Interfacing DRAM to the TMS320C80

Dave Comisky
Systems Engineer — 'C8x Applications

SPRA056
July 1996

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

Appendixes

List of Illustrations

List of Tables

Introduction

As the need for larger and larger memory arrays has risen, dynamic random access memory (DRAM) has emerged as a perennial favorite among system designers, based largely on the ratio of its size (megabytes) to board space. The TMS320C80 has integrated DRAM support directly into its memory interface, allowing for simple connection to one or more memory banks, each of which can be of different size and speed. The ability to access multiple memory banks and an on-chip refresh controller reduces system cost.

This application report discusses an interface between the ’C80 and the TMS417400 DRAM(s). The TMS417400s are interfaced in the form of two TM497BBK32 single inline memory modules (SIMMs). Each SIMM is organized as a 4MB x 32-bit, or 16MB, DRAM memory module. The interface described in this report also allows connection to 4MB and 8MB SIMMs. A timing analysis for the proposed design is presented on page 14, and a complete set of schematics is provided in Appendix B.

DRAM Functional Description

DRAM is organized as an array of memory cells and comes in many sizes, which are specified in two dimensions. The TMS417400s discussed in this report are 4MB x 4-bit memories. The first dimension (4MB) identifies the number of memory cells per bit plane, and the second dimension (4-bit) identifies the number of bit planes per device. Pages of memory are constructed by grouping like numbered rows across multiple columns of the bit planes. This is illustrated in the following figure, which shows a graphical representation of a single DRAM:

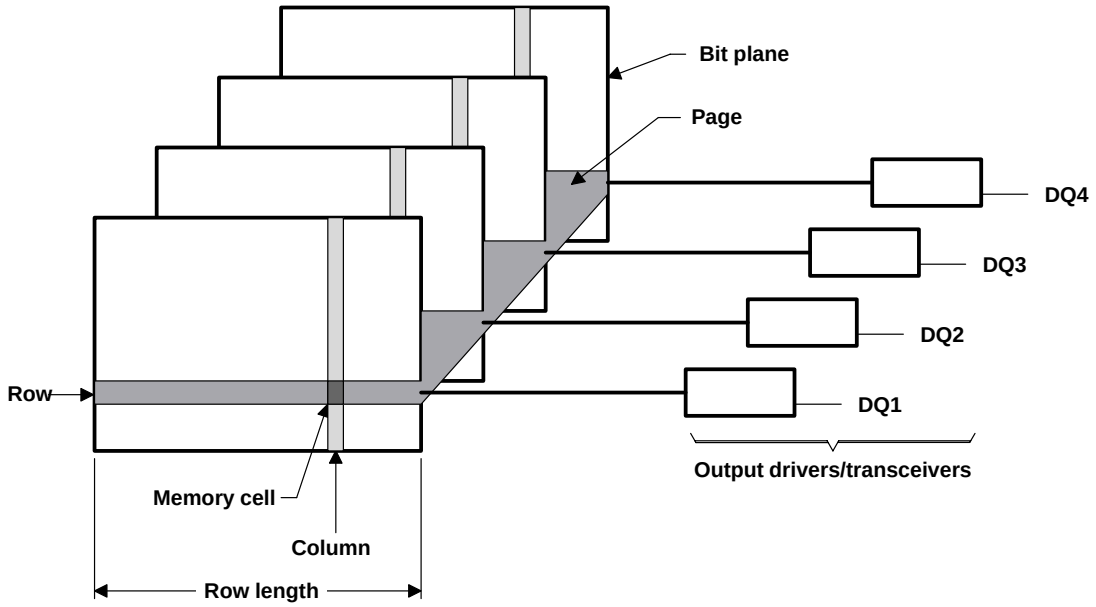

Figure 1. DRAM Architecture

In order to access the memory cells of DRAM, the desired location in the array must be identified. This is accomplished by decoding the row and column address applied to the device. Row and column addresses to DRAM are latched with the row address strobe ($\overline{\text{RAS}}$) and column address strobe ($\overline{\text{CAS}}$) inputs. A memory access to DRAM starts with activating a row by applying the desired row address and pulling $\overline{\text{RAS}}$ low. All subsequent column accesses to the DRAM are made in the specified row. When $\overline{\text{RAS}}$ is deasserted (high), the row is deactivated. This is called *precharging*.

Because of the way that DRAM is organized, and also as a pin saving measure, the inputs for the row and column addresses are the same. This means that external hardware must be capable of multiplexing the row and column address bits to the same physical location on the address bus. With many processors, this requires a complex array of buffers and latches. The ’C80 has DRAM support (up to 256MB x *n*) built into the on-chip transfer controller (TC). Address multiplexing, as well as the generation of $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and write enable ($\overline{\text{W}}$), is provided directly by the ’C80, greatly simplifying the DRAM interface.

Page Mode

Page-mode operation increases bandwidth by allowing faster memory accesses. This is accomplished by keeping row addresses latched in by $\overline{\text{RAS}}$ and strobing random column addresses into the DRAM, eliminating the time required to set up and strobe the row addresses for each access. The maximum number of columns that can be accessed is determined by the row length and the maximum allowable $\overline{\text{RAS}}$ low time.

Enhanced page mode is a feature that further increases bandwidth by allowing an access to begin as soon as the column address is valid, rather than waiting for $\overline{\text{CAS}}$ to drop as with conventional paged memories. The $\overline{\text{CAS}}$ lines for enhanced page-mode devices serve as strobes to transparent latches, as well as output enables for the data drivers on the DRAM.

Refresh

Because DRAM is a dynamic memory, it must be refreshed to retain data. DRAM refresh cycles can take several forms:

- A normal read or write cycle refreshes all the bits in the selected row.
- A $\overline{\text{RAS}}$-only refresh is realized by cycling $\overline{\text{RAS}}$ with $\overline{\text{CAS}}$ held high. This is a power conserving mode since the outputs are not enabled. In this mode, external hardware must provide the refresh address.
- A hidden refresh can be performed following a read. A hidden refresh is similar to a $\overline{\text{RAS}}$-only refresh, but $\overline{\text{CAS}}$ is low and the address is generated internally.
- $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ (CBR) refresh cycles, wherein the $\overline{\text{CAS}}$ input is forced low prior to $\overline{\text{RAS}}$. The DRAM ignores the external address, and the refresh address is generated internally.

The ’C80 institutes $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh cycles only. The refresh rate is user-programmable, and decoding hardware can be added to support multiple banks of volatile memory (for example, one DRAM, two VRAM, and one SDRAM bank). During refresh, the ’C80 outputs a pseudo-address on A[31:16], which can be decoded to designate multiple banks of dynamic memory. The pseudo-address is decremented for every refresh that is performed. In this application report, the “refresh map” is defined with bits A16 and A17 (latched) of the ’C80 designating four areas of volatile memory. Since the ’C80 begins every row access, including refresh, by examining the cycle configuration inputs, banks of different speed are supported directly.

’C80 Address Subcycles

The ’C80 read and write cycles are divided into two sections – row time and column time. States within each of these sections are numbered “r*i*” and “c*i*” respectively, where *i* is an integer. The length of each section is a function of the memory speed (as determined by the CT[2:0] inputs) and the length of the access (column time only). During row time, the ’C80 outputs the entire 32-bit address on the address bus as well as status information on STATUS[5:0]. External logic is required to decode the address and status

information and return cycle configuration information to the ’C80. The row time status codes are shown in Table 1.

Table 1. Row Time Status Codes

STATUS[5:0]	CYCLE TYPE	STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	Normal Read	1 0 0 0 0 0	Reserved
0 0 0 0 0 1	Normal Write	1 0 0 0 0 1	Reserved
0 0 0 0 1 0	Refresh	1 0 0 0 1 0	Reserved
0 0 0 0 1 1	SDRAM DCAB	1 0 0 0 1 1	Reserved
0 0 0 1 0 0	Peripheral Device PT Read	1 0 0 1 0 0	XPT1 Read
0 0 0 1 0 1	Peripheral Device PT Write	1 0 0 1 0 1	XPT1 Write
0 0 0 1 1 0	Reserved	1 0 0 1 1 0	XPT1 PDPT Read
0 0 0 1 1 1	Reserved	1 0 0 1 1 1	XPT1 PDPT Write
0 0 1 0 0 0	Reserved	1 0 1 0 0 0	XPT2 Read
0 0 1 0 0 1	Block Write PT	1 0 1 0 0 1	XPT2 Write
0 0 1 0 1 0	Reserved	1 0 1 0 1 0	XPT2 PDPT Read
0 0 1 0 1 1	Reserved	1 0 1 0 1 1	XPT2 PDPT Write
0 0 1 1 0 0	SDRAM MRS	1 0 1 1 0 0	XPT3 Read
0 0 1 1 0 1	Load Color Register	1 0 1 1 0 1	XPT3 Write
0 0 1 1 1 0	Reserved	1 0 1 1 1 0	XPT3 PDPT Read
0 0 1 1 1 1	Reserved	1 0 1 1 1 1	XPT3 PDPT Write
0 1 0 0 0 0	Frame 0 Read Transfer	1 1 0 0 0 0	XPT4/SAM1 Read
0 1 0 0 0 1	Frame 0 Write Transfer	1 1 0 0 0 1	XPT4/SAM1 Write
0 1 0 0 1 0	Frame 0 Split Read Transfer	1 1 0 0 1 0	XPT4/SAM1 PDPT Read
0 1 0 0 1 1	Frame 0 Split Write Transfer	1 1 0 0 1 1	XPT4/SAM1 PDPT Write
0 1 0 1 0 0	Frame 1 Read Transfer	1 1 0 1 0 0	XPT5/SOF1 Read
0 1 0 1 0 1	Frame 1 Write Transfer	1 1 0 1 0 1	XPT5/SOF1 Write
0 1 0 1 1 0	Frame 1 Split Read Transfer	1 1 0 1 1 0	XPT5/SOF1 PDPT Read
0 1 0 1 1 1	Frame 1 Split Write Transfer	1 1 0 1 1 1	XPT5/SOF1 PDPT Write
0 1 1 0 0 0	Reserved	1 1 1 0 0 0	XPT6/SAM0 Read
0 1 1 0 0 1	Reserved	1 1 1 0 0 1	XPT6/SAM0 Write
0 1 1 0 1 0	Reserved	1 1 1 0 1 0	XPT6/SAM0 PDPT Read
0 1 1 0 1 1	Reserved	1 1 1 0 1 1	XPT6/SAM0 PDPT Write
0 1 1 1 0 0	PT Read Transfer	1 1 1 1 0 0	XPT7/SOF0 Read
0 1 1 1 0 1	PT Write Transfer	1 1 1 1 0 1	XPT7/SOF0 Write
0 1 1 1 1 0	Reserved	1 1 1 1 1 0	XPT7/SOF0 PDPT Read
0 1 1 1 1 1	Idle	1 1 1 1 1 1	XPT7/SOF0 PDPT Write

Cycle configuration information is input on the AS[2:0], BS[1:0], CT[2:0], and PS[3:0]. These inputs determine the following for the requested section (page) of memory:

- Address shift amount (AS)
- Bus size (BS)
- Column timing (CT)
- Page size (PS)

After the cycle configuration information is input, the 'C80 proceeds with the access at the requested cycle timing and bus size for the remainder of the page.

Address Shifting

The 'C80 on-chip TC supports page memories up to 256MB × *n* size paged memories. This is accomplished by shifting the position of logical address bits on the physical address bus during column time to line up with the memory bank. In this manner, both the row and column address are multiplexed on the same physical address lines. The amount of shifting applied to the logical address bits is controlled by the AS[2:0] cycle configuration inputs, as shown in Table 2.

Table 2. Column Time Address Multiplexing

AS[2:0] at Row Time	Column Time Address Bits																														RAM Type
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0 0 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Static
0 0 1	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	2	1	0	64 x n
0 1 0	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	X	2	1	0	256K x n
0 1 1	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	X	X	2	1	0	1M x n
1 0 0	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	X	X	X	2	1	0	4M x n
1 0 1	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	X	X	X	X	2	1	0	16M xn
1 1 0	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	X	X	X	X	X	2	1	0	64M x n
1 1 1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X	X	X	X	X	X	X	X	X	2	1	0	256M xn

Page Size

The TC performs page-mode cycles whenever possible, and therefore must track the boundary of a given row. If a row boundary is crossed during an access, the TC deactivates (precharges) the row by pulling $\overline{\text{RAS}}$ high, then starts a new cycle beginning with the r1 state. This boundary is tracked in the LASTPAGE register of the TC, which is updated at the beginning of a memory cycle based on the address and PS[3:0] inputs. The LASTPAGE register is 26 bits in length, the upper 12 of which are always compared to the current address on an access-by-access basis. The lower 14 bits of LASTPAGE, corresponding to bits 19–6 of the logical address, may or may not be compared, depending on the PS[3:0] input values. Table 3 illustrates the address bit comparison based on the PS[3:0] inputs.

Table 3. Page Size Codes

PS[3:0]				ADDRESS BITS COMPARED	PAGE SIZE (BYTES)
0	0	0	0	A(31:6)	64
0	0	0	1	A(31:7)	128
0	0	1	0	A(31:8)	256
0	0	1	1	A(31:9)	512
0	1	0	0	A(31:10)	1K
0	1	0	1	A(31:18)	256K
0	1	1	0	A(31:19)	512K
0	1	1	1	A(31:20)	1M
1	0	0	0	A(31:0)	1–8†
1	0	0	1	A(31:11)	2K
1	0	1	0	A(31:12)	4K
1	0	1	1	A(31:13)	8K
1	1	0	0	A(31:14)	16K
1	1	0	1	A(31:15)	32K
1	1	1	0	A(31:16)	64K
1	1	1	1	A(31:17)	128K

† When PS[3:0] = 1000, page mode cycles are disabled. Page size is therefore equal to bus size.

Cycle Timing

The minimum length of row time is determined by the CT[2:0] inputs. Neglecting possible user-inserted wait states (using the READY input), row time is either 4, 5, or 6 CLKOUT cycles long for read accesses of 1, 2, and 3 cycles per column. Write cycles always have at least six row time states, which allows time for the ’C80 to turn on its data drivers. All subsequent column accesses in the given row are performed at the indicated cycles/column timing; that is, column time accesses are 1, 2, or 3 CLKOUT cycles long.

