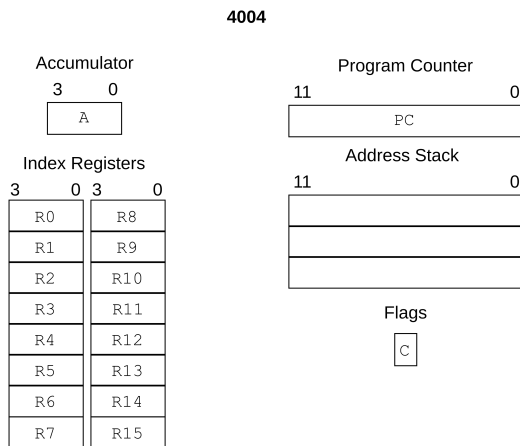


Figure 10: Interne Register des 4004

Die MCS-4 Serie war einer der ersten Schritte in der bis heute über 40 Jahre langen Geschichte der Mikroprozessoren und Intel spielte eine große Rolle während allen diesen Jahren. Die in 1971 veröffentlichte MCS-4 Serie wurde vor allem für das Rechnen mit Binary-Coded Decimals konzipiert. Im Gegensatz zu vielen ihrer Nachfolgern verwendet die MCS-4 Serie eine Harvard Architektur, das heißt, dass Instruktionen und Daten in getrennten Speichern stehen. Die Programm Instruktionen werden im 4001 ROM gespeichert, dessen Inhalt bei der Produktion festgelegt werden muss. Dabei können maximal 4096 8-bit Worte im 4001 adressiert werden. Bis zu 1024 4-bit Datenworte können im RAM (dem 4002) adressiert werden. Der 4004 besitzt 16 Index Register und 1 Akkumulator Register, welche jeweils 4 Bit breit sind. 4 Bit wurde gewählt, da dies die Anzahl an Bits war die nötig war um eine Dezimalziffer darzustellen. Bei einer Clock Frequenz von bis zu 740 kHz können bis zu 92500 Instruktionen pro Sekunde abgearbeitet werden [10].

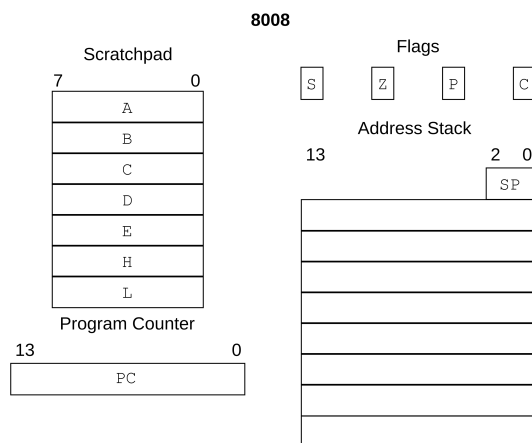


Figure 11: Interne Register des 8008

In 1972 enthüllte Intel die mit 8-Bit Daten arbeitende MCS-8 Serie. Die 8008 ist konzeptionell ähnlich aufgebaut wie die 4004, wurde jedoch auf CRT Terminal Anwendungen, das heißt Verarbeitung 8-bit breiter Zeichen, spezialisiert. Die MCS-8 Serie hat jedoch im Gegensatz zur MCS-4 Serie keinen separaten Adressbereich für Instruktionen und Daten, sondern kann sowohl RAM als auch ROM im selben 14-Bit breiten Adressbereich ansprechen. Der 8008 besitzt 7 Register, darunter 1 Akkumulator Register (A), 4 General Purpose Register (B, C, D, E) und 2 Register die zusammen als Adresse in den Speicher verwendet werden können (H, L). Alle Register sind hierbei 8 Bit breit. Die Konkatenation von H und L zeigt auf ein Wort im Speicher, das ebenfalls als Operand verwendet werden kann (M). Der 8008 wurde mit Maximalfrequenzen von bis zu 800 kHz produziert und kann so bis zu 80000 Instruktionen verarbeiten [8] [11] [13].

