Integrated Device Technology, Inc.

DESIGNING A DISCRETE DRAM CONTROLLER FOR THE R3051 RISCONTROLLER™ FAMILY

APPLICATION NOTE AN-90

By Bob Napaa

INTRODUCTION

The IDT R3051™ RISController™ family utilizes a high-performance computing core to achieve high performance across a variety of applications. Further, the amount of cache incorporated in the R3051 family allow these CPUs to achieve very high performance even with simple, low speed, low cost memory sub-systems.

The R3051 RISController CPU family includes a full R3000A core RISC processor, and thus is fully software compatible with the standard MIPS processor. In order to provide high-bandwidth to the CPU core, the family also incorporates on-chip up to 8 kB of instruction cache and 2 kB of data cache. The external memory interface from the R3051 family is very flexible, and allows a wide variety of implementations according to the price / performance goals of the application. For a detailed reference to the system interface of the R3051 family, the reader is advised to refer to the "R3051 Family Hardware User's Manual".

This applications note is a design example on the interface to a non-interleaved DRAM memory sub-system. The goals of this sub-system are to provide a simple, extensible memory interface using off-the-shelf components, and to illustrate basic design techniques for systems using an R3051 family CPU.

GENERAL DESCRIPTION OF THE DRAM SYSTEM

Figure 1 illustrates a typical system based on the R3051 RISController family. The R3051 family uses a double-frequency input clock for its internal operation and provides a nominal frequency reference clock output for the external system. This output clock, $\overline{\text{SysClk}}$,synchronizes the external memory sub-systems to the R3051.

Memory transactions from the R3051 use a single, time multiplexed 32-bit address and data bus and a simple set of control signals. External logic then performs address demultiplexing and decoding, memory control, interface timing, and data path control.

The system shown in Figure 1 runs at 25 Mhz (2x clock = 50 Mhz). The R3051 interfaces to a DRAM system as the main

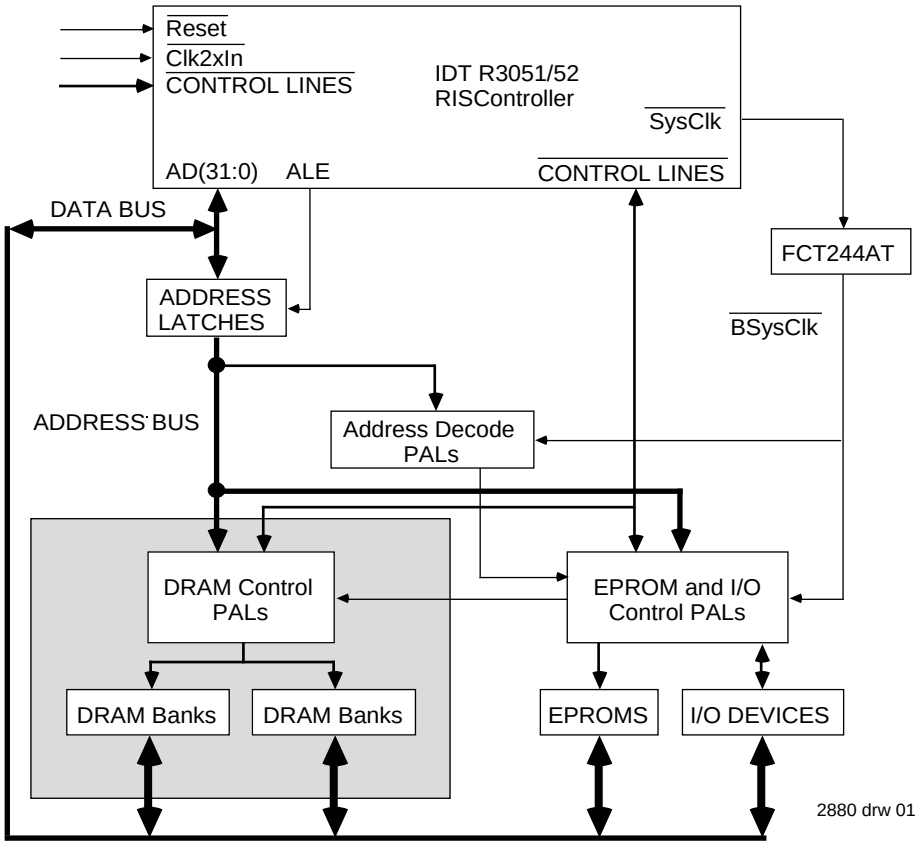


Figure 1. R3051 RISController Family Based System

memory, to an EPROM system and to various I/O devices and controllers. Address latches decouple the address bus from the data bus. Address decoders select among the various external modules. The output clock from the R3051 ($\overline{SysClk}$) is buffered ($\overline{BSysClk}$) to reduce the loading effect and to provide clock drive capability with minimum clock skew for the system. This applications note will focus on the DRAM control and data path sub-system.

The main DRAM memory system is based on 1 to 4 banks of non-interleaved DRAMs with 80 nsec of access time (t_{rac} = 80 nsec). The density of the DRAMs used is 256K x 4 to provide a maximum memory space of 4 Mbytes. The DRAM memory space occupies the lower 4 Mbytes of the physical memory space (A21:A0). Figure 2 illustrates the architecture of the main DRAM memory system.

Table 1 illustrates the decoding scheme used in accessing the DRAM memory space. To simplify address decoding, software will insure that all references to the DRAM memory occur with address bit A(22) low, and thus only that bit will be used in the decoding. Address bits A(21:20) will select among the four banks, and the $\overline{Rd}$ and $\overline{Wr}$ outputs from the R3051 differentiate between read and write accesses.

Each 1MB bank of DRAMs is individually controlled by separate $\overline{RAS}$ and $\overline{CAS}$ control signals. Thus, each bank may be independently selected. The banks are arranged so that each bank represents a single, contiguous range of 1MB (as opposed to an interleaved memory structure).

Data buffers isolate the DRAM banks from the R3051 data bus to reduce the loading effect and to prevent any bus contentions between the R3051 and the DRAMs from occurring. Note that this also alleviates concerns about the relatively slow tri-state times associated with DRAM devices. The data buffers selected are actually bi-directional latching transceivers; the use of a latching transceiver greatly simplified the timing control of the DRAM accesses, as will be described later.

DRAM addresses are provided by multiplexing the latched R3051 address bus, using IDT FBT2827B memory drivers. This device type was chosen based on its ability to drive large capacitive loads, such as that found when driving 32 DRAMs. A single FBT output has sufficient drive to drive all four banks of the DRAM sub-system.

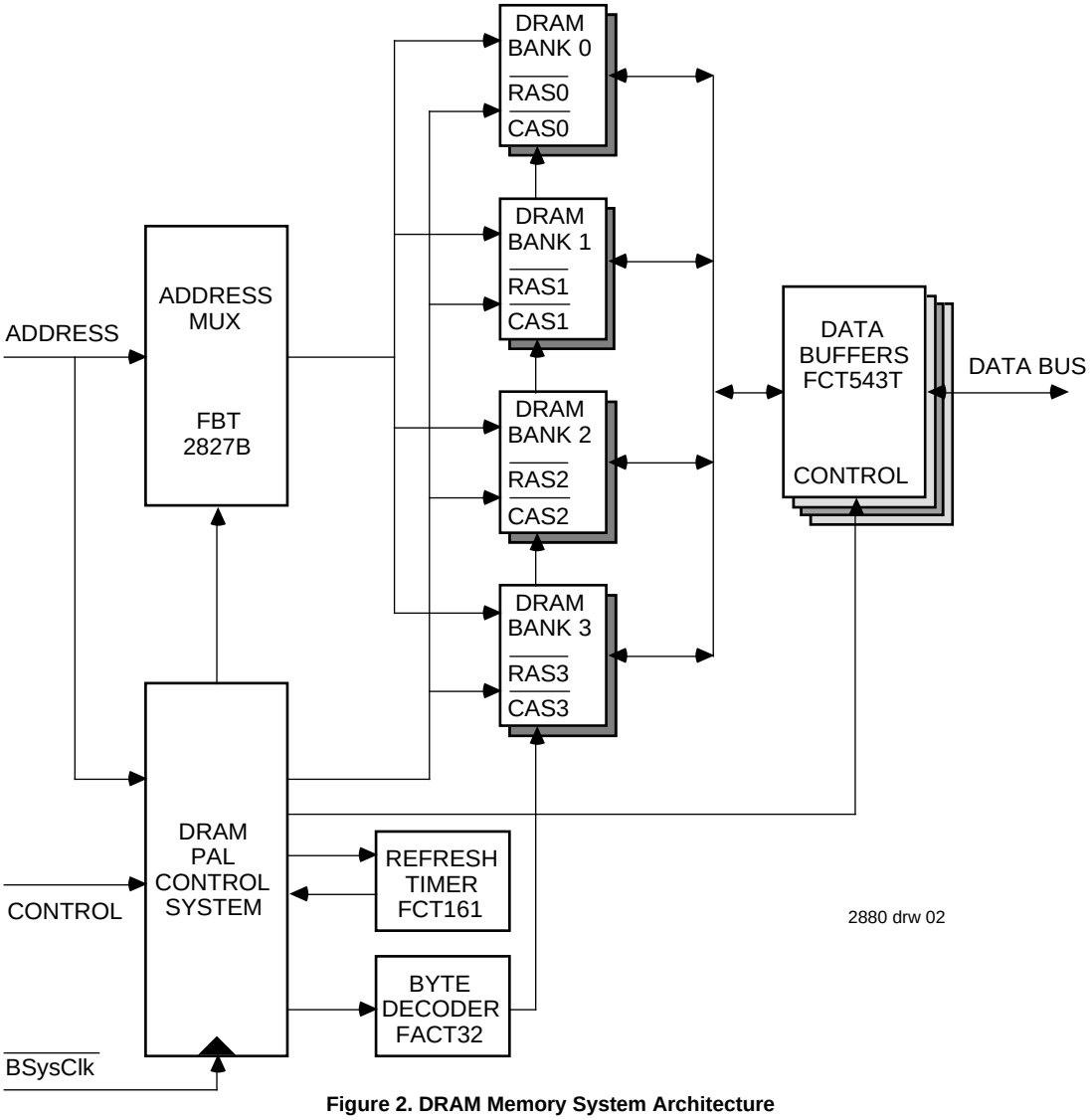


Figure 2. DRAM Memory System Architecture

A22	0	0	0	0	0	0	0	0	1	1	X
A21	0	0	1	0	0	0	1	1	X	X	X
A20	0	1	0	1	0	1	0	0	X	X	X
$\overline{WR}$	1	1	1	1	0	0	0	0	1	0	1
$\overline{RD}$	0	0	0	0	1	1	1	1	0	1	1
SELECTION	**READ BANK 0**	**READ BANK 1**	**READ BANK 2**	**READ BANK 3**	**WRITE BANK 0**	**WRITE BANK 1**	**WRITE BANK 2**	**WRITE BANK 3**	**READ OUTSIDE DRAM SPACE**	**WRITE OUTSIDE DRAM SPACE**	**NO ACCESS**

Table 1. DRAM Memory Space Decoding

In an R3051 system, it is possible to perform a 32-bit read access even when smaller data elements are requested. However, on writes, it is important to enable only those bytes which are actually being written by the CPU. The R3051 bus interface provides four individual byte enables to indicate which byte lanes are involved in a particular transfer. The DRAM sub-system uses a byte decoder (OR gate) to individually select from 1 to 4 bytes for write accesses. Each write byte enable is connected to those DRAMs which reside on that particular byte lane (across the multiple banks)

An 8-bit refresh timer requests the refreshing of the DRAMs every 9.6 µsec. Although this is more frequent than is actually required by the DRAMs, the use of this value simplified the control logic associated with page mode write. DRAMs require that $\overline{RAS}$ be maintained low no longer than 10µsec; by choosing a refresh value smaller than this maximum time, the system is assured that maximum $\overline{RAS}$ low time will not be violated.The operation of the DRAM memory system is synchronized by $\overline{BSysClk}$.

STATE MACHINE IMPLEMENTATION

A simple state machine is used to perform the major aspects of DRAM control. The state machine uses a simple four-bit counter (C(3:0)) to dictate the timing for the DRAM control and CPU response, and is sequenced using $\overline{BSysClk}$. There are nine major states to the state machine, as illustrated in figure 3; these states are dictated by the type of transfer requested and the state the DRAM control logic was left in by the prior transfer. Three PALs are required to implement the entire DRAM control logic.