Beyond the aforementioned two-dimensional naming convention of DRAMs, DRAM is also identified with a speed code. Typical speed codes are –60, –70, and –80, which indicate a latency from $\overline{\text{RAS}}$ falling to valid data out of 60, 70, and 80 ns (reads). The speed code is also indicative of other timing characteristics of the DRAM, including cycle time. The ’C80’s CT inputs allow the user to specify 1, 2, or 3 cycle/column accesses, where cycle refers to an integral number of CLKOUT periods. The CT codes are shown in Table 4.

Table 4. Column Timing Codes

CT[2:0]			CYCLE TIMING
0	0	0	Pipelined (Burst Length 1) SDRAM CAS Latency of 2
0	0	1	Pipelined (Burst Length 1) SDRAM CAS Latency of 3
0	1	0	Interleaved (Burst Length 2) SDRAM CAS Latency of 2
0	1	1	Interleaved (Burst Length 2) SDRAM CAS Latency of 3
1	0	0	Pipelined 1 Cycle/Column
1	0	1	Nonpipelined 1 Cycle/Column
1	1	0	2 Cycle/Column
1	1	1	3 Cycle/Column

Bus Size

The maximum theoretical bandwidth of a system is a function of clock frequency, the cycles/column selected, and the last cycle configuration input parameter, bus size. Bus size is indicated on the BS[1:0] inputs to the ’C80. Sizes of 1, 2, 4, or 8 bytes can be specified for a given row. The TC activates the appropriate $\overline{\text{CAS}}$ lines and drives or reads the appropriate data lines according to the data size, address, and current endian mode. Bus size codes are shown in Table 5.

Table 5. Bus Size Codes

BS[1:0]		Bus Size (bits)
0	0	8
0	1	16
1	0	32
1	1	64

Column Time

When a row is activated ($\overline{\text{RAS}}$ falls), column accesses can begin. For the ’C80, this occurs after 1–3 row time states, depending on the cycles/column setting and direction of the access. During column time, the logical address is shifted to the appropriate bits on the physical address bus as specified by the AS inputs sampled at row time. The meaning of the STATUS bus is altered during column time, as shown in Table 6. Although not required by DRAM, the status codes can be decoded by external logic for application-specific functions. The column-time status code 011111 indicates an idle cycle, and is useful for detecting bubbles in the column-time pipeline.

Table 6. Column Time Status Codes

STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	PP0 Low Priority Packet Transfer
0 0 0 0 0 1	PP0 High Priority Packet Transfer
0 0 0 0 1 0	PP0 Instruction Cache
0 0 0 0 1 1	PP0 DEA
0 0 0 1 0 0	PP1 Low Priority Packet Transfer
0 0 0 1 0 1	PP1 High Priority Packet Transfer
0 0 0 1 1 0	PP1 Instruction Cache
0 0 0 1 1 1	PP1 DEA
0 0 1 0 0 0	PP2 Low Priority Packet Transfer
0 0 1 0 0 1	PP2 High Priority Packet Transfer
0 0 1 0 1 0	PP2 Instruction Cache
0 0 1 0 1 1	PP2 DEA
0 0 1 1 0 0	PP3 Low Priority Packet Transfer
0 0 1 1 0 1	PP3 High Priority Packet Transfer
0 0 1 1 1 0	PP3 Instruction Cache
0 0 1 1 1 1	PP3 DEA
0 1 0 0 0 0	MP Low Priority Packet Transfer
0 1 0 0 0 1	MP High Priority Packet Transfer
0 1 0 0 1 0	MP Urgent Packet Transfer (Low)
0 1 0 0 1 1	MP Urgent Packet Transfer (High)
0 1 0 1 0 0	XPT/VCPT in Progress
0 1 0 1 0 1	XPT/VCPT Complete
0 1 0 1 1 0	MP Instruction Cache (Low)
0 1 0 1 1 1	MP Instruction Cache (High)
0 1 1 0 0 0	MP DEA (Low)
0 1 1 0 0 1	MP DEA (High)
0 1 1 0 1 0	MP Data Cache (Low)
0 1 1 0 1 1	MP Data Cache (High)
0 1 1 1 0 0	Frame 0
0 1 1 1 0 1	Frame 1
0 1 1 1 1 0	Refresh
0 1 1 1 1 1	Idle

STATUS[5:0]	CYCLE TYPE
1 0 0 0 0 0	Reserved
1 0 0 0 0 1	Reserved
1 0 0 0 1 0	Reserved
1 0 0 0 1 1	Reserved
1 0 0 1 0 0	Reserved
1 0 0 1 0 1	Reserved
1 0 0 1 1 0	Reserved
1 0 0 1 1 1	Reserved
1 0 1 0 0 0	Reserved
1 0 1 0 0 1	Reserved
1 0 1 0 1 0	Reserved
1 0 1 0 1 1	Reserved
1 0 1 1 0 0	Reserved
1 0 1 1 0 1	Reserved
1 0 1 1 1 0	Reserved
1 0 1 1 1 1	Reserved
1 1 0 0 0 0	Reserved
1 1 0 0 0 1	Reserved
1 1 0 0 1 0	Reserved
1 1 0 0 1 1	Reserved
1 1 0 1 0 0	Reserved
1 1 0 1 0 1	Reserved
1 1 0 1 1 0	Reserved
1 1 0 1 1 1	Reserved
1 1 1 0 0 0	Reserved
1 1 1 0 0 1	Reserved
1 1 1 0 1 0	Reserved
1 1 1 0 1 1	Reserved
1 1 1 1 0 0	Reserved
1 1 1 1 0 1	Reserved
1 1 1 1 1 0	Reserved
1 1 1 1 1 1	Write Drain / SDRAM DCAB

Low = MP operating in low (normal) priority mode
High = MP operating in high priority mode

DRAM System Overview

This report discusses a system in which the DRAM bank is 32MB in size and consists of two 16MB SIMMs. The glue logic associated with this interface is implemented in PAL/GALs, but the equations presented here are valid for ASIC and FPGA designs as well. Figure 2 shows a block digram of the DRAM system. See page 14 for a timing analysis of this design, and refer to Appendix B for a complete set of schematics.

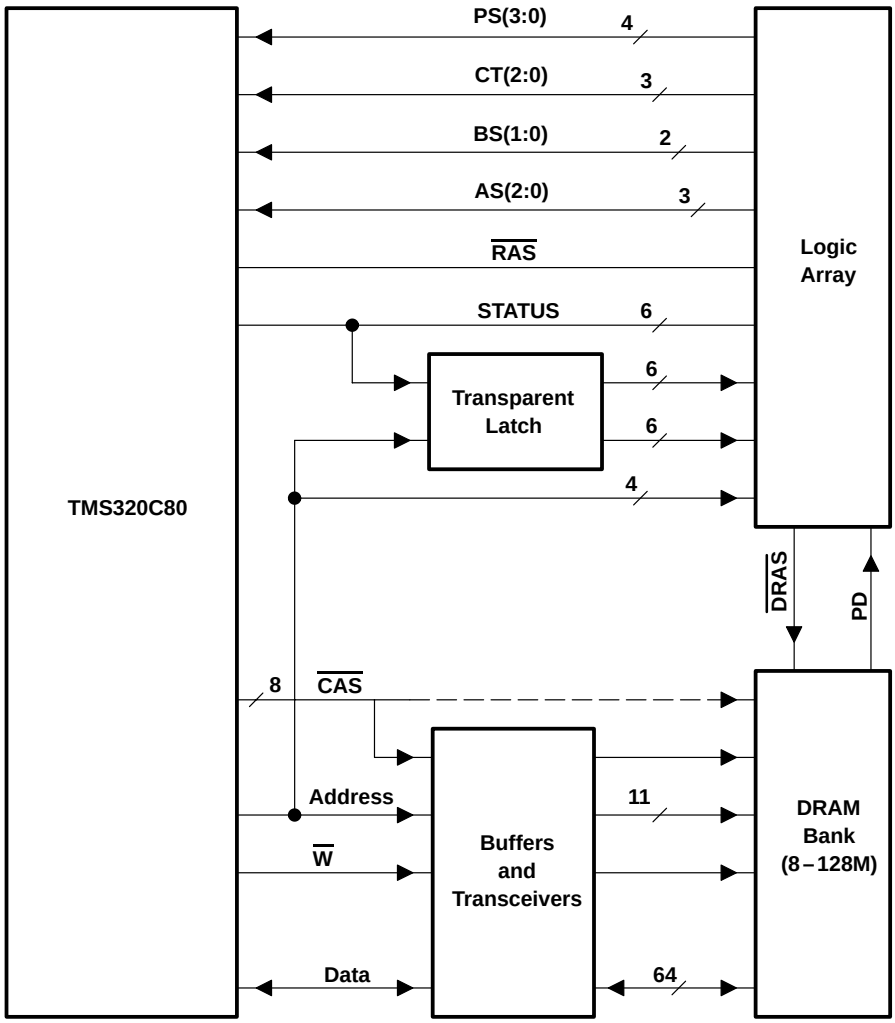

Figure 2. DRAM System Block Diagram

This overview provides detailed information about the following portions of the proposed interface:

- Presence detects
- Bank selection
- $\overline{\text{CAS}}$ strobe
- $\overline{\text{W}}$ strobe
- Data bus
- Address bus

Presence Detects

The TM497BBK32 memory modules, like many SIMMs, have four special outputs which identify the size and speed characteristics of the module. These outputs are known as the *presence detects* (PD1–4), which can be used for any number of application-specific purposes. In this design, considerations for both a 40 and 50 MHz are made. The PD bits are used to adjust the cycles/column setting for accesses, thus allowing the system to adjust to the DRAM speed provided. Similarly, the PD bits can be used to adjust the address inputs for connecting to 16MB and 32MB SIMMs (see *Changing the Bank Size* on page 20). The presence detect bits are internally connected on the SIMM, as shown in Table 7.

Table 7. Presence Detect Codes

PRESENCE DETECT					
MEMORY MODULE	**SPEED**	**PD1**	**PD2**	**PD3**	**PD4**
TM497BBK32 (16MB)	80 ns	V_{SS}	NC	NC	V_{SS}
	70 ns	V_{SS}	NC	V_{SS}	NC
	60 ns	V_{SS}	NC	NC	NC
TM893CBK32 (32MB)	80 ns	NC	V_{SS}	NC	V_{SS}
	70 ns	NC	V_{SS}	V_{SS}	NC
	60 ns	NC	V_{SS}	NC	NC

Bank Selection

DRAM accesses begin with activating a row by addressing the desired row and pulling $\overline{\text{RAS}}$ low. Because DRAMs do not have chip select inputs, selection between one device and another is made by applying or inhibiting the $\overline{\text{RAS}}$ signal to one or more DRAM banks. The $\overline{\text{RAS}}$ signals for a multi-chip DRAM bank must be generated accordingly. Since the ’C80 has no way of knowing how many chips are in a particular bank, this $\overline{\text{RAS}}$ decoding must be implemented in external logic.

The two TM497BBK32 modules used in this design contain sixteen TMS417400 DRAM memories (eight per module). The TMS417400s (4MB × 4-bit) are organized in pairs with one $\overline{\text{CAS}}$ line per pair, thus four $\overline{\text{CAS}}$ lines are present at each SIMM interface. Each $\overline{\text{CAS}}$ accesses one byte. The TM497BBK32 provides two $\overline{\text{RAS}}$ inputs, each of which controls four TMS417400s. In this design, four $\overline{\text{RAS}}$ inputs (two per SIMM) are connected to the SIMM socket to allow for expansion to 8MB and 32MB SIMMs, which are populated with DRAM on both sides. Table 8 shows the connections for the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals on the memory modules.

Table 8. SIMM Connection Table

DATA BLOCK	$\overline{RASx}$		$\overline{CASx}$
	SIDE 1	SIDE 2†	
DQ0–DQ7	$\overline{RAS0}$	$\overline{RAS1}$	$\overline{CAS0}$
DQ8–DQ15	$\overline{RAS0}$	$\overline{RAS1}$	$\overline{CAS1}$
DQ16–DQ23	$\overline{RAS2}$	$\overline{RAS3}$	$\overline{CAS2}$
DQ24–DQ31	$\overline{RAS2}$	$\overline{RAS3}$	$\overline{CAS3}$

† Side 2 applies only to 8MB and 32 MB modules.

Generation of the $\overline{RAS}$ signals for the DRAM bank is straightforward; however, take care to ensure that multiple SIMM sizes are supported. There are two $\overline{RAS}$ signals that must be generated. These are designated as $\overline{DRAS0}$ and $\overline{DRAS1}$ (active low signals). $\overline{DRAS0}$ connects to $\overline{RAS0}$ and $\overline{RAS2}$ of the SIMMs; $\overline{DRAS1}$ controls $\overline{RAS1}$ and $\overline{RAS3}$. Thus, when using 4MB or 16MB SIMMs, only $\overline{DRAS0}$ needs to be generated. For 8MB and 32MB SIMMs, both $\overline{DRAS}$ signals need to be generated.

The signals are generated by conditionalizing the ’C80’s $\overline{RAS}$ signal on a valid address and status decode, masking out SDRAM MRS and SDRAM DCABs if SDRAM also is present.

When 8MB and 32MB SIMMs are used, the bank decode must be augmented by an additional latched address bit to distinguish the two sides of the modules. This latched bit serves as a bank select. For 32MB SIMMs, the MSB of address is C80A24; therefore, C80A25 is the bank select (see *Address Bus* on page 11). Similarly, for 8MB SIMMs, C80A23 serves as the bank select. Refreshes are separated into two areas (accounts for memory expansion), 0 and 2 of the refresh map, which is decoded with C80A16 and C80A17 of the refresh pseudo-address. An expression for the $\overline{DRAS}$ signals might look like the following:

```
!_DRAS0    =   (!RAS & DRAM_AV & !SDRAM_cyc & !refresh &
               (!PD1a #
               PD2a & PD1a & !LC80A23 #
               !PD2a & PD1a & !LC80A25 ))
               #(!RAS & refresh & !LC80A17 & !LC80A16);
!_DRAS1    =   (!RAS & DRAM_AV & !SDRAM_cyc & !refresh &
               (PD2a & PD1a & LC80A23 #
               !PD2a & PD1a & LC80A25 ))
               #(!RAS & refresh & LC80A17 & !LC80A16);
where
S5         = STATUS5; externally latched by RL
S4         = STATUS4; externally latched by RL
S3         = STATUS3; externally latched by RL
S2         = STATUS2; externally latched by RL
S1         = STATUS1; externally latched by RL
S0         = STATUS0; externally latched by RL
DRAM_AV    = LC80A31 & LC80A30 & !LC80A29 & LC80A28;
SDRAM_cyc = !S5 & !S4 &!S3 &!S2 & S1 & S0
            #!S5 & !S4 & S3 & S2 & !S1 & !S0;
refresh    = !S5 & !S4 &!S3 &!S2 & S1 & !S0;
```

The previous example assumes that both SIMMs are identical and that DRAM occupies sections 0 and 2 of the refresh map.

