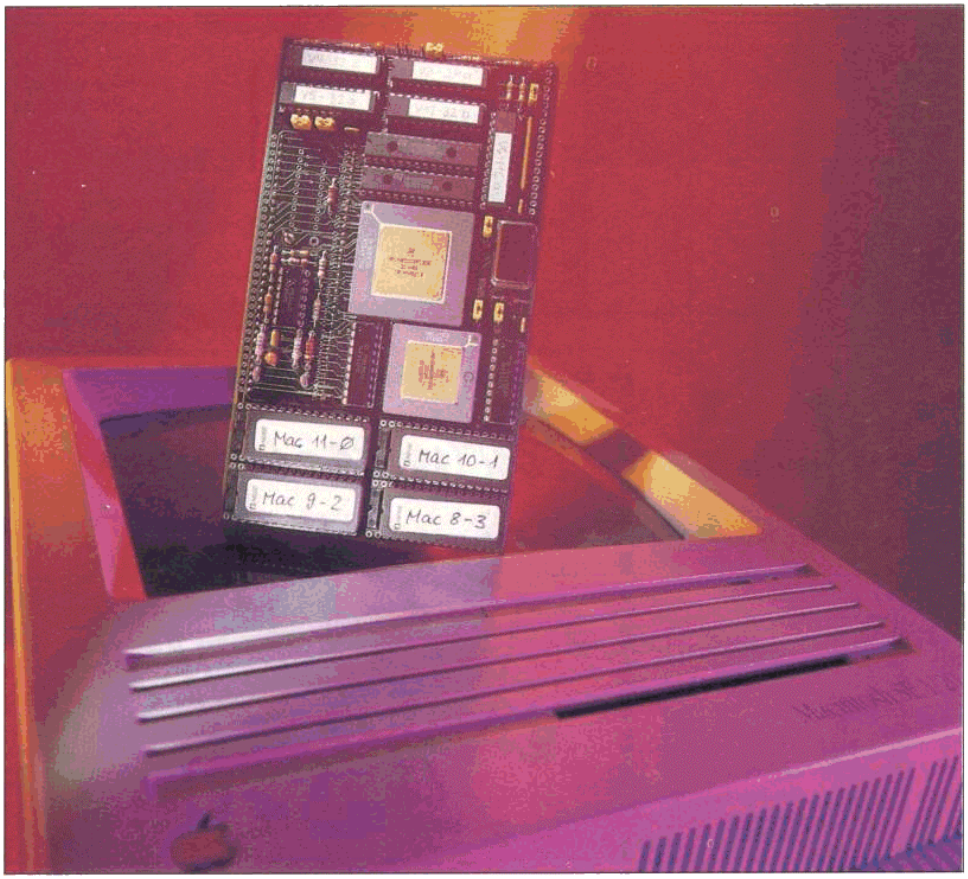
Die PAK-68 im Amiga.

Scans der Prozessor-Austausch-Karte aus der Zeitschrift c’t



Doppel-PAK

PAK-68/3 mit Cache und 68020/030 für 68000-Rechner

Holger Zimmermann

Ganz im Zeichen der fünften Zweierpotenz präsentiert sich die nunmehr dritte Auflage unseres Dauerbrenners PAK-68: 32 MHz, 32 Bit und 32 KByte Cache sind die gewichtigen Gründe, den optimalen 68000-Beschleuniger zum Selbstbau vorzustellen.

Die PAK-68 hat seit ihrer (nicht ganz ohne Komplikationen verlaufenen) Geburt im Juli 1987 viele Freunde gefunden. Anfangs [1] noch mit gemächlichen 8 MHz, später mit 16 MHz [2] und nun mit Taktfrequenzen fast nach Belieben getaktet, bleibt die Prozessor-Austauschkarte der konsequenteste Weg, einem vorhandenen Rechner zeitgemäße Leistung einzuhauchen.

Und die ist nicht von Pappe: Neben der eklatanten Beschleunigung kommen Atari-ST-Freunde in den Genuß der modernen TOS-Versionen 2.06 und (mit kleinen Patches) 3.06 im 32-Bit-Zugriff; mit der 68030-Version gibt's zusätzlich eine PMMU fürs Multi-TOS. Kompakt-Macs profitieren unter dem ressourcenfressenden System 7 besonders von der – je nach Ausstattung – Verdrei- bis Versechsfachung der Arbeitsgeschwindigkeit. Dabei blieb es unser Bestreben, so 68000-kompatibel wie möglich zu bleiben, damit auch kritische Rechner (Atari mit IMP-Chipsätzen, Amiga) nicht außen vor bleiben – obwohl es sicher immer noch Applikationen und Erweiterungen geben wird, die nicht 68020/030-konform aufgebaut sind. Erkundigen Sie sich also vor der PAK-Anschaffung, ob Ihr Liebligs-Editor auch auf größeren Maschinen (Atari TT, Falcon, Mac II) 'spielt'.

Umweltfreundlich

Ein 68020 mit 8 oder 16 MHz ist aber mittlerweile auch nicht mehr ganz 'state of the art'. Ganz oben auf der Wunschliste stand daher ein 68030, gefolgt von höherer Taktfrequenz und einem Second-Level-Cache. Zum Glück gibt es heute preiswerte Bauteile, die eine Implementation derlei Begierden ohne nennenswerte Hardwareverrenkungen und ohne Kontoaustrocknung ermöglichen. Zum alten Eisen respektive Plastik gehört Ihr ST oder Schuhkarton-Mac also noch lange nicht.

Hier ist sie also, die PAK-68/3. Bei nahezu gleichen Abmessungen wie die PAK/2 hat sich 'unter der Haube' einiges getan: Wahlweise eine 68020- oder 68030-CPU mit 16 bis 33 MHz Taktfrequenz bei 8 MHz Systemtakt, eine 68881- oder 68882-FPU, 256 oder 512 KByte ROM sowie ein 32 Bit breiter und 32 KByte großer Second-Level-Cache.

Beim Nachbau der PAK/3 steht eine schwerwiegende Entscheidung gleich zu Anfang an: Soll es ein 68020 oder ein 68030 sein? Da es für beide Prozessoren jeweils eine eigene Platinenversion gibt, ist kein nachträglicher Wechsel möglich. Gewissermaßen als Ausgleich dazu können aber alle anderen Features auch nachträglich durch Austauschen der GALs hinzugefügt oder weggelassen werden. In Anbetracht der mitunter sehr preiswert angebotenen Gebrauchtprozessoren (siehe Kleinanzeigen) empfehlen wir bei nicht schon vorhandener alter PAK-68 die 030er-Version, die gegenüber der 020er bei gleicher Taktfrequenz einen Geschwindigkeitsvorteil von rund 15 Prozent bringt. Ansonsten können Sie die meisten Bauteile der alten PAK weiterverwenden, die GALs allerdings nur, wenn Sie eine geeignete Programmiereinrichtung besitzen. Natürlich muß der Prozessor die gewünschte Taktfrequenz verkraften.

Große Pläne

Der Schaltplan der 030er-Version der PAK/3 deckt im Prinzip die 020er mit ab. Da beim 68020 die Anschlüsse /STERM, /CIIN, /CIOUT, /MMUDIS, /CBREQ und /CBACK nicht vorhanden sind, entfallen in diesem Fall die entsprechenden Leitungen mitsamt zugehörigen Pullups und Jumpern. Die Pinbelegung beider CPUs ist leider deutlich unterschiedlich, was sich im Schaltplan nur im Kleingedruckten bemerkbar macht, beim Platinenlayout aber im Endeffekt zu den beiden unterschiedlichen Platinenversionen geführt hat. Bis auf die GALs, R1, R2 und eben

PAK-68/3-Jumper

Jumper	Name		Funktion
J1	MMU disable	gesteckt	MMU disabled
		offen	enabled
J2	CPU Cache disable	gesteckt	CPU-interner Cache disabled
		offen	enabled
J3	FPU enable	gesteckt	FPU aktiviert
		offen	desaktiviert oder nicht vorhanden
J4	L2 Cache disable	gesteckt	Second-Level-Cache ausgeschaltet
		offen	eingeschaltet
J5	PAK enable	gesteckt	PAK ist am 68000 Bus aktiv
		offen	alle Ausgänge hochohmig
J6	BR	gesteckt	Normalbetrieb
		offen	Betrieb mit 68000 auf der PAK (extra Verdrahtung notwendig)
J7	ROM	1-2	ROM auf der PAK aktiviert
		2-3	desaktiviert oder nicht vorhanden
J8	ROM_CS	1-2	/ROM_CS auf GND
			für langsame EPROMs
		2-3	mit /ROM_OE verbunden
			spart Strom
J9	CPUCLK	1-2	CPU-Takt 32 MHz (asynchron)
		2-3	16 MHz (synchron)
J9	FPUCLK	1-2	FPU-Takt 32 MHz (asynchron)
		2-3	16 MHz (synchron)
			(die FPU sollte nicht langsamer laufen als die CPU)
J11	Cache Control	1-2	L2-Cache wird über die MMU kontrolliert
		2-3	Umschaltung über Schreibzugriff auf das ROM
		offen	L2-Cache immer an

die CPU sind beide Platinenversionen absolut gleich bestückt.

