Replacement of MC68000 Processor with MC68020 or MC68030 with Coprocessor Option for MC68881 and MC68882: MACH110

Application Note
by Christoph Niessen, Germany

Advanced Micro Devices

INTRODUCTION

The MC68000 processor is one of the most widely used processors in minicomputers[1], peripherals[2] and controllers. The new processors in this family (MC68010, MC68020, MC68030 and MC68040) are of course, software compatible, but in addition it is not a trivial matter to integrate them into old hardware environments. After the 68020 for example, all buses have to be 32 bits wide and the processors can be expanded by a well thought-out coprocessor interface.

The new processors offer the following advantages:

- Higher clock frequency (up to 50 MHz for the MC68030)
- Doubled, external data buswidths (32 bit)
- Doubled, internal buswidths (32 bit)
- Fewer cycles per memory access (4 for the MC68000, 3 for the MC68020, 2 for the MC68030)[3]
- Improved microprogram with far fewer cycles per command
- Integrated cache

These advantages can be utilized only in part when a processor of this type is adapted to hardware that was designed only for the MC68000. The higher clock frequency and the doubled data buswidth for example, normally can not be implemented any more readily than the reduced number of clock cycles per memory access. The bus bandwidth of the MC68000 of 8 MHz amounts to a maximum of 16 bits*8 MHz/4 cycles = 4 Mbytes/s, in contrast to 32 bit*50 MHz/2 cycles = 100 Mbyte/s of an MC68030 at 50 MHz clock frequency. It is evident that the hardware of the computer can not keep pace with a processor of this type. For these reasons, the first attempts to replace the MC68000 were carried out only with the boundary conditions of equal clock frequency and 16-bit data buswidth. The first publications originate from the year 1985 from Motorola Corp. and for their adaptation, they needed the processors (MC68020/881) and an additional three PAL16R4A-15 and six resistors.

[1] Atari, Commodore, Apple etc.
[2] Laser printer
[3] Minimal values

Other designs required up to 13 logic components. In spite of the expenses, the solutions could only be implemented at the thus yield enhanced performance of less than 50% in normal computation mode without FPU; in programs with FPU support, the performance rises considerably.

With the new, highly integrated logic components, like the MACH Chips by AMD, the entire circuit can be created in one component. With little extra expense, the following expansions can also be implemented:

- Bus interface for local expansions with the entire bus bandwidth
- Differing clock frequency for computer, microprocessor and coprocessor(s)
- Programmable (reduced) access times to parts of the computer that allow this

INCOMPATIBILITIES OF THE PROCESSORS

To replace an MC68000 by an MC68020, the differences in the buses have to be compensated. These differences consist of the following points:

- The MC68020 does not have a synchronous bus interface for the peripheral components of the MC68xxx family
- The MC68020 reads the data right in the third clock cycle of a data transfer
- A different protocol is used for read-modify-write cycles
- The autovector-interrupt protocol is different

MACH110

The MACH110 component by AMD is a programmable logic component with a typical PAL-structure. The complexity of two PAL22V16 is attained with two blocks of 16 macrocells each and three additional inputs. The 44-pin PLCC-housing takes up very little board space. Like all newer logic components, the macrocells are variously programmable (D- or T-flip-flop, inverter, OE...).

Now the pin-compatible MACH210 is twice as complex and can be used when the logic design will no longer fit into the MACH110-series after a "redesign."

Publication# 17752 Rev. A Amendment/0
Issue Date: May 1993

ABEL

The ABEL development environment by DATA I/O offers the possibility to design, optimize and simulate

- Designs in equation form
- As state diagram or
- As value table

With an auxiliary module, these designs can then be "fitted" into a MACH component; that is, the allocation of logic to the component resources can be implemented.

DESIGN

The design consists essentially of a series of synchronous and asynchronous state machines. All state transitions are secured against hazards and races. The function of the single modules has been demonstrated with test vectors via the simulator. The processors MC68020 and MC68030 are compatible for the applications stated here, so that this design is suitable for both.

The MC68000 internally doubles the clock, so that the external signals will be activated with both clock flanks. For this reason, not only the normal processor clock, but also the inverted processor clock has to be fed to the MACH component. In order that the local bus is expandable, several lines have to be equipped with OK-drivers. For the processor and coprocessor, timing sources, perhaps also differing timing sources, will have to be made available.

