Bank Interleaved Memory System for an Am29030™ Microprocessor

Advanced Micro Devices

Application Note

by David Stoenner

This application note explains how to modify the EZ-030 demonstration board to increase the bus speed from 16 MHz to 30 MHz.

INTRODUCTION

Although memory has grown faster over the years, memory speed has not kept up with the speed of processors or the demand for even faster microprocessor systems, be they desktop or embedded. The EZ-030 demonstration board addresses the need for a low-cost and simple memory system for the AMD Family of Am29030 microprocessor RISC-based machines, yet it is limited to a maximum bus speed of 16 MHz (see the *EZ-030 Demonstration Board Theory of Operation* application note for more information). When faster memories, faster PAL® devices, and faster multiplexers (74F157s) are substituted in the EZ-030 board design, 20 MHz is possible, but only in PGA versions.

Somewhere in the gray area of 20 to 25 MHz is the point where memory can no longer be single-cycled out of one memory bank. The limiting factor is the CAS access time and the processor setup time for data. However, DRAMs have a shorter data valid time from their output enable. So, if two banks are connected at the data pins and then independently cycled and the output enable (OE) of each bank is used to multiplex the values together, the memories can support the single-cycle access for which the 29K™ Family is so famous. This technique is called *bank interleaving*.

This application note explains how a few simple enhancements to the EZ-030 logic design, along with another memory bank, can extend the upper frequency limit to 30 MHz.

CHANGES TO DESIGN

To achieve a 30-MHz speed, make the following changes to the EZ-030 logic design:

- Replace the 16-MHz Am29030 processor with a 33-MHz Am29030 processor
- Add a memory bank
- Add an extra 16V8 to the lower address lines of the Am29030 processor for advanced counting
- Change the MSTR_CON PAL device from a 16V8 to a 22V10
- Change the CAS_DEC PAL device from a 16V8 to a 22V10

Add a Memory Bank and a Counter

In a bank interleaved memory design, two memory systems with their outputs connected together form a single data bus. While one bank provides data for the current cycle, the other bank gets the next value. This requires the new bank to be ahead of the current bank, which requires an address one greater than what the processor provides. This is accomplished in the design by adding a counter on the lower processor address lines that drives the CAS addresses in the memory bank. Processor address A2 is used to select between banks while A9–A3 drive CAS A6–A0. This means the count can be loaded, incremented, or held in the counter. Since the CAS address is not needed until the second clock cycle, there is plenty of time to get the counter up-to-date with the correct address.

Note that this counter needs to be pre-incremented if the burst starts at an address boundary where A2 of the access is 1. In this case, the odd memory bank starts up first but the counter is quickly incremented to the next value, so the new CAS can start the even bank access at the next larger A9–A3 value. No address counting past A9 is needed since internal address comparators on the Am29030 processor stop all burst transactions on the bus at 1K boundaries.

Change the MSTR_CON PAL Device

The second design change is in the master state machine. Since the clock is now higher in frequency but the memories are not any faster, the first access must have a greater number of clock cycles to have the same time value as those at 16 MHz. The memories used are 80-ns RAS access DRAMs, so at least a 3 clock-cycle access is needed once a RAS is started. RAS is not started until a $\overline{\text{REQ}}$ from the processor has been asserted for one clock cycle, which makes the first access in this memory system 4 clocks.

The other value that must be accounted for is RAS precharge, the value that RAS is de-asserted before the next RAS access. This value is 50 ns in 80-ns memory, which was 1 clock at 16 MHz but is now 2 clocks at 30 MHz. Both of these requirements add two more states to the master control PAL device and thereby necessitate a larger number of outputs than was in the previous design. This is accomplished by changing the

Publication# **18478** Rev. **A** Amendment **/0**
Issue Date: **May 1994**

16V8 MSTR_CON PAL device in the EZ-030 board design to a 22V10. In addition, the RDY term in the EZ-030 board MSTL_CTL PAL device is moved outside all the PAL devices and put in a simple 74F08 used to OR the two sources of $\overline{\text{RDY}}$ (MEM_RDY and SERIAL_RDY) for the processor.

Change the CAS_DEC PAL Device

The third design change centers around the memory subsystem. The two banks are common in their RAS generation and address multiplexing, but the CAS generation must take into account the two separate banks as well as the need to be byte-writable on an individual basis. This requires 10 outputs for this function, so the CAS_DEC PAL device in the EZ-030 board design is now replaced with another 22V10.

These are the needed design changes at a high level; the next section looks at the PAL devices in more detail.

THEORY OF OPERATION

The MSTR_CON, CAS_DEC, REFRESH, and ADD_CTR PAL devices are discussed in the following sections. The PAL equations referenced are provided in Appendix B at the end of the document. (The SERIAL PAL device is the same as that discussed in the EZ-030 board application note; the SERIAL PAL equations are shown in Appendix B.) Appendix A contains the schematics for the design.

MSTR_CON PAL Device

The state machine in the MSTR_CON PAL device represents the heart of the bank interleaved memory design. It controls the main memory functions of both the DRAM access and the refresh of the DRAMs, and also provides decoding and timing for the ROM access. This is done in a way that hides the refreshes so they do not impact performance; for example, DRAM refresh can be performed while ROM is being accessed. Additionally, state bits are shared between state machines so that extra flip-flops are not needed. For example, this sharing is done for ROM access because a 3-cycle access on ROM is needed at 30 MHz, while in 16 MHz only 2 cycles are needed. So a simple division by 2 needs to change to a division by 3, which requires one more state bit.

The state machine in Figure 1 shows the normal path of the EZ-030 board design, where IDLE is the state in which the machine is at rest and is ready to arbitrate a decision as to which path is chosen.

A refresh has priority in the arbiter, so a REF_ACCESS is taken regardless of the state of $\overline{\text{REQ}}$ in the Am29030 processor. MEM_ACCESS is taken only when there is no REF_REQ with a REQ, and when the address lines A31=1 and A30=0 are valid. The last state of CAS in the MEM_ACCESS holds and MEM_RDY is driven True in this state until the $\overline{\text{BURST}}$ signal from the Am29030 processor goes False.

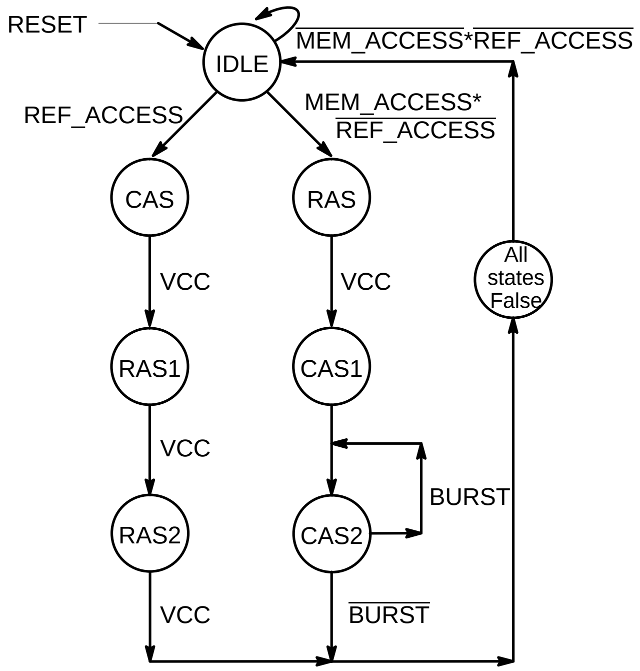

RESET
IDLE
$\overline{\text{MEM\_ACCESS}}$*$\overline{\text{REF\_ACCESS}}$
REF_ACCESS
MEM_ACCESS*
$\overline{\text{REF\_ACCESS}}$
CAS
RAS
VCC
VCC
All states False
RAS1
CAS1
VCC
BURST
RAS2
CAS2
VCC
$\overline{\text{BURST}}$
18478A-1

Figure 1. State Diagram

If the $\overline{\text{BURST}}$ signal is False when $\overline{\text{REQ}}$ is asserted at the beginning of an access, this is a single access and the machine goes to the bottom and cycles back around. An extra precharge clock cycle is needed at the end of a memory cycle to ensure that no two accesses are too close. This is accomplished by waiting until all state bits go False before moving to the IDLE state. Once the state machine moves to IDLE, it can re-arbitrate. Because the IDLE cycle occurs before either RAS or CAS can be asserted, this ensures that 2 clock cycles are performed on back-to-back accesses with RAS False, which meets the precharge requirement.

OE_BANK0 serves as a dual-function signal to the state machine. It is used on MEM_ACCESS to enable the even bank and is normally a division by 2 in a burst sequence. However, a new state is needed in REF_ACCESS to time the memory cycle so, in a refresh cycle, OE_BANK0 is used to signal the last state (RAS2) of a refresh cycle. OE_BANK0 and OE_BANK_1 are also used in MEM_ACCESS to provide an indication of the last state as well as provide the output enable (OE) multiplexing for the DRAMs.

The remaining signals used for a DRAM access are RAS, INC, and MEM_RDY. RAS is a term that is registered to provide the highest speed to the memory interface. RAS has the same timing as MEM_ACCESS, or is delayed by one clock so that a REF_ACCESS can provide the CAS-before-RAS signaling for a refresh.

INC controls the incrementing of the address counter on the Am29030 processor's A9–A3 bits. INC is a registered term for ease of state control. It becomes True in an access that starts with A2=0, as MEM_RDY becomes True at the 3-clock cycle from RAS. INC then starts a division by 2 until all accesses have been completed. If A2=1 on the first access, INC becomes True one clock cycle before MEM_RDY becomes True, so as to increment the CAS address to the next higher bank before the even bank CAS can be asserted.