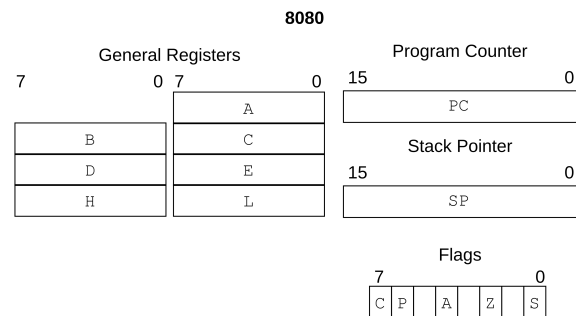
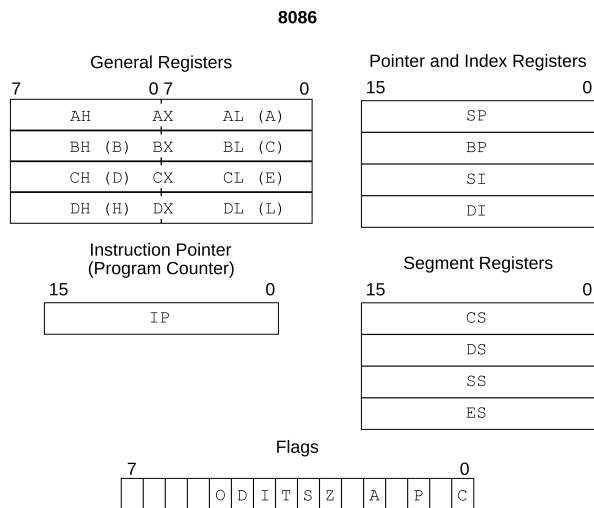


Figure 12: Interne Register des 8080

Nach den relativen Erfolgen der MCS-4 und MCS-8, welche eigentlich für spezielle Anwendungen gedacht waren, entschied Intel einen Mikroprozessor für den entwickelnden Mikroprozessor Markt zu entwerfen. Dabei war es ihnen wichtig, vor allem um Abwärtskompatibilität zu wahren, dass der Instruktionssatz des 8080 eine Obermenge des Instruktionssatzes des 8008 darstellte, was für Assembly-Level Kompatibilität mit dem 8008 erlaubte. Um auch Anwendungen ähnlich des 4004s auszuführen wurden auch Instruktionen für die Verarbeitung von Packed BCD Daten hinzugefügt. Der in 1974 veröffentlichte 8080 erweiterte den Adressraum auf 64 KiByte mit 16-Bit Adressen. Der Registersatz des 8008 wurde größtenteils übernommen, wobei jedoch, um 16-bit Berechnungen durchzuführen, manche Register zu Paaren zusammengesetzt wurden. Diese Paare sind BC, DE und HL. Das A Register wird wie auch beim 8008 als Operand und Ziel in logischen und arithmetischen Operationen verwendet. Das HL Registerpaar adressiert ebenfalls wieder ein Wort im Speicher. Im Gegensatz zum 8008 und 4004 die einen stark begrenzten Stack auf dem Prozessor haben, besitzt der 8080 ein Stackpointer-Register das auf den Stack im Speicher zeigt. Um den 8080 Mikroprozessor einfach zu halten sind nicht alle Register zu allen Aufgaben fähig. Dies erschwerte jedoch das Programmieren für den 8080 [13] [12].

endfigure

In 1978 veröffentlichte Intel den 8086. Dieser baute, um weiter Abwärtskompatibilität zu wahren, auf dem 8080 auf, erweiterte diesen jedoch in vielen Hinsichten. Ein wichtiger Punkt den der 8086 über dem 8080 verbesserte war die Symmetrisierung des Registersatzes. Das Registermodell, das Registern unterschiedliche Funktionen erlaubt macht



**Figure 13: Interne Register des 8086**

das Programmieren um einiges komplexer und so wurden die Register des 8086 größtenteils gleichmächtig gestaltet. Der 8086 ist zu 16-bit Arithmetik fähig. Der Adressraum wurde auf 20-bit Adressen erweitert, genug um 1 MiByte an Speicher zu adressieren. Der 8086 besitzt jedoch nur 16-Bit Arithmetik. Um den gesamten Adressraum so zu adressieren führte Intel 4 Segmentregister ein. Diese, genannt Code Segment Register (CS), Data Segment Register (DS), Stack Segment Register (SS) und Extra Segment Register (ES), sind auch 16-Bit breit. Beim Speicherzugriff über ein solches Segment wird der Wert des jeweiligen Segment Registers um 4 Bit nach links verschoben. Dies stellt die Basisadresse des Segments dar. Die Offset Adresse des zu adressierenden Wortes wird auf die Basisadresse des Segments addiert um die physische Adresse des Wortes im Speicher zu finden. Zusätzlich zu dem Stackpointer (SP) wurden noch Register für einen Base Pointer (BP), Source Index (SI) und Destination Index (DI) eingeführt. Der Base Pointer wurde meist verwendet um bei Funktionsaufrufen einen Funktionsrahmen aufzubauen, was rekursive Funktionen erleichtert. Source Index und Destination Index werden vor allem für String Instruktionen verwendet. Die Arithmetischen Register wurden auf 4 16-Bit breite Register reduziert (AX, BX, CX, DX). Dabei sind jedoch die jeweils oberen und unteren 8 Bit des Registers separat adressierbar (AH, AL, BH, BL, CH, CL, DH, DL). Diese (mit Ausnahme von AH) stellen jeweils ein Register des 8080 dar. Der 8086 wurde auch mit einigen neuen Instruktionen ausgestattet. Eine wichtige Erweiterung des Befehlssatzes war das Hinzufügen von Multiplikations- und Divisionsinstruktionen. Neben diesen stehen auch Instruktionen zur Verarbeitung von Byte-Strings zur Verfügung. Diese können durch Präfixe bei Speziellen Byte-String Anwendungen ganze Schleifen mit einer Instruktion darstellen. Des Weiteren wurden die BCD Instruktionen des 8080 um Instruktionen zum Verarbeiten von Unpacked BCD erweitert. Unpacked BCD enthalten jeweils eine Ziffer pro Byte, das zusätzlich mit einem Offset versehen werden kann. Ein Beispiel für Unpacked BCD wäre eine Zahl aus ASCII Zeichen. Der 8086 kann arithmetische Operationen mit solchen Zahlen durch-