Replacement of MC68000 Processor with MC68020 or MC68030 with Coprocessor Option for MC68881 and MC68882

Advanced Micro Devices

Application Note

by Christoph Neissen, Germany

EINLEITUNG

Der Prozessor MC68000 ist einer der am weitesten verbreiteten Prozessoren in Kleinrechnern[1], Peripheriegeraten[2] und Steuerungen. Die neuen Prozessoren dieser Familie (MC68010, MC68020, MC68030 und MC68040) sind zwar softwarekompatibel aber nicht trivial in alte Hardware–Umgebungen zu integrieren. Ab dem 68020 sind zum Beispiel alle Busse 32-Bit breit und die Prozessoren sind durch ein gut durchdachtes Koprozessorinterface erweiterbar.

Die neuen Prozessoren bieten folgende Vorteile:

- Höhere Taktfrequenz (bis zu 50 MHz beim MC68030)
- Doppelte externe Datenbusbreite (32-Bit)
- Doppelte interne Busbreiten (32-Bit)
- Weniger Zyklen pro Speicherzugriff (4 beim MC68000, 3 beim MC68020, 2 beim MC68030)[3]
- Verbessertes Mikroprogramm mit deutlich weniger Zyklen pro Befehl
- Integrierte Caches

Diese Vorteile lassen sich nur teilweise nutzen wenn ein solcher Prozessor an eine Hardware angepaßt wird, die nur für einen MC68000 konzipiert ist. Die höhere Taktfrequenz und die doppelte Datenbusbreite lassen sich zum Beispiel normalerweise ebensowenig implementieren wie die reduzierte Anzahl der Taktzyklen pro Speicherzugriff. Die Busbandbreite des MC68000 mit 8 MHz beträgt maximal 16 Bit*8 MHz/4 Zyklen=4 Mbyte/s im Gegensatz zu 32 Bit*50 MHz/2 Zyklen=100 Mbyte/s eines MC68030 bei 50 MHz Taktfrequenz. Es ist offensichtlich, daß die Hardware des Rechners mit einem solchen Prozessor nicht mithalten kann. Aus diesen Gründen wurden die ersten Versuche den MC68000 zu ersetzen nur mit den Randbedingungen gleiche Taktfrequenz und 16-Bit Datenbusbreite durchgeführt. Die erste Veröffentlichung stammt aus dem Jahr 1985 von MOTOROLA und benötigt für die Anpassung außer den Prozessoren (MC68020/881) noch 3 PAL16R4A–15 und 6 Widerstände. Andere Konzepte benötigen bis zu 13 Logikbausteine. Trotz des Aufwandes sind die Lösungen nur bei der Taktfrequenz des Rechners einsetzbar und bringen so Leistungssteigerungen von weniger als 50% im normalen Rechenbetrieb ohne FPU; bei Programmen mit FPU–Unterstützung steigt die Leistung dagegen erheblich.

Mit den neuen hochintegrierten Logikbausteinen wie den MACH-Chips von AMD ist die gesamte Schaltung in einem Baustein zu realisieren. Mit wenig Zusatzaufwand können auch noch folgende Erweiterungen implementiert werden:

- Busschnittstelle für lokale Erweitereungen mit der vollen Busbandbreite
- Unterschiedliche Taktfrequenz von Rechner, Mikroprozessor und Koprozessor(en)
- Programmierbare (reduzierte) Zugriffszeiten auf Teile des Rechners die dieses erlauben

INKOMPATIBILITÄTEN DER PROZESSOREN

Für den Ersatz eines MC68000 durch einen MC68020 müssen die Unterschiede der Busse ausgeglichen weden. Diese Unterschiede bestehen in folgenden Punkten:

- Der MC68020 hat kein synchrones Businterface für die Peripheriebausteine der MC68xx–Familie
- Der MC68020 liest bereits im dritten Taktzyklus eines Datentransfers die Daten
- Bei Read-modify-write-Zyklen wird ein anderes Protokoll benutzt
- Das Autovektor-Interrupt-Protokoll ist unterschiedlich

MACH110

Der Baustein MACH110 von AMD ist ein programmierbare Logikbaustein mit einer typischen PAL–Struktur. Mit zwei Blöcken mit je 16 Makrozellen und drei zusätzlichen Eingängen wird die Komplexität von 2 PAL22V16 erreicht. Das 44-pin PLCC-Gehäuse belegt nur wenig Platinenfläche. Wie bei allen neueren Logikbausteinen sind die Makrozellen vielfältig programmierbar (D-oder T-Flip-Flop, Inverter, OE...).

Mit dem Pin-kompatiblen MACH210 steht ein doppelt so komplexer Baustein zu Verfügung, der dann eingesetzt

[1] Atari, Commodore, Apple etc.

