Understanding Static RAM Operation

Overview

This document describes basic synchronous SRAM operation, including some of the most commonly used features for improving SRAM performance.

Fast, Faster, Fastest

As microprocessors and other electronics applications get faster and faster, the need for large quantities of data at very high speeds increases, while providing the data at such high speeds gets more difficult to accomplish. As microprocessor speeds increase from 25 MHz to 100 MHz, to 250 MHz and beyond, systems designers have become more creative in their use of cache memory, interleaving, burst mode and other high-speed methods for accessing memory.

The old systems sporting just an on-chip instruction cache, a moderate amount of DRAM and a hard drive have given way to sophisticated designs using multilevel memory architectures. One of the primary building blocks of the multi-level memory architecture is the data cache.

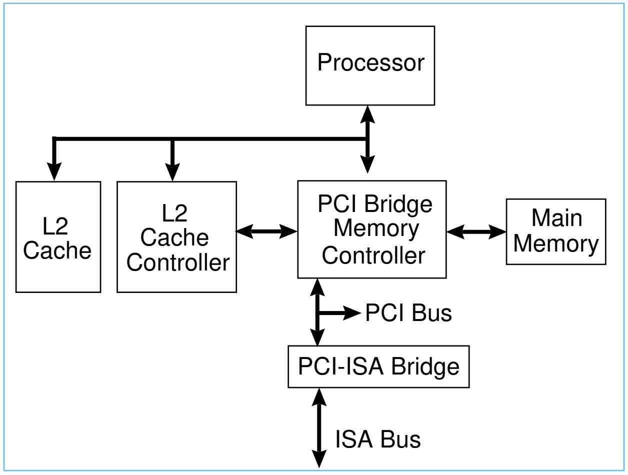
What is a Cache?

A cache is memory used to temporarily store data. In many computer systems, whether a PC, a RISC workstation, or a mainframe, caches are used to:

- Store the instructions and data for the processor. Called the Level 1 or L1 cache, this small memory is found on the microprocessor chip and runs at processor speed.
- Optimize the amount of time required to provide data to the CPU when there is a miss on L1. This is called the Level 2 (L2) cache. It is many times larger than the L1 cache and is usually designed so that 90% of the time, the processor can find the data it needs there. In PC applications, the L2 cache runs at one half to one third of the processor speed. In workstations, the L2 cache may be required to operate at processor speed. The L2 cache resides on the board with the processor or on SIMMs (Single In-Line Memory Modules) that are located near the processor. Data stored by the processor in the L2 cache is periodically off-loaded to the DRAM (extended or Level 3 memory) or to one of the disk drives.

Increasingly, the designers of high performance systems, including PC and RISC based computers and high speed telecommunications applications, rely on synchronous SRAMs to design caches that provide data at the speeds they require.

Figure 1. Basic Cache System



Why use an SRAM?

There are many reasons to use an SRAM or a DRAM in a system design. Design tradeoffs include density, speed, volatility, cost, and features. All of these factors should be considered before you select a RAM for your system design.

- **Speed.** The primary advantage of an SRAM over a DRAM is its speed. The fastest DRAMs on the market still require five to ten processor clock cycles to access the first bit of data. Although features such as EDO and Fast Page Mode have improved the speed with which subsequent bits of data can be accessed, bus performance and other limitations mean the processor must wait for data coming from DRAM. Fast, synchronous SRAMs can operate at processor speeds of 250 MHz and beyond, with access and cycle times equal to the clock cycle used by the microprocessor. With a well-designed cache using ultra-fast SRAMs, conditions in which the processor has to wait for a DRAM access become rare.