MEM_RDY is also a registered term and provides all ready generation for both DRAM and ROM access. In MEM_ACCESS, MEM_RDY becomes True once, if $\overline{\text{BURST}}$ from the Am29030 processor is False on the first access; or, if $\overline{\text{BURST}}$ is True on the first access, MEM_RDY stays True as long as $\overline{\text{BURST}}$ stays True from the Am29030 processor.

ROM access does not use the master state machine but instead uses the MEM_RDY line to provide one of the clock cycles of access timing. There are, however, three cycles that are needed.

The first cycle is a $\overline{\text{REQ}}$ signal from the Am29030 processor, with A31=0 and A30=0. Because a ROM access is taking place, the state bit INC, which controls the address pre-counter, is not needed for a REF_ACCESS. So, in the second cycle, INC is set to True. Finally, in the third cycle, MEM_RDY is set to True. Then both bits are set to False as the Am29030 processor changes the address on the bus for the next access.

Just as in the EZ-030 board design, this modified design carries forth the single ROM for boot-up and monitor code, so the $\overline{\text{RDN}}$ pin is controlled for all ROM accesses by the ROM_CS term in the MSTR_CON PAL device. During reset, this pin is False (high) so this signals the internal logic of the Am29030 processor that a $\overline{\text{RDN}}$ is 8-bits wide as opposed to 16-bits wide.

CAS_DEC PAL Device

The CAS_DEC PAL device has all the CAS signal generation as well as two extra terms: DIV_CLK and SECOND_ACCESS. DIV_CLK clocks the refresh counter and the I/O control PAL device, which is needed because of the higher clock rate of this design as compared to the EZ-030 board design. This keeps the two PAL devices compatible to their original timing.

SECOND_ACCESS is used in the CAS pulse generation and controls the INC term in the MSTR_CON PAL device. It goes True after the first access to DRAM has taken place and stays True until MEM_ACCESS goes False.

The CAS pulse generation is the most complicated of the modifications. There are eight CAS lines: one set of four for the even bank and one set of four for the odd bank. Each set of four controls byte-enable access, especially for writes to memory. Reads are always performed 32 bits (one word) at a time. The equations are then essentially copies of each other, with only the corresponding byte enable (WEx) used for writes and OE_BANKx used for bank control (see Equation 1).

The equations are divided into three parts. Part 1 is one line [1.a] and supports CAS before RAS refresh. Part 3 is one line [1.h] that controls writes. Writes are simply an AND of the MEM_CLK and the controlling OE_BANK term and the WEx term that controls that byte. Part 2 is the middle six terms [1.b–1.g] and forms the read pulse shaping. The CAS pulse starts out with SECOND_ACCESS False and the OE_BANK control True [1.b]. CAS is then held True as a regenerative latch [1.c] if it does go True. It is also held True through an entire cycle if SECOND_ACCESS is False and MEM_RDY is True [1.d]. When the SECOND_ACCESS term is True, the next three lines [1.e–1.g] form a pulse that makes CAS for this bank False for half a clock cycle on the opposite bank access [1.e]. This CAS False gets the new address in the CAS address latch on the inside of the DRAMs. Then the next line [1.f] provides a cross term to prevent output glitches while the OE_BANK term switches, and then the CAS is held True for the entire cycle that its own OE_BANK is True. All these equations [1.e–1.g] then provide a CAS pulse that is False for half a clock cycle, and True for the rest of the clock cycle and then the next full clock cycle, making the 2 clock-cycle access. Because of the higher clock frequency that provides a half clock cycle of 16.5 ns, no delay line is needed in this design as was needed in the EZ-030 board design. Note also in the PAL equation section that minimization is turned off for all the CAS equations because the cross term is logically redundant (but not from a timing standpoint) and if minimized out would create glitches on the CAS line that would adversely effect the DRAMs.

REFRESH PAL Device

The REFRESH PAL device is identical to the one used in the EZ-030 board design and is a 7-bit counter that

$$
\begin{array}{lll}
\text{BANK0\_CAS0} = & \text{REF\_ACCESS} & [1.a] \\
 & +\ \text{MEM\_ACCESS*}\overline{\text{SECOND\_ACCESS}}\text{*}\overline{\text{WRITE}}\text{*OE\_BANK0*}\overline{\text{MEM\_RDY}}\text{*}\overline{\text{MEMCLK}} & [1.b] \\
 & +\ \text{MEM\_ACCESS*}\overline{\text{SECOND\_ACCESS}}\text{*}\overline{\text{WRITE}}\text{*OE\_BANK0*BANK0\_CAS0} & [1.c] \\
 & +\ \text{MEM\_ACCESS*}\overline{\text{SECOND\_ACCESS}}\text{*}\overline{\text{WRITE}}\text{*OE\_BANK1*MEM\_RDY*}\overline{\text{MEMCLK}} & [1.d] \\
 & +\ \text{MEM\_ACCESS*SECOND\_ACCESS*MEM\_RDY*OE\_BANK0*}\overline{\text{WRITE}} & [1.e] \\
 & +\ \text{MEM\_ACCESS*SECOND\_ACCESS*MEM\_RDY*OE\_BANK1*}\overline{\text{WRITE}}\text{*}\overline{\text{MEMCLK}} & [1.f] \\
 & +\ \text{MEM\_ACCESS*SECOND\_ACCESS*MEM\_RDY*}\overline{\text{WRITE}}\text{*BANK0\_CAS0*MEMCLK} & [1.g] \\
 & +\ \text{MEM\_ACCESS*MEM\_RDY*OE\_BANK0*WRITE*WE0*}\overline{\text{MEMCLK}} & [1.h]
\end{array}
$$

Equation 1. CAS_DEC PAL Equation

counts off 128 DIV_CLK before setting REF_REQ. REF_REQ is held active until MSTR_CON acknowledges this request by issuing a REF_ACCESS. This then resets the REF_REQ bit. The count of 256 33-ns cycles gives a refresh of 8.45 µs, which is an over refresh but accounts for the possibility of an Am29030 processor burst losing a whole cycle count in the REFRESH PAL device.

ADDR_CTR PAL Device

The last PAL device to examine is the ADDR_CTR PAL device. This device is a 7-bit counter that is in between the A9–A3 address bits of the Am29030 processor and the 74F157 address multiplexers. This PAL device has two control terms: LOAD and INC. All the bits are registered with the MEM_CLK, making this PAL device a clocked, loadable, and controllable counter. LOAD is controlled by IDLE of the MSTR_CON PAL device and when the master state machine is in IDLE, the output registers are constantly being updated by their corresponding address bits. When a MEM_ACCESS starts, the address is held until MSTR_CON gives an INC control and then the outputs count up by 1.

Additional Considerations

One hidden caveat in both the original and modified designs is that the write signal passes through the 74F157 multiplexers. This is because refresh has priority over memory accesses, and because 4-Mbyte DRAMs and above have a new test mode such that if write is Low during a CAS before RAS refresh, DRAM is put in a special test mode. By not making MUX True during a REF_ACCESS and by putting $\overline{\text{WRITE}}$ from the processor into the multiplexer, the problem is not only prevented but the write signal to the DRAM is buffered.

TIMING ANALYSIS

The timing analysis is simple in this design. There are four critical paths to examine: the ROM access, the RAS access, the CAS access, and the OE access.

ROM Access Time

The ROM access is controlled by the processor address valid time, plus the address access time of the ROM, plus the Am29030 processor setup time. The equation is as follows:

$$T_{\text{clk–q 29030}} + T_{\text{ROM access}} + T_{\text{setup 29030}} \leq 3 \text{ MEMCLKs}$$

This results in a value of 13 + ROM access + 9 $\leq$ 99, making a ROM of access time 77 ns or less acceptable. AMD makes 70-ns EPROMs so that speed is used for this design.

RAS Access Time

RAS access time is given by the following equation:

$$T_{\text{clk–q 22V10–7}} + T_{\text{RAS access}} + T_{\text{setup 29030}} \leq 3 \text{ MEMCLKs}$$

This results in a value of 5 + 80 + 9 $\leq$ 99, which is acceptable.

CAS Access Time

CAS access time is given by the following equation:

$$T_{\text{clk–q 22V10–7}} + T_{\text{CAS access}} + T_{\text{setup 29030}} \leq 2 \text{ MEMCLKs}$$

This results in a value of 5 + 35 + 9 $\leq$ 66, which is acceptable.

OE Access Time

The real critical-path item is the OE access time. Its equation is given by the following:

$$T_{\text{clk–q 22V10–7}} + T_{\text{OE access}} + T_{\text{setup 29030}} \leq \text{MEMCLK}$$

This results in a value of 5 + 20 + 9 $\leq$ 33, or 34 $\leq$ 33. Although the result is off by 1 ns, it can still be used if all of the memory subsystem is kept very close to the processor to minimize transmission line issues, knowing that the worst-case timing is always at high temperature, low voltage, and worst-case parts, simultaneously. This could be guaranteed in a manufacturing environment by running the design at 33 MHz for all system tests and then shipping at 30 MHz as designed.