[2] Laserdrucker

[3] Minimalwerte

werden kann, wenn der Logikentwurf nach einem "Redesign" nicht mehr in den MACH110er paßt.

ABEL

Die Entwicklungsumgebung ABEL von DATA I/O bietet die Möglichkeit

- Designs in Gleichungsform,
- Als Zustandsdiagramm oder
- Als Wertetabelle

Zu enwerfen, optimieren und simulieren. Mit einem Zusatzmodul können diese Designs anschließend in einen MACH-Baustein "gefittet" werden, das bedeutet, daß die Zuordnung der Logik zu den Bausteinrecourcen durchgeführt wird.

ENTWURF

Der Entwurf besteht im Wesentlichen aus einer Reihe von synchronen und asynchronen Zustandmaschinen. Alle Zustandsübergänge sind gegen Hazards und Races gesichert. Die Funktion der einzelnen Module ist mit Testvektoren durch den Simulator nachgewiesen. Die Prozessoren MC68020 und MC68030 sind für die hier betroffenen Anwendungen kompatibel, so das dieser Entwurf für beide geeignet ist.

Der MC68000 verdoppelt intern den Takt, so daß die externen Signale mit beiden Taktflanken aktiviert werden. Aus diesem Grund muß dem MACH-Baustein nicht nur der normale Prozessortakt, sondern auch der invertierte zugeführt werden. Damit der lokale Bus erweiterbar ist, müssen einige Leitungen mit OK-Treibern versehen weden. Für den Prozessor und Koprozessor müssen—eventuell auch unterschiedliche—Taktquellen bereitgestellt werden.