Das State-Machine-Konzept der PAK/2 hat sich in der Praxis bewährt, so daß es im Kern auch für die PAK/3 beibehalten werden konnte. CPU, FPU und die ROMs sind weitgehend wie beim Vorgänger verschaltet. Die synchrone Takterzeugung mit einem 74F86 als Verdoppler für den 8-MHz-Systemtakt ist ebenfalls identisch. Auch der Quarzoszillator war schon auf der PAK/2 zu finden, dort aber lediglich für die FPU zuständig. Bei entsprechender GAL-Bestückung kann die PAK nun auch asynchron zum Systemtakt (getestet bis 33 MHz) betrieben werden; mehr dazu in der nächsten c't. Die GALs U4, U5 und U6 haben ihre Funktion (und ihre Namen) weitgehend beibehalten. Bei den Gleichungen für U5 wird man noch einiges wiedererkennen, während bei U4 kaum ein Stein auf dem anderen geblieben ist. Gänzlich neu sind GAL U1, das die Kontrolle über die lokalen Buszyklen übernimmt (also bei ROM-Read und bei Cache-Hit) und GAL U2, das den Cache koordiniert.

Der 68020 hatte bereits einen Cache für Befehle in der CPU integriert, beim 68030 ist noch ein Datencache dazugekommen. Leider sind die internen Caches mit 256 Bytes nicht besonders üppig bemessen. Benchmarks benutzen häufig kurze Schleifen zum Testen der Performance, hier liegt der CPU-Cache noch gut im Rennen; in der Praxis bleibt davon jedoch nicht allzuviel übrig. Die PAK/3 hat deshalb einen 32 KByte großen Second-Level-Cache spendiert bekommen, und wer ihn mal probehalber ausschaltet, merkt, wie richtig diese Entscheidung war. Typische Anwendungen laufen bei Maximalausbau (32 MHz plus L2-Cache) in wenigstens sechsfacher 68000-Geschwindigkeit, einige Benchmarks zeigen gar zweistellige Performance-Gewinne. Im Vergleich zur alten PAK, die rund 250 Prozent der Leistung eines 68000 brachte, ein nennenswerter Fortschritt.

Hucke-PAK

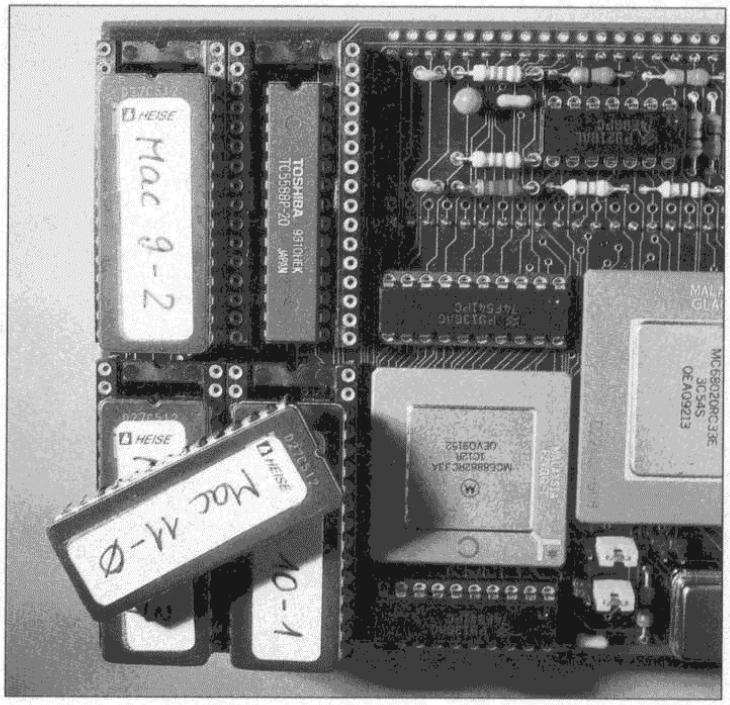
Für den L2-Cache sind zwei Tag-RAMs, vier schnelle SRAMs und zwei Datenbustreiber hinzugekommen. Die SRAMs (schmale Bauform, 300 mil) benötigen weitgehend dieselben Signale wie die ROMs (Breite 600 mil), so daß sich hier eine zweigeschossige Bauweise anbot. Nur dadurch war es auch möglich, die Abmessungen der PAK/2 beizubehalten. Nebenbei konnten wir auch einen kleinen Schönheitsfehler beseitigen: Die PAK überdeckt nun nicht mehr den PDS-Steckplatz im Mac SE.

Die vier SRAMs (je 8 KBit × 8, also 8 KByte) bilden zusammen einen 32 KByte großen Cache-Speicher, auf den die CPU sehr schnell und in voller Breite zugreifen kann. GAL U2 sorgt dafür, daß die aus dem Hauptspeicher geholten Daten sozusagen nebenbei im Cache abgelegt werden. Daten beispielsweise von der Hauptspeicheradresse 1 landen so auch im Cache unter Adresse 1. Da der Cache aber nur 32 768 Byte umfaßt, landen auch Daten von der Hauptspeicheradresse 32768+1 im Cache unter Adresse 1, wobei der vorherige Inhalt des Cache natürlich überschrieben wird. Wie kann das gutgehen?

Schildchen-Speicher

An dieser Stelle kommen die Tag-RAMs ins Spiel, das sind zunächst einmal ebenfalls statische RAMs mit 8K × 8, deren Datenleitungen aber an die CPU-Adreßleitungen A15 bis A21 angeschlossen sind. Jedesmal, wenn Daten in den Cache übernommen werden, kommen die dazugehörigen oberen Adreßbits im Tag-RAM zu liegen. Zu jedem einzelnen Datenwort im Cache steht hier also die Information, zu welcher 32-KByte-Page des Hauptspeichers der Inhalt des Cache gehört.

Wenn die CPU einen Lesezugriff initiiert, wird das Tag-RAM jetzt nicht ausgelesen, sondern im sogenannten Compare-Modus betrieben, wobei es die von außen angelegten Adreßbits A15 bis A21 mit den gespeicherten Adreßbits vergleicht. Der dazu notwendige Komparator ist im Tag-RAM integriert und meldet das Ergebnis über den Anschluß MATCH nach außen, im Falle der Übereinstimmung erhält man hier *high*-Pegel. Das ist das Zeichen für die CPU und ihre Read-Logik, daß sich die gesuchten Daten bereits im Cache befinden. Meldet das Tag-RAM keinen Treffer (MATCH ist *low*), dann gehört das Datenwort im Cache zu einer anderen Page, woraufhin die CPU jetzt doch im Hauptspeicher nachsehen muß. In der Hoffnung, das gerade geholte Datenwort werde in naher Zu-



Die Cache-SRAMs liegen unter den EPROM-Fassungen. Probieren Sie vor dem Einsatz von SRAM-Sockeln aus, ob die 'Türmchen' in Ihren Rechner passen. Vor allem im Kompakt-Mac geht es recht eng zu.

GAL-Bestückung

U1	P12_16	68020 mit 16 MHz
	P12_32	68020 mit 32 MHz
	P13_16	68030 mit 16 MHz
	P13_32	68030 mit 32 MHz
U2	P2_JP	für Mac, L2-Cache wird über Jumper geschaltet bleibt im Normalfall immer eingeschaltet
	P2_RST	für Atari ST, L2-Cache bleibt nach Reset bis zum ersten DMA-Zugriff ausgeschaltet
U4	P4_16	für 16 MHz
	P4_32	für 32 MHz
U5	P5_16	für 16 MHz
	P5_32	für 32 MHz
U6	P6_MAC	für Mac
	P6_ST	für Atari

Beispiel: Sie wollen im Atari eine 030-PAK mit 16 MHz einsetzen. Benötigt werden also P13_16, P2_RST, P4_16, P5_16 und P6_ST.