CONCLUSION

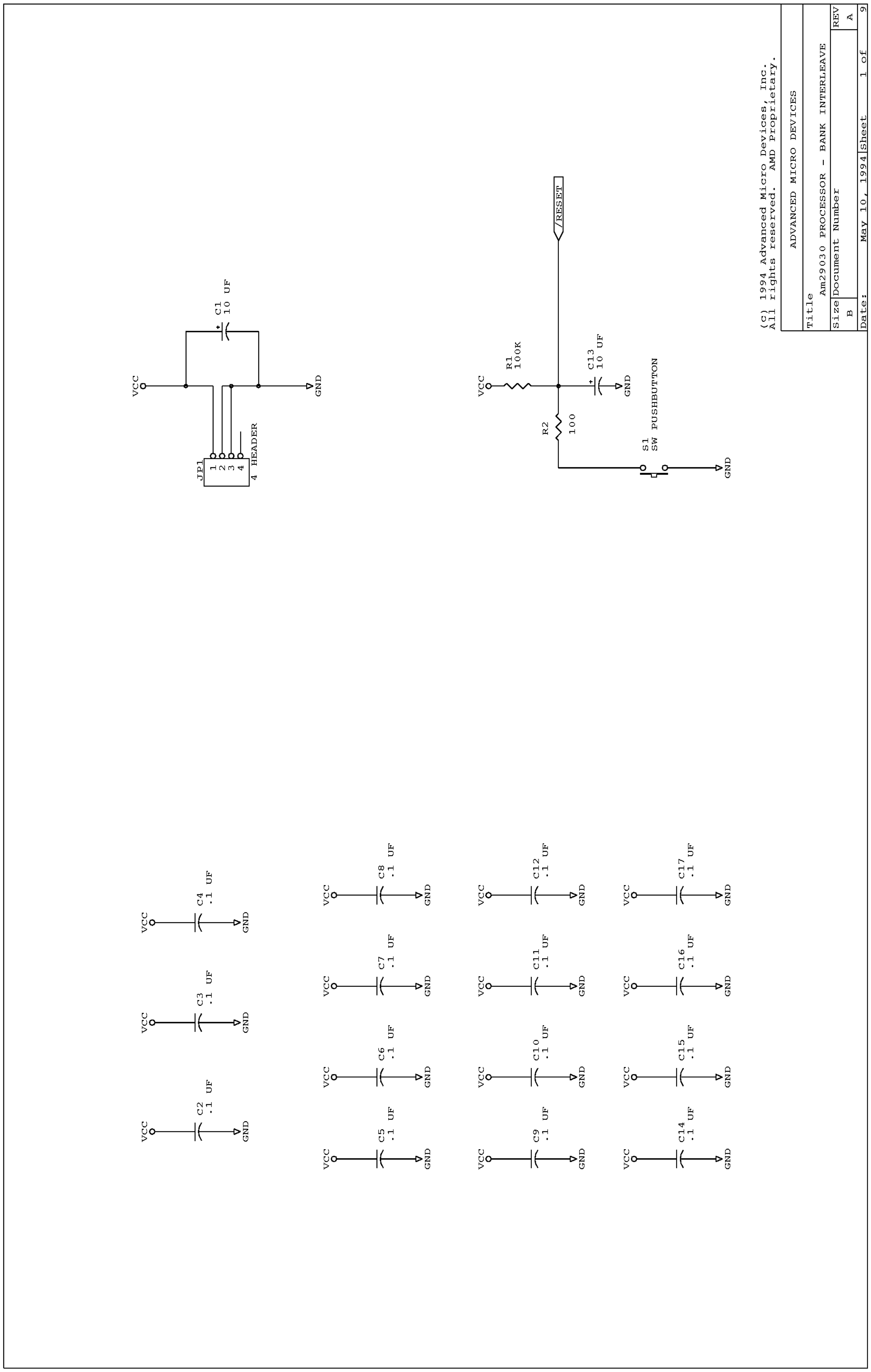
Although bank interleaving is not without an attendant cost factor and complexity of design, the design presented in this application note shows how easily an extension to the original EZ-030 board design can be made to bring the speed up to double the original design. The increased complexity turns out to be only one extra PAL device. A delay line from the original design was even eliminated. The performance of this modified design is a 4 clock-cycle first access and single-cycle burst, while the original EZ-030 board design is a 3 clock-cycle first access and single-cycle burst. The extra cost of the design is only the increase in the cost of 7.5-ns PAL devices and the extra memory array, plus the cost of a 33-MHz Am29030 processor as opposed to a 16-MHz Am29030 processor. The performance, while not quite double, comes close in a very compact design.

SUGGESTED REFERENCE

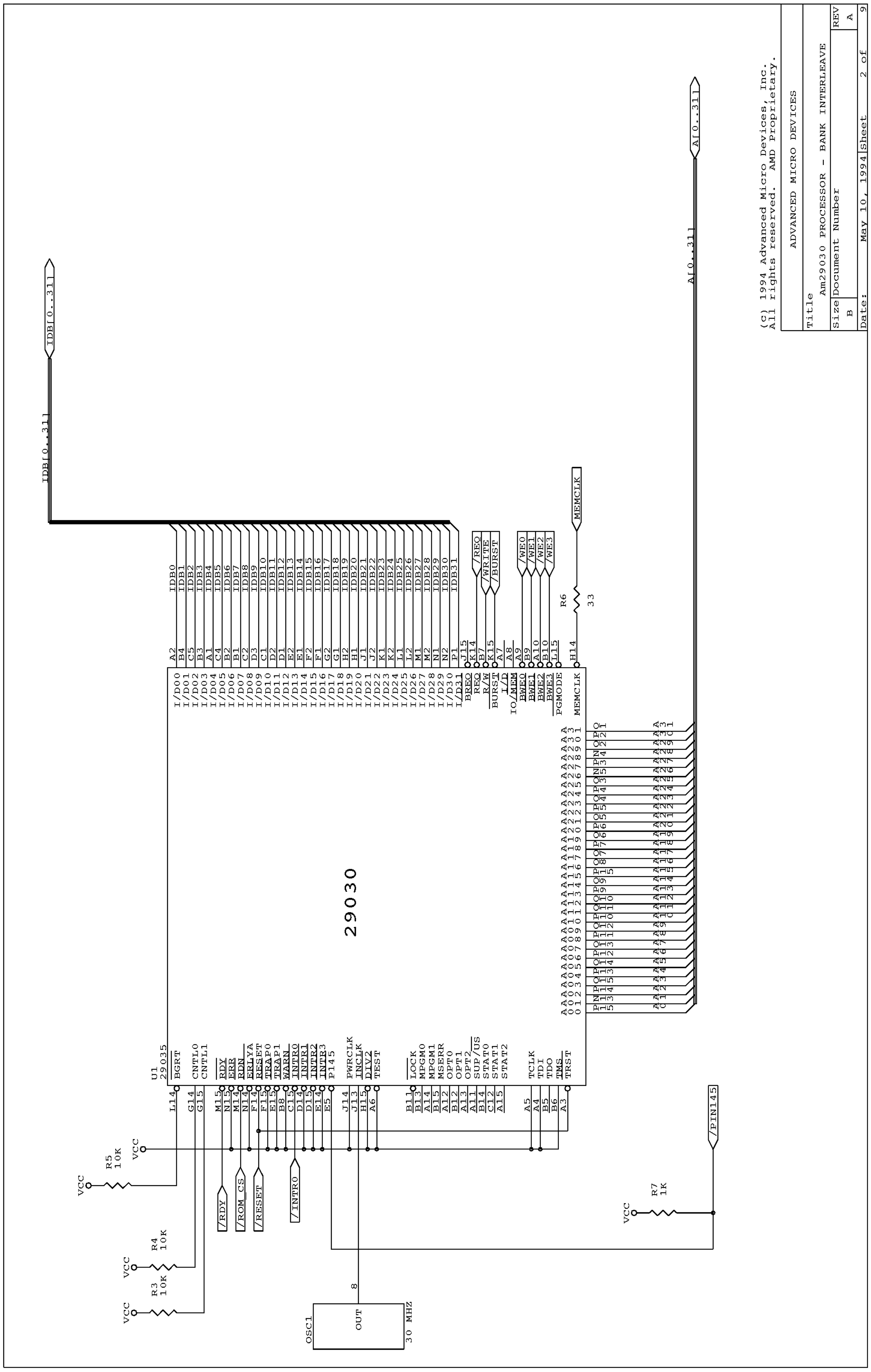
- *EZ-030 Demonstration Board Theory of Operation* application note, order# 17580, Advanced Micro Devices
- *Redesigning the EZ-030 Demonstration Board with a PCI I/O Bus* application note, order# 18468, Advanced Micro Devices