$\overline{\text{CAS}}$ Lines

The ’C80 provides eight $\overline{\text{CAS}}$ lines, which serve as byte strobes for the 64-bit data bus. $\overline{\text{CAS0}}$ always controls bits D0–7, and $\overline{\text{CAS7}}$ always controls bits D56–63, regardless of the current endian mode. The TM497BBK32s tie the TMS417400s’ output enables ($\overline{\text{OE}}$) to ground directly on the SIMM, so the $\overline{\text{CAS}}$ lines serve directly as the module output enables.

In large systems, the $\overline{\text{CAS}}$ lines are typically one of the most heavily loaded, usually second only to some of the address lines. Because of this, it is normally desirable to buffer the $\overline{\text{CAS}}$ lines to the DRAM bank. The TM497BBK32 specifies an input capacitance of just 14 pF (28 pF total) on the $\overline{\text{CAS}}$ inputs, which can be interfaced directly to the ’C80 with little adverse effect upon the rest of the system. In this particular design, the decision is a question of operating speed. For a 40 MHz system, a 2 cycle/column access can be obtained using a –60 DRAM module. At 50 MHz the column cycle time is violated, so a 3 cycle/column access must be specified.

The critical timing parameter in this decision is t_{CP}, the $\overline{\text{CAS}}$ high time between page-mode accesses. During a 2 cycle/column access, the ’C80 inserts an analog delay into the $\overline{\text{CAS}}$ generating logic, which assures a high time of t_H -2 ns (10.5 ns @ 40 MHz). Most DRAMs, including the TMS417400s used here, specify a t_{CP} minimum of 10 ns. For this reason, the inclusion of buffers on the $\overline{\text{CAS}}$ lines can cause a timing violation under worst case conditions, as shown below:

With $\overline{\text{CAS}}$ buffers–(40 MHz)

$$\begin{aligned} t_{CP} &= t_{W(CASH)PG} - t_{PLH244MAX}{}^{*} + t_{PHL244MIN}{}^{*} \\ &= (t_H - 2) - 4.1 + 1 \\ &= 10.5 - 4.1 + 1 \\ &= 7.4 \text{ ns} \end{aligned}$$

*using an LTV16244 buffer on $\overline{\text{CAS}}$ lines

With the 3 cycle/column accesses that are required for 50 MHz systems (or systems using slower DRAMs), there is no constraint, since the $\overline{\text{CAS}}$ high time is extended to a full $2t_H$. In the design presented here, the $\overline{\text{CAS}}$ buffers are included because a 50 MHz interface is discussed.

$\overline{\text{W}}$ Strobe

Since the $\overline{\text{W}}$ strobe has to interface with all eight DRAMs on each SIMM, it is heavily loaded at 56 pF. The ’C80 drives this signal during the r6 state, so the loading is not a speed issue. The $\overline{\text{W}}$ signal, however, is generally used by all memories (or logic which interfaces to memory) so it is a good idea to buffer this signal. The address lines also need to be buffered, and there is plenty of room left over on a 16-bit buffer to accomplish this.

Data Bus

The data bus of the ’C80 must not be driven above V_{DD3} + 0.3 (excepting the $\overline{\text{HREQ}}$ condition). As with 5 V peripherals, the TMS417400s’ outputs must be buffered to interface with the ’C80. This is easily accomplished using LVT16245 data transceivers. The enable and direction inputs of these transceivers is provided directly by the ’C80 through the $\overline{\text{DBEN}}$ and $\overline{\text{DDIN}}$ outputs respectively.

Address Bus

The address bus is the last portion of the interface to the DRAM bank. The TM497BBK32s are arranged as 4MB × 32-bit, or 16MB, memory modules. The individual TMS417400s on each SIMM are set up as 4MB × 4-bit memories, thus requiring eleven address bits to decode a memory cell location ($2^{11} \times 2^{11}$ = 4MB). Since the $\overline{\text{CAS}}$ strobes (8 total) serve to select bytes, the lower three logical address bits of the ’C80

are disregarded ($2^3 = 8$). Thus, bit 3 of the logical address bus is the LSB for each SIMM. Table 2 shows that for a 4MB × *n* device, bit 3 lines up with ’C80 address bit A14. Therefore, address bits A14–A24 of the ’C80 should interface to the DRAM bank when using 16 MB (and 32MB) SIMMs. (See *Changing the Bank Size* for a discussion of support for 4MB and 8MB SIMMs.)

The ’C80 address bus is output only, so it is not necessary to buffer the address bus to the DRAM bank for electrical reasons because the V_{IH} specification of the DRAM is met. However, since the address bits are the most heavily load lines in a typical design, they are buffered. Other 5-V peripherals can be connected to the buffered address bus provided that the V_{IH} specification of those devices is met.

Cycle Configuration Inputs

Like any other peripheral or memory bank that the ’C80 is to interface with, external logic must be used to generate the cycle configuration inputs that tell the ’C80 how to proceed with an access. The logic must be capable of decoding this infomation for the other sections of memory and peripherals as well. This information can be obtained from the address and status codes output at row time at the beginning of the r1 state. It should be noted, however, that the information must be taken from the *unlatched* address and STATUS lines; otherwise, setup time may not be met.

As far as the DRAM bank is concerned, the cycle configuration inputs must be valid for DRAM bank reads, DRAM bank writes, DRAM bank refreshes, plus any application specific cycles, such as XPTs or frame memory cycles. Expressions for the cycle configuration inputs might look like those on the following page.

```
Equations
AS2 =(DRAM_AV) & !(SDRAM_cyc # refresh)
    # (SYSTEM_SPECIFIC_REQUIRING_AS2=1);

AS1 =(SYSTEM_SPECIFIC_REQUIRING_AS1=1);
AS0 =( SYSTEM_SPECIFIC_REQUIRING_AS0=1);

BS1 = (DRAM_AV) & !(SDRAM_cyc)
    #(SYSTEM_SPECIFIC_REQUIRING_BS1=1);

BS0 = (DRAM_AV) & !(SDRAM_cyc)
    & (TWO_SIMMS)
    #(SYSTEM_SPECIFIC_REQUIRING_BS0=1);

CT2 = (DRAM_AV) & !(SDRAM_cyc # refresh)
    # DRAM_refresh
    #(SYSTEM_SPECIFIC_REQUIRING_CT2=1);

CT1 =(DRAM_AV) & !(SDRAM_cyc # refresh)
    # DRAM_refresh
    #(SYSTEM_SPECIFIC_REQUIRING_CT1=1);

CT0 = (DRAM_AV) & !(SDRAM_cyc # refresh)
    & (SEVENTY # EIGHTY)
    #((DRAM_refresh)
    & (SEVENTY # EIGHTY)
    #(SYSTEM_SPECIFIC_REQUIRING_CT0=1);

PS3 = (DRAM_AV) & !(SDRAM_cyc # refresh)
    #(SYSTEM_SPECIFIC_REQUIRING_PS3=1);

PS2 = (DRAM_AV) & !(SDRAM_cyc # refresh)
    #(SYSTEM_SPECIFIC_REQUIRING_PS2=1);

PS1 = (SYSTEM_SPECIFIC_REQUIRING_PS1=1);
```

CYCLE CODE	VALUE
AS[2:0]	100 (4MB × *n*)
BS[1:0]	11 (64-bit bus)
CT[2:0]	111 or 110, depending on DRAM speed
PS[3:0]	1100 (16K)

```
Constants and alias names
S5  = STATUS5;
S4  = STATUS4;
S3  = STATUS3;
S2  = STATUS2;
S1  = STATUS1;
S0  = STATUS0;
DRAM_AV     =   C80A31 & C80A30 & !C80A29 &
                C80A28;
SDRAM_cyc   =   !S5 & !S4 &!S3 &!S2 & S1 &
                S0 #!S5 & !S4 & S3 & S2 &
                !S1 & !S0;
refresh     =   !S5 & !S4 &!S3 &!S2 & S1 &
                !S0;
DRAM_refresh    =   !S5 & !S4 &!S3 &!S2 & S1
                    & !S0 &((C80A17 & !C80A16)
                    # (!C80A17 & !C80A16));
SEVENTY     =   !PD3;
EIGHTY      =   !PD4;
```

NOTES:
1. SYSTEM_SPECIFIC_REQUIRING_xxx conditions should not set the cycle configuration inputs for the DRAM cycles. If two SIMMs of different speeds are mixed, the system operates at the slower DRAM speed. SIMMs of different sizes should not be mixed.
2. Assumes 64-bit bus. Page size assumes 16/32MB SIMMs installed. (Setting to 8K handles 4MB, 8MB, 16MB, and 32MB SIMMs.)
3. The equations have not been simplified.

Timing Analysis

The DRAM interface has a considerable number of timing parameters that must be met. Among these are the following:

- Access times from both $\overline{RAS}$ and $\overline{CAS}$
- Address setup and hold times
- Data setup and hold times
- Propagation delays from input to input for read, write, and refresh cycles

Figures 3 through 5 show the parameters of interest in timing diagrams for page-mode reads, page-mode writes, and refresh cycles. Tables 10 through 15 contain equations and values for each of the calculated parameters.

Delays are considered for the data transceivers, the $\overline{W}$ strobe, and the address buffers (and the $\overline{CAS}$ buffer for the 50 MHz design). Also, since $\overline{RAS}$ signals for the DRAM bank are generated in external logic, a maximum logic delay is considered. The parameters shown in Table 9 are used in the timing calculations for these delays.

Table 9. Component Delays

DESCRIPTION	MNEUMONIC	PARAMETER
Maximum logic delay	$t_{PROPPALMAX}$	7.5 ns
Maximum tranceiver delay	$t_{Pxx245MAX}$†	4.1 ns
Minimum tranceiver delay	$t_{Pxx245MIN}$	1 ns
Maximum buffer delay	$t_{Pxx244MAX}$	4.1 ns
Minumum buffer delay	$t_{Pxx244MIN}$	1 ns

† "x" is replaced by L or H where applicable. For example, t_{PHL} indicates high-to-low transition; t_{PLH} indicates low-to-high transition.

In addition to the DRAM interface timing, the timing for the 'C80 cycle configuration inputs should also be evaluated. Since these signals are generated in external logic and feed directly back to the 'C80, the analysis is simple:

$$t_{a(MIDV-CFGV)} = t_{PROPPAL} = 7.5 \text{ ns}$$

The 'C80 only requires that $t_{a(MIDV-CFGV)}$ be less than 20 ns ($3t_H$–10) at 50 MHz.

Table 10. DRAM Timing Parameters – Access Times (2 cyc/col at 40 MHz)

PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	'C80 SPECIFICATION (ns)	40 MHz RESULT (ns)
Access time (address)	$t_{AA} + t_{Pxx244MAX} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 4t_H - 9 = 41$	38.2
Access time $\overline{CASL}$	$t_{CAC} + t_{Pxx245MAX}$	$t_{a(CASL-DV)} = 3t_H - 12 = 25.5$	19.1
Access time $\overline{CASH}$	$t_{CPA} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 4t_H - 9 = 41$	39.1
Access time $\overline{RASL}$	$t_{RAC} + t_{PROPPALMAX} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 8t_H - 8 = 92$	71.6

Table 11. DRAM Timing Parameters – Setup and Hold Times (2 cyc/col at 40 MHz)

PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	'C80 SPECIFICATION (ns)	40 MHz RESULT (ns)
Cycle time, random	$(t_{h(OUTV-OUTV)} = 14t_H - 5.5) - t_{PROPPALmax} + t_{PROPPALmin}$	$t_{RC} = 100$	162
Cycle time, page mode	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5)$	$t_{PC} = 40$	43.5
$\overline{RAS}$ low	$(t_{w(OUTV)} = 8t_H - 5.5) - t_{PROPPALmax} + t_{PROPALmin}$	$t_{RASP} = 60$	87
$\overline{CAS}$ low	$(t_{w(CASL)} = 3t_H - 11)$	$t_{CAS} = 15$	26.5
$\overline{CAS}$ high	$(t_{w(CASH)} = t_H - 2)$	$t_{CP} = 10$	10.5
Pulse duration, $\overline{W}$	$(t_{w(OUTV)} = 7t_H - 5.5) - t_{PHL244max} + t_{PLH244min}$	$t_{WP} = 15$	78.9
Setup time, address (column)	$(t_{h(OUTV-CASL)} = t_H - 4.5) - t_{Pxx244max}$	$t_{ASC} = 0$	1.4
Hold time, address (column)	$(t_{h(CASL-OUTV)} = 3t_H - 11) + t_{Pxx244min}$	$t_{CAH} = 10$	27.5
Setup time, address (row)	$(t_{su(OUTV-OUTV)} = 6t_H - 6.5) - t_{Pxx244max} + t_{PROPPALmin}$	$t_{ASR} = 0$	64.4
Hold time, address (row)	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{PROPPALmax} + t_{Pxx244min}$	$t_{RAH} = 10$	37
Setup time, data	$(t_{h(OUTV-CASL)} = t_H - 5) - t_{Pxx245max}$	$t_{DS} = 0$	0.9
Hold time, data	$(t_{h(CASL-OUTV)} = 3t_H - 11) + t_{Pxx245min}$	$t_{DH} = 10$	27.5
Setup time, $\overline{W}$ (to $\overline{CAS_H}$)	$(t_{h(OUTV-OUTV)} = 7t_H - 6.5) - t_{PHL244max}$	$t_{CWL} = 15$	76.9
Setup time, $\overline{W}$ (to $\overline{RAS_H}$)	$(t_{h(OUTV-OUTV)} = 7t_H - 5.5) - t_{PROPPALmax} + t_{PLH244min}$	$t_{RWL} = 15$	75.5
Setup time, $\overline{W}$ (refresh)	$(t_{h(OUTV-OUTV)} = 5t_H - 5.5) + t_{PROPPALmin} - t_{PLH244max}$	$t_{WSR} = 10$	52.9
Hold time, $\overline{W}$ (refresh)	$(t_{h(OUTV-OUTV)} = 15t_H - 5.5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{WHR} = 10$	175.5
Hold time, read from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 9t_H - 5.5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{RRH} = 5$	100.5
Setup time, read to $\overline{CAS}$	$(t_{h(OUTV-CASL)} = 10t_H - 3.5) - t_{PLH244max}$	$t_{RCS} = 0$	117.4
Hold time, read from $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 9t_H - 6.5) - t_{Pxx244max} + t_{PHL244min}$	$t_{RCH} = 0$	102.9