Stückliste

Halbleiter

U8	MC68020/68030, Taktfrequenz ab 16 MHz	
U22	74F86	
U1, U2, U4, U5, U6	GAL 20V8-15 Prog.	
U23	Quarz-Osz. 32 MHz	(Option 32 MHz)
U9	MC68882	(Option FPU)
U10, U11, U12, U13	27C010-120 oder 27C512-120	(Option ROM)
U14, U15, U16, U17	Cypress CY7C185-25, oder Toshiba TC5588-25	(Option Cache)
U18, U19	SGS MK48S74-25, oder SGS MK48S80-25	(Option Cache)
U20, U21	74F541	

Widerstände

RN1	Array 4k7 × 8
R1, R4, R7, R10, R23…R27	4k7
R8	2k2
R2, R5, R6	1k
R31, R32	330R
R41, R43	560R
R44…R47	33R

Kondensatoren

C1…C15	100n ker. RM2.5
C20…C24	Ta 10uF/16V

Sonstiges

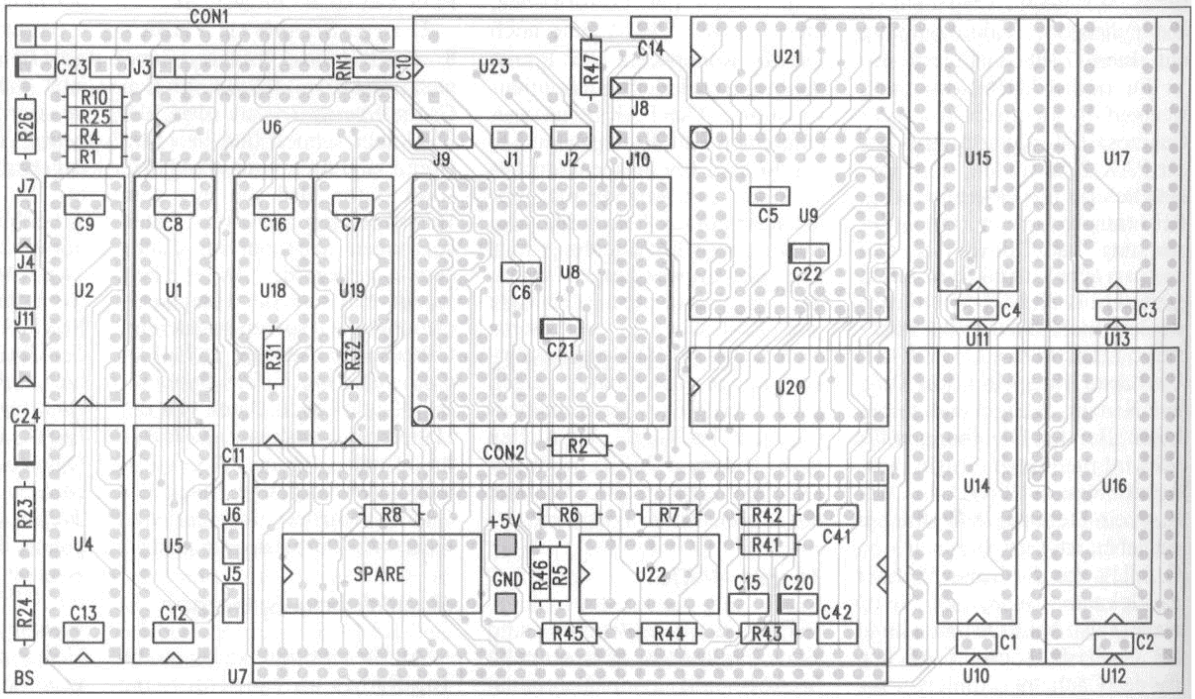
1 IC-Fassung 128pol. PGA (evt. 124pol.)
1 IC-Fassung 68pol. PGA
1 Adapterfassung 64pol. DIL (oder SIL-Stecker o.ä.)
8 Fassung 16pol. SIL für ROMs oder
4 IC-Fassungen 32pol.
6 IC-Fassung 28pol. DIL schmal (oder 2 × 14er)
5 IC-Fassung 24pol. DIL schmal
2 IC-Fassung 20pol. DIL
2 IC-Fassung 14pol. DIL
1 Platine PAK-68/3-20 oder PAK-68/3-30
Pfostenleisten einreihig, Jumper nach Bedarf

kunft nochmals gebraucht, legt es die Steuerlogik dann auch gleich im Cache ab.

Wegen des 16 Bit breiten Sytembusses muß der Cache wortweise verwaltet werden, für den 32 Bit breiten Cache der PAK werden also zwei Tag-RAMs benötigt. Die sieben Adreßbits, die das Tag-RAM auswertet, decken 128 Pages, also den Adreßraum von 0 bis 4 MByte ab. Im Bereich oberhalb $40 0000 ist der Second-Level-Cache unwirksam, was bei den in Frage kommenden Macs und Ataris keine Einschränkung bedeutet; die CPU hat das ROM auf der PAK ja ohnehin im direkten Zugriff.

Bei der Bestückung sollten Sie in der Reihenfolge der Optionen vorgehen und jede Variante gründlich testen.

Das achte Bit der Tag-RAMs bildet das Valid-Bit und liegt daher an V_{CC}. Um alle Daten im Cache für ungültig zu erklären (z. B. bei DMA-Zugriffen oder Reset), werden die Tag-RAMs über den Anschluß /TCLR gelöscht und damit alle Bits auf *low* gesetzt. Damit er-



Gleicher Plan für zwei Versionen: Bis auf die CPU sind beide neuen PAKs identisch.Der 64polige Sockel zur Verbindung mit der Hauptplatine gehört auf die Lötseite in die DIL-Reihen, die zur Platinenmitte weisen (siehe auch Detailfoto). R1 und R2 werden nur bei der 030-Version bestückt.

```
PAK 68/3, GAL 1:  CPU-Clk State Machine für 68020 mit 16 MHz
26-08-93   V12_16a   Holger Zimmermann @ PE
%ID
      P12_16
%TYP
      GAL20V8A
%PINS
 CLK  mat2  16m  a1  !as_00  !tclr  !wr  !as_20  !prom  !csp19  g1p11
 !OE  !dram  !roe  !as_21  !doe  !as_22  !sterm  !cyc_00
 !dsack1  !dsack0  mat0
%LOGIC

 sterm.OE  = GND;

 as_21.OE  = VCC;
 as_21     = as_20 * 16m
           + as_21 * !16m
           + as_20 * as_21;

 as_22     = as_20 * as_21 * !16m
           + as_22 * !16m
           + as_20 * as_22;

 doe      = as_20 * !as_00 * dram * !tclr * !wr * mat0 * mat2 * as_22
          + as_20 * !as_00 * dram * !tclr * !wr *   a1 * mat2 * as_22;

 !cyc_00   = !as_21
           + csp19
           + !as_00 * dram * !tclr * !wr * mat0 * mat2
           + !as_00 * dram * !tclr * !wr *   a1 * mat2
           + prom * !wr;

 dsack0.OE = as_21 * !cyc_00 * !csp19;
 dsack0    = as_20 * dram * !tclr * !wr * mat0 * mat2 * 16m
           + as_20 * dram * !tclr * !wr *   a1 * mat2 * 16m
           + as_20 * dram * !tclr * !wr * mat0 * mat2 * dsack1
           + as_20 * dram * !tclr * !wr *   a1 * mat2 * dsack1
           + as_20 * prom * !wr * 16m
           + as_20 * prom * !wr * dsack1;

 dsack1.OE = as_21 * !cyc_00 * !csp19;
 dsack1    = as_20 * dram * !tclr * !wr * mat0 * mat2 * 16m
           + as_20 * dram * !tclr * !wr *   a1 * mat2 * 16m
           + as_20 * dram * !tclr * !wr * mat0 * mat2 * dsack1
           + as_20 * dram * !tclr * !wr *   a1 * mat2 * dsack1
           + as_20 * prom * !wr * 16m
           + as_20 * prom * !wr * dsack1;

 roe       = as_20 * prom * !wr * !OE;

%END
```

```
PAK 68/3, GAL U2:  Cache Control Logic, L2-Cache über Jumper schaltbar
08-09-93   V2_JPa     Holger Zimmermann @ PE
%ID
      P2_JP
%TYP
      GAL20V8A
%PINS
 !rom  mat2  a0  a1  !as_00  !bgack_20  !wr  !as_20
 !dsack  !reset  siz1
 siz0  !dram  !twe2  !dwe3  !dwe1  !c2off  !dwe2  !tclr
 !dwe0  !twe0  mat0
%LOGIC

 dwe0.OE = VCC;
 dwe0  = dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * !a0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat0 * !a1 * dsack
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe0 * !a1 * dsack;

 dwe1  = dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * siz1
       + dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * a0
       + dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * !siz0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat0 * !a1 * dsack
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe1 * !a1 * dsack;

 dwe2  = dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * !a0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat2 *  a1 * dsack
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe2 *  a1 * dsack;

 dwe3  = dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * siz1
       + dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * a0
       + dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * !siz0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat2 *  a1 * dsack
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe3 *  a1 * dsack;

 twe0  = dram * !tclr * as_20 *  as_00 * !wr *  dwe0 * !a1 * dsack
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe1 * !a1 * dsack;

 twe2  = dram * !tclr * as_20 *  as_00 * !wr *  dwe2 *  a1 * dsack
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe3 *  a1 * dsack;

 tclr  =  as_20 * tclr
       + !as_20 * bgack_20
       + !as_20 * reset
       + !as_20 * c2off;

 c2off.OE = GND;                                    'SLC off per Jumper

%END
```

```
PAK 68/3, GAL U6:  Adressdekoder, Atari ST
16-09-93   V6_STc1   Holger Zimmermann @ PE
%ID
   P6_ST
%TYP
   GAL20V8A
%PINS
 !vpa  fc0  fc1  !as_20  a21  a20  a17  a18  a16  a19  a22
 !berr_00  a23  !avec  !dram  !ciin  !word  !fpucs  !rom  !csp19
 !berr_20  !jp3
%LOGIC

 berr_20.OE = VCC;
 berr_20    = as_20 * fc1 * fc0 * !a19 * !a17              'PMMU, BKPT
            + as_20 * fc1 * fc0 * !a19 *  a17 * !jp3     'FPU disabled
            + as_20 * berr_00;

 fpucs      = fc1 * fc0 * !a19 * !a18 *  a17 * !a16 * jp3;
              'FPU enabled

 avec       = fc1 * fc0 * vpa;

 csp19      = fc1 * fc0 * !a19;

 dram       = !fc0 * !a23 * !a22
            + !fc1 * !a23 * !a22;

 rom        = a23 * a22 * a21 *  a20 *  a19 *  a18 * !a17    'TOS 1.04
            + a23 * a22 * a21 *  a20 *  a19 *  a18 * !a16    'TOS 1.04
            + a23 * a22 * a21 * !a20 * !a19;                 'TOS x.06

 word       = a23 * a22 * a21 *  a20 *  a19 *  a18 * !a17
            + a23 * a22 * a21 *  a20 *  a19 *  a18 * !a16
            + a23 * a22 * a21 * !a20 * !a19
            + !fc0 * !a23 * !a22
            + !fc1 * !a23 * !a22;

 !ciin      = a23 * a22 * a21 *  a20 *  a19 *  a18 * !a17         'ROM
            + a23 * a22 * a21 *  a20 *  a19 *  a18 * !a16         'ROM
            + a23 * a22 * a21 * !a20 * !a19                       'ROM
            + !fc0 * !a23 * !a22                                  'RAM
            + !fc1 * !a23 * !a22;                                 'RAM

%END
```

```
PAK 68/3, GAL U2:  Cache Control Logic, L2-Cache nach Reset aus bis 1. DMA
18-09-93   V2_RSTb    Holger Zimmermann @ PE
%ID
      P2_RST
%TYP
      GAL20V8A
%PINS
 !ciout  mat2  a0  a1  !as_00  !bgack_20  !wr  !as_20
 !dsack  !reset  siz1
 siz0  !dram  !twe2  !dwe3  !dwe1  !c2off  !dwe2  !tclr
 !dwe0  !twe0  mat0
%LOGIC

 dwe0.OE = VCC;
 dwe0  = dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * !a0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat0 * !a1 * dsack *
         !ciout
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe0 * !a1 * dsack *
         !ciout;

 dwe1  = dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * siz1
       + dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * a0
       + dram * !tclr * as_20 *  as_00 *  wr *  mat0 * !a1 * !siz0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat0 * !a1 * dsack *
         !ciout
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe1 * !a1 * dsack *
         !ciout;

 dwe2  = dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * !a0
       + dram * !tclr * as_20 *  as_00 * !wr * !mat2 *  a1 * dsack *
         !ciout
       + dram * !tclr * as_20 *  as_00 * !wr *  dwe2 *  a1 * dsack *
         !ciout;

%END
```