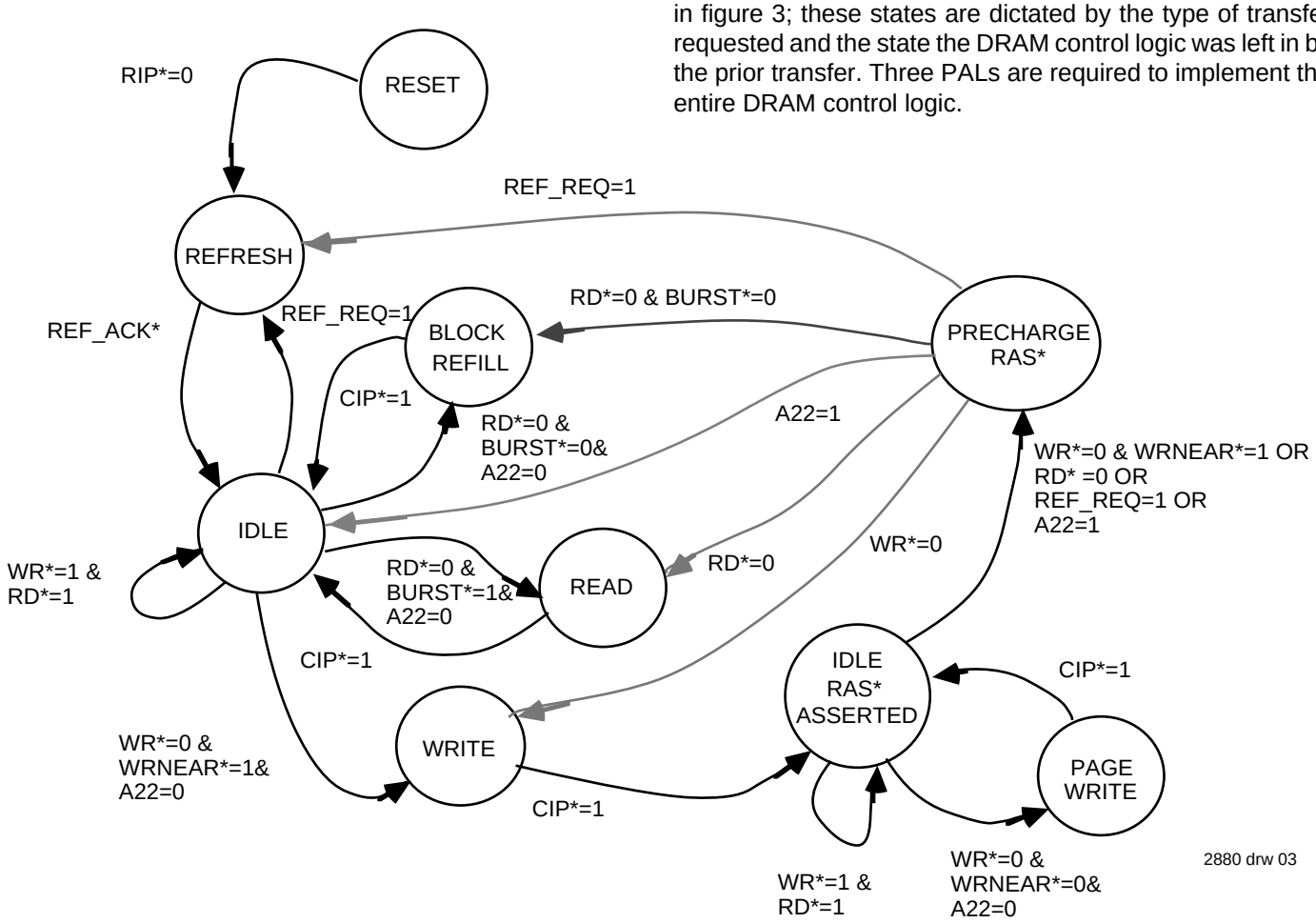


Figure 3. State Machine

The state machine uses the Reset pulse to reset its internal states and to synchronize its operation to the R3051. During the RESET state, it also performs one refresh cycle before entering the IDLE state.

In the IDLE state, the state machine arbitrates between a refresh cycle and a bus access. A DRAM bus access is started whenever Rd or Wr are asserted and A22 is low. A refresh request is detected using the REF_REQ (Refresh_Request) pulse from the refresh timer.

The state machine supports 4 types of bus accesses: “Block refill read”, “Single read”, “Single write” and “Page write”, according to the types of transfers which the R3051 may request.

After a “Single write” or a “Page write” access, the machine enters the IDLE RAS ASSERTED state. This state is very much analogous to the IDLE state, except that the RAS control signal to the DRAMs remains asserted. This state allows subsequent “near” writes to be retired using page mode accesses, which are much quicker than standard accesses. When the IDLE RAS ASSERTED state must be exited (i.e. an action other than near write is requested) the RAS signal must be pre-charged prior to another DRAM transaction.

THE DRAM MEMORY SYSTEM IMPLEMENTATION DETAIL

The DRAM memory system consists of the control system, the address path and the data path as illustrated earlier in Figure 2.

PAL System

The state machine and control PAL system consists of 3 standard speed PALs: Pal 1 (PAL22V10-10), Pal 2 (PAL20R8-10) and Pal 3 (PAL16R8-10). Figure 4 illustrates the control system and the address path. The PAL equations are included in the appendix to this applications note.

Pal 1 is driven by SysClk directly. This allows the CIP line to detect transitions on the Rd and Wr signals from the R3051. Signals generated by Pal 1 include:

- 4 RAS signals (one per DRAM bank)
- The DRAM_ACK and DRAM_RDCEN response signals to the R3051 family CPU.
  These signals are used to provide termination response to the processor.
- The CIP (Cycle_In_Progress) indicates to the rest of the control system that a bus access is being performed.
- The DRAM_WN (DRAM_WrNear) signal indicates that the RAS signals are kept asserted after a “Single write” or a “Page write” access.

Pal 2 is also driven by SysClk directly. Pal 2 generates:

- 4 CAS signals (one per DRAM bank)
- DRAM_LE (DRAM_Latch_Enable), which latches the read data into the data buffers.
- The S (Select) controls the memory drivers selection.
- The T/R (Transmit/Receive) controls the data buffers during read accesses.
- The DRAM_WR (DRAM_Write), used during write accesses.

Pal 3 uses the buffered CIP signal (BCIP) which is delayed with respect to CIP by the buffer propagation delay. This is important to ensure the proper operation of Pal 3, which is driven by the buffered SysClk (BSysClk). Pal 3 generates the master 4-bit counter. It also generates:

- The RIP (Reset_In_Progress), which indicates that a reset cycle is being performed.
- The REF_ACK (Refresh_Acknowledge) signals that a refresh cycle is being performed.
- The GATE_COUNTER controls the operation of the counter when transitioning between bus accesses and refresh accesses.

Refresh Timer

The refresh timer consists of 2 “74FCT161” counters cascaded together as shown in Figure 4. The refresh timer issues a REF_REQ pulse every 9.6 µsec. The refresh timer is loaded with the value b00001111 after each refresh. It is incremented by one for every clock cycle. At value b11111111, it will issue the REF_REQ pulse. This amounts to a total count of 240 which at 25 Mhz reflects a 9.6 µsec refresh period.

The refresh period is set to be shorter than the maximum 15.5 µsec refresh period that most DRAM require. The refresh interval has been set to 9.6 µsec in order not to violate the RAS maximum pulse width of 10 µsec (t_{ras} = 10 µsec max). In an IDLE RAS ASSERTED state, the RAS signals are left asserted while the CAS signals are de-asserted.

Byte Decoding

The byte decoding uses a “74FACT32” OR gate to OR the BE signals from the R3051 with the DRAM_WR signal to produce the write-byte signals WB(3:0). The DRAM_WR signal ensures that the WB(3:0) are only asserted during DRAM write accesses and that the WB(3:0) meet the “write command hold time” (t_{wch} = 20 nsec) of the DRAMs. It also ensure that the WB(3:0) are asserted before the CAS signals for “Early Write” accesses. Every WB signal enables one byte of the DRAM banks and of the data buffers during write accesses to allow for partial word write operations. The WB(3:0) are always issued one clock cycle before the CAS signals are asserted, in order to meet the timing requirements for a DRAM “Early Write” cycle.

Address Path

The DRAM address path consists of 2 “74FBT2827B” memory drivers to multiplex the row and column address of the DRAMs. The “FBT2827” have a 25 Ω series resistance incorporated in the output buffers and are used to drive multiple memory banks with large capacitive loading. The S bit from Pal 2 selects between the row address and the column address that drive all the DRAM banks. Figure 4 illustrates the address path architecture. The address to the DRAMs is always set one clock cycle before the assertion of either the RAS or the CAS signals, in order to guarantee proper address set-up time to the DRAMs.

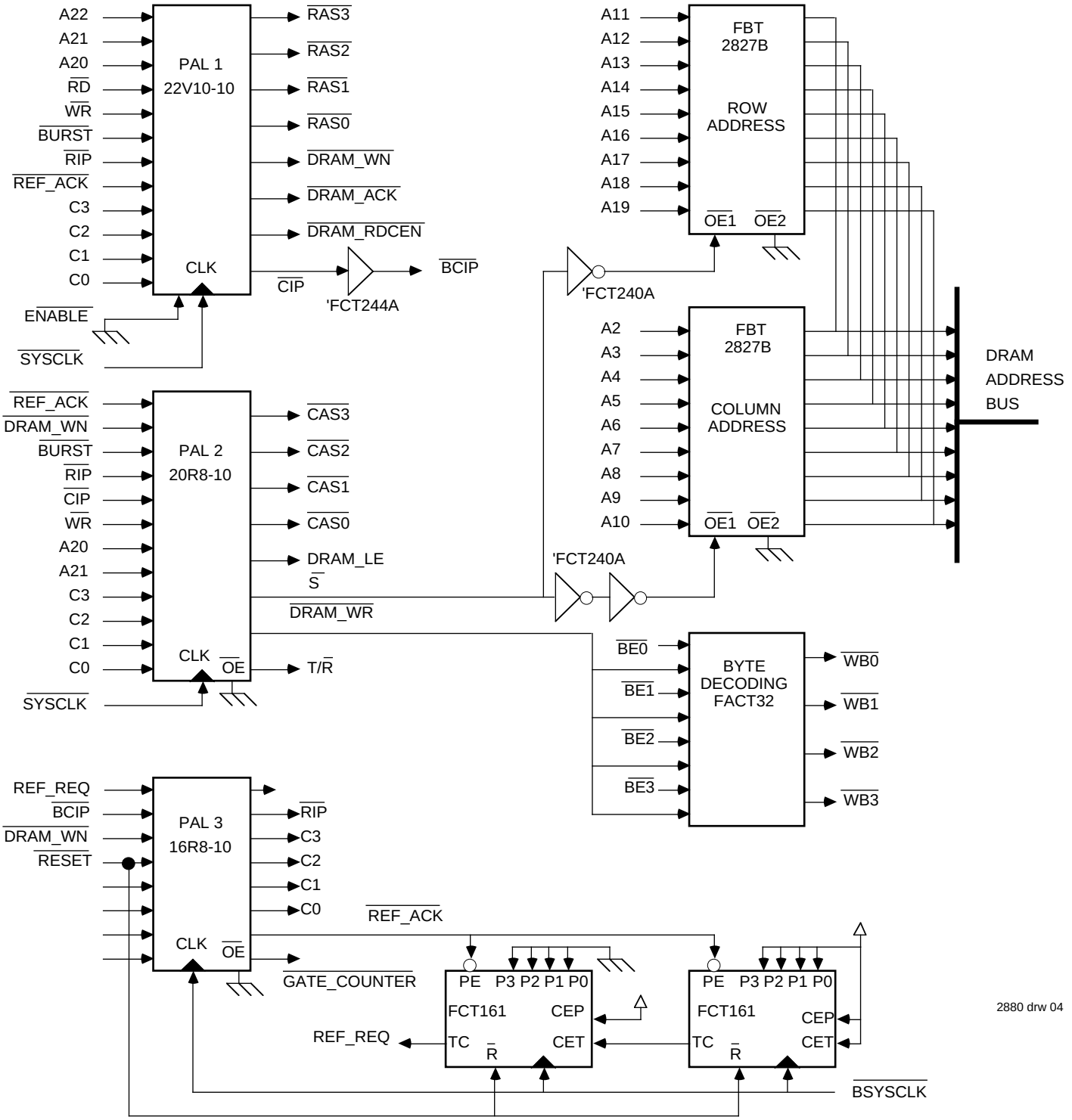


Figure 4. Control System and Address Path

Data Path

The data path consists of the DRAM banks and 4 74FCT543 latched transceivers. Figure 5 illustrates the architecture of the data path and of the data buffers. Latching transceivers are used to allow more access time to the DRAMs; the data is captured by the latches one-half cycle before they are needed by the CPU. During this half-cycle, the data propagates through the buffer; if traditional buffering transceivers had been used, the buffer propagation delay would have occurred at the expense of the DRAM access time.

Up to four banks of DRAMs are used, with each bank having its own set of $\overline{RAS}$ and $\overline{CAS}$ signals to minimize the loading impact of multiple DRAM devices. Address bits A21 and A20 determine the bank selection.

The latched transceivers serve three roles in the DRAM sub-system: they isolate the DRAMs from the A/D bus of the

R3051 to minimize loading; they latch the data from the DRAMs on reads to allow a better timing model; and they are used to prevent bus contention from occurring at the end of a read (as the processor begins another transaction). The R3051 is connected to the A bus of the transceivers, and the DRAM system is connected to the B bus.

In a processor write access, the R3051 drives both the address and the data. In this case the latches are left transparent to pass the processor data through directly to the DRAMs. Only those transceivers whose byte lanes are involved in the write are output enabled, since only those DRAMs will be written into. DRAMs not accessed in this write will output the current contents of their memory at that location, since the $\overline{OE}$ of the DRAMs is asserted. $\overline{DRAM\_WR}$ controls the $\overline{LEAB}$, leaving the latch transparent throughout the write. $\overline{WB}$(3:0) controls the $\overline{OEAB}$ of the latches, thus enabling only those bytes that are written.