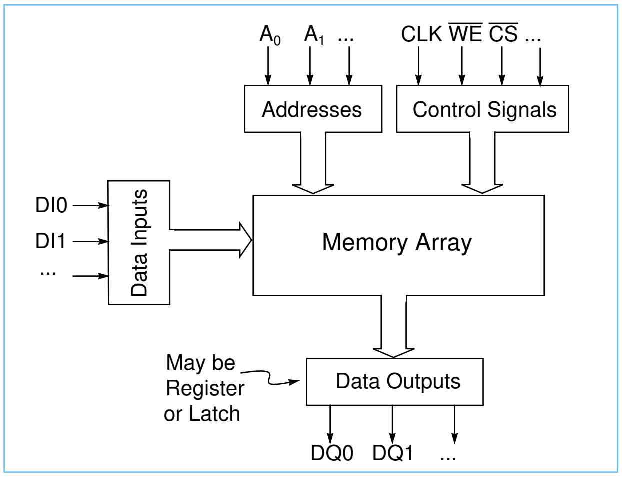
- **Density.** Because of the way DRAM and SRAM memory cells are designed, readily available DRAMs have significantly higher densities than the largest SRAMs. Thus, when 64 Mb DRAMs are rolling off the production lines, the largest SRAMs are expected to be only 16 Mb.
- **Volatility.** While SRAM memory cells require more space on the silicon chip, they have other advantages that translate directly into improved performance. Unlike DRAMs, SRAM cells do not need to be refreshed. This means they are available for reading and writing data 100% of the time.
- **Cost.** If cost is the primary factor in a memory design, then DRAMs win hands down. If, on the other hand, performance is a critical factor, then a well-designed SRAM is an effective cost performance solution.
- **Custom features.** Most DRAMs come in only one or two flavors. This keeps the cost down, but doesn't help when you need a particular kind of addressing sequence, or some other custom feature. IBM's SRAMs are tailored, via metal and substrate, for the processor or application that will be using them. Features are connected or disconnected according to the requirements of the user. Likewise, interface levels are selected to match the processor levels. IBM provides processor specific solutions by producing a chip with a standard core design, plus metal mask options to define feature sets.

Basic Architecture

The basic architecture of a static RAM includes one or more rectangular arrays of memory cells with support circuitry to decode addresses, and implement the required read and write operations. Additional support circuitry used to implement special features, such as burst operation, may also be present on the chip.

Figure 2 shows a basic block diagram of a synchronous SRAM. As you read, you may wish to refer to the diagram to help you visualize how the SRAM works.

Figure 2. Block Diagram of a Synchronous SRAM



Memory Arrays

SRAM memory arrays are arranged in rows and columns of memory cells called wordlines and bitlines, respectively. In IBM SRAMs, the wordlines are made from polysilicon while the bitlines are metal. Each memory cell has a unique location or address defined by the intersection of a row and column. Each address is linked to a particular data input/output pin. The number of arrays on a memory chip is determined by the total size of the memory, the speed at which the memory must operate, layout and testing requirements, and the number of data I/Os on the chip.