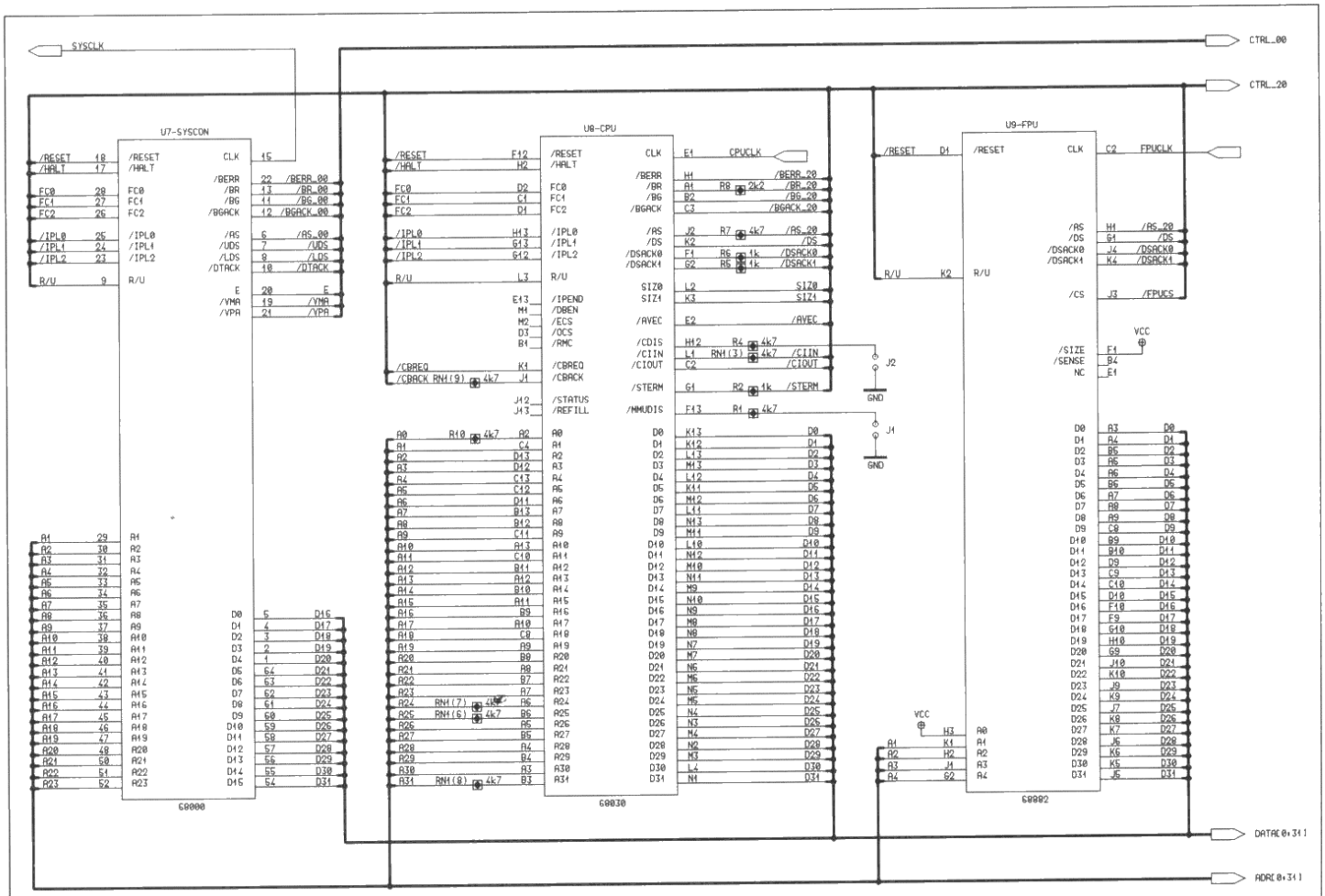
```
26  dwe3  = dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * siz1
27        + dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * a0
28        + dram * !tclr * as_20 *  as_00 *  wr *  mat2 *  a1 * !siz0
29        + dram * !tclr * as_20 *  as_00 * !wr * !mat2 *  a1 * dsack *
            !ciout
30        + dram * !tclr * as_20 *  as_00 * !wr *  dwe3 *  a1 * dsack *
            !ciout;
31
32  twe0  = dram * !tclr * as_20 *  as_00 * !wr *  dwe0 * !a1 * dsack *
            !ciout
33        + dram * !tclr * as_20 *  as_00 * !wr *  dwe1 * !a1 * dsack *
            !ciout;
34
35  twe2  = dram * !tclr * as_20 *  as_00 * !wr *  dwe2 *  a1 * dsack *
            !ciout
36        + dram * !tclr * as_20 *  as_00 * !wr *  dwe3 *  a1 * dsack *
            !ciout;
37
38  tclr  =  as_20 * tclr
39        + !as_20 * bgack_20
40        + !as_20 * reset
41        + !as_20 * c2off;
42
43  c2off.OE = !bgack_20;
44  c2off    = reset                'SLC off nach Reset
45           + c2off * !bgack_20;   'notwendig für TOS 2.06 Speichertest
46
47  %END
```