In a processor read access, the DRAM system drives the data bus. The DRAM system is synchronized to the rising edge of $\overline{BSysClk}$, and the R3051 samples the input data on the falling edge of $\overline{SysClk}$ before terminating the access. Thus, the DRAM control design, which drives the $\overline{RAS}$ and $\overline{CAS}$ signals on the rising edge of $\overline{SysClk}$, actually removes $\overline{CAS}$ one-half cycle before the data is sampled by the CPU. Thus, data output by the DRAMs is actually latched by the transceivers, and remains valid when the CPU samples the A/D bus one-half clock cycle later.

The DRAM_LE from the DRAM controller is connected to the $\overline{LEBA}$ pin, which latches the data into the transceivers. The T/$\overline{R}$ signal connected to the $\overline{CEBA}$ pin, which controls the direction of the bi-directional transceiver. The $\overline{DataEn}$ signal from the R3051 is connected directly to the $\overline{OEBA}$ pin to control the timing of the output enable onto the A/D bus. This ensures that the output buffers are tri-stated before the next R3051 access starts and prevents any bus contention.

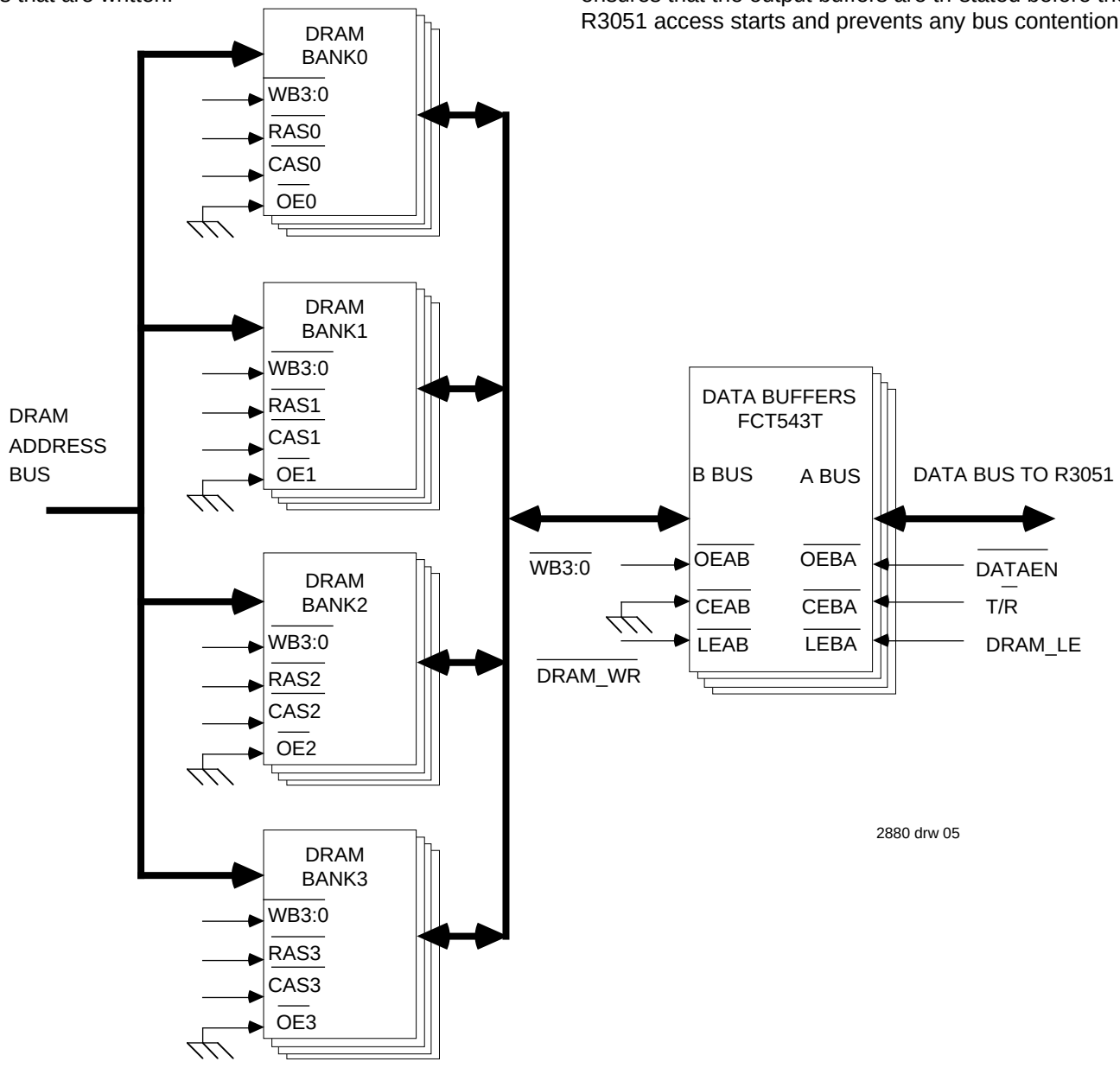


Figure 5. DRAM Banks and Data Buffers

THE DRAM MEMORY SYSTEM TIMING

The R3051 system interface allows this DRAM interface to be simply constructed. Features of the R3051 which are used in this DRAM system include:

- On-chip four-deep read and a four-deep write buffers. These buffers decouple the system interface speed from the speed of the execution engine on-chip.
- Single word reads and four-word refills. Block refills amortize the relatively long latency of DRAMs over multiple words, taking advantage of high-bandwidth capabilities (e.g. Page Mode) offered by DRAMs.
- The $\overline{\text{WrNear}}$ signal, which informs the external DRAM sub-system that two consecutive writes have the same upper 22 address bits (equivalent to a local page of 256 words), and can be written using a Page Mode access.

For the system running at 25 Mhz, the clock period is 40 nsec. DRAMs with 80 nsec of access time require 160 nsec (t_{rc} =160 nsec) to complete one read access (as per DRAM data sheet). A 5 clock cycles (200 nsec) read access time allows an acceptable margin for address decoding, control signal propagation, and bus interface.

For a 4 word block refill read, the initial latency (time to read the first word) is the same as for a single word read access (200 nsec). For the next 3 consecutive words, the DRAM memory system provides a word every 2 clock cycles (every 80 nsec). A block refill access can be completed in 11 clock cycles (440 nsec), which is an average of 110 nsec per word. Thus, block refill, with this simple scheme, provides a significant improvement in the average access time per word (over 2 clock cycles per word savings).

The state machine to manage write operations takes advantage of two features of the R3051:

- On a write cycle, the write data from the processor is held one full clock cycle after the clock edge where the processor samples its ACK input. Thus, the DRAM system can give an early acknowledge, and still rely on the CPU to continue driving data.
- The $\overline{\text{WrNear}}$ output from the CPU, which indicates that this write may be retired using a page mode write. This reduces the number of cycles required to perform write-intensive operations, such as building the program stack or flushing the write buffer.

The state machine for single word writes is optimized to allow subsequent near writes to be retired using page mode accesses. The DRAM memory system takes advantage of the $\overline{\text{WrNear}}$ signal from the R3051 by defaulting to the case that any single write to the DRAM system will be followed by another write with the same upper 22 address bits (within the local page of 256 words). Given this assumption, the $\overline{\text{RAS}}$ signals must be kept asserted after every write access to remain in the page mode of the DRAMs.

Thus, an initial single write can be performed in 4 clock cycles (160 nsec) since the $\overline{\text{RAS}}$ signals are not de-asserted and the $\overline{\text{RAS}}$ precharge time (t_{rp} = 70 nsec) will be deferred until the end of the page write mode. Note that this is faster than a single read; the state machine takes advantage of the fact that the processor will drive data a full clock cycle after acknowledge is given.

A consecutive write to the same DRAM page can be performed in 3 clock cycles (120 nsec) since the $\overline{\text{RAS}}$ signal is already asserted and doesn't need to be precharged. When this state is exited (when a write outside of page or a different type of access occurs) the $\overline{\text{RAS}}$ signal needs to be precharged for 2 clock cycles (80 nsec) before responding to the pending access.

Single Write Cycle/Page Write Cycle

Figure 6 illustrates the timing diagrams for a single write access followed by a page write. The R3051 initiates a single DRAM write access by the assertion of $\overline{\text{Wr}}$ and with A22 low. Since the state machine is in the IDLE state, $\overline{\text{RAS}}$ is deasserted and the ROW addresses are flowing through the address multiplexer. The $\overline{\text{CIP}}$ is issued on the next clock edge to inform the rest of the machine that the write is being processed, thus preventing the commitment of any other state (e.g. refresh). The appropriate $\overline{\text{RAS}}$ signal is issued on the same edge as the $\overline{\text{CIP}}$. The $\overline{\text{DRAM\_ACK}}$ is issued on the following edge and the $\overline{\text{CAS}}$ signal on the 4th edge to terminate the write access. At the end of the access, the $\overline{\text{CIP}}$ is removed while the $\overline{\text{RAS}}$ signal is kept asserted in anticipation of a consecutive write access within the same page. At the end of an initial write access, the $\overline{\text{DRAM\_WN}}$ signal remains asserted. This signal informs the rest of the state machine that the $\overline{\text{RAS}}$ signals are kept asserted.

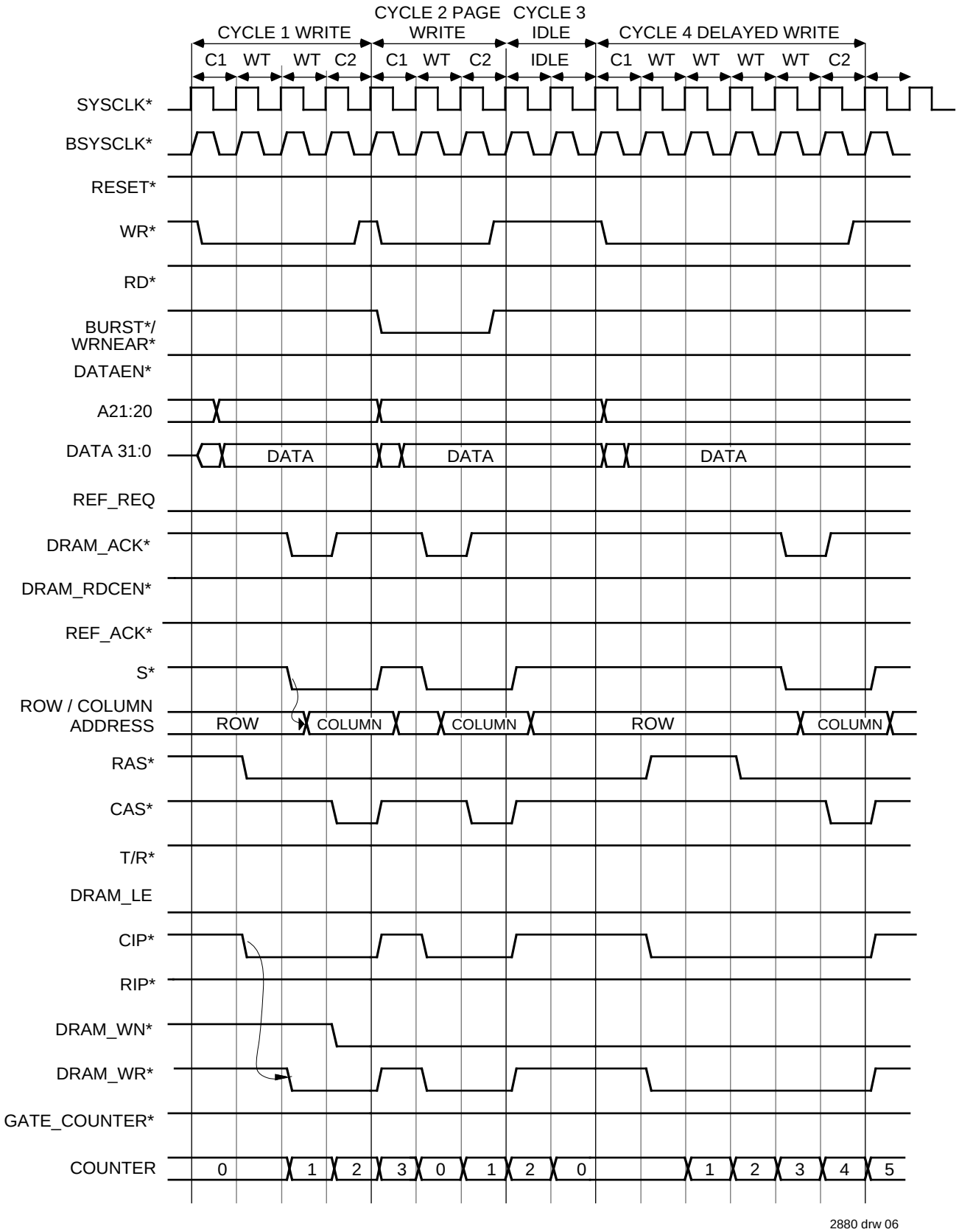
Idle, $\overline{\text{RAS}}$ Asserted State

At the end of a write access the state machine enters this state where a $\overline{\text{RAS}}$ signal is kept asserted while the state machine awaits a subsequent transaction. If the next access is a local write ($\overline{\text{WrNear}}$ from the R3051 is asserted) the state machine enters the page write mode. If a different access type occurs (read, block refill, not local write) or a refresh is pending, the state machine exits this state.