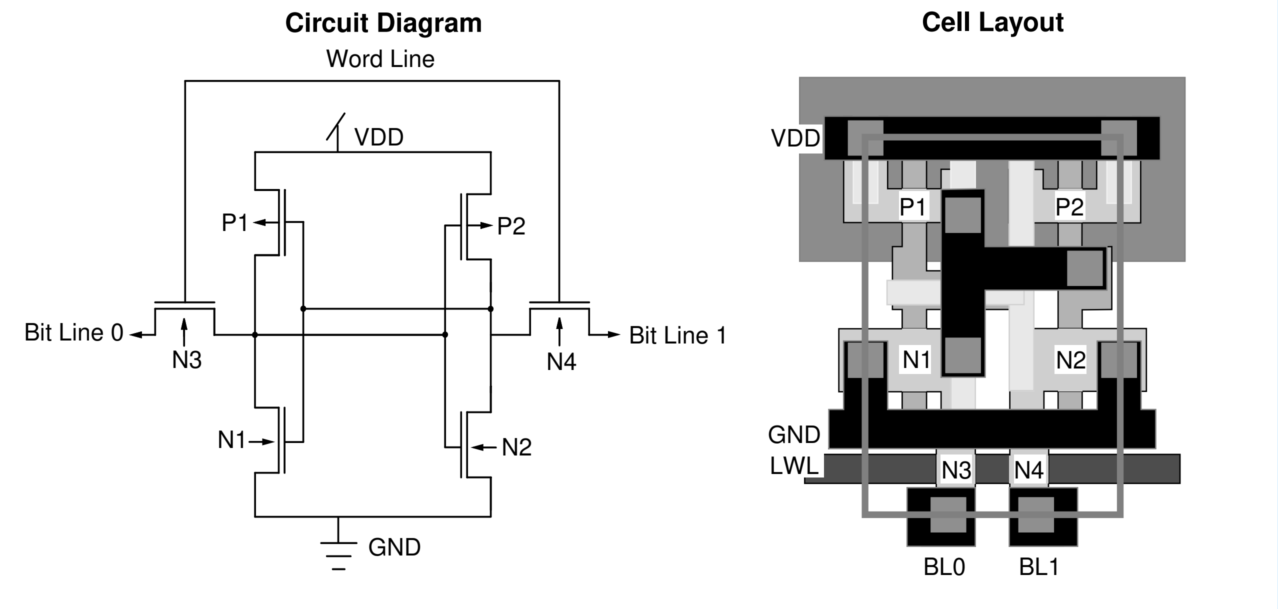
Memory Cell

An SRAM memory cell is a bi-stable flip-flop made up of four to six transistors. The flip-flop may be in either of two states that can be interpreted by the support circuitry to be a 1 or a 0.

Many of the SRAMs on the market use a four transistor cell with a polysilicon load. Suitable for medium to high performance, this design has a relatively high leakage current, and consequently high standby current. Four transistor designs may also be more susceptible to various types of radiation-induced soft errors. IBM's SRAMs all use a six transistor memory cell (also called a six-device cell) that is highly stable, relatively impervious to soft errors, and has low leakage and standby currents.

Recognizing the superiority of the six-device cell while trying to avoid using the extra chip real estate, many industry SRAM producers are migrating slowly toward the six-device cell via intermediate designs. IBM's commitment to the six-device cell stems from its functional superiority and reliability.

Figure 3. IBM's 6-Transistor Memory Cell



Support Circuitry

The memory chip's support circuitry allows the user to read the data stored in the memory's cells, and write data to the cells. This circuitry generally includes:

- Address logic to select rows and columns.
- Translation logic that "reads" the data in a cell and sends that data to the data I/O.
- Write logic that takes the user data applied at the input and stores it in a memory cell.
- Output enable logic to prevent data from appearing at the outputs unless specifically desired.
- Internal counters and registers to keep track of burst address sequences, pipelined data, and other control functions on the chip.
- Clock circuitry to control the timing of the read and write operations and all of their variations.

Silicon Technology

High performance SRAMs currently on the market are manufactured using one of three basic processes. A small number of offerings are made using bipolar circuits. Most are made using a BiCMOS technology in which most of the internal circuitry is CMOS while the interface circuits (including the drivers) are bipolar. A few, including all of IBM Microelectronics' high performance, synchronous SRAMs, are made entirely with CMOS circuits.

Packaging

IBM Microelectronics' SRAMs come in three different packages depending on their cost and performance. PC-compatible SRAMs are usually sold in 52-pin PLCC or 100-pin TQFP packages. These packages comply with the JEDEC standards for pinout and footprint. Ultra-high performance SRAMs, including most of the SRAMs geared to high-end workstations, require a higher performance package — the 119-pin Ball Grid Array (BGA) package. The BGA package has superior performance characteristics such as shorter leads from chip to package and internal planes that result in lower inductance and reduced package-related noise.

Understanding the SRAM Timing Diagram

Synchronous or Asynchronous — SRAMs come in a variety of architectures and speeds, and in synchronous and asynchronous designs. Asynchronous SRAMs respond to changes at the device's address pins by generating a clock signal that is used to time the SRAM's internal circuitry during a read or write operation. Although commonly used, this type of design runs into limitations at the high end of the performance range. For this reason, the fastest SRAMs are generally synchronous. Synchronous SRAMs use one or more external clocks to time the SRAM's operations. Because of the improved timing control possible with this method, access times and cycle times can be reduced to match the clock cycles of the fastest PC and RISC processors on the market today. All of IBM Microelectronics' SRAM offerings are synchronous.