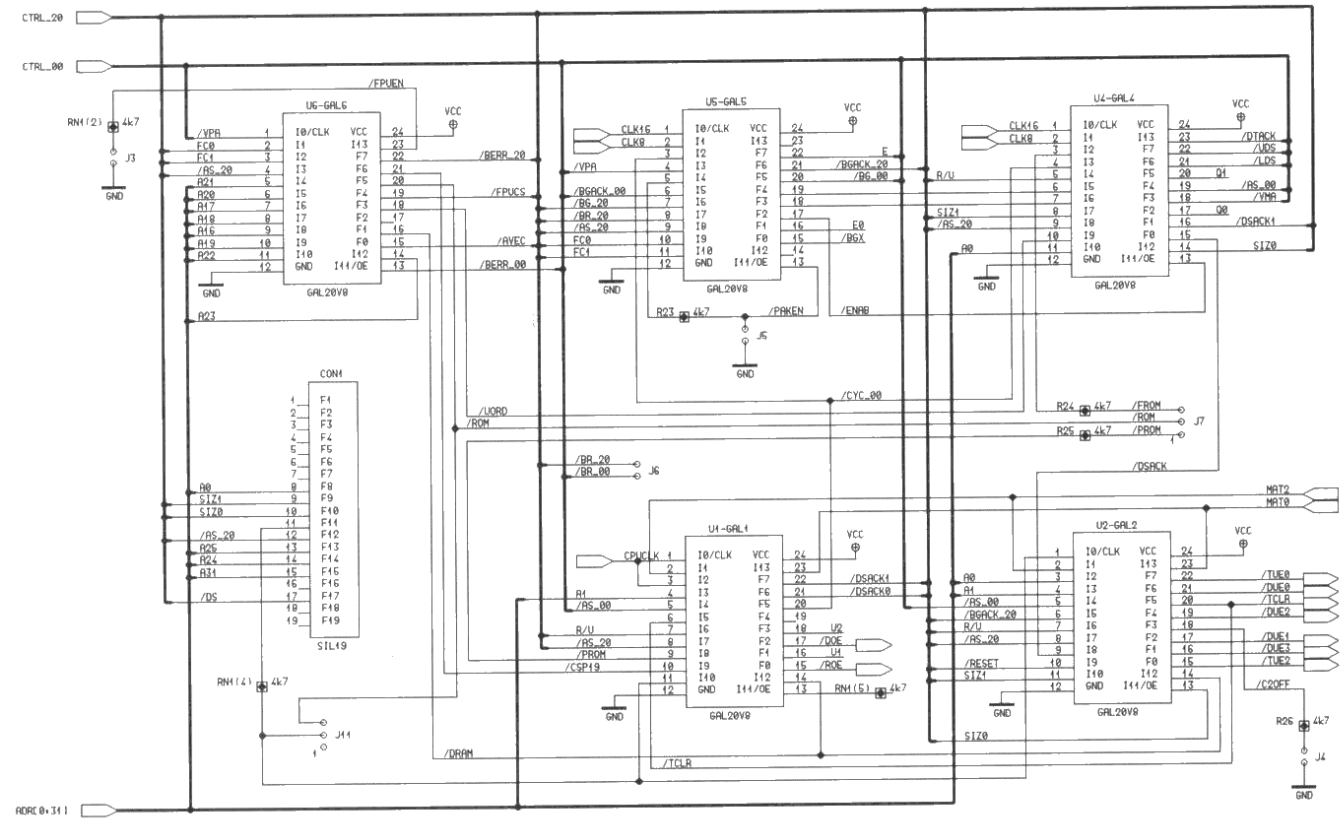
```
 0  PAK 68/3, GAL U4:  State Machine 1 für 68000 Businterface, 16 MHz
 1  26-08-93   V4_16b   Holger Zimmermann @ PE
 2  %ID
 3    P4_16
 4  %TYP
 5    GAL20V8A
 6  %PINS
 7    CLK  clk8  !from  !cyc_00  rd  e2  e1  siz1  !as_20  !word  a0
 8    !OE  siz0  !dsack  !dsack1  q0  !vma  !as_00  q1  !lds  !uds
      !dtack
 9  %LOGIC
10
11  dsack.OE = VCC;
12  dsack    = as_20 * dsack1;
13
14  dsack1 <-  as_00 * !dsack1 *  q0 * !q1 * as_20 * rd * !clk8 * dtack
15           +  as_00 * !dsack1 *  q0 * !q1 * as_20 * rd * !clk8 * vma *
               !e1 * e2
16           +  as_00 * !dsack1 *  q0 *  q1
17           +  as_00 *  dsack1       * !q1;
18
19       q0 <- !as_00 * !dsack1 * !q0 * !q1
20           + !as_00 * !dsack1 *  q0 * !q1 * !as_20
21           + !as_00 * !dsack1 *  q0 * !q1 *  as_20 * !clk8
22           + !as_00 * !dsack1 *  q0 * !q1 *  as_20 * !cyc_00
23           +  as_00 * !dsack1       * !q1 *  as_20;
24
25       q1 <-  as_00 * !dsack1 *  q0 * !q1 * as_20 * !rd * !clk8 * dtack
26           +  as_00 * !dsack1 *  q0 * !q1 * as_20 * !rd * !clk8 * vma *
                !e1 * e2
27           +  as_00 *  dsack1 *  q0 * !q1;
28
29      vma <- !clk8 * !e1 * !e2
30           + vma * !e2
31           + vma * !e1
32           + vma * clk8;
33
34    as_00 <- !as_00 * !dsack1 *  q0 * !q1 * as_20 * clk8 * cyc_00
35           +  as_00 * !dsack1 * !q0 * !q1 * as_20
36           +  as_00          *  q0 * !q1 * as_20
37           +  as_00 * !dsack1 *  q0       * as_20;
38
39      uds <- !as_00 * !dsack1 *  q0 * !q1 * as_20 *  rd * clk8 * cyc_00
                * word
40           + !as_00 * !dsack1 *  q0 * !q1 * as_20 *  rd * clk8 * cyc_00
                * !a0
41           +  as_00                * !q1 * as_20 *  rd * uds
42           +  as_00 * !dsack1 *  q0       * as_20 * !rd * !a0;
43
44      lds <- !as_00 * !dsack1 *  q0 * !q1 * as_20 *  rd * clk8 * cyc_00
                * word
45           + !as_00 * !dsack1 *  q0 * !q1 * as_20 *  rd * clk8 * cyc_00
                * a0
46           + !as_00 * !dsack1 *  q0 * !q1 * as_20 *  rd * clk8 * cyc_00
                * !siz0
47           + !as_00 * !dsack1 *  q0 * !q1 * as_20 *  rd * clk8 * cyc_00
                * siz1
48           +  as_00                * !q1 * as_20 *  rd * lds
49           +  as_00 * !dsack1 *  q0       * as_20 * !rd * a0
50           +  as_00 * !dsack1 *  q0       * as_20 * !rd * !siz0
51           +  as_00 * !dsack1 *  q0       * as_20 * !rd * siz1;
52
53  %END
54
```