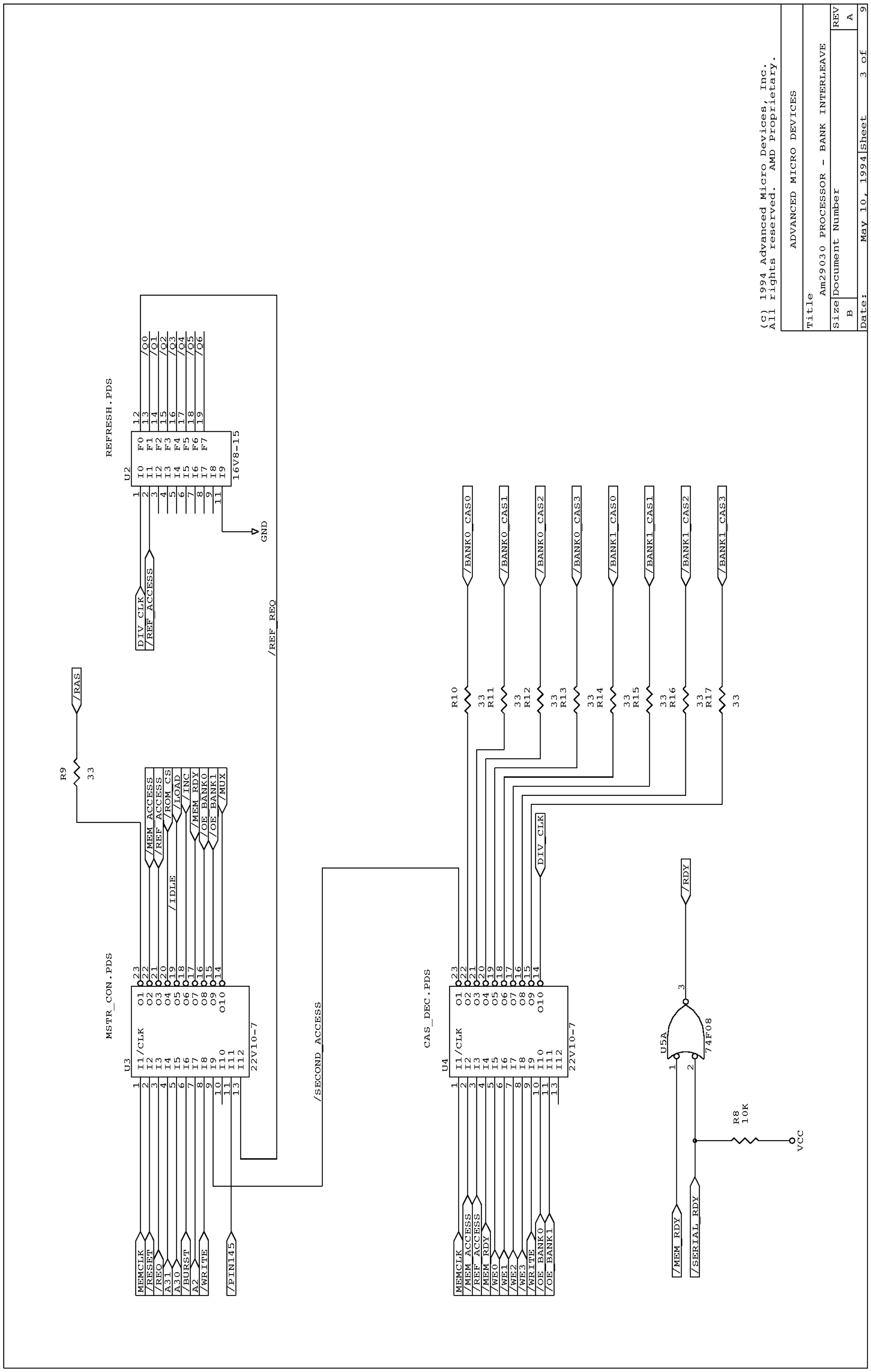
Appendix A. Schematics

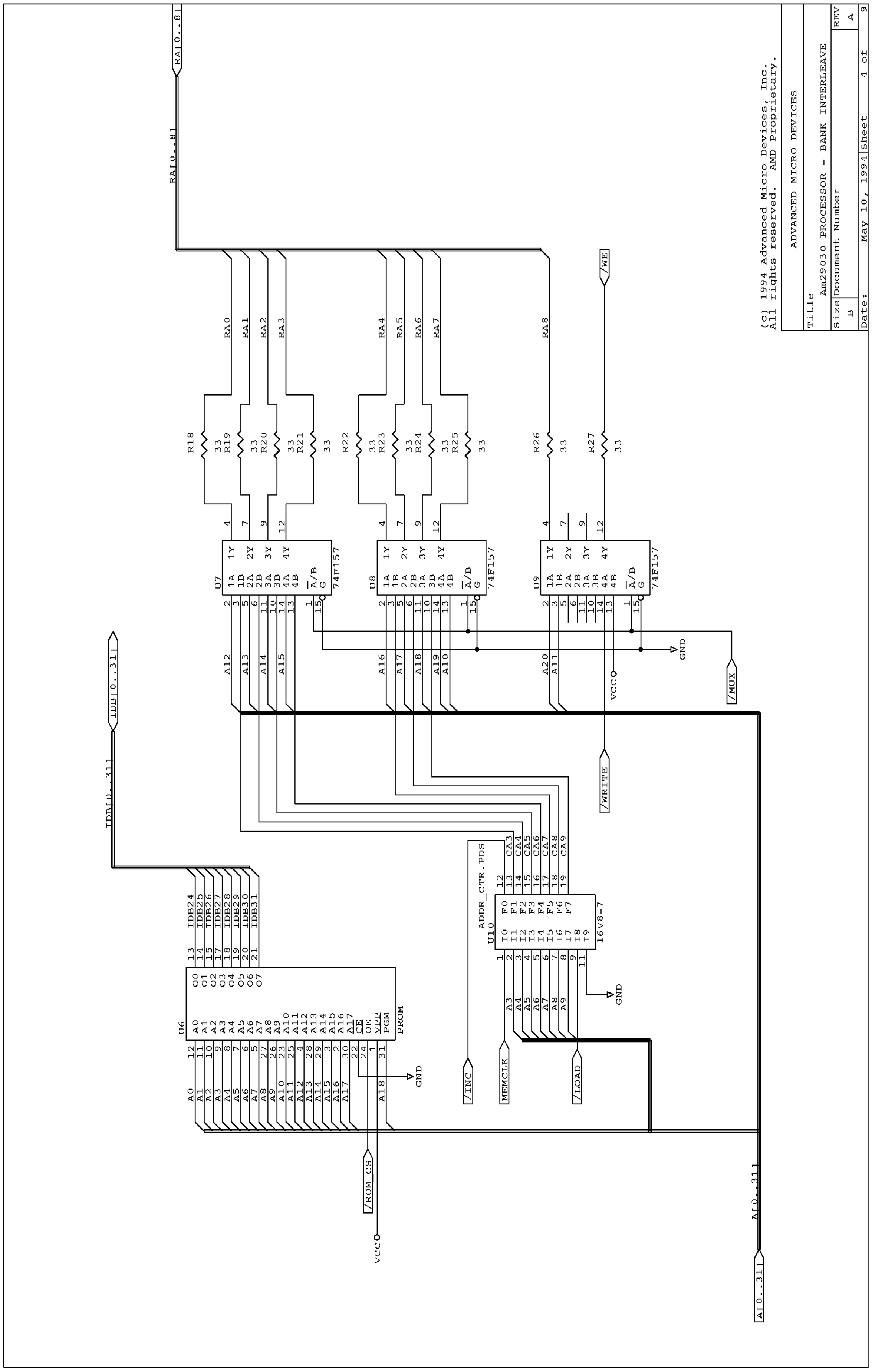
The schematics for this design are shown on the pages that follow.