Table 12. DRAM Timing Parameters – Delay Times (2 cyc/col at 40 MHz)

PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	DRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
$\overline{RAS_L}$ to $\overline{CAS_H}$ (refresh)	$(t_{h(OUTV-OUTV)} = 8t_H - 6.5) - t_{PROPPALmax}$	$t_{CHR} = 20$	86
$\overline{CAS_H}$ to $\overline{RAS_L}$	$(t_{h(OUTV-OUTV)} = 6t_H - 5.5 + t_{PROPPALmin}$	$t_{CRP} = 5$	69.5
$\overline{RAS_L}$ to $\overline{CAS_H}$	$(t_{h(OUTV-OUTV)} = 8t_H - 6.5) - t_{PROPPALmax}$	$t_{CSH} = 60$	86
$\overline{CAS_L}$ to $\overline{RAS_L}$ (refresh)	$(t_{h(OUTV-OUTV)} = 3t_H - 5.5) + t_{PROPPALmin}$	$t_{CSR} = 10$	32
$\overline{RAS_L}$ to column address	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{PROPPALmax} + t_{Pxx244min}$	$t_{RAD} = 15$	37
Column address to $\overline{RAS_H}$	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{Pxx244max} + t_{PROPPALmin}$	$t_{RAL} = 30$	40.4
Column address to $\overline{CAS_H}$	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{Pxx244max}$	$t_{CAL} = 30$	39.4
$\overline{RAS_L}$ to $\overline{CAS_L}$	$(t_{h(OUTV-CASL)} = 5t_H - 3.5) - t_{PROPPALmax}$	$t_{RCD} = 20$	51.5
$\overline{RAS_H}$ to $\overline{CAS_L}$ (refresh)	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{PROPPALmax}$	$t_{RPC} = 0$	36
$\overline{CAS_L}$ to $\overline{RAS_H}$	$(t_{h(CASL-OUTV)} = 3t_H - 11) + t_{PROPPALmin}$	$t_{RSH} = 15$	26.5
$\overline{RAS}$ hold from precharge	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) + t_{PROPPALmin}$	$t_{CPRH} = 35$	44.5

Table 13. DRAM Timing Parameters – Access Times (3 cyc/col at 50 MHz)

PARAMETER	FORMULA (WITH $\overline{CAS}$ BUFFERS)	DRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
Access time (address)	$t_{AA} + t_{Pxx244max} + t_{Pxx245max}$	$t_{a(OUTV-DV)} = 6t_H - 7 = 53$	38.2
Access time ($\overline{CAS_L}$)	$t_{CAC} + t_{PHL244max} + t_{Pxx245max}$	$t_{a(OUTV-DV)} = 4t_H - 7 = 33$	23.2
Access time ($\overline{CAS_H}$)	$t_{CPA} + t_{PLH244max} + t_{Pxx245max}$	$t_{a(OUTV-DV)} = 6t_H - 7 = 53$	43.2
Access time ($\overline{RAS_L}$)	$t_{RAC} + t_{PROPPALmax} + t_{Pxx245max}$	$t_{a(OUTV-DV)} = 10t_H - 6.5 = 93.5$	71.6

Table 14. DRAM Timing Parameters – Setup and Hold Times (3 cyc/col at 50MHz)

PARAMETER	FORMULA (WITH $\overline{CAS}$ BUFFERS)	DRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
Cycle time, random	$(t_{h(OUTV-OUTV)} = 18t_H - 5) - t_{PROPPALmax} + t_{PROPPALmin}$	$t_{RC} = 100$	168.5
Cycle time, page mode	$(t_{h(OUTV-OUTV)} = 6t_H - 5) - t_{PLH244max} + t_{PLH244min}$	$t_{PC} = 40$	51.9
$\overline{RAS}$ low	$(t_{w(OUTV)} = 10t_H - 5) - t_{PROPPALmax} + t_{PROPALmin}$	$t_{RASP} = 60$	87.5
$\overline{CAS}$ low	$(t_{w(OUTV)} = 4t_H - 5.5) - t_{PHL244max} + t_{PLH244min}$	$t_{CAS} = 15$	31.4
$\overline{CAS}$ high	$(t_{w(OUTV)} = 2t_H - 5.5) - t_{PLH244max} + t_{PHL244min}$	$t_{CP} = 10$	11.4
Pulse duration, $\overline{W}$	$(t_{w(OUTV)} = 8t_H - 5) - t_{PHL244max} + t_{PLH244min}$	$t_{WP} = 15$	71.9
Setup time, address column	$(t_{h(OUTV-OUTV)} = 2t_H - 5.5) - t_{Pxx244max} + t_{PHL244min}$	$t_{ASC} = 0$	11.4
Hold time, address (column)	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{PHL244max} + t_{Pxx244min}$	$t_{CAH} = 10$	31.4
Setup time, address (row)	$(t_{h(OUTV-OUTV)} = 8t_H - 5) - t_{Pxx244max} + t_{PROPPALmin}$	$t_{ASR} = 0$	70.9
Hold time, address row	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{PROPPALmax} + t_{Pxx244min}$	$t_{RAH} = 10$	28
Setup time, data	$(t_{h(OUTV-OUTV)} = 2t_H - 5.5) + t_{PHL244min} - t_{Pxx245max}$	$t_{DS} = 0$	11.4
Hold time, data	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{PHL244max} + t_{Pxx245min}$	$t_{DH} = 10$	30.4
Setup time, $\overline{W}$ to $\overline{CAS}_H$	$(t_{h(OUTV-OUTV)} = 7t_H - 5.5) - t_{PHL244max} + t_{PLH244min}$	$t_{CWL} = 15$	61.4
Setup time, $\overline{W}$ to $\overline{RAS}_H$	$(t_{h(OUTV-OUTV)} = 7t_H - 5) - t_{PROPPALmax} + t_{PLH244min}$	$t_{RWL} = 15$	58.5
Setup time, $\overline{W}$ refresh	$(t_{h(OUTV-OUTV)} = 7t_H - 5) + t_{PROPPALmin} - t_{PLH244max}$	$t_{WSR} = 10$	60.9
Hold time, $\overline{W}$ refresh	$(t_{h(OUTV-OUTV)} = 17t_H - 5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{WHR} = 10$	158.5
Hold time, read from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 11t_H - 5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{RRH} = 5$	98.5
Setup time, read to $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 13t_H - 5.5) + t_{PHL244min} - t_{PLH244max}$	$t_{RCS} = 0$	121.4
Hold time, read from $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 11t_H - 5) - t_{PLH244max} + t_{PHL244min}$	$t_{RCH} = 0$	101.9

Table 15. DRAM Timing Parameters – Delay Times (3 cyc/col at 50MHz)

PARAMETER	FORMULA (WITH $\overline{CAS}$ BUFFERS)	DRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
$\overline{RAS}_L$ to $\overline{CAS}_H$ (refresh)	$(t_{h(OUTV-OUTV)} = 10t_H - 5.5) - t_{PROPPALmax} + t_{PLH244min}$	$t_{CHR} = 20$	88
$\overline{CAS}_H$ to $\overline{RAS}_L$	$(t_{h(OUTV-OUTV)} = 8t_H - 5 - t_{PLH244max} + t_{PROPPALmin}$	$t_{CRP} = 5$	70.9
$\overline{RAS}_L$ to $\overline{CAS}_H$	$(t_{h(OUTV-OUTV)} = 10t_H - 5.5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{CSH} = 60$	88
$\overline{CAS}_L$ to $\overline{RAS}_L$ (refresh)	$(t_{d(OUTV-OUTV)} = 3t_H - 5) - t_{PHL244max} + t_{PROPPALmin}$	$t_{CSR} = 10$	20.9
$\overline{RAS}_L$ to column address	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{PROPPALmax} + t_{Pxx244min}$	$t_{RAD} = 15$	28
Column address to $\overline{RAS}_H$	$(t_{h(OUTV-OUTV)} = 6t_H - 5) - t_{Pxx244max} + t_{PROPPALmin}$	$t_{RAL} = 30$	50.9
Column address to $\overline{CAS}_H$	$(t_{h(OUTV-OUTV)} = 6t_H - 5.5) - t_{Pxx244max} + t_{PLH244min}$	$t_{CAL} = 30$	51.4
$\overline{RAS}_L$ to $\overline{CAS}_L$	$(t_{h(OUTV-OUTV)} = 5t_H - 5.5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{RCD} = 20$	38
$\overline{RAS}_H$ to $\overline{CAS}_L$ (refresh)	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{PROPPALmax} + t_{PHL244min}$	$t_{RPC} = 0$	28
$\overline{CAS}_L$ to $\overline{RAS}_H$	$(t_{h(OUTV-OUTV)} = 4t_H - 5) - t_{PHL244max} + t_{PROPPALmin}$	$t_{RSH} = 15$	30.9
Hold time, $\overline{RAS}$ from precharge	$(t_{h(OUTV-OUTV)} = 6t_H - 5) - t_{PLH244max} + t_{PROPPALmin}$	$t_{CPRH} = 35$	50.9

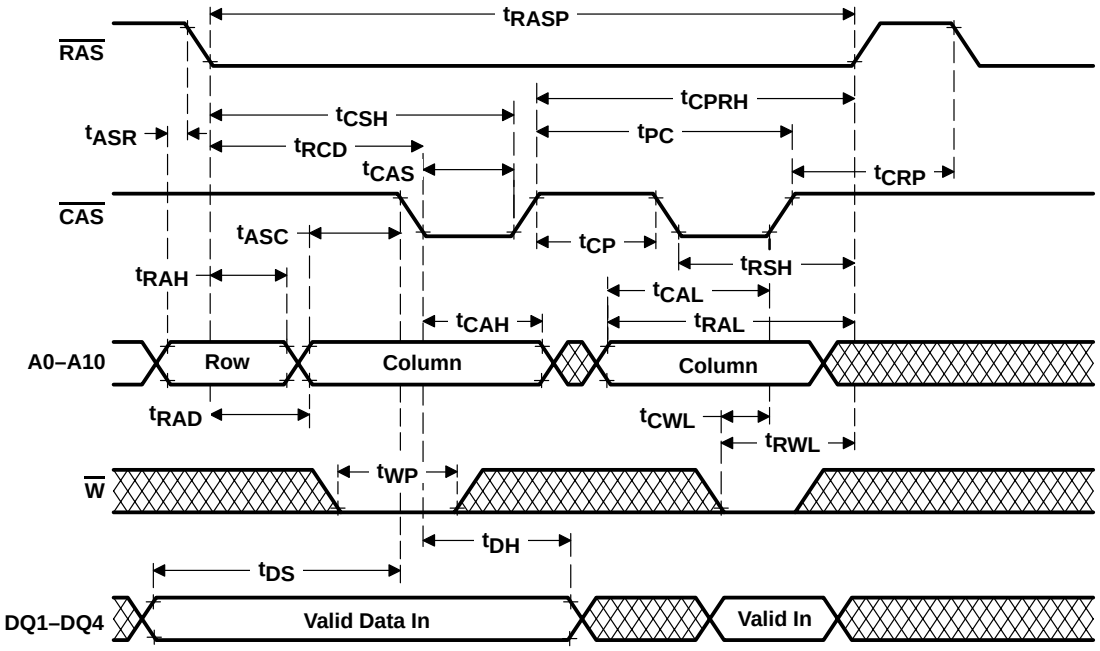

Figure 3. DRAM Write Cycle (page-mode)

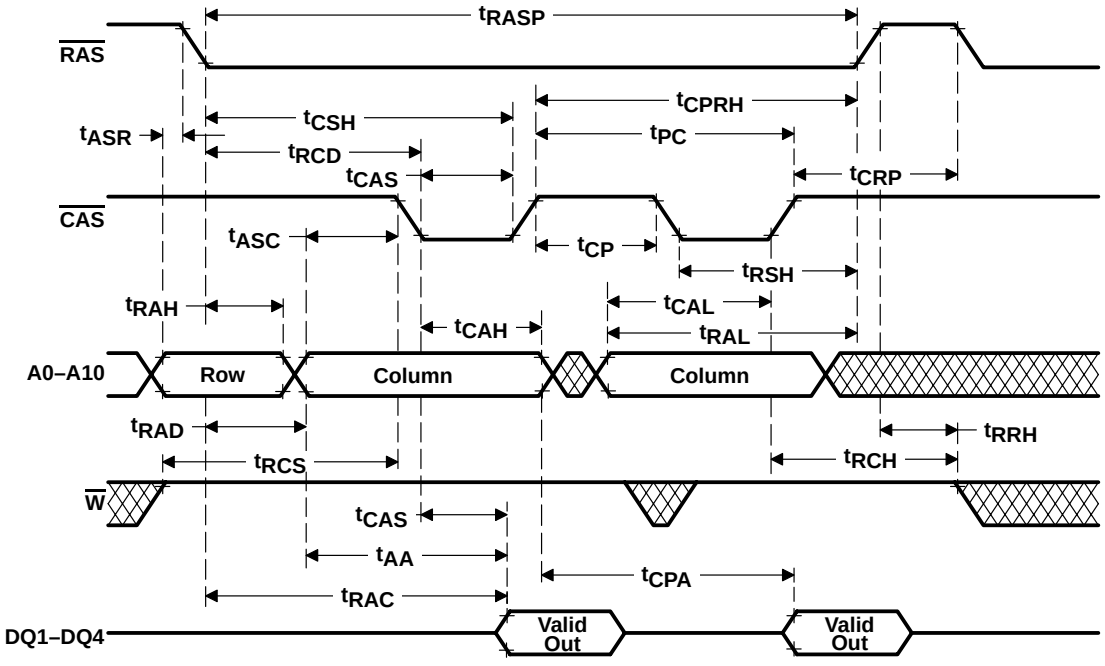

Figure 4. DRAM Read Cycle (page-mode)