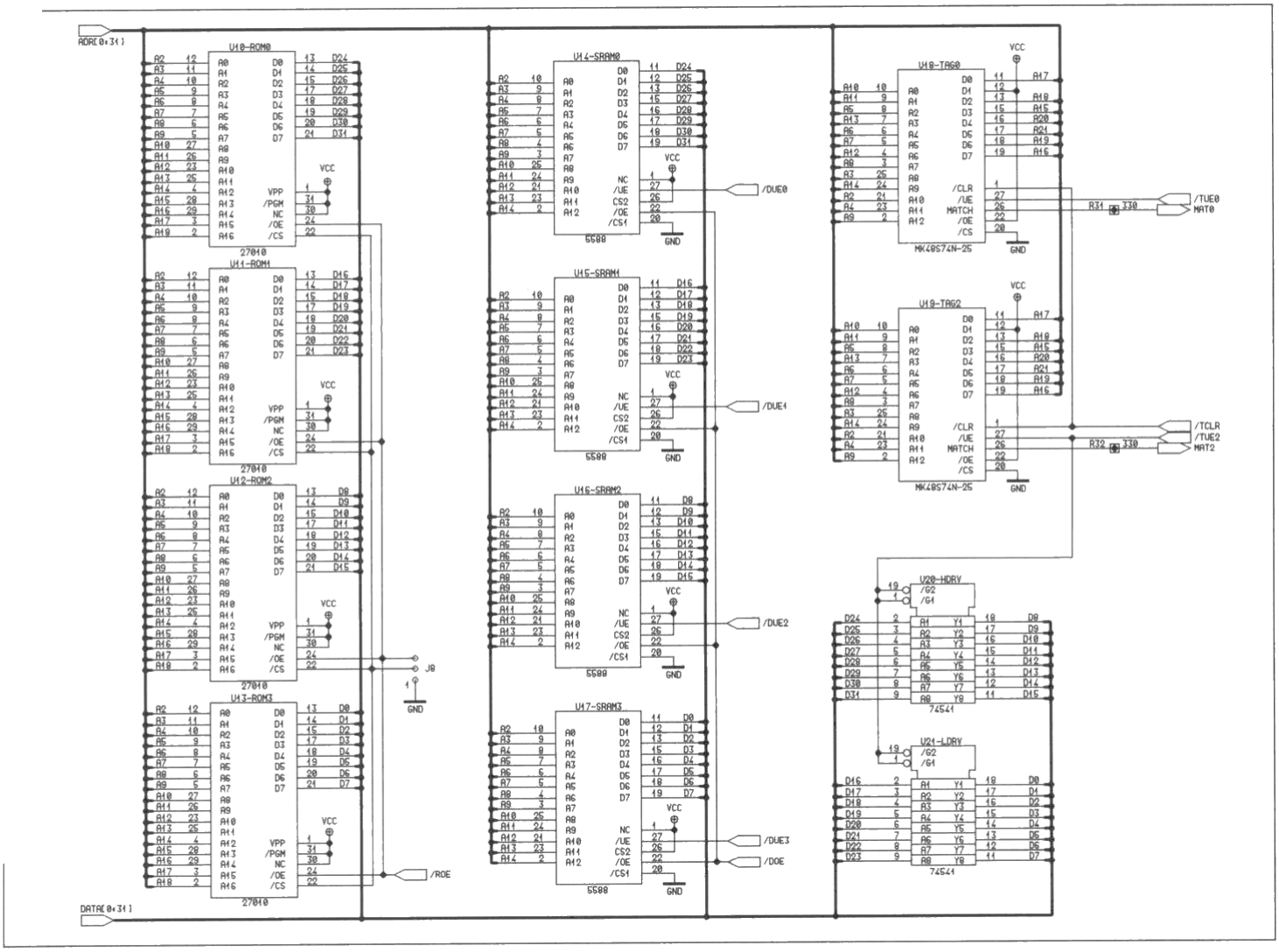
```
 0  PAK 68/3, GAL U5:  State Machine 2 für 68000 Businterface, 16 MHz
 1  21-08-93   V5_16a   Holger Zimmermann @ PE
 2  %ID
 3    P5_16
 4  %TYP
 5    GAL20V8A
 6  %PINS
 7    CLK  clk8  !cyc_00  !vpa  !pak_en  !bgack_00  !bg_20  !br_20  !as_20
 8    fc0  fc1
 9    !OE  nc  !bgx  e0  !enab  e1  e2  !bg_00  !bgack_20  e  nc
10  %LOGIC
11
12    enab.OE   = VCC;
13    enab      = !bg_00 * !bgack_20 * pak_en * cyc_00
14              + !bg_00 * !bgack_20 * pak_en * enab * !clk8;
15
16    bgx       <-  clk8 * bg_20 * !as_20
17              + !clk8 * bgx;
18
19    bg_00     <-  clk8 * bg_20 * bgx * br_20
20              +  clk8 * bg_20 * bgx * bgack_00
21              + !clk8 * bg_00;
22
23    bgack_20 <-         bgack_00
24              + !clk8 * bgack_20;
25
26    e         <-  e       * !e1
27              +  e             * !e0
28              +  e                   *  clk8
29              +      !e2 *  e1 *  e0 * !clk8
30              +      !e2 * !e1 * !e0 * !clk8;
31
32    e0        <-                 e0 *  clk8
33              +                 !e0 * !clk8;
34
35    !e1       <- !e *  e2 * !e1 * !e0
36              +  e       * !e1 *  e0
37              +            !e1       *  clk8
38              + !e *  e2 *  e1 *  e0 * !clk8
39              +  e * !e2       *  e0 * !clk8;
40
41    !e2       <-      !e2 *  e1 * !e0
42              + !e * !e2 *  e1
43              +      !e2             *  clk8
44              + !e       * !e1 *  e0 * !clk8
45              + !e *  e2 * !e1       * !clk8 * !fc0 * vpa
46              + !e *  e2 * !e1       * !clk8 * !fc1 * vpa;
47
48  %END
```

```
 0  PAK 68/3, GAL U6:  Adressdekoder, Mac SE und Mac Plus
 1  01-09-93   V6_MCa    Holger Zimmermann @ PE
 2  %ID
 3    P6_MAC
 4  %TYP
 5    GAL20V8A
 6  %PINS
 7   !vpa  fc0  fc1  !as_20  a21  a20  a17  a18  a16  a19  a22
 8   !berr_00  a23  !avec  !dram  !ciin  !word  !fpucs  !rom  !csp19
 9   !berr_20  !jp3
10  %LOGIC
11
12   berr_20.OE = VCC;
13   berr_20    = as_20 * fc1 * fc0 * !a19 * !a17              'PMMU, BKPT
14              + as_20 * fc1 * fc0 * !a19 *  a17 * !jp3      'FPU disabled
15              + as_20 * berr_00;
16
17   fpucs      = fc1 * fc0 * !a19 * !a18 *  a17 * !a16 * jp3;
                  'FPU enabled
18
19   avec       = fc1 * fc0 * vpa;
20
21   csp19      = fc1 * fc0 * !a19;
22
23   dram       = !fc0 * !a23 * !a22                                  'RAM
24              + !fc1 * !a23 * !a22;
25
26   rom        = !a23 * a22 * !a21 * !a20 * !a19;                    'ROM
27
28   word       = !a23 * a22 * !a21 * !a20                            'ROM
29              + !fc0 * !a23 * !a22                                  'RAM
30              + !fc1 * !a23 * !a22;
31
32   !ciin      = !a23 * a22 * !a21 * !a20                            'ROM
33              + !fc0 * !a23 * !a22                                  'RAM
34              + !fc1 * !a23 * !a22;
35
36  %END
```

Diese sieben GAL-Listings gelten für die 16-MHz-Version mit 68020 im Atari und Mac.


SYSCLK
CTRL_00
CTRL_20
U7-SYSCON
U8-CPU
U9-FPU
/RESET
/HALT
CLK
/BERR
/BR
/BG
/BGACK
FC0
FC1
FC2
/IPL0
/IPL1
/IPL2
/AS
/UDS
/LDS
/DTACK
R/U
E
/VMA
/VPA
/BERR_00
/BR_00
/BG_00
/BGACK_00
/AS_00
/UDS
/LDS
/DTACK
CPUCLK
/BERR_20
/BR_20
/BG_20
/BGACK_20
/AS_20
/DS
/DSACK0
/DSACK1
/AVEC
/CIIN
/CIOUT
/STERM
/FPUCS
FPUCLK
/SIZE
/SENSE
VCC
GND
J1
J2
G0000
G0030
G0002
DATA[0..31]
ADR[0..31]


CTRL_20
CTRL_00
/FPUEN
U6-GAL6
U5-GAL5
U4-GAL4
U1-GAL4
U2-GAL2
GAL20V8
CON1
SIL19
/VPA
/BERR_20
/FPUCS
/AVEC
/BERR_00
/BGACK_00
/BG_20
/BR_20
/AS_20
/BG_00
/DTACK
/UDS
/LDS
/AS_00
/VMA
/DSACK1
/DSACK0
SIZ1
SIZ0
/CYC_00
/VORD
/ROM
/FROM
/PROM
/DSACK
/CSP19
/DRAM
/TCLR
/PAKEN
/ENAB
/BGX
/DUE0
/DUE1
/DUE2
/TCLR
/C2OFF
MAT2
MAT0
ADR[0..31]



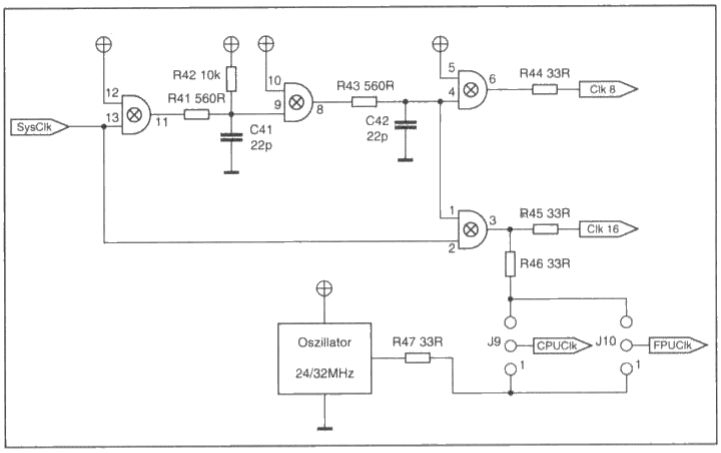
Bis auf den Prozessor und sein Pinout sind 020er- und 030er-PAK gleich. Hier finden Sie die Schaltung deshalb nur einmal.

kennt der Komparator beim nächsten Lesezugriff in jedem Fall ein Cache Miss (kein Treffer) und lädt daraufhin den Cache neu.

UngeBursted

Im Ablauf eines Buszyklus gibt es auf der PAK/3 keine Unterschiede zwischen 68020 und 68030. Der synchrone Busmodus des 68030, der einst den Mac IIci so beschleunigte (Burst-Mode mit /STERM), darf hier leider nicht genutzt werden; zum einen sind die ROMs und Tag-RAMs bei 32 MHz zu langsam, zum anderen gäbe es ein Problem, wenn nach einem Lesezugriff auf den Hauptspeicher ein lokaler Lesezugriff auf der PAK/3 folgte, weil die Datenbustreiber des Mainboards (im Mac genauso wie im Atari) noch vom vorherigen Zugriff her eingeschaltet wären und somit dazwischenfunken würden.

Zur Veranschaulichung der Abläufe soll zunächst eine PAK/3 mit 16 MHz Taktfrequenz herhalten. Ein Buszyklus beginnt, indem die CPU die Adresse anlegt und, nachdem die Suche im CPU-internen Cache erfolglos verlaufen ist, /AS_20 mit fallender Flanke von CLK16 aktiviert. Falls dies ein ROM-Zugriff ist (/ROM von U6 aktiv) und sich ROMs auf der PAK befinden (J7 auf 1-2), schaltet U1 die ROMs ein und bestätigt sofort mit /DSACK0 und /DSACK1. Mit der nächsten fallenden Flanke von CLK16 wird dies von der CPU erkannt, worauf sie den Buszyklus genau einen CLK16-Takt später durch Desaktivieren von /AS_20 beendet. U1 hält /CYC_00 die ganze



Der Taktverdoppler für den synchronen 16-MHz-Betrieb wurde von der alten PAK übernommen.

Zeit auf *high*, so daß U4 keinen Zugriff auf das Mainboard startet.