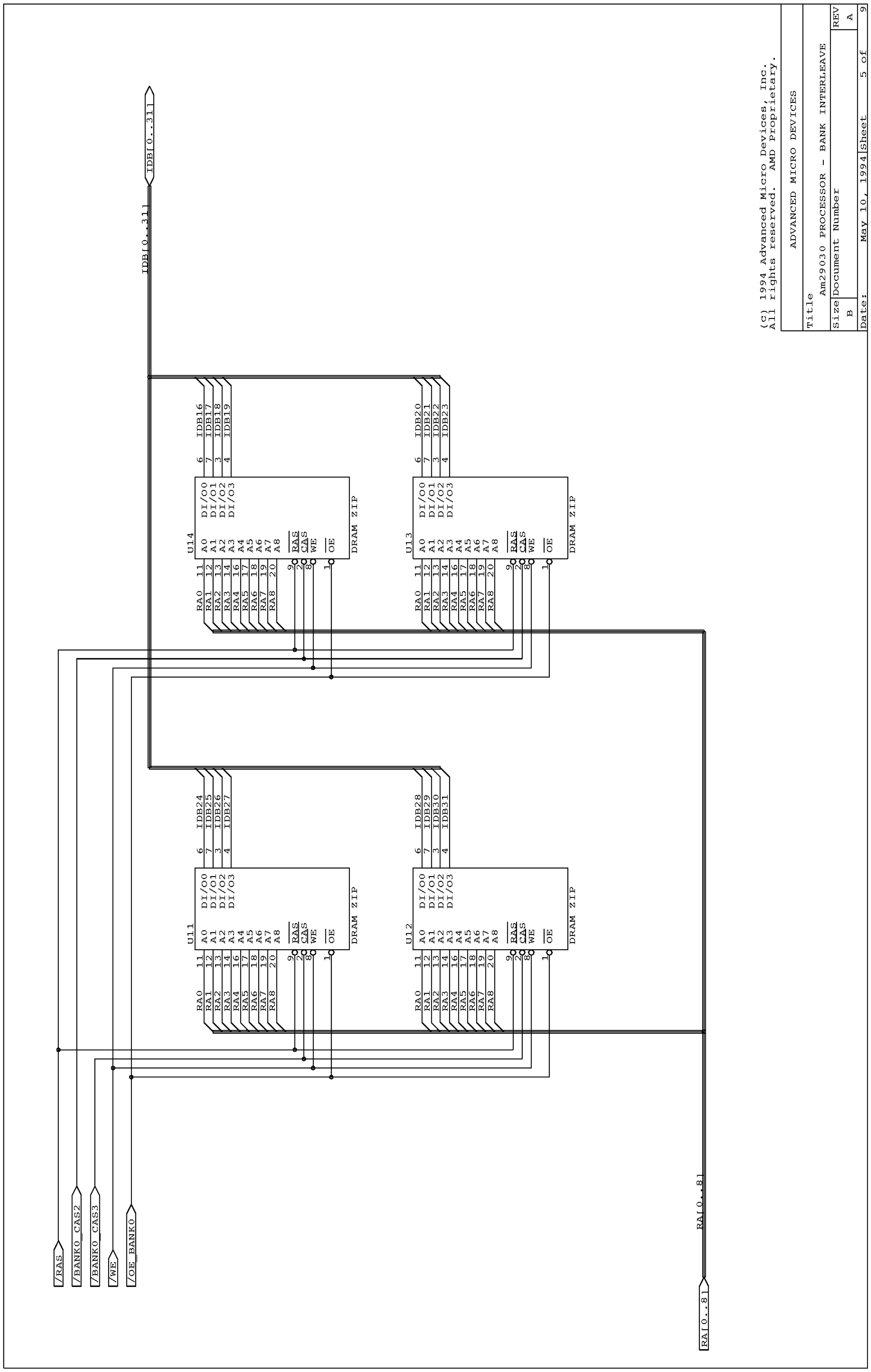


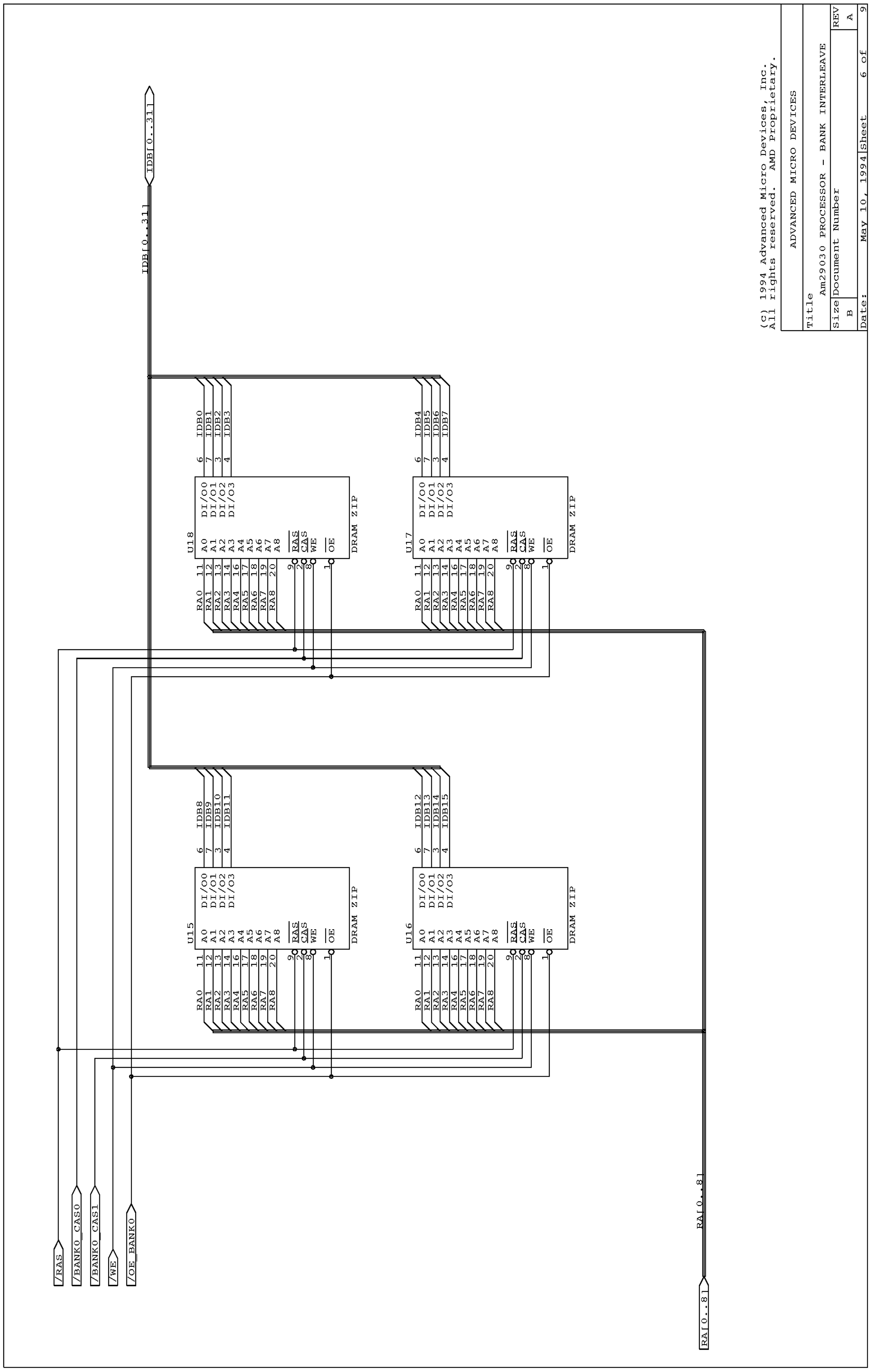
(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 1 of 9
/RESET
VCC
R1
100K
C13
10 UF
GND
R2
100
S1
SW PUSHBUTTON
GND
VCC
C1
10 UF
GND
JP1
1
2
3
4
4 HEADER
VCC
C2
.1 UF
GND
C3
.1 UF
C4
.1 UF
C5
.1 UF
C6
.1 UF
C7
.1 UF
C8
.1 UF
C9
.1 UF
C10
.1 UF
C11
.1 UF
C12
.1 UF
C14
.1 UF
C15
.1 UF
C16
.1 UF
C17
.1 UF

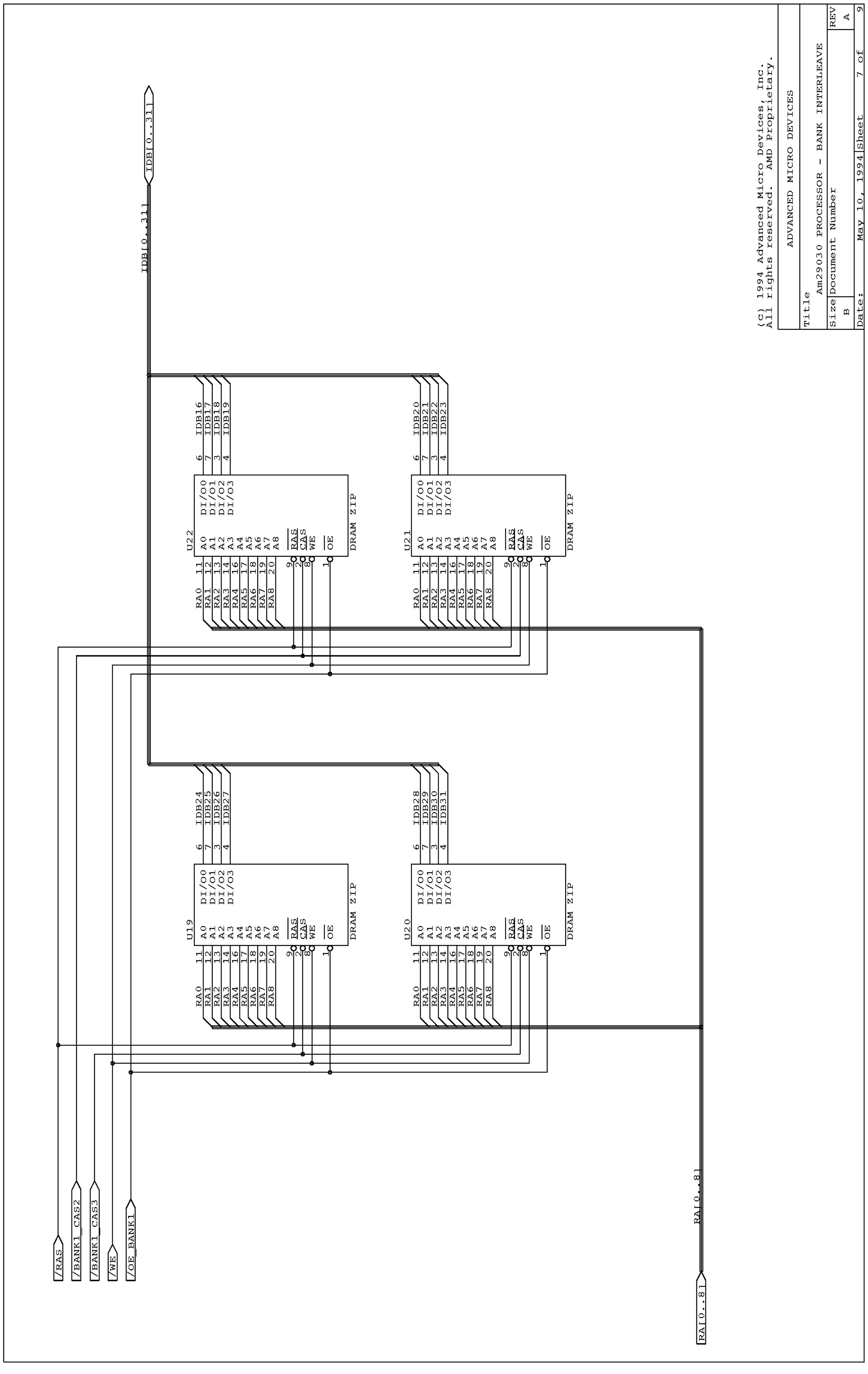

(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 2 of 9
29030
U1 29035
IDB[0..31]
A[0..31]
MEMCLK
R6 33
/REQ
/WRITE
/BURST
/WE0
/WE1
/WE2
/WE3
/RDY
/ROM_CS
/RESET
/INTR0
/PIN145
R7 1K
R5 10K
R4 10K
R3 10K
VCC
OSC1
OUT
30 MHZ
8