Performance Specifications — SRAM access and cycle times are routinely used to compare the offerings of various manufacturers. The quoted access time is the minimum amount of time required to read a bit of data from the memory, measured with respect to the initial rising clock edge in the SRAM read operation. This access time specification is measured under specific load, temperature, and power supply conditions, in which all critical timings meet the requirements set out in the product specification.

Access times may also be measured with respect to other signal transitions such as the Output Enable. This type of specification gives the amount of time required to obtain a signal at the data output pin after the signal switches at the inputs.

The cycle time is the amount of time required to perform a single read or write operation and reset the internal circuitry so that another operation can begin. This time is usually designated by one complete clock cycle. (See Figure 4.) In some cases, the access time and cycle time are equal; in others, access times may be greater or smaller than the cycle time.

Synchronous SRAM Control Signals

Synchronous SRAM operation is relatively straightforward. All operations are controlled by one or more clock signals. These clock signals are generated externally using clock chips or circuits. To operate properly, each of the control signals must be valid when the appropriate clock edge occurs.

The following signals control an SRAM's operation. Other signal pins, used to implement special features such as sleep mode and specific burst addressing sequences, or to satisfy testing requirements, may also be present on a particular SRAM.

Address (ADDR or **SAx)** The address inputs are used to select a memory location on the chip. In actuality, when you select an address, you choose a memory location for every I/O on the chip. On a chip with 18 data I/Os, you choose 18 memory locations simultaneously. For performance reasons, there are row and column address pins so that both may be selected at once. The number of address input pins depends on the size of the memory and how it is organized. For example, a 32K by 36 SRAM has 15 address input pins (2^{15} = 32K) and 36 data I/O pins.

Data Inputs and Outputs (DQs or **I/Os)** The DQ pins are used for data input and output. DQs on SRAMs come in two forms — on some devices, the inputs and outputs are separate, on others, they are common, with the input and output using the same pin on the memory device. All of IBM Microelectronics' SRAM offerings have common I/O.

During a write operation, a data signal is applied at the data input pin. This data is translated into the appropriate signal and stored in the selected memory cell. During a read operation, data from the selected memory cell appears at the data output pin once access is complete and the output is enabled. At most other times, the DQs are in a high impedance state (also called tri-state); they do not source or sink any current, and do not present a signal to the system. This also prevents DQ contention when two or more devices are dotted together.

Output Enable ($\overline{\textbf{OE}}$ or **$\overline{\textbf{G}}$)** When $\overline{\text{OE}}$ is high, the data outputs (DQs) are always tri-stated. When $\overline{\text{OE}}$ is low, the outputs are active, that is, data, if available, can appear at the output pins. During a read operation, this signal is used to prevent data from appearing at the output until needed. Prior to a write operation, $\overline{\text{OE}}$ is sometimes used to tri-state the data bus to avoid data contention. $\overline{\text{OE}}$ is ignored during a write operation. $\overline{\text{OE}}$ is an asynchronous signal. It can be switched at any time and the SRAM will respond to the signal. Use of the output enable, although recommended, is optional.

K Clock The K clock is the primary clock. On single clock SRAMs, this clock controls when input signals are latched at the beginning of a read or write operation, and when output signals appear at the output pins. On dual clock SRAMs, the K clock controls only the input signals; the output signals are controlled by the C clock. On late write SRAMs, the K clock is a differential input (the input pins are K and $\overline{\text{K}}$).

C Clock The C clock is only used in dual clock SRAMs. The C clock is used to time the appearance of data at the output pins. A K clock is used to control the input signals. On late write SRAMs, the C clock is a differential input (the input pins are C and $\overline{\text{C}}$).

Chip Select ($\overline{\textbf{CS}}$ or $\overline{\textbf{SS}}$) The Chip Select is used to block or allow input signals to the chip. When $\overline{\text{CS}}$ is high, input signals applied to the chip's input pins are ignored. When $\overline{\text{CS}}$ is low, input signals may be applied to the chip's input pins, and the signals will be latched at the appropriate time in the clock cycle.