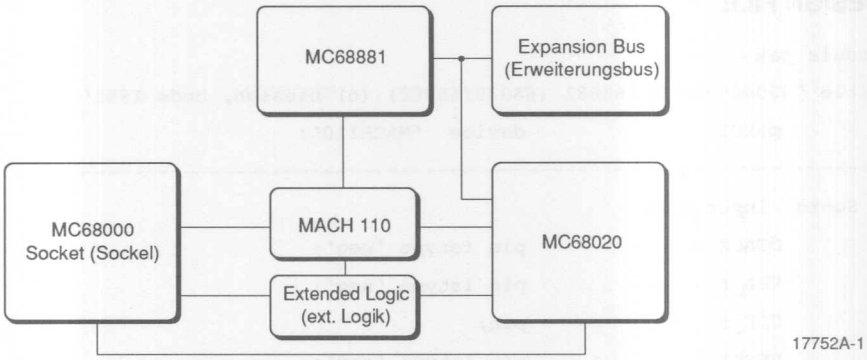

Figure 4-1. Block Diagram

DESIGN FILE

```
module pak
title '68000->68020/68881 (68030/68882) (c) niessen, bode 1991'
        pak01                   device  'MACH110';
"-------------------------------------------------------------
" Socket input pins
        DTACK_s                 pin istype 'neg';
        VPA_s                   pin istype 'neg';
        CLK_s                   pin;
        RESET_s                 pin istype 'neg';
" Socket output pins
        AS_s                    pin istype 'neg,reg';
        LDS_s                   pin istype 'neg,reg';
        UDS_s                   pin istype 'neg,reg';
        VMA_s                   pin istype 'neg';
        BG_s                    pin istype 'neg';
        E_p                     pin istype 'buffer,reg';
"-------------------------------------------------------------
" Processor input pins
        BG_p                    pin istype 'neg';
        AS_p                    pin istype 'neg';
        DS_p                    pin istype 'neg';
        RnotW_p                 pin;
        RMC_p                   pin istype 'neg';
        SIZ1_p                  pin;
        SIZ0_p                  pin;
        FC2_p                   pin;
        FC1_p                   pin;
        FC0_p                   pin;
        A31_p                   pin;
        A24_p                   pin;
        A19_p                   pin;
        A18_p                   pin;
        A17_p                   pin;
        A16_p                   pin;
        A15_p                   pin;
        A14_p                   pin;
        A13_p                   pin;
```

```
        A0_p                    pin;
" Processor output pins
        DSACK1_p                pin istype 'neg';
        AVEC_p                  pin istype 'neg';
"-------------------------------------------------------------
        CLK_neg                 pin;" CLK_s auf Platine invertiert
        CS_FPU                  pin istype 'neg';" Output
        CS_RAM                  node istype 'neg';
"-------------------------------------------------------------
        DSACK1_syn              node istype 'buffer,reg';
        DTERR                   node;
        LDS                     node istype 'buffer,reg';
        UDS                     node istype 'buffer,reg';
        AS                      node istype 'buffer,reg';
"
" Zustandsregister fuer die Verzoegerung des Datenzugriffs:
"
        z1,z0                   node istype 'buffer,reg';
        data_valid_00           =[z1,z0];
        Z0                      =[0,0];
        Z1                      =[0,1];
        Z2                      =[1,1];
        Z3                      =[1,0];
"
" Die folgenden Register werden fuer die E_p Erzeugung benutzt. Die
" Zustandskodierung ist so gewaehlt, dass alle Zustandsuebergaenge jeweils
" nur ein Bit aendern und dass das linke Bit direkt als E_p-Signal
" benutzt werden kann.
"
        c2,c1,c0                node istype 'buffer,reg';
        s0                      =[1,0,0,0];
        s1                      =[1,0,0,1];
        s2                      =[1,0,1,1];
        s3                      =[1,0,1,0];
        s4                      =[0,0,1,0];
        s5                      =[0,1,1,0];
        s6                      =[0,1,1,1];
        s7                      =[0,0,1,1];
        s8                      =[0,0,0,1];
```

```
        s9                  =[0,0,0,0];
        s10                 =[0,1,0,1];
        s11                 =[0,1,0,0];
        s12                 =[1,1,0,1];
        s13                 =[1,1,0,0];
        s14                 =[1,1,1,1];
        s15                 =[1,1,1,0];
"-------------------------------------------------------------
"AMDMACH property 'GROUP A LDS_s UDS_s AS_s BG_s';
AMDMACH property 'GROUP B DTERR DTACK_s z0 z1 SIZ0_p SIZ1_p A0_p';