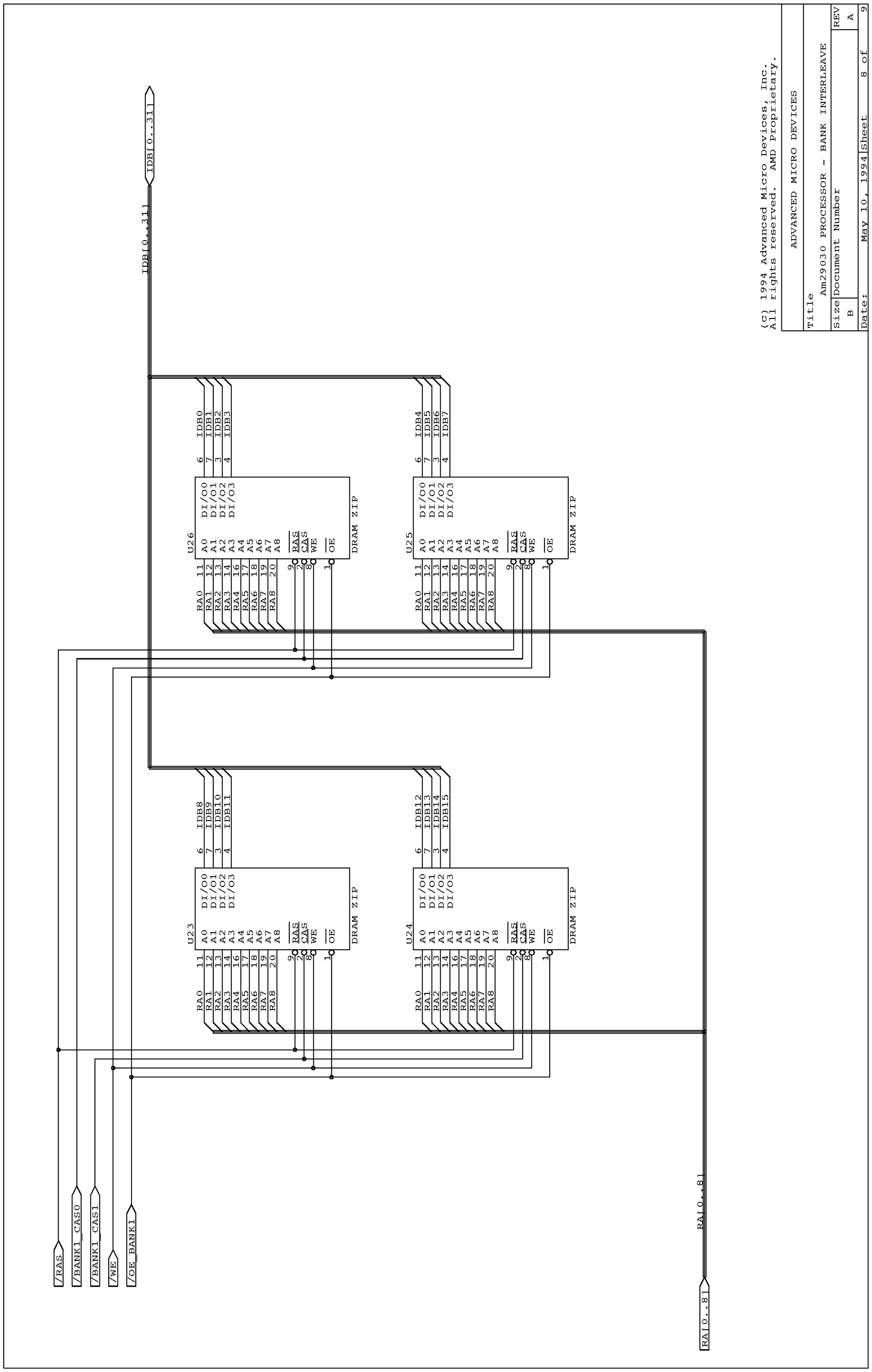

(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 3 of 9
REFRESH.PDS
U2
16V8-15
/REF_REQ
DIV_CLK
/REF_ACCESS
GND
/Q0
/Q1
/Q2
/Q3
/Q4
/Q5
/Q6
/RAS
R9
33
MSTR_CON.PDS
U3
22V10-7
/MEM_ACCESS
/REF_ACCESS
/ROM_CS
/LOAD
/INC
/MEM_RDY
/OE_BANK0
/OE_BANK1
/MUX
/IDLE
/SECOND_ACCESS
MEMCLK
/RESET
/REQ
A31
A30
/BURST
A2
/WRITE
/PIN145
CAS_DEC.PDS
U4
22V10-7
/BANK0_CAS0
/BANK0_CAS1
/BANK0_CAS2
/BANK0_CAS3
/BANK1_CAS0
/BANK1_CAS1
/BANK1_CAS2
/BANK1_CAS3
R10
R11
R12
R13
R14
R15
R16
R17
33
DIV_CLK
MEMCLK
/MEM_ACCESS
/REF_ACCESS
/MEM_RDY
/WE0
/WE1
/WE2
/WE3
/WRITE
/OE_BANK0
/OE_BANK1
/RDY
U5A
74F08
R8
10K
VCC
/MEM_RDY
/SERIAL_RDY

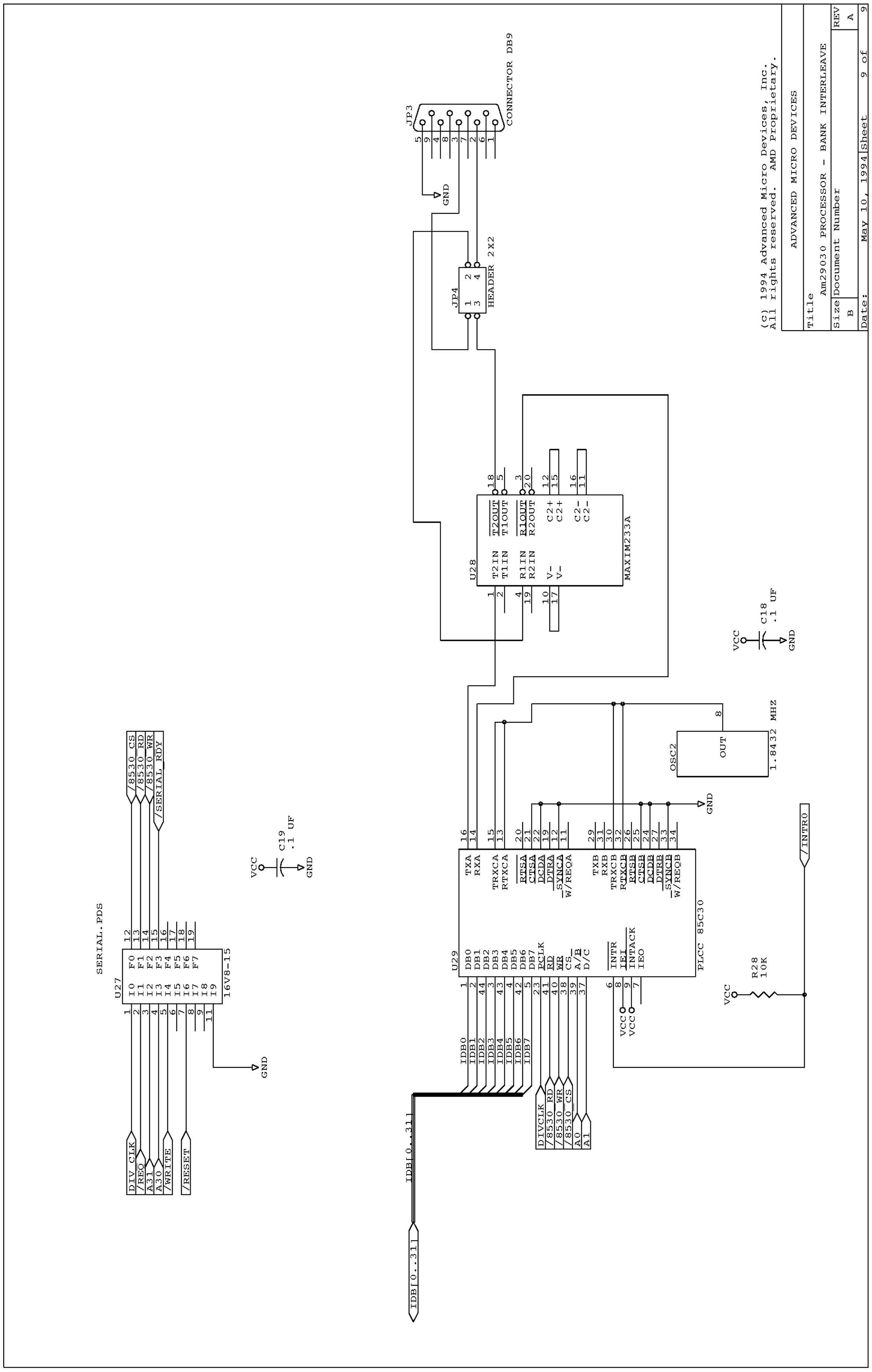

(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 4 of 9
RA[0..8]
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
33
/WE
U7
U8
U9
74F157
1A 1B 2A 2B 3A 3B 4A 4B
1Y 2Y 3Y 4Y
A/B
G
GND
VCC
/MUX
/WRITE
A12
A13
A14
A15
A16
A17
A18
A19
A10
A20
A11
IDB[0..31]
IDB24
IDB25
IDB26
IDB27
IDB28
IDB29
IDB30
IDB31
U6
PROM
O0 O1 O2 O3 O4 O5 O6 O7
CE
OE
VPP
PGM
/ROM_CS
U10
ADDR_CTR.PDS
16V8-7
I0 I1 I2 I3 I4 I5 I6 I7 I8 I9
F0 F1 F2 F3 F4 F5 F6 F7
CA3
CA4
CA5
CA6
CA7
CA8
CA9
/INC
MEMCLK
/LOAD
A[0..31]