Upon exiting this state, the machine precharges the $\overline{\text{RAS}}$ signal before responding to the pending access. For the ease of discussion, any access that requires the $\overline{\text{RAS}}$ signals to be precharged before the access is processed will be referred to as "delayed" access. If an access outside the DRAM space is detected ($\overline{\text{Wr}}$ or $\overline{\text{Rd}}$ asserted while A22=1) the $\overline{\text{RAS}}$ signals are immediately de-asserted and the machine goes into the IDLE state. This is an important condition; an intervening write to another memory location causes the R3051 to report subsequent writes as "near" to that other memory location, and thus the DRAM controller should not process these writes as near writes.

Page Write Cycle

A page write cycle is a write access to the DRAM following another write with the same upper 22 address bits. Figure 6 illustrates the timing diagram for a page write access. The R3051 initiates a page write cycle by the assertion of $\overline{\text{Wr}}$, $\overline{\text{WrNear}}$ and A22 = 0. On the following clock edge $\overline{\text{CIP}}$ and $\overline{\text{DRAM\_ACK}}$ are issued, and on the 3rd clock edge $\overline{\text{CAS}}$ is asserted. and the access is terminated ($\overline{\text{CIP}}$ is negated). The $\overline{\text{RAS}}$ and $\overline{\text{DRAM\_WN}}$ signals are kept asserted, allowing



2880 drw 06

Figure 6. Single Write, Page Write and Delayed Write Timing Diagrams

subsequent page writes to be rapidly processed. The state machine exits this state into the IDLE RAS ASSERTED state to await subsequent page mode writes.

Delayed Write Cycle

The delayed write cycle has exactly the same sequence as a single write but is delayed by two clock cycles. A delayed write is a "non-near" write detected in the IDLE RAS ASSERTED state. Figure 6 illustrates the timing diagrams for a delayed write access.

The R3051 initiates a delayed write access by the assertion of Wr and A22 = 0 while RAS and DRAM_WN are asserted. On the next clock edge RAS is de-asserted while the DRAM_WN is kept asserted. The precharging of the RAS signal takes two clock cycles. The DRAM_WN signal is kept asserted to inform the state machine that the control signals for this access have to be delayed by two clock cycles. This is true for all the delayed accesses.

Single Read Cycle

A single read cycle is a read access to the DRAM following an IDLE state in which the RAS and the DRAM_WN are not asserted. Figure 7 illustrates the timing diagrams for a read access. The R3051 initiates a single read access by the assertion of Rd with A22 low while the state machine is IDLE and all RAS outputs are de-asserted. The CIP is issued on the next clock edge to inform the rest of the machine that a cycle is ongoing, thus preventing the commitment of any other state. The appropriate RAS signal is issued on the same edge as the CIP. Two clock cycles later, the CAS, DRAM_RDCEN and the DRAM_ACK are issued to terminate the cycle.

For a read access both the DRAM_ACK and the DRAM_RDCEN are required to end the cycle. The processor will not actually sample RdCEn until one-clock after the clock edge used to generate DRAM_RDCEN, and thus will not sample the data until one and one-half clock cycles after the edge used to generate DRAM_RDCEN. From the timing diagrams it is clear that the CAS and the RAS signals are removed half a clock cycle before the falling edge of the clock when the R3051 samples the data. DRAM_LE latches the DRAM data into the transceivers and holds it for one clock cycle. At the end of the access the CIP is removed.

Delayed Read Cycle

The timings of a delayed read are exactly the same as for a single read but shifted by two clock cycles to accommodate RAS pre-charge time. A delayed read cycle is a read access to the DRAM following an IDLE RAS ASSERTED state in which the RAS and the DRAM_WN are still asserted. Figure 8 illustrates the timing diagrams for a delayed read access. Once a read access is detected, the RAS signal is de-asserted while the DRAM_WN is kept asserted. The RAS signal is precharged for two clock cycles. At the end of a delayed read, the DRAM_WN and the CIP are removed and the machine enters the IDLE state.

Block Refill Cycle

A block refill cycle is a 4 word read access to the DRAM following an IDLE state. Figure 7 illustrates the timing diagrams for 4-word block refill access. The R3051 indicates a block refill read access by the assertion of Rd and Burst with A22 low. The DRAM control sub-system handles block refill accesses using the Throttled Block Refill mode of the R3051. In a throttled read, RdCEn is used to control the data rate of memory back to the CPU. The Ack input is not provided back to the processor until the transfer has sufficiently progressed such that the last word of the transfer is clocked into the on-chip read buffer before the processor core requires it.

In the block refill access the first word read takes the same time as a single read while the 3 subsequent words are read into the read buffer at the rate of 1 word every two clock cycles. The DRAM_RDCEN is issued with every word being read to cause the R3051 to latch the data into the read buffer. The DRAM_ACK is issued between the second and the third word read. This ensures that for 4 subsequent falling edges of SysClk the read buffer can provide data to the R3000A core at the rate of a word every clock cycle.

Block refill uses the page mode characteristics of the DRAM to obtain subsequent words at a high data rate. In this access, the RAS signal is kept asserted while the CAS signal is toggled 4 times to produce 4 data words. Every word from the DRAM system is latched into the transceivers as for a single read operation, using the DRAM_LE to clock the latched transceivers. At the end of the access RAS and CIP are de-asserted, and the state machine returns to the IDLE state.

In the block refill access, address lines Addr(3:2) from the R3051 act as a two-bit counter to provide the address of 4 consecutive words. These two lines are incremented on the falling edge of SysClk. This timing could prove critical at high-frequencies: this is only half a clock margin (20 nsec) before the CAS signals are asserted, in which address set-up time to CAS must be provided. These two lines are part of the address path and are driving large capacitive loads. Two minimize additional delay due to loading, two sets or more of memory address drivers could then be used to minimize the effect of the capacitive loads and to ensure proper operation.

Delayed Block Refill Cycle

A delayed block refill cycle is a block refill access to the DRAM following an IDLE RAS ASSERTED state in which the RAS and the DRAM_WN are asserted. Figure 9 illustrates the timing diagrams for a delayed block refill access. A delayed block refill is exactly the same as a block refill with the exception that the access is shifted by two clock cycles to accommodate RAS precharge requirements. The DRAM_WN signals to the machine that the access has a delayed timing. At the end of the access, the DRAM_WN and the CIP are de-asserted.