AMDMACH property 'GROUP A VPA_s VMA_s E_p c2 c1 c0';
"-------------------------------------------------------------
        CPU_SPACE MACRO {(FC0_p&FC1_p&FC2_p)};
"       CP_COMMU  MACRO {([A19_p,A18_p,A17_p,A16_p]==[0,0,1,0])};
        CP_COMMU  MACRO {(!A31_p)};
        CP_ID     MACRO {([A15_p,A14_p,A13_p]==[0,0,1])};
        BGINH     MACRO {(CS_FPU#CS_RAM)};
"       IACKN     MACRO {(CPU_SPACE&AS_p&A19_p&A18_p&A17_p&A16_p)};
        IACKN     MACRO {(CPU_SPACE&AS_p&A31_p)};
        RESET_all MACRO {test_vectors(RESET_s->E_p);1->0;0->0;};
"-------------------------------------------------------------
equations
        CS_FPU=CPU_SPACE&CP_COMMU&CP_ID;
        CS_RAM=(!CPU_SPACE&!A31_p&A24_p);
        VMA_s.OE=!BGINH;
        BG_s=BG_p&!BGINH;BG_s.OE=BG_p&!BGINH;
        !AVEC_p=!IACKN#!VPA_s;
" VMA bestaetigt einen synchronen Zyklus der mit VPA eingeleitet wurde.
" VPA wird immer nur drei fallende Flanken vor E_p^ erkannt
        VMA_s=VPA_s&(([E_p.FB,c2.FB,c1.FB,c0.FB]==s11)#VMA_s);
        DSACK1_syn.clk=CLK_neg;
        DSACK1_syn.T=!AS_p&DSACK1_syn;" Ausschalten wenn AS_p inaktiv
"-------------------------------------------------------------
equations
" DTERR ist aktiv, wenn DTACK laenger als AS aktiv ist:
        DTERR=!AS_p&DTACK_s#DTERR&DTACK_s;
        AS:=AS_p&!DTERR#RMC_p;
        AS_s=AS&(AS_p#RMC_p);
        DSACK1_p=DSACK1_syn&AS_p;
```

```
        [UDS_s,LDS_s,AS_s].OE=!BGINH;
        [UDS,LDS,AS].CLK=CLK_s;
        LDS:=!DTERR&DS_p&(SIZ1_p#!SIZ0_p#A0_p);
        UDS:=!DTERR&DS_p&!A0_p;
        LDS_s=LDS&DS_p;
        UDS_s=UDS&DS_p;
"-------------------------------------------------------------------
equations
        data_valid_00.CLK=CLK_neg;
        data_valid_00.AR=RESET_s;
"-------------------------------------------------------------------
"
state_diagram data_valid_00
state Z0:if AS_s then Z1 else Z0;           "Start bei AS_s
state Z1:if DTACK_s then Z2 else Z1;        "DTACK abtasten
state Z2:goto Z3;                           "1 Zyklus verzoegern
state Z3:DSACK1_p=1;IF DS_p then Z3 else Z0;"Ende wenn DS inaktiv
"-------------------------------------------------------------------
equations
        [E_p,c2,c1,c0].CLK=CLK_neg;
        [E_p,c2,c1,c0].AR=RESET_s;
"-------------------------------------------------------------------
state_diagram [E_p,c2,c1,c0]
state s0:goto s1;
state s1:goto s2;
state s2:DSACK1_syn.T=VMA_s&AS_p&RnotW_p&!DSACK1_syn;goto s3; "Lesen
state s3:DSACK1_syn.T=VMA_s&AS_p&!RnotW_p&!DSACK1_syn;goto s4;"Schreiben
state s4:goto s5;
state s5:goto s6;
state s6:if VPA_s then s10 else s7;"synchron: erzeuge in 2 Zyklen VMA (s11)
state s7:goto s8;
state s8:goto s9;
state s9:goto s0;
"
state s10:goto s11;"VPA erkannt
state s11:goto s9;"VMA setzen
"
state s12:goto s0;
state s13:goto s0;
```

```
state s14:goto s0;
state s15:goto s0;
"-------------------------------------------------------------------
" Datenzugriff
RESET_all;
test_vectors
        ([CLK_s,CLK_neg,AS_p,DTACK_s,DS_p]->[data_valid_00,AS,AS_s,DSACK1_p]);
        [.x.,.x.,0,0,0]->[Z0,0,0,0];
        [.c.,.x.,0,0,0]->[Z0,0,0,0];
        [.x.,.x.,1,0,1]->[Z0,0,0,0];
        [.x.,.c.,1,0,1]->[Z0,0,0,0];
        [.x.,.x.,1,0,1]->[Z0,0,0,0];
        [.c.,.x.,1,0,1]->[Z0,1,1,0];" AS_p ist jetzt mit CLK synchronisiert
        [.x.,.x.,1,0,1]->[Z0,1,1,0];
        [.x.,.c.,1,0,1]->[Z1,1,1,0];" AS_s erkannt
        [.x.,.x.,1,0,1]->[Z1,1,1,0];
        [.c.,.x.,1,1,1]->[Z1,1,1,0];
        [.x.,.x.,1,1,1]->[Z1,1,1,0];
        [.x.,.c.,1,1,1]->[Z2,1,1,0];" DTACK_s erkannt aber erst im
        [.x.,.x.,1,1,1]->[Z2,1,1,0];" naechsten Zyklus weitergeben
        [.c.,.x.,1,1,1]->[Z2,1,1,0];
        [.x.,.x.,1,1,1]->[Z2,1,1,0];
        [.x.,.c.,1,1,1]->[Z3,1,1,1];" DSACK1 aktiviert
        [.c.,.x.,1,1,1]->[Z3,1,1,1];
        [.x.,.c.,1,1,1]->[Z3,1,1,1];
        [.x.,.x.,0,1,0]->[Z3,1,0,1];
        [.c.,.x.,0,1,0]->[Z3,0,0,1];