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 5 of 9
IDB[0..31]
U14
U13
U11
U12
DRAM ZIP
DI/O0
DI/O1
DI/O2
DI/O3
A0 A1 A2 A3 A4 A5 A6 A7 A8
RAS
CAS
WE
OE
IDB16 IDB17 IDB18 IDB19
IDB20 IDB21 IDB22 IDB23
IDB24 IDB25 IDB26 IDB27
IDB28 IDB29 IDB30 IDB31
RA0 RA1 RA2 RA3 RA4 RA5 RA6 RA7 RA8
RA[0..8]
/RAS
/BANK0_CAS2
/BANK0_CAS3
/WE
/OE_BANK0


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 6 of 9
IDB[0..31]
U18
U17
U15
U16
DRAM ZIP
DI/O0
DI/O1
DI/O2
DI/O3
A0
A1
A2
A3
A4
A5
A6
A7
A8
RAS
CAS
WE
OE
IDB0
IDB1
IDB2
IDB3
IDB4
IDB5
IDB6
IDB7
IDB8
IDB9
IDB10
IDB11
IDB12
IDB13
IDB14
IDB15
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8
RA[0..8]
/RAS
/BANK0_CAS0
/BANK0_CAS1
/WE
/OE_BANK0


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 7 of 9
IDB[0..31]
U22
U21
U19
U20
DRAM ZIP
IDB16
IDB17
IDB18
IDB19
IDB20
IDB21
IDB22
IDB23
IDB24
IDB25
IDB26
IDB27
IDB28
IDB29
IDB30
IDB31
DI/O0
DI/O1
DI/O2
DI/O3
A0
A1
A2
A3
A4
A5
A6
A7
A8
RAS
CAS
WE
OE
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8
RA[0..8]
/RAS
/BANK1_CAS2
/BANK1_CAS3
/WE
/OE_BANK1


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size B
Document Number
REV A
Date: May 10, 1994
Sheet 8 of 9
IDB[0..31]
RA[0..8]
/RAS
/BANK1_CAS0
/BANK1_CAS1
/WE
/OE_BANK1
U26
U25
U23
U24
DRAM ZIP
IDB0
IDB1
IDB2
IDB3
IDB4
IDB5
IDB6
IDB7
IDB8
IDB9
IDB10
IDB11
IDB12
IDB13
IDB14
IDB15
DI/O0
DI/O1
DI/O2
DI/O3
A0
A1
A2
A3
A4
A5
A6
A7
A8
RAS
CAS
WE
OE
RA0
RA1
RA2
RA3
RA4
RA5
RA6
RA7
RA8


(C) 1994 Advanced Micro Devices, Inc.
All rights reserved. AMD Proprietary.
ADVANCED MICRO DEVICES
Title
Am29030 PROCESSOR - BANK INTERLEAVE
Size
B
Document Number
REV
A
Date: May 10, 1994
Sheet 9 of 9
CONNECTOR DB9
JP3
GND
JP4
HEADER 2X2
U28
MAXIM233A
T2IN
T1IN
R1IN
R2IN
V-
T2OUT
T1OUT
R1OUT
R2OUT
C2+
C2-
C18
.1 UF
VCC
GND
OSC2
OUT
1.8432 MHZ
U29
PLCC 85C30
TXA
RXA
TRXCA
RTXCA
RTSA
CTSA
DCDA
DTRA
SYNCA
W/REQA
TXB
RXB
TRXCB
RTXCB
RTSB
CTSB
DCDB
DTRB
SYNCB
W/REQB
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
PCLK
RD
WR
CS
A/B
D/C
INTR
IEI
INTACK
IEO
/INTR0
R28
10K
IDB[0..31]
IDB0
IDB1
IDB2
IDB3
IDB4
IDB5
IDB6
IDB7
DIVCLK
/8530 RD
/8530 WR
/8530 CS
A0
A1
U27
16V8-15
SERIAL.PDS
I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
F0
F1
F2
F3
F4
F5
F6
F7
/8530 CS
/8530 RD
/8530 WR
/SERIAL_RDY
DIV_CLK
/REQ
A31
A30
/WRITE
/RESET
C19
.1 UF

APPENDIX B. PAL EQUATIONS

This appendix shows the PAL equations in PALASM® software syntax for: the MSTR_CON equations, the CAS_DEC equations, the REFRESH equations, the ADDR_CTR equations, and the SERIAL equations.

MSTR_CON PAL EQUATIONS

```
;PALASM Software Design Description

;----------------------------------- Declaration Segment ------------
TITLE    MEMORY CYCLE GENERATOR FOR 32-BIT INTERLEAVED Am29030 PROCESSOR DESIGN
PATTERN  MSTR_CON.PDS
REVISION B
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     06/22/92

CHIP  U3  PALCE22V10 DEVICE

;----------------------------------- PIN Declarations ---------------
PIN  1          MEMCLK
PIN  2          /RESET
PIN  3          /REQ
PIN  4          A31
PIN  5          A30
PIN  6          /BURST
PIN  7          A2
PIN  8          /WRITE
PIN  9          /SECOND_ACCESS
PIN  10         NC
PIN  11         /PIN145
PIN  12         GND
PIN  13         /REF_REQ
PIN  14         /MUX
PIN  15         /OE_BANK1                              REGISTERED
PIN  16         /OE_BANK0                              REGISTERED
PIN  17         /MEM_RDY                               REGISTERED
PIN  18         /INC                                   REGISTERED
PIN  19         /IDLE                                  REGISTERED
PIN  20         /ROM_CS
PIN  21         /REF_ACCESS                            REGISTERED
PIN  22         /MEM_ACCESS                            REGISTERED
PIN  23         /RAS                                   REGISTERED
PIN  24         VCC

;------------------------------------ Boolean Equation Segment ------
EQUATIONS

ROM_CS = REQ*/A31*/A30*/PIN145 + RESET

IDLE :=   /IDLE*/MEM_ACCESS*/REF_ACCESS
        + RESET
        + /IDLE*/MEM_ACCESS*/REF_ACCESS*/RAS
        + IDLE*/(REQ*A31*/A30)*/REF_REQ
        + PIN145*/REF_REQ

REF_ACCESS :=   IDLE*REF_REQ*/REF_ACCESS
              + REF_ACCESS*REF_REQ
              + REF_ACCESS*/OE_BANK0
```

```
MEM_ACCESS :=   IDLE*REQ*A31*/A30*/REF_REQ*/MEM_ACCESS*/PIN145
              + MEM_ACCESS*/MEM_RDY
              + MEM_ACCESS*BURST

MEM_RDY :=    REQ*/A31*/A30*INC*/MEM_RDY*/PIN145
            + MEM_ACCESS*(OE_BANK0 + OE_BANK1)*/MEM_RDY
            + MEM_ACCESS*MEM_RDY*BURST

RAS :=    IDLE*REQ*A31*/A30*/REF_REQ*/MEM_ACCESS*/PIN145
        + MEM_ACCESS*/MEM_RDY
        + MEM_ACCESS*BURST
        + REF_ACCESS*/RAS
        + REF_ACCESS*RAS*/OE_BANK0

MUX =     MEM_ACCESS*RAS*/MEMCLK
       +  MEM_ACCESS*RAS*MUX

INC :=   REQ*/A31*/A30*/INC*/MEM_RDY*/PIN145
       + REQ*/A31*/A30*INC*/MEM_RDY*/PIN145
       + MEM_ACCESS*/WRITE*/SECOND_ACCESS*A2*/INC*BURST
       + MEM_ACCESS*/WRITE*/SECOND_ACCESS*(OE_BANK0 + OE_BANK1)*/A2*BURST*/INC
       + MEM_ACCESS*OE_BANK1*WRITE*/SECOND_ACCESS*A2*/INC*BURST
       + MEM_ACCESS*MEM_RDY*/INC*BURST

OE_BANK0 :=    REF_ACCESS*RAS
             + MEM_ACCESS*/A2*/MEM_RDY*/OE_BANK0
             + MEM_ACCESS*OE_BANK0*/MEM_RDY
             + MEM_ACCESS*OE_BANK1*MEM_RDY*BURST

OE_BANK1 :=    MEM_ACCESS*A2*/MEM_RDY*/OE_BANK1
             + MEM_ACCESS*OE_BANK1*/MEM_RDY
             + MEM_ACCESS*OE_BANK0*MEM_RDY*BURST

;-------------------------------------------------------------------
```