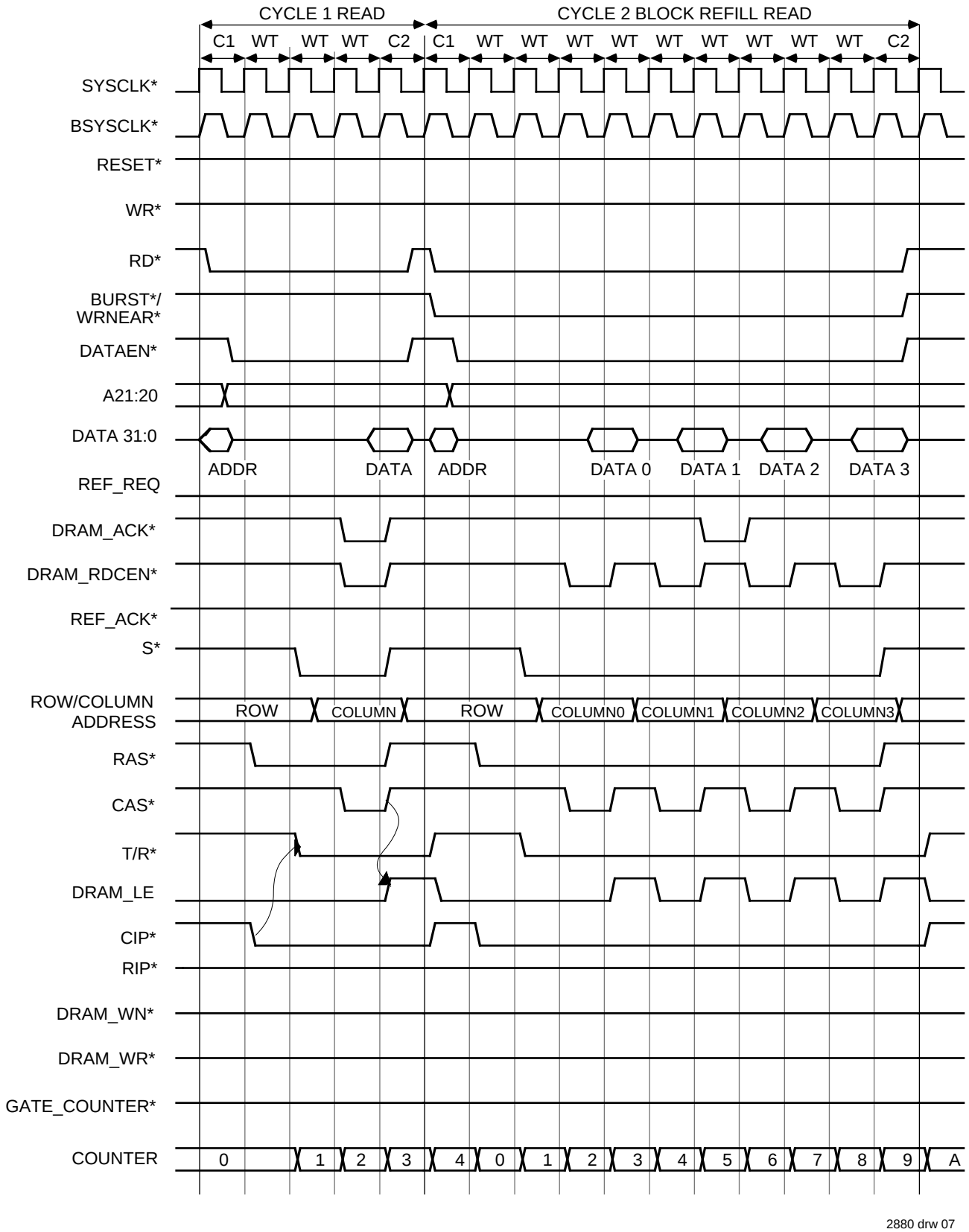


Figure 7. Single Read and Block Refill Read Timing Diagrams

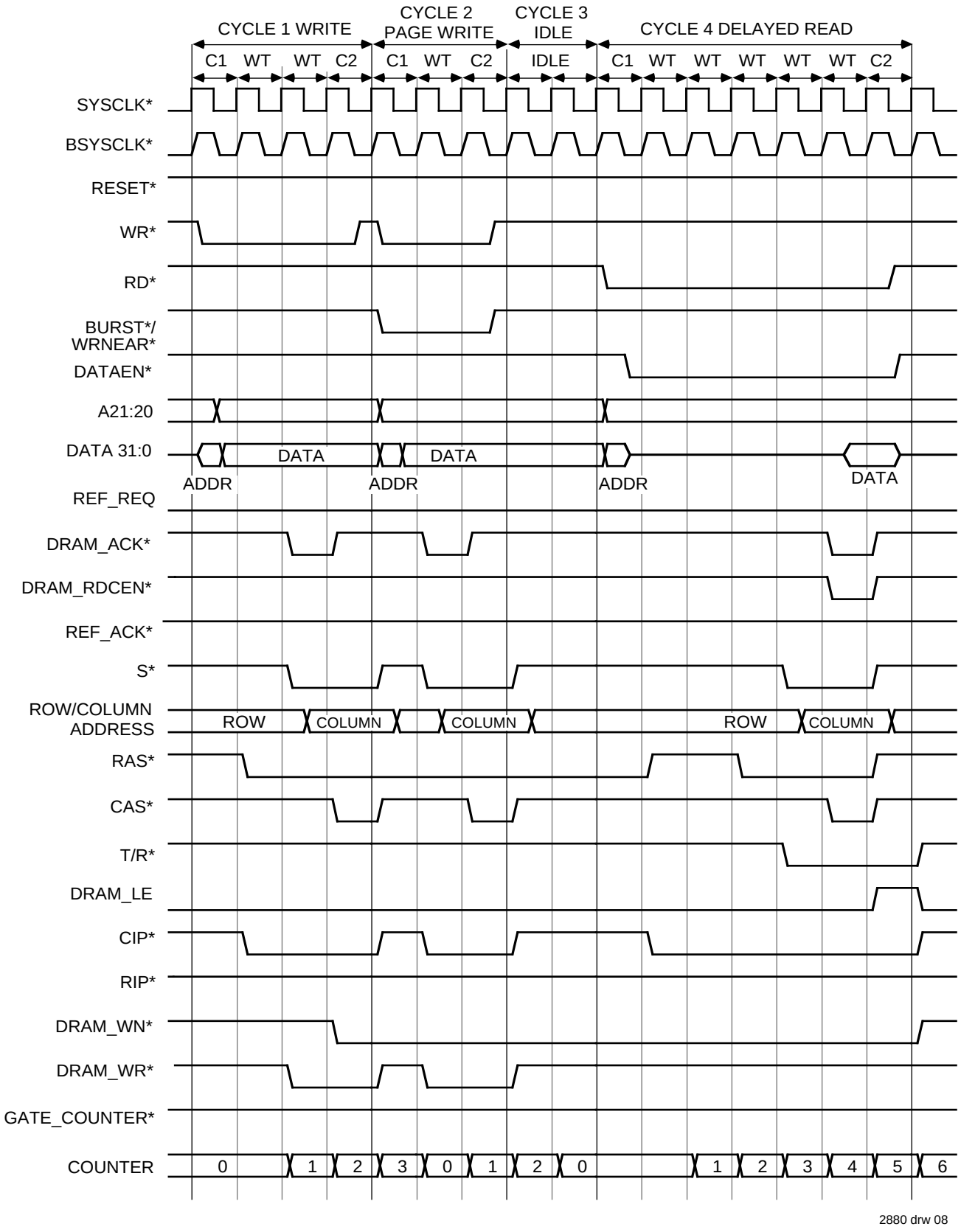


Figure 8. Delayed Single Read Timing Diagrams

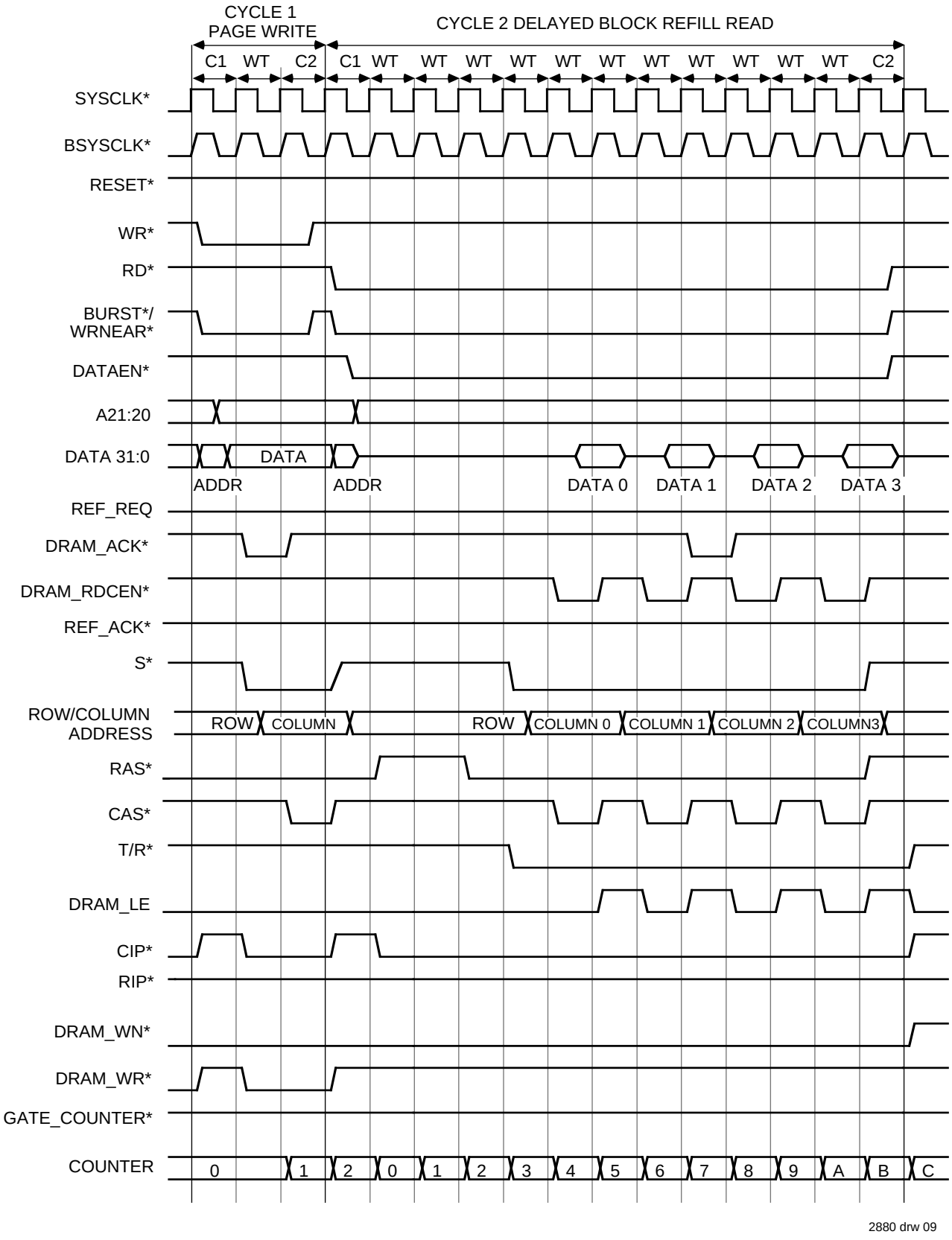


Figure 9. Delayed Block Refill Read Timing Diagrams

Refresh Cycle

A refresh cycle is initiated every time a REF_REQ pulse is detected. The state machine responds immediately by asserting the $\overline{\text{REF\_ACK}}$ signal on the following clock edge. This disables the refresh timer until the refresh access is completed. Figure 10 illustrates the timing diagrams for a refresh arbitration and the actual refresh access.

If a REF_REQ occurs during an access or at the same time as an access, the refresh is delayed until the access is terminated (signaled by $\overline{\text{CIP}}$ de-asserted). Asserting $\overline{\text{REF\_ACK}}$ at the detection of REF_REQ ensures that the following access will be a refresh access and prevents the commitment of any other state. Delaying a refresh request until the end of a bus access doesn't affect the DRAM operation, since the refresh period selected is much less than the maximum refresh period of a DRAM row. The refresh period is every 9.6 μsec and the longest access is the delayed block refill with 14 clock cycles (until $\overline{\text{CIP}}$ is removed) which is 0.56 μsec. Thus, the refresh will be serviced at a maximum of 10.16 μsec, which is substantially below the maximum 15.5 μsec refresh requirement of the DRAMs. By the same reasoning, if the granted access is a delayed access, the $\overline{\text{RAS}}$ signal will be precharged prior to the 10 μsec $\overline{\text{RAS}}$ pulse width maximum requirements. If a Page Mode Write is granted, it will be retired in 3 cycles, or .12μsec, and thus $\overline{\text{RAS}}$ will be precharged for the refresh no longer than 9.72μsec after it was asserted.

The refresh access is a $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh in which all four $\overline{\text{CAS}}$ and $\overline{\text{RAS}}$ signals are issued. The $\overline{\text{CAS}}$ signal is issued 1 clock cycle before the $\overline{\text{RAS}}$ signal. A refresh access takes 10 clock cycles. This time is long enough to allow the $\overline{\text{RAS}}$ signals to be precharged if needed (delayed refresh). A delayed refresh has then the same timing as a refresh access.

Figure 11 shows the timing diagrams for the delayed refresh cycles. $\overline{\text{GATE\_COUNTER}}$ controls the operation of the 4-bit counter when transitioning between bus accesses and refresh accesses. It is mainly used in the arbitration phase when a bus access and refresh access are requested at the same time.

Reset Cycle

A reset cycle is initiated by the assertion of the $\overline{\text{Reset}}$ signal. This is a hardware reset and is used to initialize the PALs to the correct IDLE state. The $\overline{\text{RIP}}$ signal is asserted on the following clock edge to inform the machine that a reset cycle is in progress. After the $\overline{\text{Reset}}$ signal is de-asserted, the $\overline{\text{RIP}}$ stays asserted and one refresh access is initiated. At the end of this refresh access, the $\overline{\text{RIP}}$ is removed and the state machine enters the IDLE state. Figure 12 illustrates the timing diagrams of the reset operation.

Most DRAMs require at least 8 $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ refresh accesses prior to a regular access, to insure proper initialization. The actual state machine provides only one refresh access. It is the responsibility of the software to ensure that no DRAM access is made prior to the elapsing of 8 refresh periods from the refresh timer. This can typically be insured by normal operation of the boot PROM; however, software could "spin-lock" for a pre-determined number of loops to insure that sufficient time has elapsed.

Idle State

The IDLE state is the state in which the machine is not performing any bus access or a refresh access but is constantly monitoring the bus for any access request. All the signals are de-asserted and the 4-bit counter operation is halted.

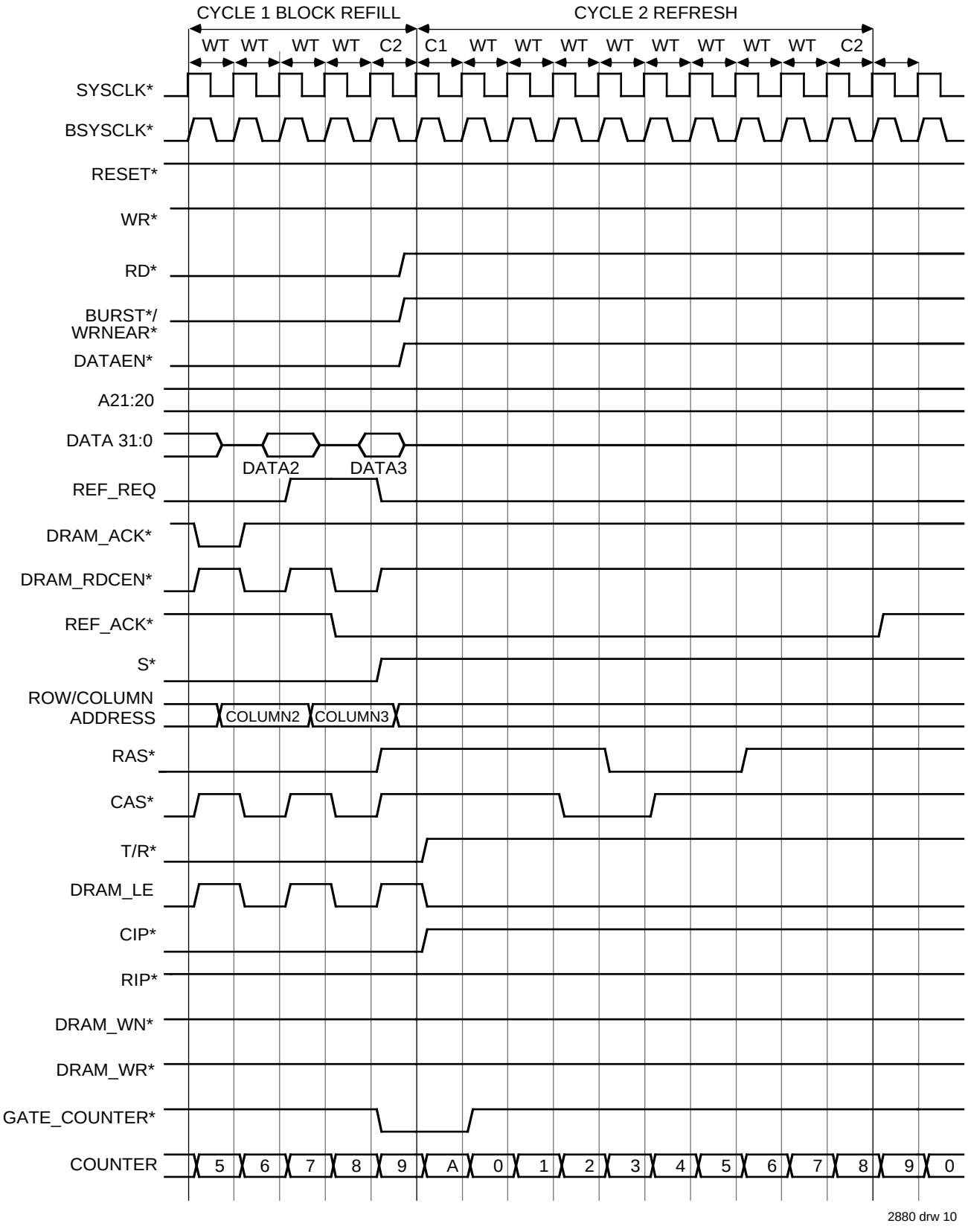
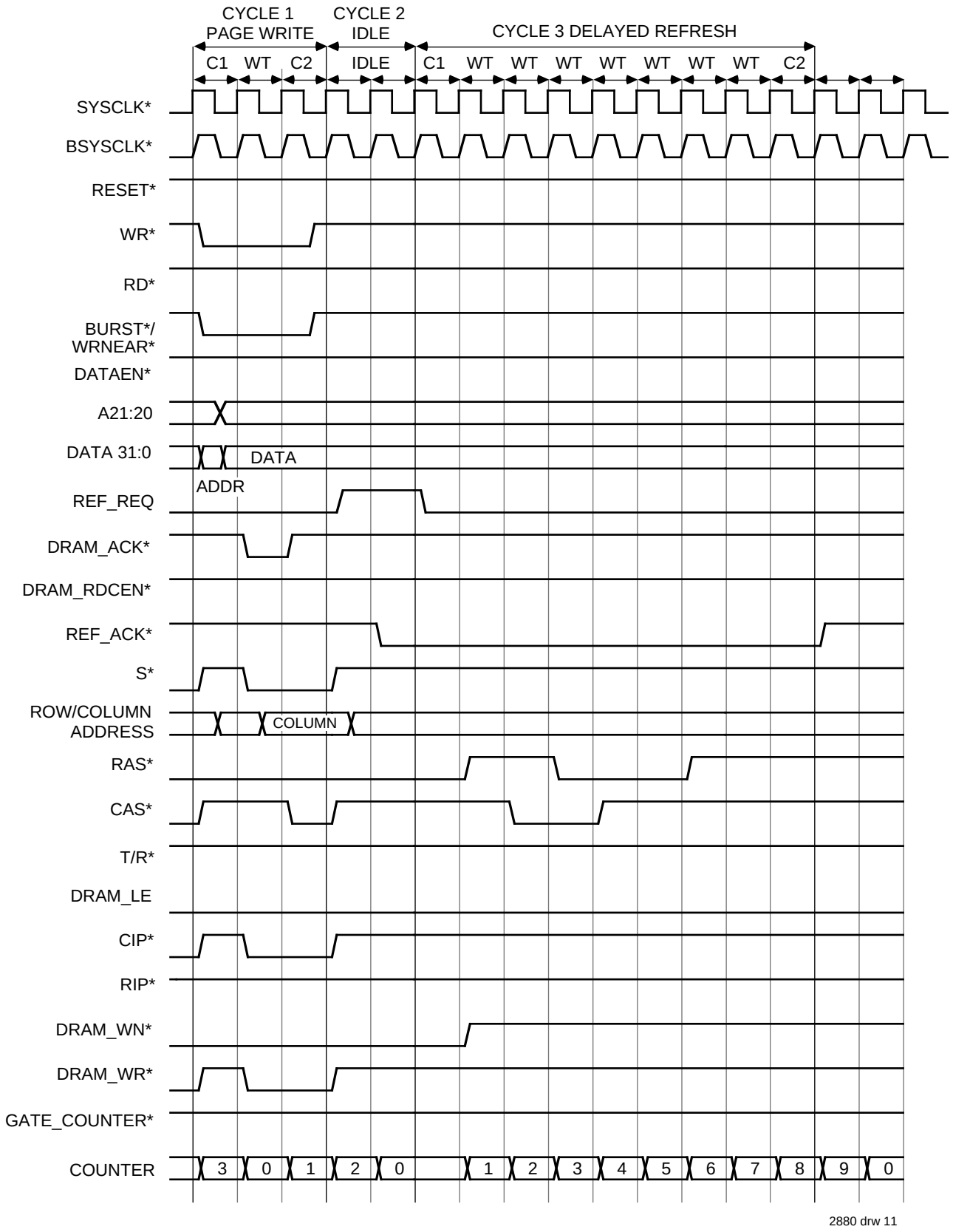