        [.x.,.c.,0,0,0]->[Z0,0,0,0];" Ende weil DS_p inaktiv
        [.c.,.x.,0,0,0]->[Z0,0,0,0];
        [.x.,.c.,0,0,0]->[Z0,0,0,0];
        [.x.,.x.,0,0,0]->[Z0,0,0,0];
"-------------------------------------------------------------------
" DTERR-Test
RESET_all;
test_vectors([AS_p,DTACK_s]->DTERR);
        [0,0]->0;" Start
        [1,0]->0;" Adressen gueltig
        [1,1]->0;" Daten uebernommen
        [0,1]->1;" Adressen nicht mehr gueltig
```

```
        [0,1]->1;" Fehlerbedingung
        [0,0]->0;" ab jetzt wieder alles ok
        [1,0]->0;" neuer Zyklus
        [1,1]->0;
        [0,1]->1;
        [0,1]->1;
        [1,1]->1;" neuer AS muss maskiert werden
        [1,0]->0;" neuer AS ist erlaubt wenn DTACK fehlt
        [1,0]->0;
        [0,0]->0;" z. B. Busfehler
"-----------------------------------------------------------------
RESET_all;
test_vectors([DS_p,SIZ1_p,SIZ0_p,A0_p] -> [LDS.D,UDS.D])
        [0,.x.,.x.,.x.] -> [0,0];
        [1,0,0,0]       -> [1,1];
        [1,0,0,1]       -> [1,0];
        [1,0,1,0]       -> [0,1];
        [1,0,1,1]       -> [1,0];
        [1,1,0,0]       -> [1,1];
        [1,1,0,1]       -> [1,0];
        [1,1,1,0]       -> [1,1];
        [1,1,1,1]       -> [1,0];
"-----------------------------------------------------------------
" VMA-Test: best case read
RESET_all;
test_vectors"best case"([CLK_neg,AS_p,RnotW_p,!VPA_s]
                    -> [E_p,!VMA_s,DSACK1_syn])
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[0,1,0];
        [.c.,1,1,1]->[0,1,0];
        [.c.,1,1,1]->[0,1,0];
        [ 0 ,1,1,0]->[0,1,0];
        [.c.,1,1,0]->[0,1,0];" hier wird VPA_s getestet und erkannt...
        [.c.,1,1,0]->[0,0,0];" und in diesem Takt dann VMA_s aktiviert
        [.c.,1,1,0]->[0,0,0];
        [.c.,1,1,0]->[1,0,0];
```

```
        [.c.,1,1,0]->[1,0,0];
        [.c.,1,1,0]->[1,0,0];
        [.c.,1,1,0]->[1,0,1];" Datenuebernahme
        [.c.,1,1,0]->[0,0,1];
        [.x.,0,1,0]->[0,0,1];
        [.x.,0,0,1]->[0,1,1];
        [.x.,0,0,1]->[0,1,1];
        [.c.,0,0,1]->[0,1,0];
        [.c.,0,0,1]->[0,1,0];
"-----------------------------------------------------------------
" VMA-Test: best case write
RESET_all;
test_vectors"best case"([CLK_neg,AS_p,!RnotW_p,!VPA_s]
                    -> [E_p,!VMA_s,DSACK1_syn])
        [.c.,0,0,1]->[1,1,0];
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[1,1,0];
        [.c.,1,1,1]->[0,1,0];
        [.c.,1,1,1]->[0,1,0];
        [.c.,1,1,1]->[0,1,0];
        [.c.,1,1,0]->[0,1,0];" hier wird VPA_s getestet
        [.c.,1,1,0]->[0,0,0];
        [.c.,1,1,0]->[0,0,0];
        [.c.,1,1,0]->[1,0,0];
        [.c.,1,1,0]->[1,0,0];
        [.c.,1,1,0]->[1,0,0];
        [.c.,1,1,0]->[1,0,0];
        [.c.,1,1,0]->[0,0,1]; " Datenuebernahme
        [.c.,0,0,1]->[0,1,0];
        [.c.,0,0,1]->[0,1,0];
"-----------------------------------------------------------------
" VMA-Test: worst case
RESET_all;
test_vectors"worst case"([CLK_neg,!VPA_s] -> [E_p,!VMA_s])
        [.c.,1]->[1,1];
        [.c.,1]->[1,1];
        [.c.,1]->[1,1];
        [.c.,1]->[1,1];
```

```
        [.c.,1]->[0,1];
        [.c.,1]->[0,1];
        [.c.,1]->[0,1];
        [.c.,1]->[0,1];" hier wird VPA_s getestet
        [ 0 ,0]->[0,1];
        [.c.,0]->[0,1];
        [.c.,0]->[0,1];
        [.c.,0]->[1,1];
        [.c.,0]->[1,1];
        [.c.,0]->[1,1];
        [.c.,0]->[1,1];
        [.c.,0]->[0,1];
        [.c.,0]->[0,1];
        [.c.,0]->[0,1];
        [.c.,0]->[0,1];" hier wird nocheinmal getestet und diesmal erkannt
        [.c.,0]->[0,0];" also kann jetzt auch VMA_s aktiviert werden.
        [.c.,0]->[0,0];
        [.c.,0]->[1,0];
        [.c.,0]->[1,0];
        [.c.,0]->[1,0];
        [.c.,0]->[1,0];" Datenuebernahme
        [.c.,0]->[0,0];
        [.c.,1]->[0,1];
        [.c.,1]->[0,1];
"-------------------------------------------------------------------
end
```