führen. Mit einer Clockfrequenz von bis zu 10 MHz konnte der 8086 bis zu 750000000 Instruktionen pro Sekunde durchführen [13] [14].

Binary Coded Decimals (BCD) Binary Coded Decimals, oder BCD (zu Deutsch dualkodierte Dezimalziffer) kodieren eine Dezimalziffer in 4-Bit. Eine 3-Bit Binärzahl kann nur 8 verschiedene Ziffern darstellen. Daher werden mindestens 4 Bit benötigt um eine Dezimalziffer zu repräsentieren [16]. Bei einer Addition, wie sie beim 4004 üblich war, werden die einzelnen Nibble der Reihe nach von niederwertigen Nibblen zu höherwertigen Nibblen binär addiert. Nach jeder solchen Addition wird eine DAA (Decimal Adjust Accumulator) Instruktion ausgeführt um die Invarianz nur Ziffern 0 bis 9 in einem Nibble zu halten und den korrekten Übertrag fortzuführen. Wenn der Wert im Akkumulator größer als 9 ist oder das Carry Flag gesetzt ist, wird, um das korrekte Ergebnis zu erhalten, der Akkumulator um 6 inkrementiert und das Carry Flag gesetzt [10]. Da ein Carry im Binären einen Wert von 16 darstellt im Dezimalen aber nur den Wert 10 besitzt müssen diese übrigen 6 noch berücksichtigt werden. Mit den erweiterten Fähigkeiten von modernen Gleitkommarechen-einheiten wurden BCD größtenteils obsolet. In der Finanzindustrie, Digitaluhren und einfachen Taschenrechnern werden diese jedoch noch immer verwendet [16].

Die folgenden Graphen sollen abschließend noch die Entwicklung des Prozessors seit 1970 darstellen:



## 6. REFERENCES

- [1] Intel 4004 architecture overview · CodeAbbey/intel4004-emu Wiki.
- [2] IntelIntellec\_4\_mod\_40.JPG (4752x3168).
- [3] IntelIntellec\_8\_intellec\_4\_brochure.pdf.
- [4] MCS-4 - Intel - WikiChip.
- [5] MCS-4\_manual.pdf.
- [6] MCS4\_data\_sheet\_nov71.pdf.
- [7] World's First Microprocessor.
- [8] W. Aspray. The Intel 4004 microprocessor: what constituted invention? *IEEE Annals Hist. Comput.*, 19(3):4–15, Sept. 1997.
- [9] F. Faggin. The MOS Silicon Gate Technology and the First Microprocessors. page 48.
- [10] F. Faggin, M. Hoff, S. Mazor, and M. Shima. The history of the 4004. *IEEE Micro*, 16(6):10–20, Dec. 1996.
- [11] S. Mazor. The history of the microcomputer-invention and evolution. *Proc. IEEE*, 83(12):1601–1608, Dec. 1995.
- [12] S. Mazor. Intel 8080 CPU Chip Development. *IEEE Annals Hist. Comput.*, 29(2):70–73, Apr. 2007.
- [13] Morse, Raveiel, Mazor, and Pohiman. Intel Microprocessors–8008 to 8086. *Computer*, 13(10):42–60, Oct. 1980.
- [14] S. Morse, W. Pohlman, and B. Ravenel. The Intel 8086 Microprocessor: a 16-bit Evolution of the 8080. *Computer*, 11(6):18–27, June 1978.
- [15] R. Noyce and M. Hoff. A History of Microprocessor Development at Intel. *IEEE Micro*, 1(1):8–21, Feb. 1981.
- [16] H.-J. Schneider. *Lexikon Informatik und Datenverarbeitung*. R. Oldenbourg Verlag, München / Wien, 1986.