Figure 10. Refresh Arbitration and Refresh Timing Diagrams



2880 drw 11

Figure 11. Delayed Refresh Timing Diagrams

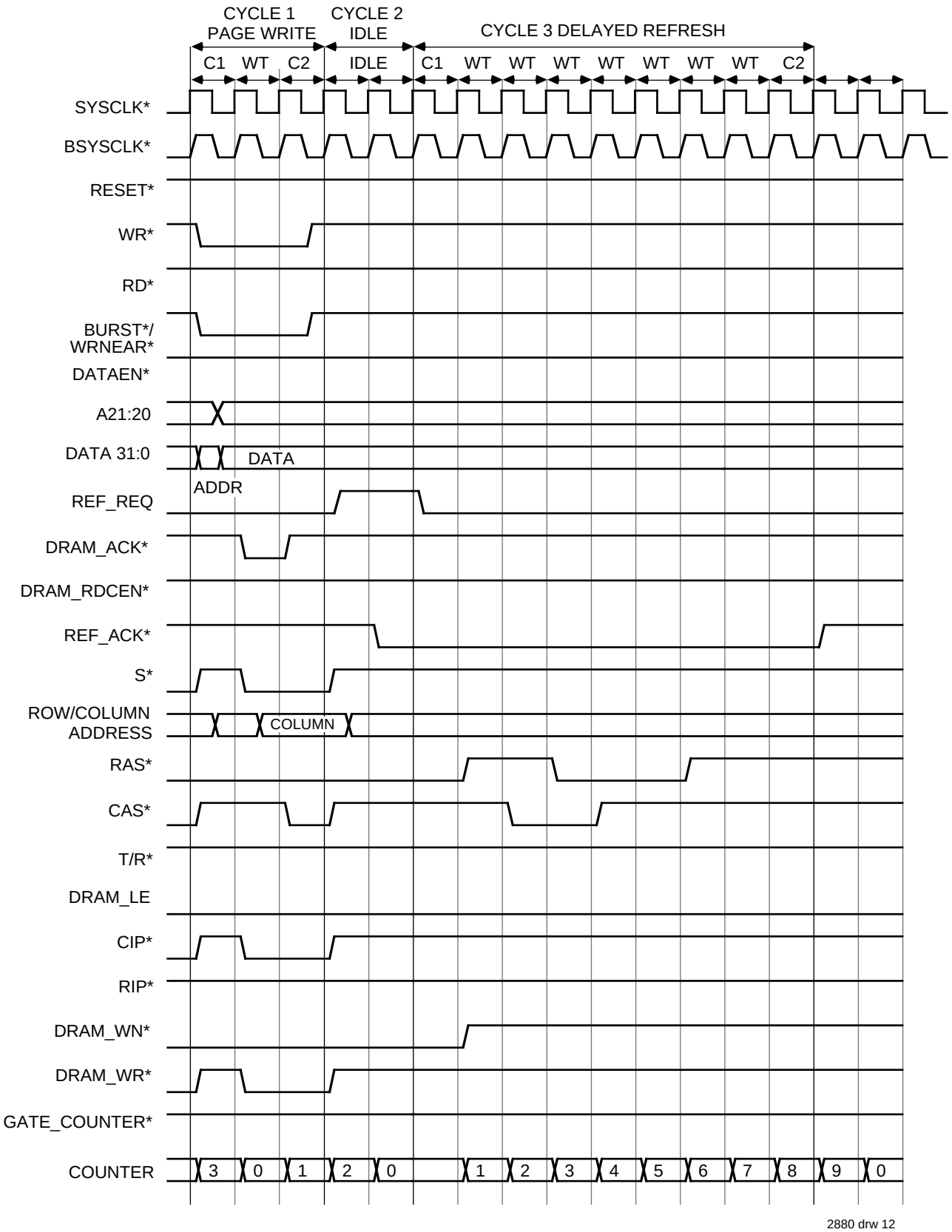


Figure 12. Reset Timing Diagrams

CRITICAL TIMING CALCULATIONS

The following is a timing analysis of some of the critical paths in the DRAM system.

DRAM Data for a Read or block Refill access

As illustrated in all the timing diagrams, the $\overline{CAS}$ signal is asserted for only 1 clock cycle for a read or a write access. For a write access there is no critical timing since the DRAM latches the data in at the $\overline{CAS}$ leading edge, and the processor insures sufficient data hold time by holding data for one cycle after $\overline{ACK}$ is detected.

For a read or a block refill access the DRAMs provide the data to the R3051 and the maximum delays must be considered. Figure 13 illustrates the detailed timing for a portion of a block refill access which is also true for a read access. The R3051 uses the $\overline{SysClk}$ for its reference with a period Tclk of 40 nsec. The $\overline{CAS}$ and the DRAM_LE signals are delayed with respect to $\overline{SysClk}$ by the Pal 2 propagation delay T1. The data is available from the DRAM after T_2 (t_{cac} = 25 nsec max). The critical path requires that the DRAM data be available and meet the setup time of the transceivers before the DRAM_LE is asserted. The timing calculation for this data path is as follows:

T_{clk}	= 40.0 nsec	
- $T_{1\ max}$	= 8.0	
	= 32.0	
- $T_{2\ max}$	= 25.0	
	= 7.0	
- T_{setup}	= 3.0	FCT543T data setup time.
	= 4.0	

The available margin is 4.0 nsec. Some 80 nsec DRAMs have T_2 (t_{cac} = 20 nsec) which could offer more margin.

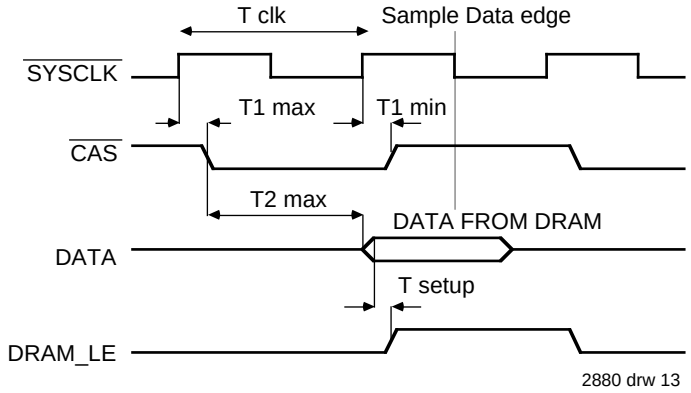


Figure 13. Read or Block Refill Access

Transceivers Turn Off time

For a read or a block refill access, the DRAMs provide the data to the R3051 through the latched transceivers. As illustrated in Figure 7, the R3051 reads the data from the bus half a clock cycle before it starts a new access in which it can drive address on the bus. This information is explained in detail in the R3051 User Manual.

The critical path requires that the transceivers be tri-stated before the R3051 starts driving the bus in the next clock cycle. The $\overline{DataEn}$ signal directly from the R3051 enables the B to A output buffers of the transceivers (FCT543T). The $\overline{DataEn}$ is delayed by T3 from the falling edge of $\overline{SysClk}$ at which the R3051 samples the data (as per R3051 data sheet). The transceivers disable the output buffers within T4. Figure 14 illustrates the timing for this path.

T_{clk} / 2	= 20.0 nsec
- $T_{3\ max}$	= 6.0
	= 14.0
- $T_{4\ max}$	= 9.0
T_{margin}	= 5.0 nsec

This margin of 5 nsec is long enough to accommodate for any $\overline{SysClk}$ skews.

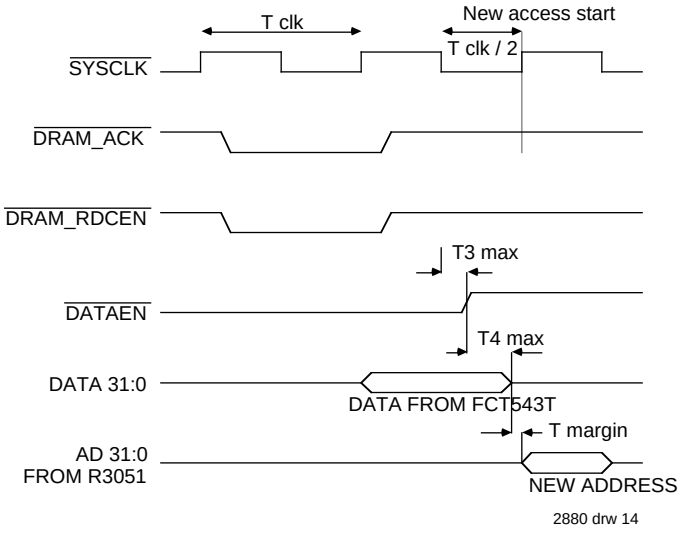


Figure 14. Termination of a Read or Block Refill Access

$\overline{\text{DRAM\_ACK}}$ and $\overline{\text{DRAM\_RDCEN}}$ Timings

The $\overline{\text{DRAM\_ACK}}$ and the $\overline{\text{DRAM\_RDCEN}}$ are issued for one clock cycle only as illustrated in the timing diagrams. They are removed by the clock edge which the R3051 uses to sample them. The R3051 requires that these two signals be held constant for a minimum of 4 nsec after the clock edge. These two signals are usually combined with similar signals from other memory sub-systems (e.g. EPROM) to form one set that is routed to the R3051. This extra delay, plus the Pal 1 minimum propagation delay are long enough to meet the R3051 required hold time.

PERFORMANCE

The performance of the different types of R3051 bus accesses to the DRAM memory is usually measured by the number of clock cycles it takes to send the $\overline{\text{Ack}}$ back to the R3051. This time is computed from the beginning of the external access. The performance of the DRAM system can be summarized as follows:

- single read: 4 clock cycles.
- block refill: 7 clock cycles.
- first write: 3 clock cycles.
- page write: 2 clock cycles.

The above numbers (with the exception of page write) will be increased by 2 in the case of delayed accesses.

Thus, relatively high memory performance is obtained with minimal external logic parts count, and low-cost commodity DRAM. More aggressive designs could utilize faster DRAMs, and techniques such as memory interleaving, to achieve still higher levels of performance.