FITTER FILE

Note: This file has been condensed in order to save trees.

```
AMD MACH FITR - MARKET RELEASE (1-24-91)
(C) - COPYRIGHT ADVANCED MICRO DEVICES INC., 1990
Flags Used:              Unplace=False        Max Packing=True
Flags Used:        Expand Small=False         Expand All=False
********************************************************************************
Mach PLD Fitter - v 1.4   68000->68020/68881 (68030/68882) (c) niessen, bode 1991
********************************************************************************
*** Timing Analysis for Signals
 Parameter   Min   Max          Signal List (Those having Max delay.)
      Tpd     1     2             AS_s          LDS_s          UDS_s
                                 VMA_s           BG_s
      Tsu     1     3               z0      DSACK1_syn
      Tco     0     1         DSACK1_p          LDS_s          UDS_s
                                   AS_s          VMA_s
      Tcr     1     2               z0      DSACK1_syn
 Key:
 Tpd - Combinatorial propagation delay, input to output
 Tsu - Combinatorial setup delay before clock
 Tco - Register clock to combinatorial output
 Tcr - Register thru combinatorial logic to setup
 All delay values are expressed in terms of array passes
*** Device Resource Checks
               Available       Used       Remaining
       Clocks:     2             2             0
         Pins:    38            30             8      ->    78%
    I/O Macro:    32             9            23
   Total Macro:    32            20            12
 Product Terms:   128            43            48      ->    62%
MACH-PLD Resource Checks OK!
Partitioning Design into Blocks...
*** Last Equations Placed in Blocks
Weakly -
Assign -            AS_s          UDS_s           BG_s        DSACK1_p
Assign -
*** Block Partitioning Results
              Array    Macros     # I/O    Buried   Product    Signal
```

```
              Inputs    Remain     Macro     Logic     Terms     Fanout
   Block-> A    22         6         5         5        40        11
   Block-> B    18         6         4         6        40         7
*** Block Signal List
Block-> A          LDS_s          CS_RAM          CS_FPU          AVEC_p
                DSACK1_syn            E_p            c0              c1
                        c2         VMA_s
Block-> B       DSACK1_p           BG_s          UDS_s           AS_s
                      AS            UDS           LDS             z0
                      z1          DTERR
|> INFORMATION F050 - Device Utilization..... *: 76 %
*** Feedback Map - 68000->68020/68881 (68030/68882) (c) niessen, bode 1991
 Gbl Inp .--.     I/O  .--+--A--+--.  I/O          I/O  .--+--B--+--.  I/O
         | 0|    FC2_p : 0|     |21| FC0_p      DTACK_s : 0|     |21| A0_p
         | 1|    FC1_p : 1|     |20| RESET_s            | 1|     |20| RESET_s
         | 2|      E_p : 2|     |19| A13_p        DTERR : 2|     |19|
         | 3|       c0 : 3|     |18| VMA_s           z0 : 3|     |18| /AS_s
         | 4|       c1 : 4|     |17| DS_p            z1 : 4|     |17| DS_p
         | 5|       c2 : 5|     |16| AS_p           UDS : 5|     |16| AS_p
         '--'  DSACK1_s  6|     |15: VPA_s      DSACK1_s  6|     |15: AS
               CS_RAM : 7|     |14: LDS         CS_RAM : 7|     |14: RMC_p
                A15_p : 8|     |13: RnotW_p           | 8|     |13: BG_p
                A24_p : 9|     |12: A31_p       SIZ0_p : 9|     |12: SIZ1_p
                A14_p :10|     |11: CS_FPU            |10|     |11: CS_FPU
                       '--+-u--u+--'                   '--+-u--u+--'
*** Logic Map - 68000->68020/68881 (68030/68882) (c) niessen, bode 1991
 Gbl Inp .--.     I/O  .--+--A--+--.  I/O          I/O  .--+--B--+--.  I/O
    AS_p| 0|    LDS_s | 0| 1   |21|              UDS_s | 0| 1   |21|
    DS_p| 1|    VMA_s | 1| 2   |20|              /AS_s | 1| 2   |20|
 CLK_neg| 2|      E_p | 2| 3   |19|              DTERR | 2| 2   |19|
 RESET_s| 3|       c0 | 3| 3   |18|                 z0 | 3| 2   |18|
   FC0_p| 4|       c1 | 4| 4   |17|                 z1 | 4| 3   |17|
   CLK_s| 5|       c2 | 5| 3   |16|                UDS | 5| 1   |16|
        '--'  DSACK1_s  6| 3  .|15|                    | 6| .  2|15| AS
              CS_RAM | 7| 3  .|14|                    | 7| .  3|14| LDS
              CS_FPU | 8| 1  .|13|                 BG_s | 8| 1  .|13|
              AVEC_p | 9| 1  .|12|              DSACK1_p  9| 2  .|12|
                     |10| .  .|11|                     |10| .  .|11|
                      '--+-u--u+--'                   '--+-u--u+--'
```

```
*** Pin Map - 68000->68020/68881 (68030/68882) (c) niessen, bode 1991

                                LDS_s
                             VMA_s  |        A15_p
                              E_p   |  |     |  A24_p
                        DTACK_s  |  |  |     |  |  A14_p
                           .  |  |  |  |     |  |  |  .
                           |  |  |  |  |     |  |  |  |
                      .-----'--'--'--'--'--o-----'--'--'--'---.
                      |                          4  4  4  4  4  |
                      |     6  5  4  3  2  1  4  3  2  1  0  |
                      | 7                                   39|A31_p
                      | 8                      G  V         38|RnotW_p
                      | 9                      n  c         37|DSACK1_p
                   AS_p|10                     d  c         36|BG_s
                   DS_p|11                                  35|CLK_s
                    Gnd |12            MACH-110              34| Gnd
                CLK_neg|13                                  33|FC0_p
                 CS_FPU|14              V  G                32|RESET_s
                 AVEC_p|15              c  n                31|FC2_p
                 SIZ0_p|16              c  d                30|FC1_p
                  A13_p|17                                   29|
                      |   1  1  2  2  2  2  2  2  2  2  2     |
                      |   8  9  0  1  2  3  4  5  6  7  8     |
                      '---.--.--.--.--------.--.--.--.--.-----'
                          |  |  |  |        |  |  |  |  |
                       VPA_s  |  |        |  |  |  |  '
                          RMC_p  |        |  |  |  A0_p
                              BG_p        |  |  '
                                SIZ1_p    |  /AS_s
                                          UDS_s
The Design Doc is  stored in ===> pak01.Rpt
The Jedec Data is  stored in ===> pak01.Jed
The Placements are stored in ===> pak01.Plc
The  Fuse Plot is  stored in ===> pak01.Xpt
%% FITR %% Error Count: 0, Warning Count: 2
%% FITR %% File Processed Successfully. - File: pak01
```