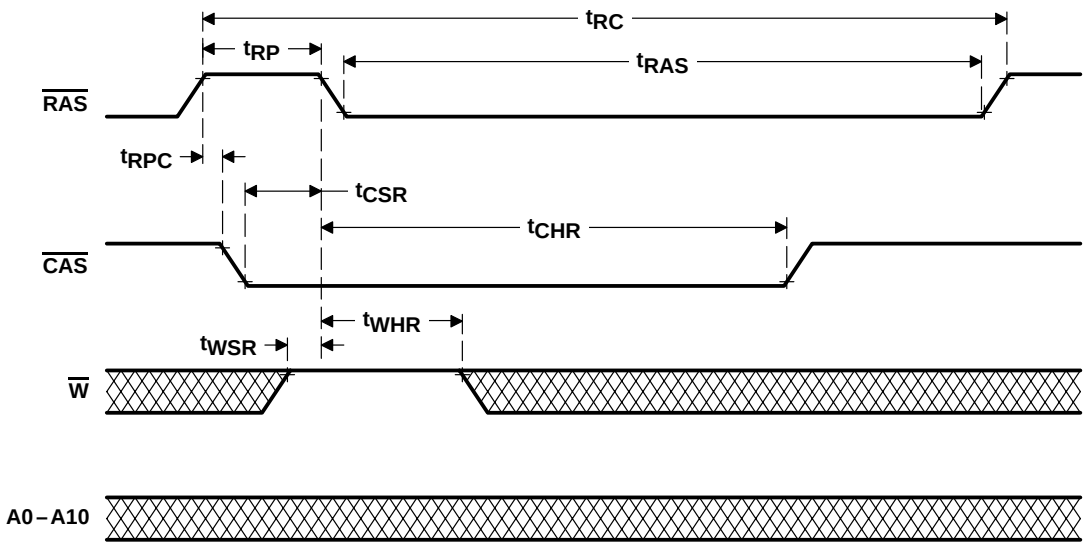

Figure 5. DRAM Refresh Cycle

Changing the Bank Size

Some systems may require that the bank size be variable, depending on the SIMMs available. If this is a possibility, then it is advantageous to build this type of support directly into the system so that the SIMMs can be interchanged and still function properly.

For the larger SIMMs (16MB and 32MB), ’C80 address pins A14–A24 must address the DRAM. Since 4MB and 8MB SIMMs are built from 1MB × *n* DRAMs, bits A13–A23 should be used when interfacing SIMMs of these sizes. Using buffer to perform an extra bit shift is a viable solution, but it is expensive and unnecessary. A much simpler solution is to generate the A0 input a little differently. For the 16MB and 32MB SIMMs, bit A0 of the DRAM bank is C80A24; for 4MB and 8MB SIMMs it should be C80A13. With 16MB and 32MB SIMMs, addressing is no longer linear; however, it is the same for both reads and writes, and thus functions perfectly. If this DRAM bank is a shared resource, it is important that both the host and the ’C80 address the bank identically.

In this design, the support for the larger SIMMs was added with a simple jumper. For designers who would like an even more automatic approach, the PD bits can be used with external logic to generate the A0 input to the DRAM bank. The only adverse effect of this is that at 40 MHz, the access time from the column address is a tight constraint, violating by 600 ps in the worst case because the logic is typically slower than with the LVT buffer that was formerly used. This constraint exists only when using the 4 MB and 8 MB SIMMs since the ’C80’s A24 bit switches on a page boundary when using 16MB and 32MB SIMMs. The row time overhead accounts for the delay.

Using the PD bits, a logic equation for the A0 input to the DRAM bank might be as follows:

```
A0 =  C80A24 & ((!PD1a & PD2a) # (PD1a & !PD2a))
      # C80A13 & ((PD1a & PD2a) # (!PD1a & !PD2a));
```

Appendix A: Bill of Materials

Table 16. Table of Materials

PART	NUMBER USED	PART TYPE	DESIGNATORS
1	45	.1 µF	C1A C1B C1C C2A C2B C2C C5A C5B C6A C6B C7A C7B C8A C8B C15A C15B C15C C15D C15E C15F C15G C15H C15I C15J C15K C15L C15M C15N C15P C15Q C17 C18A C18B C19A C19B C20A C20B C21A C21B C22A C22C C22D C23 C24 C25
2	3	4.7 µF	C15R C15S C22B
3	7	10 kΩ	R3 R4 R5 R6 R7 R8 R22
4	1	15 kΩ	RP7
5	6	22 Ω	RP1 RP2 RP3 RP4 RP5 RP6
6	2	22 µF	C59 C60
7	3	74LVT16244	U18 U19 U20
8	4	74LVT16245	U5 U6 U7 U8
9	1	74LVT16373	U21
10	1	EMULATOR HEADER	U16
11	1	JUMPER 1X3	J1
12	1	OSCILLATOR	U17
13	3	PAL22LV10PLCC	U23 U24 U25
14	1	SW SPST	S1
15	1	TL7705A	U22
16	2	TM497BBK32	U1 U2
17	1	TMS320C80–GF	U15