Angenommen, es findet ein Lesezugriff auf den Bereich $0 bis $40 0000 statt (/DRAM von U6 aktiv), der L2-Cache ist aktiviert (/TCLR auf *high*), aber mindestens eines der beiden Tag-RAMs meldet keinen Treffer (MAT0 oder MAT2 auf *low*): In diesem Fall geht /CYC_00 nach kurzer Wartezeit auf *low* und startet damit in U4 einen Zugriff auf das Mainboard, der im Prinzip wie bei der PAK/2 abläuft. Hier sorgt jetzt aber U2 dafür, daß das gelesene Datenwort auch in den SRAMs abgelegt und gleichzei-

tig die zugehörige Adresse im entsprechenden Tag-RAM festgehalten wird.

Der Umstand, daß der Datenbus auf dem Mainboard nur 16 Bit, der Cache aber 32 Bit breit ist, bedarf einer näheren Betrachtung. Liegt die Adresse eines Datenwortes nämlich nicht auf einer Langwortgrenze (also A1 = *high*), so entsteht folgendes Problem: Das Mainboard liefert das Datenwort auf den CPU-Datenleitungen D16 bis D31, die CPU erwartet es im Cache aber später auf D0 bis D15. In diesem Falle werden die beiden Bustreiber auf der PAK/3 im wahrsten Sinne des Wortes aktiv und lösen das Problem, indem sie eine temporäre Querverbindung zwischen unterem und oberem Datenwort herstellen.

Querschläger

Bis hier ist die PAK/3 noch kein Stück schneller als ihr Vorgänger. Aber beim nächsten Lesezugriff auf diese Adresse erkennt der Komparator im Tag-RAM an der Gleichheit von angelegter und abgespeicherter Adresse einen Treffer. Wenn jetzt auch noch das zweite Tag-RAM, zuständig für die andere Hälfte des 32 Bit breiten Cache, einen Treffer signalisiert (MAT0 und MAT2 beide auf *high*), dann genügt der CPU ein schneller Blick in den Cache, um die gewünschten Daten zu bekommen. In diesem Fall bleibt /CYC_00 auf *high*, U4 hält sich vornehm zurück, und das Mainboard bekommt von diesem Zugriff gar nichts mit. Statt dessen schaltet U1, analog zum ROM-Zugriff, die SRAMs ein und bestätigt mit /DSACK0 und /DSACK1. Der Rest ist Formsache. Ein solcher Lesezugriff dauert nur drei CLK16-Takte, im Vergleich zu acht CLK16-Takten, die ein Zugriff auf das Mainboard im günstigsten Fall dauert. Dabei bekommt die CPU aus dem Cache 32 Bit auf einen Schlag präsentiert, ohne Cache hingegen werden für das zweite Wort nochmal acht Takte fällig.

Bei Schreibzugriffen ist der Cache dagegen machtlos, mit der hier implementierten Write-Through-Strategie ist in jedem Fall ein Zugriff auf das Mainboard angesagt. Falls die Tag-RAMs dabei einen Treffer melden, sich das zu dieser Adresse gehörende Datenwort also auch im Cache befindet, kümmert sich U2 darum, daß der Cacheinhalt ebenfalls aktualisiert wird. Somit kann bereits der nächste Lesezugriff wieder aus dem Cache abgewickelt werden. Sogenannte Write-Back-Caches, die selbständig ihren Inhalt in den Hauptspeicher zurückschreiben und somit auch Schreibzugriffe puffern, sind in ihrer Realisierung ungleich aufwendiger und in diesem Fall – wegen des 16-Bit-Nadelöhrs – noch nicht einmal besonders wirkungsvoll.

An der CPU vorbei

Ein unerfreuliches Thema, von dem die Macs (jedenfalls die hier in Frage kommenden Typen) verschont bleiben, sind die DMA-Zugriffe. Da hier der Inhalt des Hauptspeichers an der CPU vorbei manipuliert wird, muß nach einer solchen Aktion sicherheitshalber der gesamte Cache für ungültig erklärt werden. Bei der PAK/3 wird dies erreicht, indem der Inhalt der Tag-RAMs gelöscht, also auf *low* gesetzt wird. Damit ist die Kohärenz von Cache und Hauptspeicher immer gewährleistet. Wenn jetzt aber allenthalben DMA-Zugriffe stattfinden, geht die Performance natürlich in den Keller. Aus diesem Grund ist der Blitter im Atari ST mit PAK-68/3 völlig witzlos und sollte daher ausgeschaltet oder ganz ausgebaut werden; die neue CPU ist ohnehin schneller.

Aufbau und Pflege

Für Mac und ST sind jeweils vier Permutationen in der Bestückung möglich: Völlig nackt, das heißt ohne Cache und ohne ROMs (für erste Tests empfehlenswert), mit ROMs und ohne Cache, mit Cache, aber ohne ROMs und schließlich der Vollausbau mit allem Drum und Dran, jeweils mit den zum Rechner und zur gewünschten Taktfrequenz passenden GALs. Die Optionen sind in der Stückliste angegeben. In der Tabelle neben der Stückliste finden Sie sämtliche in Frage kommenden GAL-Bezeichnungen wieder. Im Gegensatz zur alten PAK werden hier auch bei Abwesenheit der ROMs oder des Cache alle GALs bestückt, da der PAK-Logik diese Optionen durch Jumper angezeigt werden; im zweiten Teil dieses Beitrags gehen wir noch erschöpfend auf Rechnerspezifika ein.

Die Platine erreicht dank Multilayer-Technik fast die Packungsdichte eines Freibierausschanks; um die Maße der alten PAK nicht zu überschreiten, mußten die Tag-RAMs unter die EPROM-Sockel verbannt werden, und auch einige Stützkondensatoren finden bei Verwendung von geschlossenen PGA-Sockeln für die Prozessoren nur auf der Lötseite Platz. CPU und FPU lassen sich übrigens aus PGA-Typen mit Kelchfederkontakten ('Präzisionsfassungen'), wenn überhaupt, nur mit roher Gewalt entfernen. Besser sind hier normale Ausführungen, wie sie etwa AMP herstellt. Sicherheitshalber sollten Sie auch den Tag-RAMs Sockel spendieren; die Stege der darüberliegenden EPROM-Fassungen müssen vorsichtig entfernt werden.

Wie bei der alten PAK stellt ein lötseitiger 64poliger DIL-Sockel mit eingestecktem beidseitig 'piekendem' Adaptersockel oder ebensolchem SIL-Streifen die Verbindung zum Mainboard her. Aus Stabilitätsgründen ist diesmal das operative Entfernen der alten 68000-CPU – sofern nicht schon geschehen – unumgänglich, eine 64polige DIL-Fassung tritt an ihre Stelle. Ein GAL-Satz, der den Betrieb bei auf dem Mainboard verbleibender CPU gestattet, ist im Gegensatz zur PAK-68/2 nicht vorgesehen. Bewährt hat sich die Amputation sämtlicher 68000-Beinchen mit einem kleinen Seitenschneider. Einzeln lassen sich die Pins dann leicht aus der Platine ziehen. Wem bei dieser Prozedur ob der leidenden CPU die Tränen kommen, der kann dem beinlosen 68000 später eine Prothese in Form einer DIL-Fassung unterlöten und ihn für Testzwecke weiterverwenden.

Optional ist übrigens der Betrieb der 68000 umschaltbar neben dem 68020/030 möglich, wenn man einige Leitungen zur bislang leeren Spare-Fassung fädelt und ein entsprechend programmiertes GAL einsetzt (Tips dazu folgen). Leider waren die Verbindungen beim besten Willen nicht mehr auf der Platine unterzubringen, und da nicht jeder eine 'Vorher/Nachher'-Option braucht, haben wir für diese nur eine freiverdrahtbare Leerfassung vorgesehen. Die auf der Platine vorhandene 68000-Fassung auf der Bestückungsseite kann alternativ auch zur Aufnahme von MSDOS-Emulatoren und ähnlichen Erweiterungen dienen, die sich direkt der Prozessorsignale bedienen.

Der Einsatz mittelschneller CPUs (20 und 25 MHz) bei asynchroner Taktung, das heißt, bei Verwendung des 32-MHz-GAL-Satzes und entsprechend ausgelegtem Quarzoszillator, führt in der Regel zu Problemen; auch ist eine asynchron laufende 20-MHz-CPU bedingt durch unvermeidliche Wartezyklen kaum schneller als die synchron getaktete 16-MHz-Version.