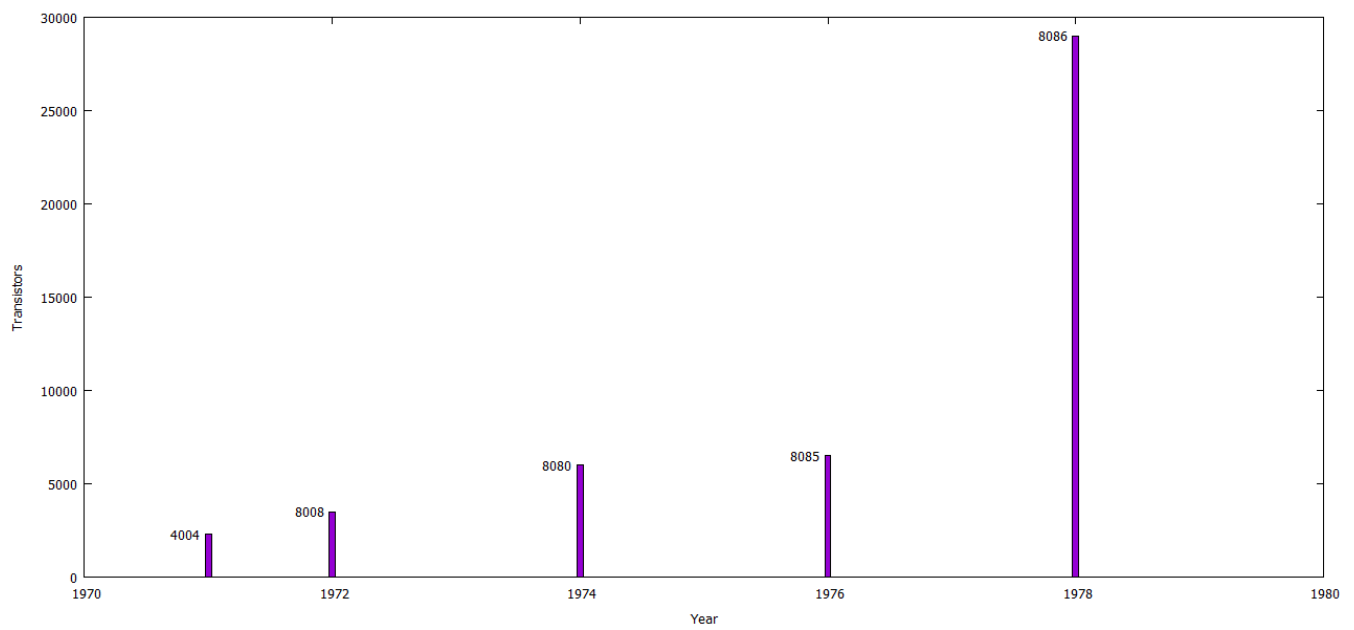


Figure 14: Entwicklung der Transistors

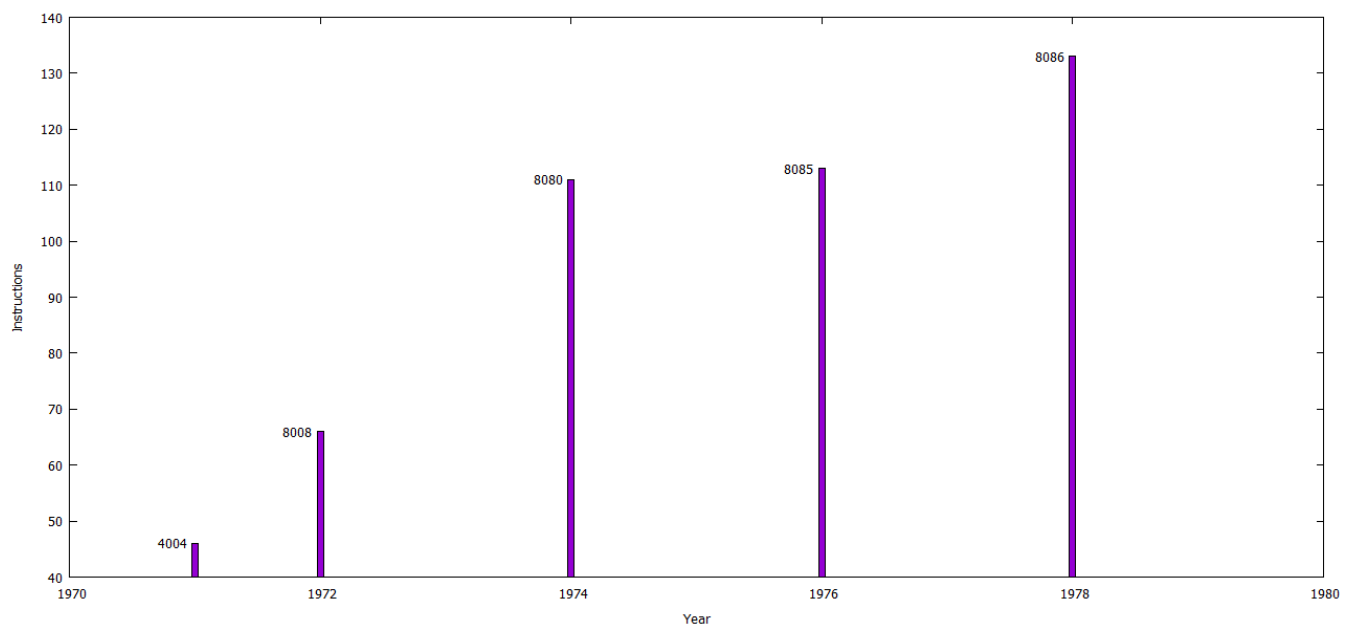