Write Enable ($\overline{\textbf{WE}}$ or $\overline{\textbf{SW}}$) Write Enable is used to choose between a read and a write operation. When $\overline{\text{WE}}$ is low, data applied at the data input pins is written into memory. When $\overline{\text{WE}}$ is high, a read operation occurs and data at the data input pins is ignored.

In IBM SRAMs, Write Enable may be implemented in two different ways. Some SRAMs have a single $\overline{\text{WE}}$ that controls the read or write operations, and several Byte Write control pins that permit the user to mask individual bytes during a write operation. Some IBM SRAMs have several Write Enable pins ($\overline{\text{WEx}}$) that combine the write enable and byte write mask functions.

Byte Write Enable ($\overline{\textbf{WEx}}$ or $\overline{\textbf{SBWx}}$) The Byte Write Enable pins permit the user to mask one or more bytes when writing data to the memory. In some IBM SRAMs, the byte write mask and the write enable functions are combined; these SRAMs have several $\overline{\text{WE}}$ pins, one for each byte of data (set of 9 DQs). Some IBM SRAMs have separate Byte Write Enable pins for each byte of data (set of 9 DQs). They must be selected along with the write enable if data is to be written. Each byte contains eight data bits and a parity bit. (Parity is not evaluated on the chip and must be provided by the user.)

On SRAMs with a $\overline{\text{WEx}}$ for each set of 9 DQs (and no $\overline{\text{WE}}$), the appropriate $\overline{\text{WE}}$ must be low to write the corresponding memory locations. To write data to the memory cells associated with all of the chip's DQs, all of the $\overline{\text{WEx}}$ pins must be low (active).

On SRAMs with a write enable and byte write enable pins, you must select $\overline{\text{WE}}$ (low) and the Byte Write Enable pins for the bytes that are to be written. If the Byte Write Enable is high, no data is written to the memory locations associated with the DQs controlled by that input.

Reading Data From Memory

Figure 4 shows the timing diagram for the simplified read operation for a flow thru part. It is used to illustrate the following example.

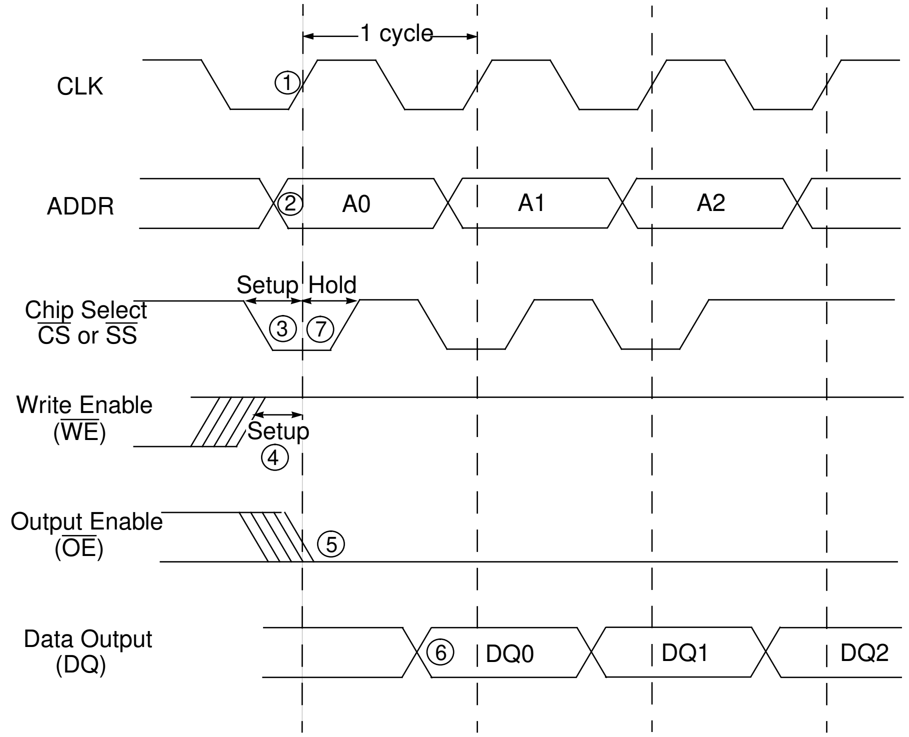
To read data from a memory cell, the cell must be selected using its row and column coordinates, the state of the cell must be determined, and the information must be sent to the data output. In terms of timing, the following steps must occur:

1. Before the clock transition (low to high) that initiates the read operation (1), the row and column addresses must be applied to the address input pins (ADDR) (2), the chip must be selected (3), and the Write Enable must be high (4).