Was es bei der Anpassung des TOS 3.06 zu beachten gilt, wie die asynchronen Zugriffe der 32-MHz-Version funktionieren und wie man den Mac zur Nutzung der FPU und des Datencache überredet, erfahren Sie in der nächsten c't. (cm)

Literatur

[1] Manfred Völkel, Carsten Meyer, Tausch mit Turbo, PAK-68/2 – mehr Dampf für 68000-Rechner, c't 10/91, S. 178

[2] Manfred Völkel, Carsten Meyer, Tausch mit Turbo, PAK-68/2 in Mac, Atari und Amiga, c't 11/91, S. 314

[3] Manfred Völkel, Tausch mit Turbo, Tips zur PAK-68/2 im ST, c't 12/91, S. 222

[4] Manfred Völkel, Turbo für den Austauschmotor, Schneller Atari mit modifizierter PAK-68, c't 3/91, S. 282

[5] Carsten Meyer, Schnelle Freundin, Amiga mit PAK und 14 MHz, c't 9/90, S. 338

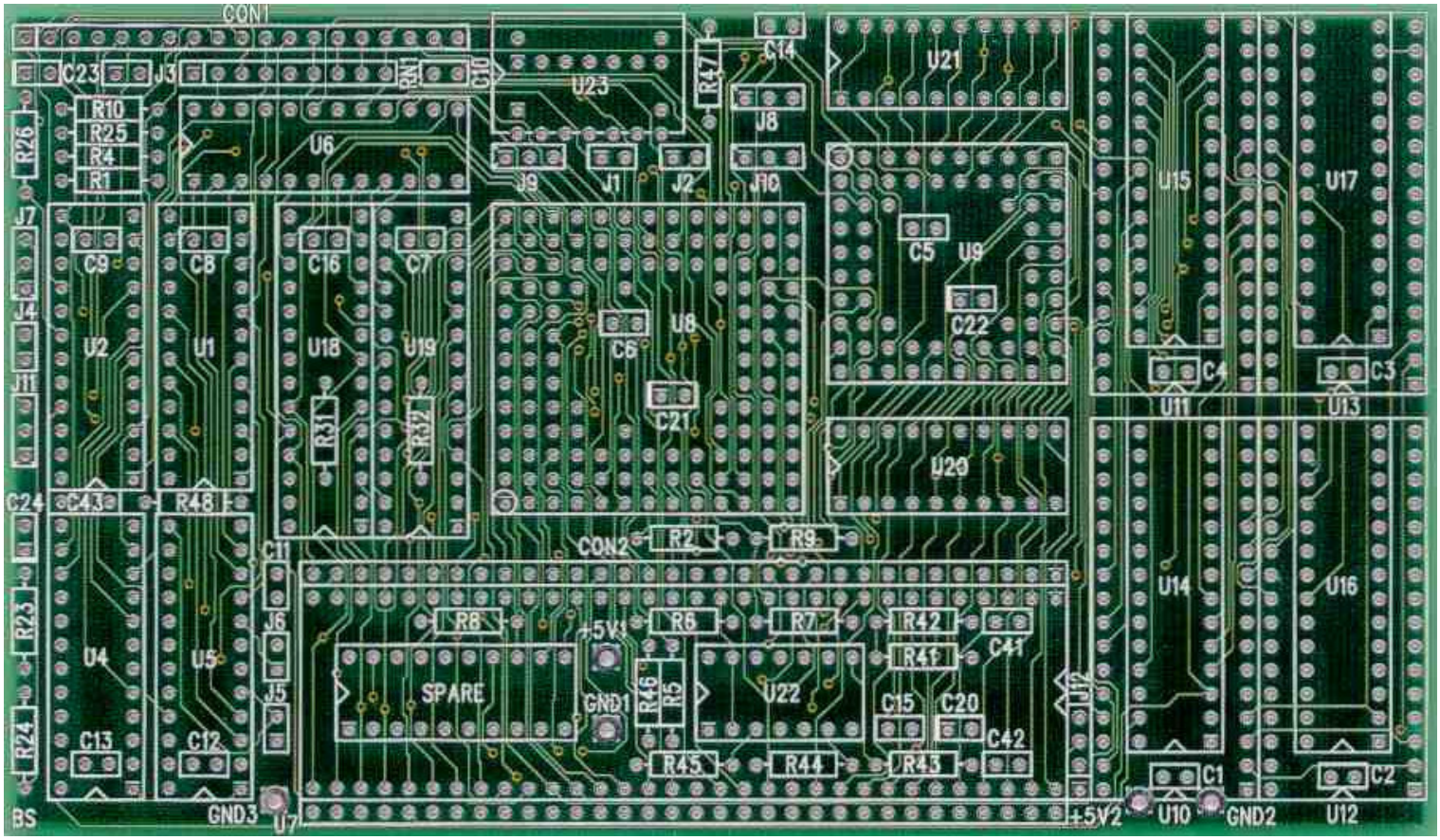
[6] Roland Beck, Hilfskraft: 68000-Prozessor trotz PAK-68/2, c't 7/92, S. 196

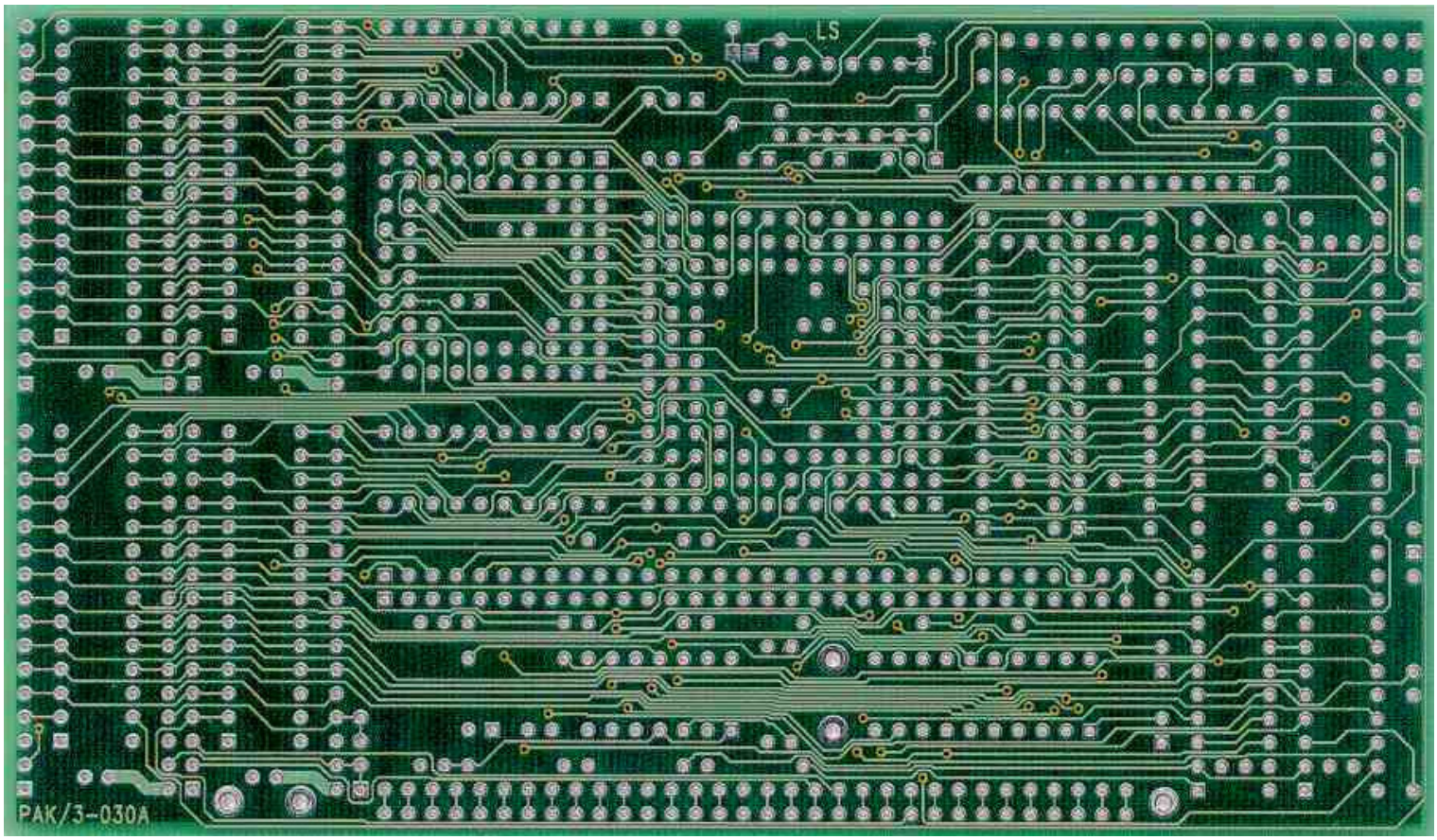
[7] Leo Drisis, Real-Renner, 68881-Einbindung für die PAK im Mac, c't 5/90, S. 190

[8] Carsten Meyer, Fast-PAK, 68000 und PAK-68 mit 16 MHz im Macintosh, c't 1/90, S. 194

[9] Johannes Assenbaum, Mehr CPU, PAK-68 – die Prozessor-Austausch-Karte für 68000-Rechner, c't 8/87, S. 68

ct


CON1
C23
J3
RN1
C10
U23
R47
C14
U21
R10
R25
R4
R1
U6
J8
J9
J1
J2
J10
U15
U17
R26
J7
C9
C8
C16
C7
C5
U9
C22
U2
U1
U18
U19
C6
U8
C4
C3
U11
U13
R31
R32
C21
U20
C24
C43
R48
CON2
R2
R9
C11
U14
U16
J6
R8
+5V1
R6
R7
R42
C41
R41
U4
U5
SPARE
GND1
R46
R5
U22
C15
C20
J5
C42
R24
R23
C13
C12
R45
R44
R43
C1
C2
BS
GND3
U7
+5V2
U10
GND2
U12


LS
PAK/3-030A