CAS_DEC PAL EQUATIONS

```
;PALASM Software Design Description

;---------------------------------- Declaration Segment ------------
TITLE    RAS AND CAS GENERATOR FOR Am29030 PROCESSOR 32-BIT INTERLEAVE
PATTERN  CAS_DEC.PDS
REVISION B
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     06/22/92

CHIP  U4  PALCE22V10 DEVICE

;---------------------------------- PIN Declarations ---------------
PIN  1          MEMCLK
PIN  2          /MEM_ACCESS
PIN  3          /REF_ACCESS
PIN  4          /MEM_RDY
PIN  5          /WE0
PIN  6          /WE1
PIN  7          /WE2
PIN  8          /WE3
PIN  9          /WRITE
PIN  10        /OE_BANK0
PIN  11        /OE_BANK1
PIN  12        GND
PIN  13        NC
PIN  14        DIV_CLK
PIN  15        /BANK1_CAS3
PIN  16        /BANK1_CAS2
PIN  17        /BANK1_CAS1
PIN  18        /BANK1_CAS0
PIN  19        /BANK0_CAS3
PIN  20        /BANK0_CAS2
PIN  21        /BANK0_CAS1
PIN  22        /BANK0_CAS0
PIN  23        /SECOND_ACCESS
PIN  24        VCC

;------------------------------------ Boolean Equation Segment ------
EQUATIONS

MINIMIZE_OFF

DIV_CLK := /DIV_CLK

SECOND_ACCESS :=   MEM_RDY
                 + SECOND_ACCESS*MEM_ACCESS

BANK0_CAS0 =   REF_ACCESS
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*/MEM_RDY*/MEMCLK
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*BANK0_CAS0
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK1*MEM_RDY*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK0_CAS0*MEMCLK
             + MEM_ACCESS*MEM_RDY*OE_BANK0*WRITE*WE0*/MEMCLK

BANK0_CAS1 =   REF_ACCESS
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*/MEM_RDY*/MEMCLK
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*BANK0_CAS1
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK1*MEM_RDY*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK0_CAS1*MEMCLK
             + MEM_ACCESS*MEM_RDY*OE_BANK0*WRITE*WE1*/MEMCLK

BANK0_CAS2 =   REF_ACCESS
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*/MEM_RDY*/MEMCLK
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*BANK0_CAS2
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK1*MEM_RDY*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK0_CAS2*MEMCLK
             + MEM_ACCESS*MEM_RDY*OE_BANK0*WRITE*WE2*/MEMCLK

BANK0_CAS3 =   REF_ACCESS
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*/MEM_RDY*/MEMCLK
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK0*BANK0_CAS3
             + MEM_ACCESS*/SECOND_ACCESS*/WRITE*OE_BANK1*MEM_RDY*/MEMCLK
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE
             + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE*/MEMCLK
```

```
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK0_CAS3*MEMCLK
              + MEM_ACCESS*MEM_RDY*OE_BANK0*WRITE*WE3*/MEMCLK

BANK1_CAS0 =    REF_ACCESS
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*(OE_BANK0 + OE_BANK1)*/MEMCLK
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*BANK1_CAS0
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE*/MEMCLK
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK1_CAS0*MEMCLK
              + MEM_ACCESS*MEM_RDY*OE_BANK1*WRITE*WE0*/MEMCLK

BANK1_CAS1 =    REF_ACCESS
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*(OE_BANK0 + OE_BANK1)*/MEMCLK
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*BANK1_CAS1
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE*/MEMCLK
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK1_CAS1*MEMCLK
              + MEM_ACCESS*MEM_RDY*OE_BANK1*WRITE*WE1*/MEMCLK

BANK1_CAS2 =    REF_ACCESS
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*(OE_BANK0 + OE_BANK1)*/MEMCLK
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*BANK1_CAS2
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE*/MEMCLK
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK1_CAS2*MEMCLK
              + MEM_ACCESS*MEM_RDY*OE_BANK1*WRITE*WE2*/MEMCLK

BANK1_CAS3 =    REF_ACCESS
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*(OE_BANK0 + OE_BANK1)*/MEMCLK
              + MEM_ACCESS*/SECOND_ACCESS*/WRITE*BANK1_CAS3
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK1*/WRITE
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*OE_BANK0*/WRITE*/MEMCLK
              + MEM_ACCESS*SECOND_ACCESS*MEM_RDY*/WRITE*BANK1_CAS3*MEMCLK
              + MEM_ACCESS*MEM_RDY*OE_BANK1*WRITE*WE3*/MEMCLK

;-------------------------------------------------------------------
```

REFRESH PAL EQUATIONS

```
;PALASM Software Design Description

;---------------------------------- Declaration Segment ------------
TITLE    REFRESH TIMER FOR Am29030 PROCESSOR
PATTERN  REFRESH.PDS
REVISION A
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     03/03/92

CHIP  U2  PALCE16V8 DEVICE

;---------------------------------- PIN Declarations ---------------
PIN  1          MEMCLK
PIN  2          /REF_ACCESS
PIN  3          NC
PIN  4          NC
PIN  5          NC
PIN  6          NC
PIN  7          NC
PIN  8          NC
PIN  9          NC
PIN  10         GND
PIN  11         /OE
PIN  12         /REF_REQ                              REGISTERED
PIN  13         /Q0                                   REGISTERED
PIN  14         /Q1                                   REGISTERED
PIN  15         /Q2                                   REGISTERED
PIN  16         /Q3                                   REGISTERED
PIN  17         /Q4                                   REGISTERED
PIN  18         /Q5                                   REGISTERED
PIN  19         /Q6                                   REGISTERED
PIN  20         VCC

;------------------------------------ Boolean Equation Segment ------
EQUATIONS

Q0 := /Q0

Q1 := Q1 :+: Q0

Q2 := Q2 :+: (Q1*Q0)

Q3 := Q3 :+: (Q2*Q1*Q0)

Q4 := Q4 :+: (Q3*Q2*Q1*Q0)

Q5 := Q5 :+: (Q4*Q3*Q2*Q1*Q0)

Q6 := Q6 :+: (Q5*Q4*Q3*Q2*Q1*Q0)

REF_REQ :=   Q6*Q5*Q4*Q3*Q2*Q1*Q0
           + REF_REQ*/REF_ACCESS

;--------------------------------------------------------------------
```

ADDR_CTR PAL EQUATIONS

```
;PALASM Software Design Description

;----------------------------------- Declaration Segment ------------
TITLE    ADDRESS COUNTER & INCREMENTER FOR 32-BIT BANK INTERLEAVED Am29030 PROCES-
SOR
PATTERN  ADDR_CTR.PDS
REVISION A
AUTHOR   DAVID STOENNER
COMPANY  AMD
DATE     06/15/92

CHIP  U10  PALCE16V8 DEVICE

;----------------------------------- PIN Declarations ---------------
PIN  1          MEMCLK
PIN  2          A3
PIN  3          A4
PIN  4          A5
PIN  5          A6
PIN  6          A7
PIN  7          A8
PIN  8          A9
PIN  9          /LOAD
PIN  10         GND
PIN  11         /OE
PIN  12         /INC
PIN  13         CA3                                     REGISTERED
PIN  14         CA4                                     REGISTERED
PIN  15         CA5                                     REGISTERED
PIN  16         CA6                                     REGISTERED
PIN  17         CA7                                     REGISTERED
PIN  18         CA8                                     REGISTERED
PIN  19         CA9                                     REGISTERED
PIN  20         VCC

;------------------------------------ Boolean Equation Segment ------
EQUATIONS

CA3 :=   INC*/CA3
      + LOAD*A3

CA4 :=   INC*(CA4 :+: CA3)
      + LOAD*A4

CA5 :=   INC*(CA5 :+: (CA4*CA3))
      + LOAD*A5

CA6 :=   INC*(CA6 :+: (CA5*CA4*CA3))
      + LOAD*A6

CA7 :=   INC*(CA7 :+: (CA6*CA5*CA4*CA3))
      + LOAD*A7

CA8 :=   INC*(CA8 :+: (CA7*CA6*CA5*CA4*CA3))
      + LOAD*A8

CA9 :=   INC*(CA9 :+: (CA8*CA7*CA6*CA5*CA4*CA3))
      + LOAD*A9

;--------------------------------------------------------------------
```

SERIAL PAL EQUATIONS

```
;PALASM Software Design Description

;----------------------------------- Declaration Segment ------------

TITLE 8530 CONTROLLER
PATTERN SERIAL.PDS
REVISION A
AUTHOR DAVID STOENNER
COMPANY AMD
DATE 03/03/1992

CHIP UXX PAL16V8 DEVICE

;----------------------------------- PIN Declarations ---------------

PIN 1           CLK
PIN 2           /REQ
PIN 3           A31
PIN 4           A30
PIN 5           /SIPW
PIN 6           NC
PIN 7           /RESET_IN
PIN 8           NC
PIN 9           NC
PIN 10          GND
PIN 11          /OE
PIN 12          /8530_CS
PIN 13          /8530_RD
PIN 14          /8530_WR
PIN 15          /8530_RDY
PIN 16          /NC1
PIN 17          /NC2
PIN 18          NC
PIN 19          /COM1INT
PIN 20          VCC

;----------------------------------- Boolean Equation Segment ------

EQUATIONS

8530_CS = REQ*A31*A30

8530_RD :=   8530_CS*/SIPW*/8530_RD*/8530_RDY
           + 8530_RD*/8530_RDY
           + RESET_IN

8530_WR :=   8530_CS*SIPW*/8530_WR*/8530_RDY
           + 8530_WR*/NC1*/8530_RDY
           + 8530_WR*/NC2*/8530_RDY
           + RESET_IN

NC1 := (8530_RD + 8530_WR)*/NC1*/8530_RDY

NC2 := (8530_RD + 8530_WR)*(NC1:+:NC2)*/8530_RDY

8530_RDY := NC1*NC2*/8530_RDY

;---------------------------------------------------------------------------
```