   Note that each of these signals must be present and valid a specified amount of time (the set up time) before the clock switches from low to high, and must remain valid for a specified amount of time (the hold time) after the clock switches (7).

   When the chip select ($\overline{\text{CS}}$) is low, the chip is selected. When it is high (inactive), the chip cannot accept any input signals. The Write Enable is used to choose between reading and writing. When it is low, a write operation occurs; when it is high, a read operation occurs.
2. On the rising edge of the clock (CLK) (1), the address is registered and the read cycle begins.
3. If the Output Enable is being used to control the appearance of data at the output, $\overline{\text{OE}}$ must go low (5). $\overline{\text{OE}}$ is an asynchronous signal; it can be activated at any time. When $\overline{\text{OE}}$ is high, the DQs are tri-stated; data from the memory will not appear on the outputs.
4. Data appears at the output pins of the SRAM (6). The time at which the data appears depends on the access time of the device, the delay associated with the Output Enable and the type of SRAM you are using. The access time of the SRAM is the amount of time required to read a bit of data from the memory when all of the timing requirements have been met.

Figure 4. Reading from Memory (Flow Thru mode)



Note: DQ0 is the data associated with Address 0 (A0). DQ1 is the data associated with Address 1 (A1).

Four Methods for Reading Data

There are four fundamental ways in which data can be read from an SRAM. They are:

- Flow thru
- Pipeline (also called Register to Register)
- Register to Latch
- Burst

With the exception of Burst mode, the main differences between these different types of SRAMs are in the relationship of the data out to the clock signal. Burst mode may be used in conjunction with flow thru and pipeline features. Figure 5 can be used to compare the relative timings. For simplicity, only the address signal, the clock, and the data output (DQ) are used to explain these methods for reading SRAM data.

Flow Thru

On flow thru SRAMs, addresses and other control signals are set up before the clock switches. Then, when the clock switches from low to high, the inputs are registered, and the read cycle begins. Some time after the clock transition, but within the same clock cycle, data appears at the outputs. The time at which the data appears depends only on the original clock transition and the speed of the internal circuitry. See (1) in Figure 5.

Pipeline (Register to Register)

On the pipelined SRAM, after all of the control signals are set up, the clock switches and the read cycle begins, As the data is read from the memory cells, it is stored in a series of output registers. The data is transferred from the output registers to the data output pins after the clock switches on the next cycle. Data at the output always appears one cycle after the address for that data was selected. See (2) in Figure 5.