CONCLUSION

The R3051 RISController family bus interface was designed to allow memory systems of differing complexity and performance to be implemented. Even a relatively simple DRAM system, as the one described here, offers very high performance. With simple modifications, this approach is applicable to higher frequencies (33 and 40 Mhz) and to interleaved memory systems yielding even higher performance.

```
{
      TITLE:              PAL1
      PURPOSE:            RAS
      AUTHOR:             BOB NAPAA, IDT INC.
      DATE:               4/5/91
}

MODULE PAL1;
TITLE PAL1;
TYPE AMD 22V10;

INPUTS;

      SYSCLKB                   NODE[PIN1];
      ENABLEB                   NODE[PIN2];
      RDB                       NODE[PIN3];
      WRB                       NODE[PIN4];
      BURSTB                    NODE[PIN5];
      RIPB                      NODE[PIN6];
      REFACKB                   NODE[PIN7];
      A22                       NODE[PIN8];
      A21                       NODE[PIN9];
      A20                       NODE[PIN10];
      C3                        NODE[PIN11];
      C2                        NODE[PIN13];
      C1                        NODE[PIN14];
      C0                        NODE[PIN15];

      {FEED BACK PINS}
      CIPB                      NODE[PIN16];
      RAS3B                     NODE[PIN17];
      RAS2B                     NODE[PIN18];
      RAS1B                     NODE[PIN19];
      RAS0B                     NODE[PIN20];
      DRAMWNB                   NODE[PIN21];
      DRAMACKB                  NODE[PIN22];
      DRAMRDCENB                NODE[PIN23];


OUTPUTS;
      CIPB                      NODE[PIN16] ATTR[RL];
      RAS3B                     NODE[PIN17] ATTR[RL];
      RAS2B                     NODE[PIN18] ATTR[RL];
      RAS1B                     NODE[PIN19] ATTR[RL];
      RAS0B                     NODE[PIN20] ATTR[RL];
      DRAMWNB                   NODE[PIN21] ATTR[RL];
      DRAMACKB                  NODE[PIN22] ATTR[RL];
      DRAMRDCENB                NODE[PIN23] ATTR[RL];

      {OUTPUT ENABLES}
      CIPBEN                    NODE[PIN16EN];
      RAS3BEN                   NODE[PIN17EN];
      RAS2BEN                   NODE[PIN18EN];
      RAS1BEN                   NODE[PIN19EN];
      RAS0BEN                   NODE[PIN20EN];
      DRAMWNBEN                 NODE[PIN21EN];
      DRAMACKBNODE[PIN22EN];
      DRAMRDCENBEN              NODE[PIN23EN];
```

```
TERMS;

      RAS3BEN          =     ENABLEB;
      RAS3B NOT        :=    RAS3B AND REFACKB AND RIPB AND DRAMWNB AND !RDB AND
                             !A22 AND A21 AND A20 {read/block refill}
                       OR    !RAS3B AND !CIPB AND !RDB AND DRAMACKB AND DRAMRDCENB
                             {keep for read/delayed read}
                       OR    RAS3B AND !CIPB AND RIPB AND !DRAMWNB AND !RDB AND
                             !A22 AND A21 AND A20 AND !C3 AND !C2 AND !C1 AND C0
                             {delayed read/delayed block refill}
                       OR    !RAS3B AND !CIPB AND !RDB AND !BURSTB AND !C3 {keep block refill}
                       OR    !RAS3B AND !CIPB AND !RDB AND !BURSTB AND !DRAMWNB AND
                             !C1 {keep delayed block refill}
                       OR    RAS3B AND REFACKB AND RIPB AND DRAMWNB AND !WRB AND
                             !A22 AND A21 AND A20 {write}
                       OR    RAS3B AND REFACKB AND RIPB AND !DRAMWNB AND !WRB AND
                             !A22 AND A21 AND A20 AND !C3 AND !C2 AND !C1 AND C0
                             {delayed write}
                       OR    !RAS3B AND !WRB AND !CIPB {keep for write}
                       OR    !RAS3B AND !DRAMWNB AND REFACKB AND RIPB AND RDB AND
                             WRB AND BURSTB {no access pending}
                       OR    !RAS3B AND !DRAMWNB AND REFACKB AND RIPB AND !WRB AND
                             !BURSTB AND !A22 AND A21 AND A20 {keep for page write}
                       OR    !REFACKB AND CIPB AND !RAS3B AND !DRAMWNB AND C0
                             {remove in refresh}
                       OR    RAS3B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND !C2
                             AND C1 AND C0 {issue for refresh}
                       OR    !RAS3B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                             AND !C1 AND !C0 {keep for refresh}
                       OR    !RAS3B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                             AND !C1 AND C0; {keep for refresh}

      RAS2BEN          =     ENABLEB;
      RAS2B NOT        :=    RAS2B AND REFACKB AND RIPB AND DRAMWNB AND !RDB AND
                             !A22 AND A21 AND !A20 {read/block refill}
                       OR    !RAS2B AND !CIPB AND !RDB AND DRAMACKB AND DRAMRDCENB
                             {keep for read/delayed read}
                       OR    RAS2B AND !CIPB  AND RIPB AND !DRAMWNB AND !RDB AND
                             !A22 AND A21 AND !A20 AND !C3 AND !C2 AND !C1 AND C0
                             {delayed read/delayed block refill}
                       OR    !RAS2B AND !CIPB AND !RDB AND !BURSTB AND !C3 {keep block refill}
                       OR    !RAS2B AND !CIPB AND !RDB AND !BURSTB AND !DRAMWNB AND
                             !C1{keep delayed block refill}
                       OR    RAS2B AND REFACKB AND RIPB AND DRAMWNB AND !WRB AND
                             !A22 AND A21 AND !A20 {write}
                       OR    RAS2B AND REFACKB AND RIPB AND !DRAMWNB AND !WRB AND
                             !A22 AND A21 AND !A20 AND !C3 AND !C2 AND !C1 AND C0
                             {delayed write}
                       OR    !RAS2B AND !WRB AND !CIPB {keep for write}
                       OR    !RAS2B AND !DRAMWNB AND REFACKB AND RIPB AND RDB AND
                             WRB AND BURSTB {no access pending}
                       OR    !RAS2B AND !DRAMWNB AND REFACKB AND RIPB AND !WRB AND
                             !BURSTB AND !A22 AND A21 AND !A20 {keep for page write}
                       OR    !REFACKB AND CIPB AND !RAS2B AND !DRAMWNB AND C0
                             {remove in refresh}
                       OR    RAS2B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND !C2
                             AND C1 AND C0 {issue for refresh}
                       OR    !RAS2B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                             AND !C1 AND !C0 {keep for refresh}
                       OR    !RAS2B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                             AND !C1 AND C0; {keep for refresh}
```

```
RAS1BEN          =     ENABLEB;
RAS1B NOT        :=    RAS1B AND REFACKB AND RIPB AND DRAMWNB AND !RDB AND
                       !A22 AND !A21 AND A20 {read/block refill}
                 OR    !RAS1B AND !CIPB AND !RDB AND DRAMACKB AND DRAMRDCENB
                       {keep for read/delayed read}
                 OR    RAS1B AND !CIPB AND RIPB AND !DRAMWNB AND !RDB AND
                       !A22 AND !A21 AND A20 AND !C3 AND !C2 AND !C1 AND C0
                       {delayed read/delayed block refill}
                 OR    !RAS1B AND !CIPB AND !RDB AND !BURSTB AND !C3 {keep block refill}
                 OR    !RAS1B AND !CIPB AND !RDB AND !BURSTB AND !DRAMWNB AND
                       !C1{keep delayed block refill}
                 OR    RAS1B AND REFACKB AND RIPB AND DRAMWNB AND !WRB AND
                       !A22 AND !A21 AND A20 {write}
                 OR    RAS1B AND REFACKB AND RIPB AND !DRAMWNB AND !WRB AND
                       !A22 AND !A21 AND A20 AND !C3 AND !C2 AND !C1 AND C0
                        {delayed write}
                 OR    !RAS1B AND !WRB AND !CIPB {keep for write}
                 OR    !RAS1B AND !DRAMWNB AND REFACKB AND RIPB AND RDB AND
                       WRB AND BURSTB {no access pending}
                 OR    !RAS1B AND !DRAMWNB AND REFACKB AND RIPB AND !WRB AND
                       !BURSTB AND !A22 AND !A21 AND A20 {keep for page write}
                 OR    !REFACKB AND CIPB AND !RAS1B AND !DRAMWNB AND C0
                       {remove in refresh}
                 OR    RAS1B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND !C2
                       AND C1 AND C0 {issue for refresh}
                 OR    !RAS1B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                       AND !C1 AND !C0 {keep for refresh}
                 OR    !RAS1B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                       AND !C1 AND C0; {keep for refresh}

RAS0BEN          =     ENABLEB;
RAS0B NOT        :=    RAS0B AND REFACKB AND RIPB AND DRAMWNB AND !RDB AND
                       !A22 AND !A21 AND !A20 {read/block refill}
                 OR    !RAS0B AND !CIPB AND !RDB AND DRAMACKB AND DRAMRDCENB
                       {keep for read/delayed read}
                 OR    RAS0B AND !CIPB AND RIPB AND !DRAMWNB AND !RDB AND
                       !A22 AND !A21 AND !A20 AND !C3 AND !C2 AND !C1 AND C0
                       {delayed read/delayed block refill}
                 OR    !RAS0B AND !CIPB AND !RDB AND !BURSTB AND !C3 {keep block refill}
                 OR    !RAS0B AND !CIPB AND !RDB AND !BURSTB AND !DRAMWNB AND
                       !C1 {keep delayed block refill}
                 OR    RAS0B AND REFACKB AND RIPB AND DRAMWNB AND !WRB AND
                       !A22 AND !A21 AND !A20 {write}
                 OR    RAS0B AND REFACKB AND RIPB AND !DRAMWNB AND !WRB AND
                       !A22 AND !A21 AND !A20 AND !C3 AND !C2 AND !C1 AND C0
                       {delayed write}
                 OR    !RAS0B AND !WRB AND !CIPB {keep for write}
                 OR    !RAS0B AND !DRAMWNB AND REFACKB AND RIPB AND RDB AND
                       WRB AND BURSTB {no access pending}
                 OR    !RAS0B AND !DRAMWNB AND REFACKB AND RIPB AND !WRB AND
                       !BURSTB AND !A22 AND !A21 AND !A20 {keep for page write}
                 OR    !REFACKB AND CIPB AND !RAS0B AND !DRAMWNB AND C0
                       {remove in refresh}
                 OR    RAS0B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND !C2
                       AND C1 AND C0 {issue for refresh}
                 OR    !RAS0B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                       AND !C1 AND !C0 {keep for refresh}
                 OR    !RAS0B AND !REFACKB AND CIPB AND DRAMWNB AND !C3 AND C2
                       AND !C1 AND C0; {keep for refresh}
```

```
      DRAMWNBEN          =     ENABLEB;
      DRAMWNB NOT        :=    DRAMWNB AND !CIPB AND RIPB AND !WRB  AND !C3   AND !C2 AND
                               !C1 AND C0 {write}
                         OR    !DRAMWNB AND !REFACKB AND CIPB AND RIPB AND !C3 AND !C2
                               AND !C1 AND !C0{remove at refresh}
                         OR    !DRAMWNB AND RIPB AND !RAS3B  {keep asserted if any RAS}
                         OR    !DRAMWNB AND RIPB AND !RAS2B
                         OR    !DRAMWNB AND RIPB AND !RAS1B
                         OR    !DRAMWNB AND RIPB AND !RAS0B
                         OR    !DRAMWNB AND RIPB AND !RDB AND !CIPB {keep for read}
                         OR    !DRAMWNB AND RIPB AND !WRB AND !CIPB;   {keep for write}

      DRAMACKBEN         =     ENABLEB;
      DRAMACKB NOT       :=    !CIPB AND !RDB AND DRAMWNB AND BURSTB AND !C3 AND !C2 AND
                               !C1 AND C0 {read}
                         OR    !CIPB AND !RDB AND !DRAMWNB AND BURSTB AND !C3 AND !C2
                               AND C1 AND C0 {delayed read}
                         OR    !CIPB AND !RDB AND DRAMWNB AND !BURSTB AND !C3 AND C2 AND
                               !C1 AND !C0 {block refill}
                         OR    !CIPB AND !RDB AND !DRAMWNB AND !BURSTB AND !C3 AND C2
                               AND C1 AND !C0 {delayed block refill}
                         OR    !CIPB AND !WRB AND DRAMWNB  AND !C3 AND !C2  AND !C1 AND !C0
                               {write}
                         OR    !CIPB AND !WRB AND !DRAMWNB AND BURSTB AND !C3 AND !C2
                               AND C1 AND !C0 {delayed write}
                         OR    !WRB AND !BURSTB AND !DRAMWNB AND REFACKB AND RIPB AND
                               CIPB AND !A22 AND !RAS3B {page write}
                         OR    !WRB AND !BURSTB AND !DRAMWNB AND REFACKB AND RIPB AND
                               CIPB AND !A22 AND !RAS2B {page write}
                         OR    !WRB AND !BURSTB AND !DRAMWNB AND REFACKB AND RIPB AND
                               CIPB AND !A22 AND !RAS1B {page write}
                         OR    !WRB AND !BURSTB AND !DRAMWNB AND REFACKB AND RIPB AND
                               CIPB AND !A22 AND !RAS0B ;{page write}

      DRAMRDCENBEN       =     ENABLEB;
      DRAMRDCENB NOT     :=    !CIPB AND !RDB AND DRAMWNB AND BURSTB AND !C3 AND !C2 AND
                               !C1 AND C0 {read}
                         OR    !CIPB AND !RDB AND !DRAMWNB AND BURSTB AND !C3 AND !C2
                               AND C1 AND C0 {delayed  read}
                         OR    !CIPB AND !RDB AND DRAMWNB AND !BURSTB AND !C3 AND C0