Appendix B: Schematics

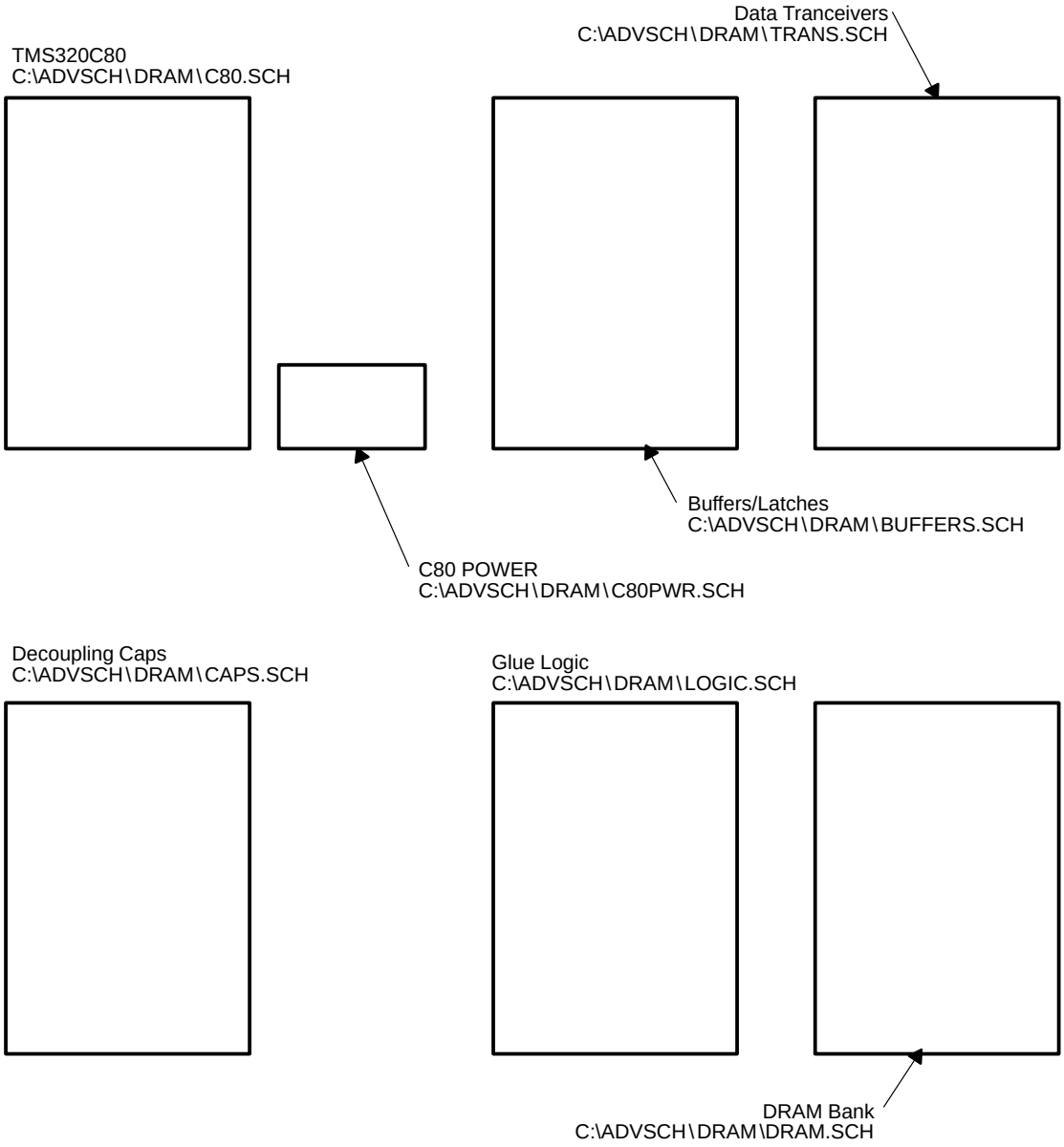

Figure 6. DRAM Application Report

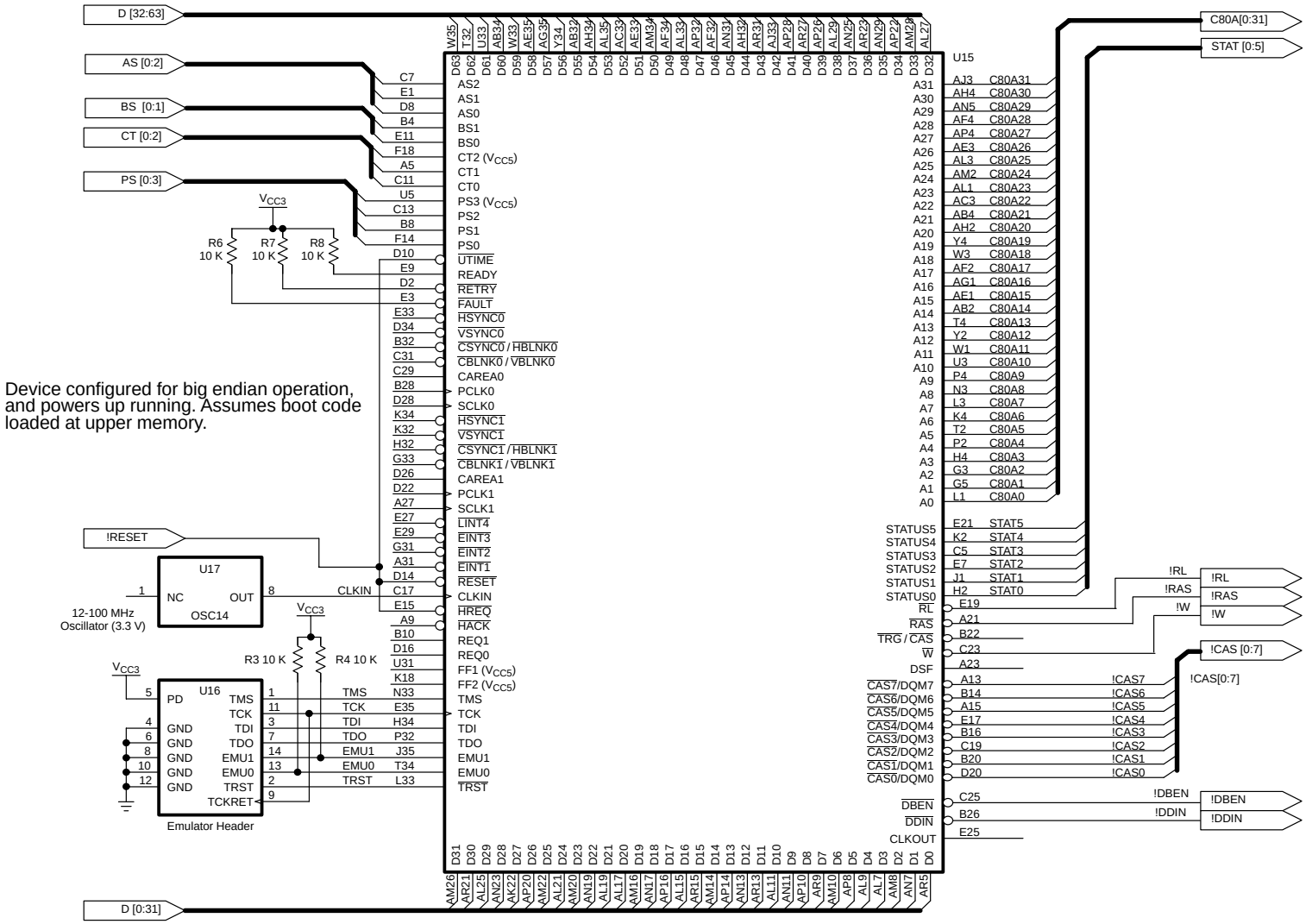

Figure 7. TMS320C80-GF

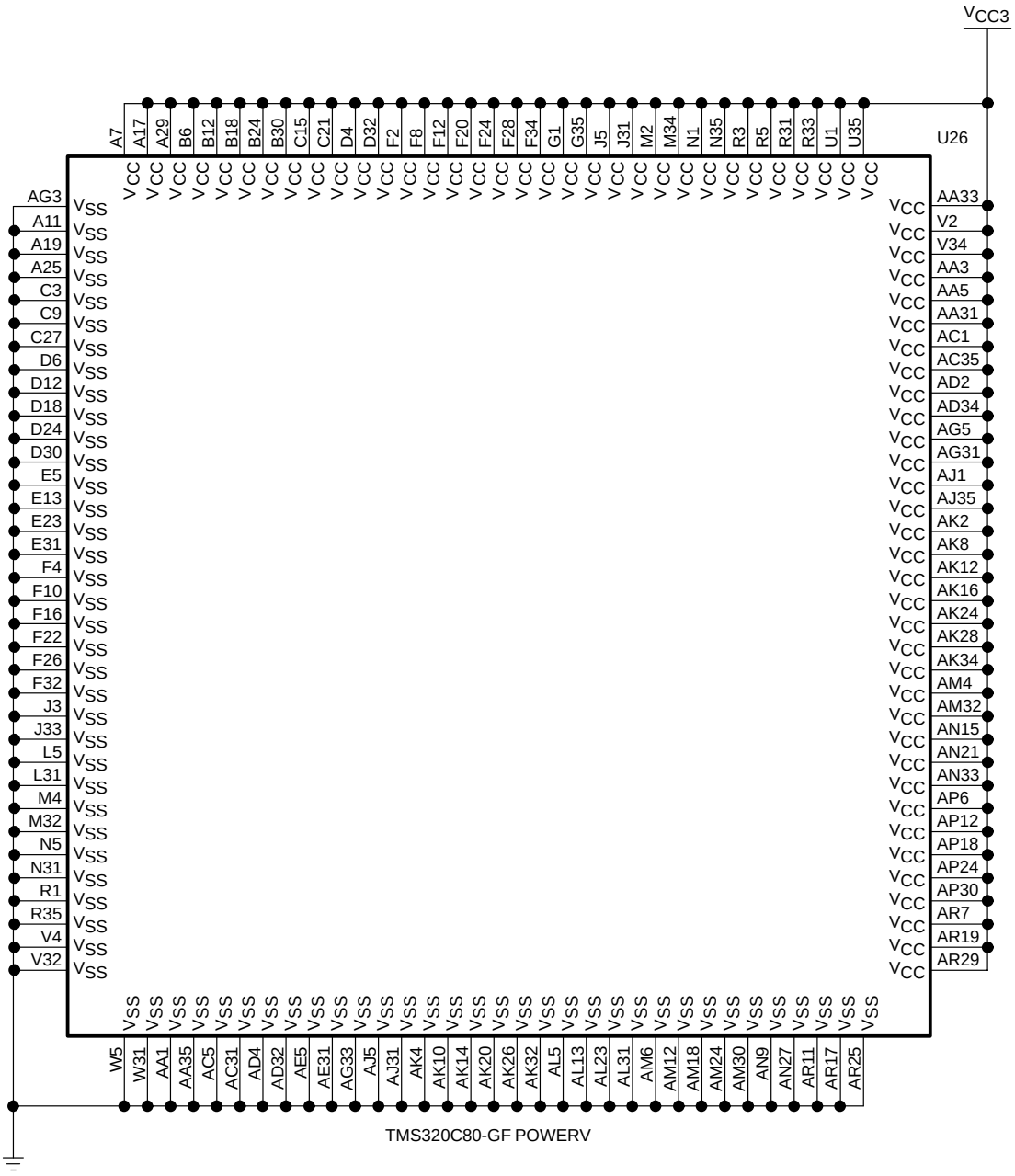

Figure 8. TMS320C80 Power and Ground Planes

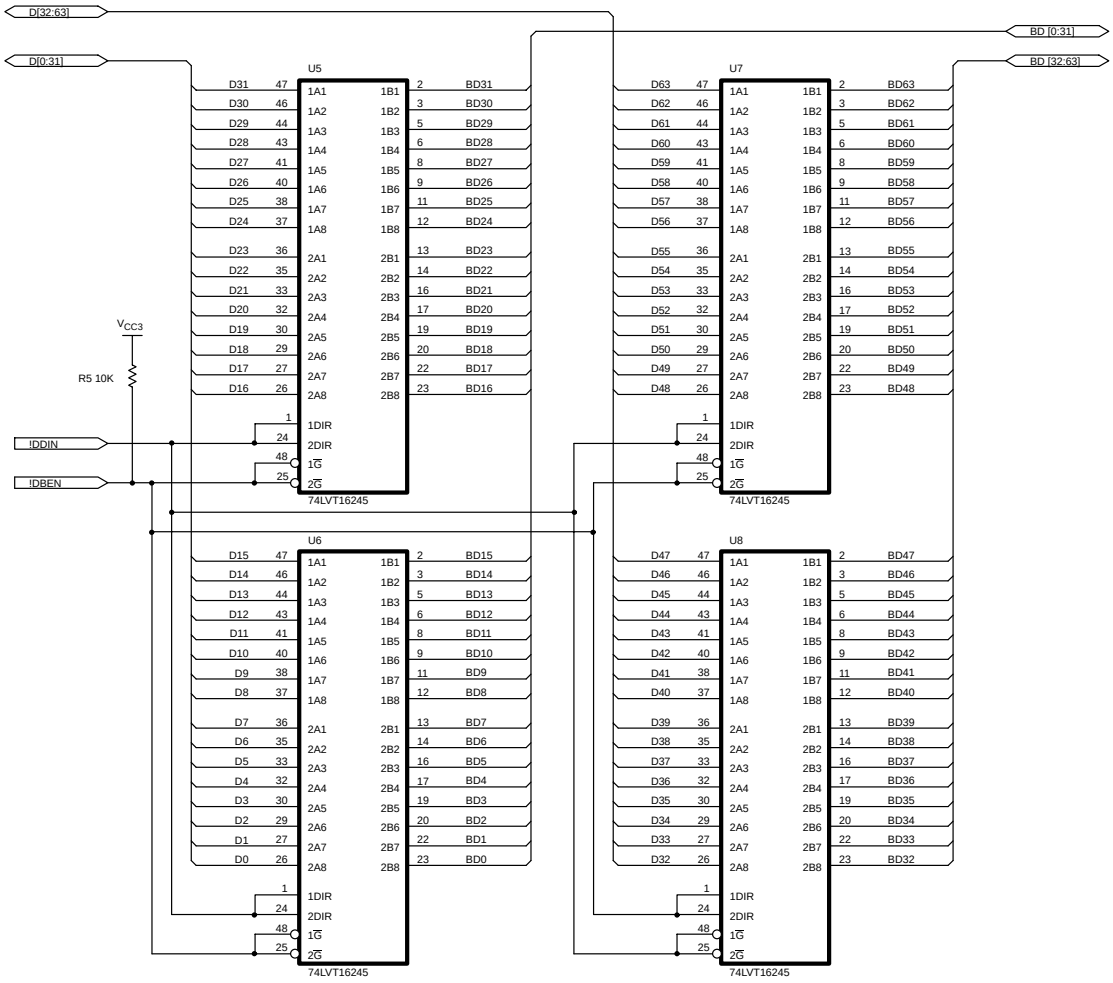

Figure 9. Data Bus Transceivers

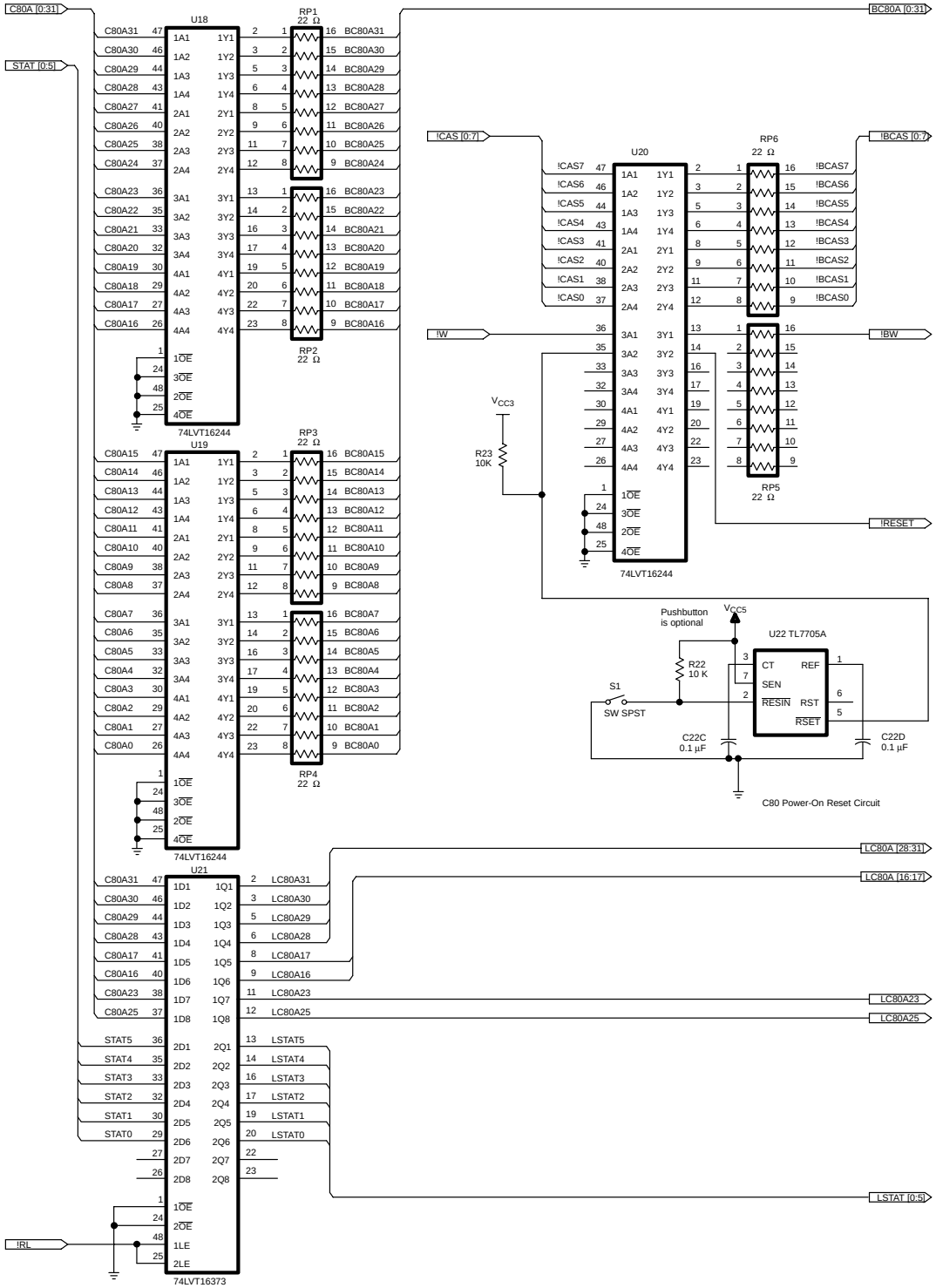

Figure 10. Address Buffers and Latches

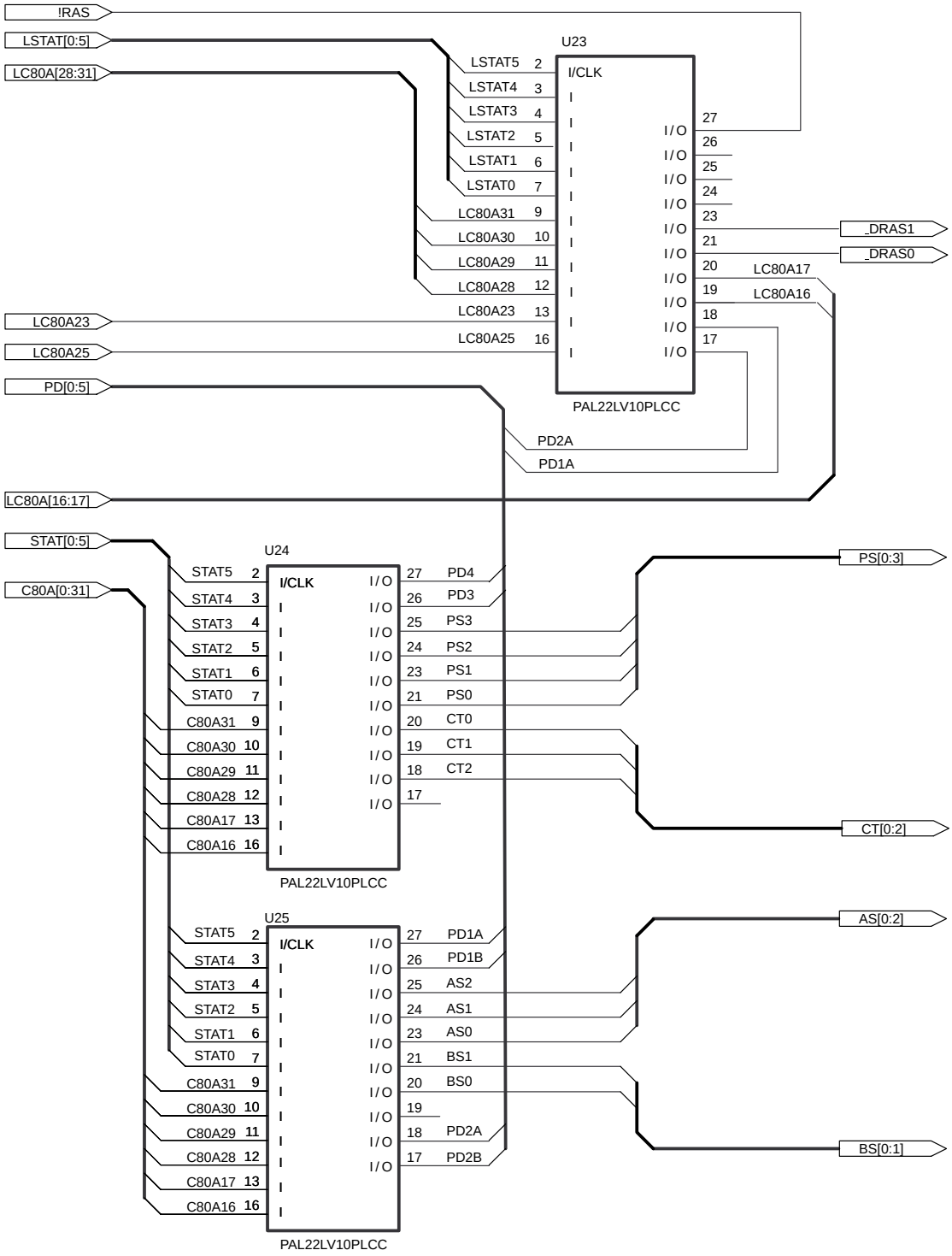

Figure 11. Glue Logic

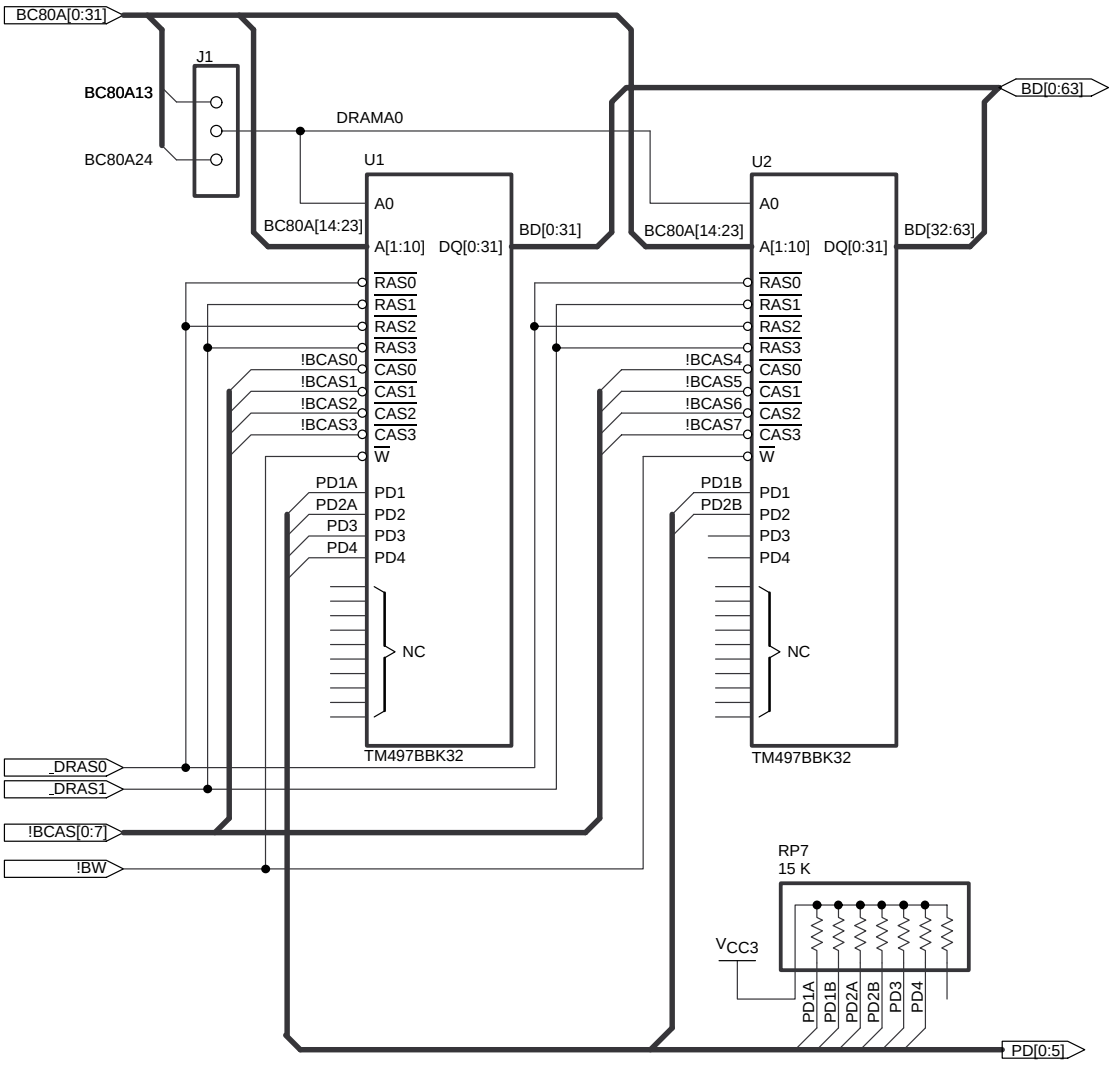

Figure 12. DRAM SIMMs

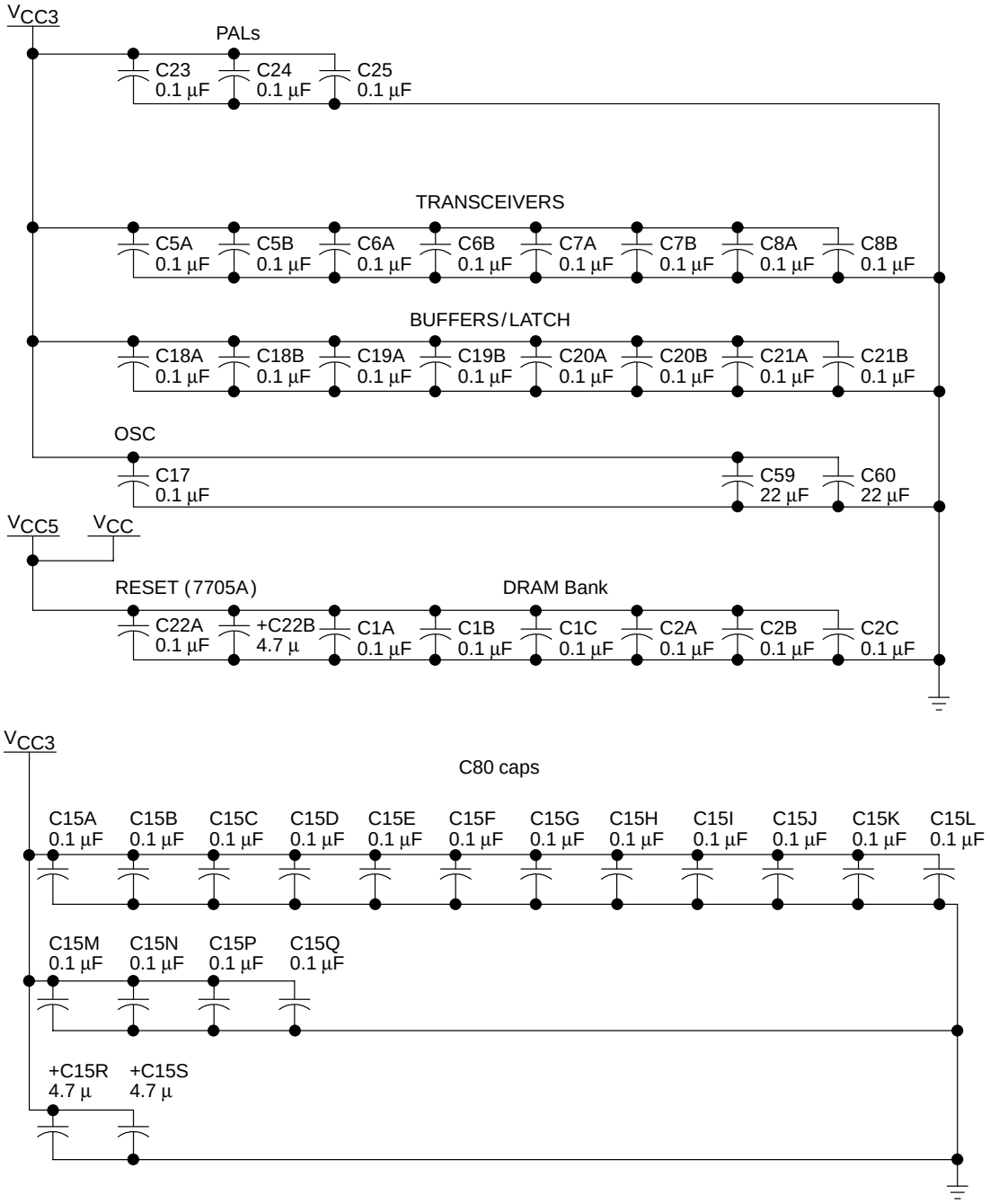

Figure 13. Decoupling Caps

Appendix C: ABEL™ Files

```
module DRASGENERATOR
title'
DWG NAME  LOGIC
PART #    U23
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C8X APPLICATIONS
DATE      07_19_95'
   DRASGEN   device 'P22V10C'; " GAL22V10–7C PLCC
   LSTAT5    Pin 2; "C80 latched STATUS5
   LSTAT4    Pin 3; "C80 latched STATUS4
   LSTAT3    Pin 4; "C80 latched STATUS3
   LSTAT2    Pin 5; "C80 latched STATUS2
   LSTAT1    Pin 6; "C80 latched STATUS1
   LSTAT0    Pin 7; "C80 latched STATUS0
   LA31      Pin 9; "C80 latched address line 31
   LA30      Pin 10; "C80 latched address line 30
   LA29      Pin 11; "C80 latched address line 29
   LA28      Pin 12; "C80 latched address line 28
   LA23      Pin 13; "C80 latched address line 23
   LA25      Pin 16; "C80 latched address line 25
   Vss       Pin 14; "Ground
   Vcc       Pin 28; "Power
   _RAS         Pin 27; "C80 _RAS
   SDRAM_cyc Pin 26;
   refresh   Pin 25;
   "NC       Pin 24;
   _DRAS1    Pin 23; "DRAM _RAS1
   _DRAS0    Pin 21; "DRAM _RAS0
   LA17      Pin 20; "latched address line 17– used for refresh decode
   LA16      Pin 19; "latched address line 16– used for refresh decode
   PD1A      Pin 18; "presence detect bit 1, SIMM A
   PD2A      Pin 17; "presence detect bit 2, SIMM A
"constants and alias names
   S5     = LSTAT5;
   S4     = LSTAT4;
```