PLACEMENT FILE

```
; Flags Used:              Unplace=False        Max Packing=True
; Flags Used:        Expand Small=False          Expand All=False
; FITR generated placements
Pin    5          DTACK_s  ; Inp        ; A  3
Pin   18            VPA_s  ; Inp        ; A 12
Pin   35            CLK_s  ; Inp        ; I  5
Pin   32          RESET_s  ; Inp        ; I  3
Pin   25            /AS_s   Comb        ; B  1
Pin    2            LDS_s   Comb        ; A  0
Pin   24            UDS_s   Comb        ; B  0
Pin    3            VMA_s   Comb        ; A  1
Pin   36             BG_s   Comb        ; B  8
Node   4            E_p_I    Reg        ; A  2
Pin   20             BG_p  ; Inp        ; A 14
Pin   10             AS_p  ; Inp        ; I  0
Pin   11             DS_p  ; Inp        ; I  1
Pin   38          RnotW_p  ; Inp        ; B 10
Pin   19            RMC_p  ; Inp        ; A 13
Pin   21           SIZ1_p  ; Inp        ; A 15
Pin   16           SIZ0_p  ; Inp        ; A 10
Pin   31            FC2_p  ; Inp        ; B  7
Pin   30            FC1_p  ; Inp        ; B  6
Pin   33            FC0_p  ; Inp        ; I  4
Pin   39            A31_p  ; Inp        ; B 11
Pin   42            A24_p  ; Inp        ; B 14
Pin   43            A15_p  ; Inp        ; B 15
Pin   41            A14_p  ; Inp        ; B 13
Pin   17            A13_p  ; Inp        ; A 11
Pin   27             A0_p  ; Inp        ; B  3
Pin   37         DSACK1_p   Comb        ; B  9
Pin   15           AVEC_p   Comb        ; A  9
Pin   13          CLK_neg  ; Inp        ; I  2
Pin   14           CS_FPU   Comb        ; A  8
Node   9           CS_RAM   Comb        ; A  7
Node   8       DSACK1_syn    Reg        ; A  6
Node  20            DTERR   Comb        ; B  2
Node  32              LDS    Reg        ; B 14
```

```
Node   23              UDS     Reg         ; B  5
Node   33               AS     Reg         ; B 15
Node   22               z1     Reg         ; B  4
Node   21               z0     Reg         ; B  3
Node    7               c2     Reg         ; A  5
Node    6               c1     Reg         ; A  4
Node    5               c0     Reg         ; A  3
Pin     4             E_p      Reg         ; A  2
; Group Mach_Seg_A LDS_s CS_RAM CS_FPU AVEC_p DSACK1_syn E_p c0 c1 c2 VMA_s
; Group Mach_Seg_B DSACK1_p BG_s UDS_s AS_s AS UDS LDS z0 z1 DTERR
```