                               {block refill}
                         OR    !CIPB AND !RDB AND !DRAMWNB AND !BURSTB AND !C3 AND !C2
                               AND C1 AND C0 {delayed block refill}
                         OR    !CIPB AND !RDB AND !DRAMWNB AND !BURSTB AND !C3 AND C2
                               AND C0 {delayed block refill}
                         OR    !CIPB AND !RDB AND !DRAMWNB AND !BURSTB AND C3 AND !C2
                               AND !C1 AND C0;  {delayed block refill}

      CIPBEN             =     ENABLEB;
      CIPB NOT           :=    CIPB AND REFACKB AND RIPB AND !RDB AND !A22 {read}
                         OR    CIPB AND REFACKB AND RIPB AND !WRB AND !A22 {write}
                         OR    !CIPB AND !RDB {keep for read}
                         OR    !CIPB AND !WRB ;{keep for write}

END;
END PAL1.
```

```
{
      TITLE:              PAL2
      PURPOSE:            CAS
      AUTHOR:             BOB NAPAA, IDT INC.
      DATE:               4/5/91
}

MODULE PAL2;
TITLE PAL2;
TYPE MMI 20R8;

INPUTS;

      {SYSCLKB                  NODE[PIN1]; }
      REFACKB                   NODE[PIN2];
      DRAMWNB                   NODE[PIN3];
      BURSTB                    NODE[PIN4];
      RIPB                      NODE[PIN5];
      CIPB                      NODE[PIN6];
      WRB                       NODE[PIN7];
      A21                       NODE[PIN8];
      A20                       NODE[PIN9];
      C3                        NODE[PIN10];
      C2                        NODE[PIN11];
      {OUTENABLEB               NODE[PIN13]; }
      C1                        NODE[PIN14];
      C0                        NODE[PIN23];

      {FEED BACK PINS}
      CAS3B                     NODE[PIN22];
      CAS2B                     NODE[PIN21];
      CAS1B                     NODE[PIN20];
      CAS0B                     NODE[PIN19];
      DRAMLE                    NODE[PIN18];
      DRAMWRB                   NODE[PIN17];
      SB                        NODE[PIN16];
      TRB                       NODE[PIN15];

OUTPUTS;

      CAS3B                     NODE[PIN22];
      CAS2B                     NODE[PIN21];
      CAS1B                     NODE[PIN20];
      CAS0B                     NODE[PIN19];
      DRAMLE                    NODE[PIN18];
      DRAMWRB                   NODE[PIN17];
      SB                        NODE[PIN16];
      TRB                       NODE[PIN15];

TABLE;

      CAS3B NOT           :=    CAS3B AND RIPB AND !CIPB  AND DRAMWNB AND (A21 AND A20
                                AND !C3 AND !C2 AND !C1 AND C0)  {read or write}
                          OR    CAS3B AND RIPB AND !CIPB  AND !DRAMWNB AND (A21 AND A20
                                AND !C3 AND !C2 AND C1 AND C0)  {delayed read/write}
                          OR    CAS3B AND RIPB AND !CIPB AND !BURSTB AND DRAMWNB AND
                                WRB AND !SB AND (A21 AND A20 AND !C3 AND C0)  {block refill}
                          OR    CAS3B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                                WRB AND !SB AND (A21 AND A20 AND C0)  {delayed block refill}
                          OR    CAS3B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                                !WRB AND !SB AND (A21 AND A20 AND !C3 AND !C2 AND !C1 AND
                                !C0)  {page write}
```

```
                    OR      CIPB AND DRAMWNB AND !REFACKB AND CAS3B AND (!C3 AND !C2
                            AND C1 AND !C0)  {refresh}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND !CAS3B AND (!C3 AND !C2
                            AND C1 AND C0); {refresh}

CAS2B NOT           :=      CAS2B AND RIPB AND !CIPB  AND DRAMWNB AND (A21 AND !A20
                            AND !C3 AND !C2 AND !C1 AND C0)  {read or write}
                    OR      CAS2B AND RIPB AND !CIPB  AND !DRAMWNB AND (A21 AND !A20
                            AND !C3 AND !C2 AND C1 AND C0)  {delayed read/write}
                    OR      CAS2B AND RIPB AND !CIPB AND !BURSTB AND DRAMWNB AND
                            WRB AND !SB AND (A21 AND !A20 AND !C3 AND C0)  {block refill}
                    OR      CAS2B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                            WRB AND !SB  AND (A21 AND !A20 AND C0)  {delayed block refill}
                    OR      CAS2B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                            !WRB AND !SB AND (A21 AND !A20 AND !C3 AND !C2 AND !C1 AND
                            !C0)  {page write}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND CAS2B AND
                            (!C3 AND !C2 AND C1 AND !C0) {refresh}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND !CAS2B AND
                            (!C3 AND !C2 AND C1 AND C0); {refresh}

CAS1B NOT           :=      CAS1B AND RIPB AND !CIPB AND DRAMWNB AND (!A21 AND A20
                            AND !C3 AND !C2 AND!C1 AND C0)   {read or write}
                    OR      CAS1B AND RIPB AND !CIPB  AND !DRAMWNB AND (!A21 AND A20
                            AND !C3 AND !C2 AND C1 AND C0)   {delayed read/write}
                    OR      CAS1B AND RIPB AND !CIPB AND !BURSTB AND DRAMWNB AND
                            WRB AND !SB AND (!A21 AND A20 AND !C3 AND C0)   {block refill}
                    OR      CAS1B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                            WRB AND !SB  AND (!A21 AND A20 AND C0)  {delayed block refill}
                    OR      CAS1B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                            !WRB AND !SB AND (!A21 AND A20 AND !C3 AND !C2 AND !C1 AND
                            !C0)  {page write}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND CAS1B AND (!C3 AND !C2
                            AND C1 AND !C0) {refresh}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND !CAS1B AND (!C3 AND !C2
                            AND C1 AND C0); {refresh}

CAS0B NOT           :=      CAS0B AND RIPB AND !CIPB AND DRAMWNB AND (!A21 AND !A20
                            AND !C3 AND !C2 AND !C1 AND C0)  AND CAS0B {read or write}
                    OR      CAS0B AND RIPB AND !CIPB  AND !DRAMWNB AND (!A21 AND !A20
                            AND !C3 AND !C2 AND C1 AND C0)  AND CAS0B {delayed read/write}
                    OR      CAS0B AND RIPB AND !CIPB AND !BURSTB AND DRAMWNB AND
                            WRB AND  !SB AND (!A21 AND !A20 AND !C3 AND C0)   {block refill}
                    OR      CAS0B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                            WRB AND !SB  AND (!A21 AND !A20 AND C0)  {delayed block refill}
                    OR      CAS0B AND RIPB AND !CIPB AND !BURSTB AND !DRAMWNB AND
                            !WRB AND !SB AND (!A21 AND !A20 AND !C3 AND !C2 AND !C1 AND
                            !C0)   {page write}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND CAS0B AND (!C3 AND !C2
                            AND C1 AND !C0) {refresh}
                    OR      CIPB AND DRAMWNB AND !REFACKB AND !CAS0B AND (!C3 AND !C2
                            AND C1 AND C0); {refresh}

DRAMLE NOT          :=      TRB AND CAS3B AND CAS2B AND CAS1B AND !CAS0B {issue after}
                    OR      TRB AND !CAS3B AND CAS2B AND CAS1B AND CAS0B {any CAS if}
                    OR      TRB AND CAS3B AND !CAS2B AND CAS1B AND CAS0B {read cycle}
                    OR      TRB AND CAS3B AND CAS2B AND !CAS1B AND CAS0B
                    OR      CAS3B AND CAS2B AND CAS1B AND CAS0B;
```

```
      DRAMWRB NOT          :=    !CIPB AND RIPB AND !WRB AND DRAMWRB {issue for write}
                           OR    !WRB AND !BURSTB AND !DRAMWNB AND DRAMWRB AND RIPB
                                 AND REFACKB {issue for page write}
                           OR    !CIPB AND !DRAMWRB AND CAS3B AND CAS2B AND CAS1B
                                 AND CAS0B AND RIPB; {keep until end of write}

      SB NOT               :=    SB AND !CIPB AND DRAMWNB AND (!C3 AND !C2 AND !C1
                                 AND !C0) {read/write/block refill}
                           OR    !SB AND !CIPB AND !BURSTB AND WRB AND !C3 {keep for block refill}
                           OR    SB AND !CIPB AND !DRAMWNB AND (!C3 AND !C2 AND C1
                                 AND !C0) {delayed read/write/block refill}
                           OR    !SB AND !CIPB AND !DRAMWNB AND !BURSTB AND WRB AND
                                 !C1 {delayed block refill}
                           OR    !SB AND !CIPB AND BURSTB AND WRB AND C0 AND CAS3B AND
                                 CAS2B AND CAS1B AND CAS0B {read and delayed read}
                           OR    !SB AND !CIPB AND !WRB  AND CAS3B AND CAS2B AND CAS1B AND
                                 CAS0B {keep for write}
                           OR    !WRB AND !BURSTB AND !DRAMWNB AND SB AND REFACKB; {page write}

      TRB NOT              :=    TRB AND !CIPB AND WRB AND DRAMWNB AND (!C3 AND !C2
                                 AND !C1 AND !C0) {read/block refill}
                           OR    TRB AND !CIPB AND WRB AND !DRAMWNB AND SB AND (!C3
                                 AND !C2 AND C1 AND !C0) {delayed read/block refill}
                           OR    !TRB AND !CIPB AND !SB; {keep asserted for read/block refill}

END;
END PAL2.
```

```
{
      TITLE:              PAL3
      PURPOSE:            COUNTER
      AUTHOR:             BOB NAPAA, IDT INC.
      DATE:               4/5/91
}

MODULE PAL3;
TITLE PAL3;
TYPE MMI 16R8;

INPUTS;

      {BSYSCLKB                 NODE[PIN1]; }
      RESETB                    NODE[PIN2];
      REFREQ                    NODE[PIN3];
      BCIPB                     NODE[PIN4];
      DRAMWNB                   NODE[PIN5];
      {OUTENABLEB               NODE[PIN11]; }

      {FEED BACK PINS}
      RIPB                      NODE[PIN18];
      C3                        NODE[PIN17];
      C2                        NODE[PIN16];
      C1                        NODE[PIN15];
      C0                        NODE[PIN14];
      REFACKB                   NODE[PIN13];
      GATECOUNTERB              NODE[PIN12];

OUTPUTS;

      RIPB                      NODE[PIN18];
      C3                        NODE[PIN17];
      C2                        NODE[PIN16];
      C1                        NODE[PIN15];
      C0                        NODE[PIN14];
      REFACKB                   NODE[PIN13];
      GATECOUNTERB              NODE[PIN12];

TABLE;

      RIPB NOT                  :=    !RESETB      {reset}
                                OR    !RIPB AND !RESETB   {keep for reset}
                                OR    !RIPB AND REFACKB   {keep for refresh}
                                OR    !RIPB AND !REFACKB AND !C3; {keep until end of refresh}

      C3 NOT                    :=    !GATECOUNTERB AND !BCIPB AND REFACKB
                                OR    !GATECOUNTERB AND BCIPB
                                OR    GATECOUNTERB AND BCIPB AND REFACKB
                                OR    !C3 AND !C2
                                OR    !C3 AND C2 AND !C1
                                OR    !C3 AND C2 AND C1 AND !C0
                                OR    C3 AND C2 AND C1 AND C0;

      C2 NOT                    :=    !GATECOUNTERB AND !BCIPB AND REFACKB
                                OR    !GATECOUNTERB AND BCIPB
                                OR    GATECOUNTERB AND BCIPB AND REFACKB
                                OR    !C2 AND !C1
                                OR    !C2 AND C1 AND !C0
                                OR    C2 AND C1 AND C0;
```

```
      C1 NOT                    :=     !GATECOUNTERB AND !BCIPB AND REFACKB
                                OR     !GATECOUNTERB AND BCIPB
                                OR     GATECOUNTERB AND BCIPB AND REFACKB
                                OR     !C1 AND !C0
                                OR     C1 AND C0;

      C0 NOT                    :=     !GATECOUNTERB AND !BCIPB AND REFACKB
                                OR     !GATECOUNTERB AND BCIPB
                                OR     GATECOUNTERB AND BCIPB AND REFACKB
                                OR     C0;

      REFACKB NOT         :=     REFACKB AND REFREQ AND RESETB    {for refreq}
                                OR     !REFACKB AND !BCIPB AND RESETB   {as long as cipb low}
                                OR     !REFACKB AND !C3 AND RESETB AND GATECOUNTERB
                          {keep asserted}
                                OR     REFACKB AND RESETB AND !RIPB {reset}
                                OR     !REFACKB AND !GATECOUNTERB; {keep for reset}

      GATECOUNTERB NOT := GATECOUNTERB AND !REFACKB  AND !BCIPB AND RIPB
                                              {issue for both refack and cipb}
                                OR     !GATECOUNTERB AND !BCIPB AND RIPB
                                             {keep as long as cipb}
                                OR     !GATECOUNTERB AND !REFACKB AND RIPB AND C3
                                OR     !GATECOUNTERB AND !REFACKB AND RIPB AND C2
                                OR     !GATECOUNTERB AND !REFACKB AND RIPB AND C1
                                OR     !GATECOUNTERB AND !REFACKB AND RIPB AND C0;

END;
END PAL3.
```