Figure 15: Entwicklung der Instruktionsanzahl in einem Prozessor

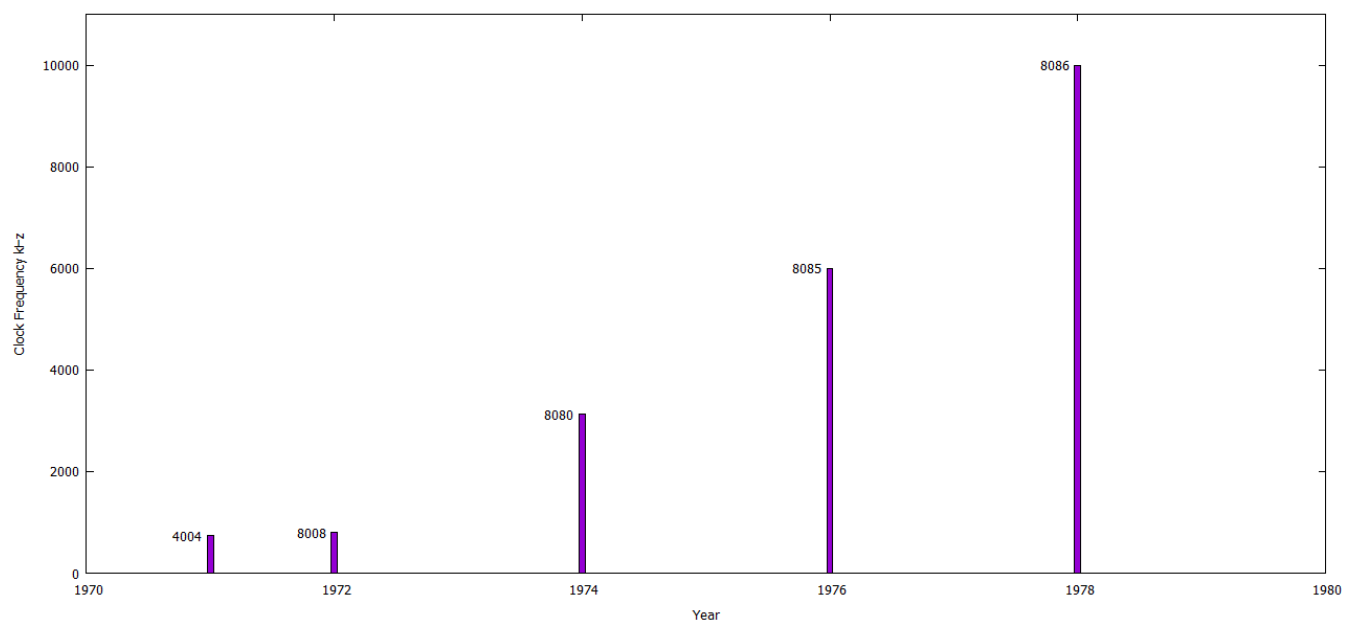


Figure 16: Entwicklung der Taktfrequenz