ABEL is a trademark of DATA I/O.

```
   S3        = LSTAT3;
   S2        = LSTAT2;
   S1        = LSTAT1;
   S0        = LSTAT0;
   DRAM_AV   = LA31 & LA30 & !LA29 & LA28;
   ADDR      = [LA31..LA28] ;
   STAT      = [LSTAT5..LSTAT0] ;
   PD        = [PD1A,PD2A] ;
   BS        = [LA25,LA23];
   RFHA      = [LA17..LA16];
equations
   !_DRAS0   = (!_RAS & DRAM_AV & !SDRAM_cyc & !refresh
              &(!PD1A #
             PD2A & PD1A & !LA23 #
             !PD2A & PD1A & !LA25))
              #(!_RAS & refresh & !LA17 & !LA16);
   !_DRAS1   = (!_RAS & DRAM_AV & !SDRAM_cyc & !refresh
             &( PD2A & PD1A & LA23 #
             !PD2A & PD1A & LA25))
             #(!_RAS & refresh & LA17 & !LA16);

   refresh   = !S5 & !S4 & !S3 & !S2 &  S1 & !S0 ;
   SDRAM_cyc = !S5 & !S4 & !S3 & !S2 &  S1 &  S0
             #!S5 & !S4 &  S3 &  S2 & !S1 & !S0 ;

test_vectors"_DRAS0
([ ADDR , STAT, PD, _RAS, BS , RFHA  ] -> [_DRAS0 ])
 [  .X. , .X. ,.X.,   1 ,.X. , .X.   ] -> [    1   ];
 [  13  ,  0  , 0 ,   0 ,.X. , .X.   ] -> [    0   ];
 [  13  ,  0  , 1 ,   0 ,.X. , .X.   ] -> [    0   ];
 [  13  ,  1  , 0 ,   0 ,.X. , .X.   ] -> [    0   ];
 [  13  ,  1  , 1 ,   0 ,.X. , .X.   ] -> [    0   ];
 [  13  ,  0  , 3 ,   0 , 0  , .X.   ] -> [    0   ];
 [  13  ,  0  , 3 ,   0 , 2  , .X.   ] -> [    0   ];
 [  13  ,  1  , 3 ,   0 , 0  , .X.   ] -> [    0   ];
 [  13  ,  1  , 3 ,   0 , 2  , .X.   ] -> [    0   ];
 [  13  ,  0  , 2 ,   0 , 0  , .X.   ] -> [    0   ];
 [  13  ,  0  , 2 ,   0 , 1  , .X.   ] -> [    0   ];
 [  13  ,  1  , 2 ,   0 , 0  , .X.   ] -> [    0   ];
```

```
 [  13  ,  1  , 2 ,   0 , 1  , .X.   ] -> [    0   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  0    ] -> [    0   ];
 [   5  , .X. ,.X.,   0 ,.X. , .X.   ] -> [    1   ];
 [   7  , .X. ,.X.,   0 ,.X. , .X.   ] -> [    1   ];
 [  .X. ,  3  ,.X.,   0 ,.X. , .X.   ] -> [    1   ];
 [  13  ,  0  , 2 ,   0 , 2  , .X.   ] -> [    1   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  1    ] -> [    1   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  2    ] -> [    1   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  3    ] -> [    1   ];
test_vectors”_DRAS1
([ ADDR , STAT, PD, _RAS, BS , RFHA  ] -> [ _DRAS1 ])
 [  .X. , .X. ,.X.,   1 ,.X. , .X.   ] -> [    1   ];
 [  13  ,  0  , 3 ,   0 , 1  , .X.   ] -> [    0   ];
 [  13  ,  0  , 3 ,   0 , 3  , .X.   ] -> [    0   ];
 [  13  ,  1  , 3 ,   0 , 1  , .X.   ] -> [    0   ];
 [  13  ,  1  , 3 ,   0 , 3  , .X.   ] -> [    0   ];
 [  13  ,  0  , 2 ,   0 , 2  , .X.   ] -> [    0   ];
 [  13  ,  0  , 2 ,   0 , 3  , .X.   ] -> [    0   ];
 [  13  ,  1  , 2 ,   0 , 2  , .X.   ] -> [    0   ];
 [  13  ,  1  , 2 ,   0 , 3  , .X.   ] -> [    0   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  2    ] -> [    0   ];
 [   5  , .X. ,.X.,   0 ,.X. , .X.   ] -> [    1   ];
 [   7  , .X. ,.X.,   0 ,.X. , .X.   ] -> [    1   ];
 [  .X. ,  3  ,.X.,   0 ,.X. , .X.   ] -> [    1   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  1    ] -> [    1   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  0    ] -> [    1   ];
 [  .X. ,  2  ,.X.,   0 ,.X. ,  3    ] -> [    1   ];
end DRASGENERATOR
```

```
module C80codes1
title'
DWG NAME  LOGIC.SCH
PAL #     U24
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      07_22_95'
   xx_001 device 'P22V10C'; " PAL22LV10–7 PLCC
   STAT5     Pin 2;  "C80 STATUS[5]
   STAT4     Pin 3;  "C80 STATUS[4]
   STAT3     Pin 4;  "C80 STATUS[3]
   STAT2     Pin 5;  "C80 STATUS[2]
   STAT1     Pin 6;  "C80 STATUS[1]
   STAT0     Pin 7;  "C80 STATUS[0]
   A31       Pin 9;  "C80 address line 31
   A30       Pin 10; "C80 address line 30
   A29       Pin 11; "C80 address line 29
   A28       Pin 12; "C80 address line 28
   A17       Pin 13; "C80 address line 17
   A16       Pin 16; "C80 address line 16
   VSS       Pin 14; "Ground
   VCC       Pin 28; "Power
   PD4       Pin 27; "presence detect 4– low indicates 16M SIMMs
   PD3       Pin 26; "presence detect 3– low indicates 32M SIMMs
   PS3       Pin 25; "C80 PS[3] input (page size)
   PS2       Pin 24; "C80 PS[2] input (page size)
   PS1       Pin 23; "C80 PS[1] input (page size)
   PS0       Pin 21; "C80 PS[0] input (page size)
   CT0       Pin 20; "C80 CT[0] input (cycle timing)
   CT1       Pin 19; "C80 CT[1] input (cycle timing)
   CT2       Pin 18; "C80 CT[2] input (cycle timing)
   "NC       Pin 17;
"constants and alias names
   ADDR      = [A31..A28] ;
   REFADD    = [A17..A16] ;
   STAT      = [STAT5..STAT0] ;
   PD        = [PD4..PD3] ;
   refr1     = 0 ; "SDRAM bank refresh pseudo–address
   refr2     = 2 ; "SDRAM bank refresh pseudo–address
```

```
   MRS       = 12 ;
   DCAB      = 3 ;
   REFRESH   = 2 ;
   READ      = 0 ;
   WRITE     = 1 ;
   SVNTY     = 2 ;
   ATY       = 1 ;
   SIXTY     = 3 ;
   SEVENTY   = !PD3 ;
   EIGHTY    = !PD4 ;
   SDRAM_cyc    =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & STAT0)
                #(!STAT5 & !STAT4 & STAT3 & STAT2 & !STAT1 & !STAT0);
   DRAM_refresh =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0)
                &(( A17 & !A16) #
                (!A17 & !A16));
   REFH         = (!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0);
   DRAM_AV      = A31 & A30 & !A29 & A28 ;

equations
   PS3    = (DRAM_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_PS3=1)

   PS2    = (DRAM_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_PS2=1)

   PS1    = 0 ;
          "(SYSTEM_SPECIFIC_REQUIRING_PS1=1);

   PS0    = 0 ;
          "#(SYSTEM_SPECIFIC_REQUIRING_PS0=1)

   CT2    = (DRAM_AV) & !(SDRAM_cyc # REFH)
           #DRAM_refresh ;
          "#(SYSTEM_SPECIFIC_REQUIRING_CT2=1)
   CT1    = (DRAM_AV) & !(SDRAM_cyc # REFH)
          #DRAM_refresh ;
          "#(SYSTEM_SPECIFIC_REQUIRING_CT1=1)
   CT0    = (DRAM_AV) & !(SDRAM_cyc # REFH)
          &(SEVENTY # EIGHTY)
```

```
      #(DRAM_refresh)
      &(SEVENTY # EIGHTY) ;
      ”#(SYSTEM_SPECIFIC_REQUIRING_CT0=1)

test_vectors”PS3
([ ADDR  , STAT  ] -> [PS3])
 [  0    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  11   , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  13   ,  0    ] -> [ 1 ] ;” DRAM addressed, reads
 [  13   ,  1    ] -> [ 1 ] ;” DRAM addressed, writes
 [  13   ,REFRESH] -> [ 0 ] ;” DRAM addressed, refresh
 [  13   ,  MRS  ] -> [ 0 ] ;” DRAM addressed, MRS cycle
 [  13   ,  DCAB ] -> [ 0 ] ;” DRAM addressed, DCAB cycle
 [  14   , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;” DRAM not addressed
test_vectors”PS2
([ ADDR  , STAT  ] -> [PS2])
 [  0    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;” DRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ;” DRAM not addressed
```

```
 [  11   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   ,  0    ] -> [ 1 ] ;" DRAM addressed, reads
 [  13   ,  1    ] -> [ 1 ] ;" DRAM addressed, writes
 [  13   ,REFRESH] -> [ 0 ] ;" DRAM addressed, refresh
 [  13   ,  MRS  ] -> [ 0 ] ;" DRAM addressed, MRS cycle
 [  13   ,  DCAB ] -> [ 0 ] ;" DRAM addressed, DCAB cycle
 [  14   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" DRAM not addressed
test_vectors"PS1
([ ADDR  , STAT  ] -> [PS1])
 [  0    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  11   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" DRAM addressed
 [  14   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" DRAM not addressed
test_vectors"PS0
([ ADDR  , STAT  ] -> [PS0])
 [  0    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;" DRAM not addressed
```

```
 [  10   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  11   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" DRAM addressed
 [  14   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" DRAM not addressed
test_vectors"CT2
([ ADDR , STAT, REFADD ] -> [ CT2 ])
 [  0   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  1   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  2   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  3   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  4   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  5   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  6   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  7   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  8   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  9   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  10  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  11  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  12  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  13  , READ,  .X.   ] -> [  1  ] ;" DRAM addressed
 [  13  ,WRITE,  .X.   ] -> [  1  ] ;" DRAM addressed
 [  14  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  15  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  .X. ,REFRESH,refr1 ] -> [  1  ] ;" DRAM refresh
 [  .X. ,REFRESH,refr2 ] -> [  1  ] ;" DRAM refresh
 [  .X. ,REFRESH,  1   ] -> [  0  ] ;" refresh- not DRAM
 [  .X. ,REFRESH,  3   ] -> [  0  ] ;" refresh- not DRAM
 [  .X. , MRS ,  .X.   ] -> [  0  ] ;" MRS
 [  .X. , DCAB,  .X.   ] -> [  0  ] ;" DCAB

test_vectors"CT1
([ ADDR , STAT, REFADD ] -> [ CT1 ])
 [  0   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  1   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  2   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  3   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  4   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
```

```
 [  5   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  6   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  7   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  8   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  9   , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  10  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  11  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  12  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  13  , READ,  .X.   ] -> [  1  ] ;" DRAM addressed
 [  13  ,WRITE,  .X.   ] -> [  1  ] ;" DRAM addressed
 [  14  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  15  , .X. ,  .X.   ] -> [  0  ] ;" DRAM not addressed
 [  .X. ,REFRESH,refr1 ] -> [  1  ] ;" DRAM refresh
 [  .X. ,REFRESH,refr2 ] -> [  1  ] ;" DRAM refresh
 [  .X. ,REFRESH,  1   ] -> [  0  ] ;" refresh- not DRAM
 [  .X. ,REFRESH,  3   ] -> [  0  ] ;" refresh- not DRAM
 [  .X. , MRS ,  .X.   ] -> [  0  ] ;" MRS
 [  .X. , DCAB,  .X.   ] -> [  0  ] ;" DCAB
test_vectors"CT0
([ ADDR , STAT, REFADD, PD    ] -> [ CT0 ])
 [  0   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  1   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  2   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  3   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  4   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  5   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  6   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  7   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  8   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  9   , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  10  , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  11  , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  12  , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  13  , READ,  .X.  , SVNTY ] -> [  1  ] ;" DRAM (-70)
 [  13  ,WRITE,  .X.  ,  ATY  ] -> [  1  ] ;" DRAM (-80)
 [  13  ,WRITE,  .X.  , SIXTY ] -> [  0  ] ;" DRAM (-60 )
 [  14  , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  15  , .X. ,  .X.  , .X.   ] -> [  0  ] ;" DRAM not addressed
 [  .X. ,REFRESH,refr1, SVNTY ] -> [  1  ] ;" refresh (-70)
```

```
 [  .X. ,REFRESH,refr2, SVNTY ] -> [  1  ] ;”refresh (–70)
 [  .X. ,REFRESH,refr1,  ATY  ] -> [  1  ] ;”refresh (–80)
 [  .X. ,REFRESH,refr2,  ATY  ] -> [  1  ] ;”refresh (–80)
 [  .X. ,REFRESH,refr1, SIXTY ] -> [  0  ] ;”refresh (–60)
 [  .X. ,REFRESH,refr2, SIXTY ] -> [  0  ] ;”refresh (–60)
 [  .X. ,REFRESH,  1  , .X.   ] -> [  0  ] ;” refresh– not DRAM
 [  .X. ,REFRESH,  3  , .X.   ] -> [  0  ] ;” refresh– not DRAM
 [  .X. , MRS ,  .X.  , .X.   ] -> [  0  ] ;” MRS
 [  .X. , DCAB,  .X.  , .X.   ] -> [  0  ] ;” DCAB

end C80codes1
```

```
module C80codes2
title'
DWG NAME  LOGIC.SCH
PAL #     U25
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      07_22_95'
   xx_001    device 'P22V10C'; " PAL22LV10–7C PLCC
   STAT5     Pin 2;  "C80 STATUS[5]
   STAT4     Pin 3;  "C80 STATUS[4]
   STAT3     Pin 4;  "C80 STATUS[3]
   STAT2     Pin 5;  "C80 STATUS[2]
   STAT1     Pin 6;  "C80 STATUS[1]
   STAT0     Pin 7;  "C80 STATUS[0]
   A31       Pin 9;  "C80 address line 31
   A30       Pin 10; "C80 address line 30
   A29       Pin 11; "C80 address line 29
   A28       Pin 12; "C80 address line 28
   A17       Pin 13; "C80 address line 17
   A16       Pin 16; "C80 address line 16
   Vss       Pin 14; "Ground
   Vcc       Pin 28; "Power
   PDone_a   Pin 27; "presence detect 1A
   PDone_b   Pin 26; "presence detect 1B
   AS2       Pin 25; "C80 AS[2] input (address shift)
   AS1       Pin 24; "C80 AS[1] input (address shift)
   AS0       Pin 23; "C80 AS[0] input (address shift)
   BS1       Pin 21; "C80 BS[1] input (bus size)
   BS0       Pin 20; "C80 BS[0] input (bus size)
   TWO_SIMMS Pin 19;
   PDtwo_a   Pin 18; "presence detect 2A
   PDtwo_b   Pin 17; "presence detect 2B
"constants and alias names
   ADDR   = [A31..A28] ;
   REFADD  = [A17..A16] ;
   STAT   = [STAT5..STAT0];
   PD     = [PDtwo_b,PDtwo_a,PDone_b,PDone_a] ;
   refr1  = 0 ; "DRAM bank refresh pseudo–address
   refr2  = 2 ; "DRAM bank refresh pseudo–address
```

```
   MRS       = 12 ;
   READ      = 0 ;
   WRITE     = 1 ;
   TWOSIMMa  = 10 ;
   TWOSIMMb  = 9 ;
   TWOSIMMc  = 6 ;
   TWOSIMMd  = 5 ;
   ONESIMMa  = 11;
   ONESIMMb  = 7 ;
   DRAM_AV   = A31 & A30 & !A29 & A28 ;
   SDRAM_cyc =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & STAT0)
              #(!STAT5 & !STAT4 & STAT3 & STAT2 & !STAT1 & !STAT0);

equations
   AS2    = (DRAM_AV) & !(SDRAM_cyc) ;
          "#(SYSTEM_SPECIFIC_REQUIRING_AS2=1)

   AS1    = 0 ;
          "#(SYSTEM_SPECIFIC_REQUIRING_AS1=1)

   AS0    = 0 ;
          "#(SYSTEM_SPECIFIC_REQUIRING_AS0=1)
   BS1= (DRAM_AV) & !(SDRAM_cyc) ;
          "#(SYSTEM_SPECIFIC_REQUIRING_BS1=1)
   BS0= (DRAM_AV) & !(SDRAM_cyc) & TWO_SIMMS ;
          "#(SYSTEM_SPECIFIC_REQUIRING_BS1=1)

test_vectors"AS2
([ ADDR  , STAT  ] -> [AS2])
 [  0    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;" DRAM not addressed
```

```
 [  10   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  11   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , .X.   ] -> [ 1 ] ;" DRAM addressed
 [  14   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , MRS   ] -> [ 0 ] ;" DRAM addressed - MRS
test_vectors"AS1
([ ADDR  , STAT  ] -> [AS1])
 [  0    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  11   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" DRAM addressed
 [  14   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , MRS   ] -> [ 0 ] ;" DRAM addressed - MRS
test_vectors"AS0
([ ADDR  , STAT  ] -> [AS0])
 [  0    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ;" DRAM not addressed
```

```
 [  11   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  12  , .X.  ] -> [ 0 ] ;" DRAM not addressed
 [  13  , .X.  ] -> [ 0 ] ;" DRAM addressed
 [  14   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" DRAM not addressed
 [  13   , MRS   ] -> [ 0 ] ;" DRAM addressed - MRS

test_vectors"BS1
([ ADDR  , STAT  ] -> [BS1])
 [  0    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  9    , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  11   , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  13   , READ  ] -> [ 1 ] ; " DRAM addressed
 [  13   , WRITE ] -> [ 1 ] ; " DRAM addressed
 [  13   , MRS   ] -> [ 0 ] ; " DRAM addressed- MRS cycle
 [  14   , .X.   ] -> [ 0 ] ; " DRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ; " DRAM not addressed
test_vectors"BS0
([ ADDR  , STAT,      PD   ] -> [BS0])
 [  0    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  1    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  2    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  3    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  4    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  5    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  6    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  7    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  8    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  9    , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
```

```
 [  10   , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  11   , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  12   , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  13   , .X. , TWOSIMMa  ] -> [ 1 ] ; " DRAM addressed (writes)
 [  13   , .X. , TWOSIMMb  ] -> [ 1 ] ; " DRAM addressed (reads)
 [  13   , .X. , TWOSIMMc  ] -> [ 1 ] ; " DRAM addressed (reads)
 [  13   , .X. , TWOSIMMd  ] -> [ 1 ] ; " DRAM addressed (reads)
 [  13   , .X. , ONESIMMa  ] -> [ 0 ] ; " DRAM addressed (writes)
 [  13   , .X. , ONESIMMb  ] -> [ 0 ] ; " DRAM addressed (writes)
 [  14   , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  15   , .X. ,      .X.  ] -> [ 0 ] ; " DRAM not addressed
 [  .X.  , MRS ,      .X.  ] -> [ 0 ] ; " MRS

end C80codes2
```