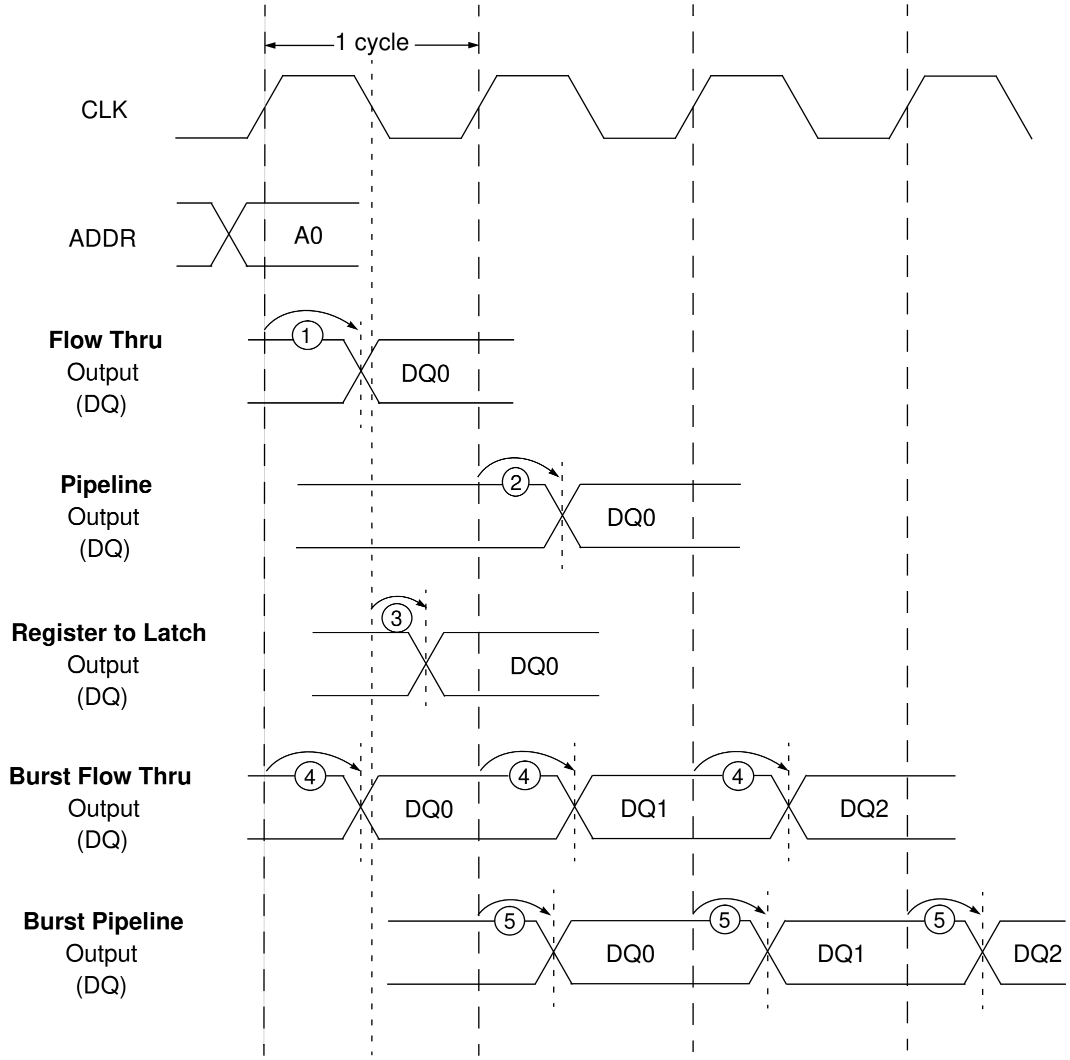
Register to Latch

Although all of IBM Microelectronics' synchronous SRAMs feature register inputs, only a few of them also offer latched outputs. On the Register to Latch SRAM, the addresses and other control signals are set up, then loaded into a register when the first clock transition occurs. The data read from the memory cells is stored in a set of latches. When, later in the same cycle, the clock switches from high to low, the data is transferred from the latches to the data output pins. This method permits the designer to control the time at which data appears at the output by adjusting the width of the clock pulse. See (3) in Figure 5.

Burst Mode

Many PC compatible SRAMs feature a burst mode operation for improved performance. In burst mode, several bits of data are selected using a single address, which is incremented using an on-chip counter. Both flow thru and pipelined SRAMs may have the burst feature. Figure 5 shows how the burst flow thru (4) and burst pipelined (5) SRAMs compare to the previously described parts. Several burst addressing sequences are supported including those designed for PowerPC and Pentium-based systems.

Figure 5. Types of SRAMs — Read Timings



Notes: DQ0 is the data associated with Address 0 (A0).
Cycle time is longer than Flow Thru access time.

Writing Data To Memory

Figure 6 shows a simplified timing diagram for the write operation used in the following example. This example uses a standard write, flow thru SRAM.

To write data to a memory cell, the cell must be selected using its row and column coordinates, the data to be stored must be applied at the data input pins, and the information must be stored in the selected memory cell. In terms of timing, the following steps must occur:

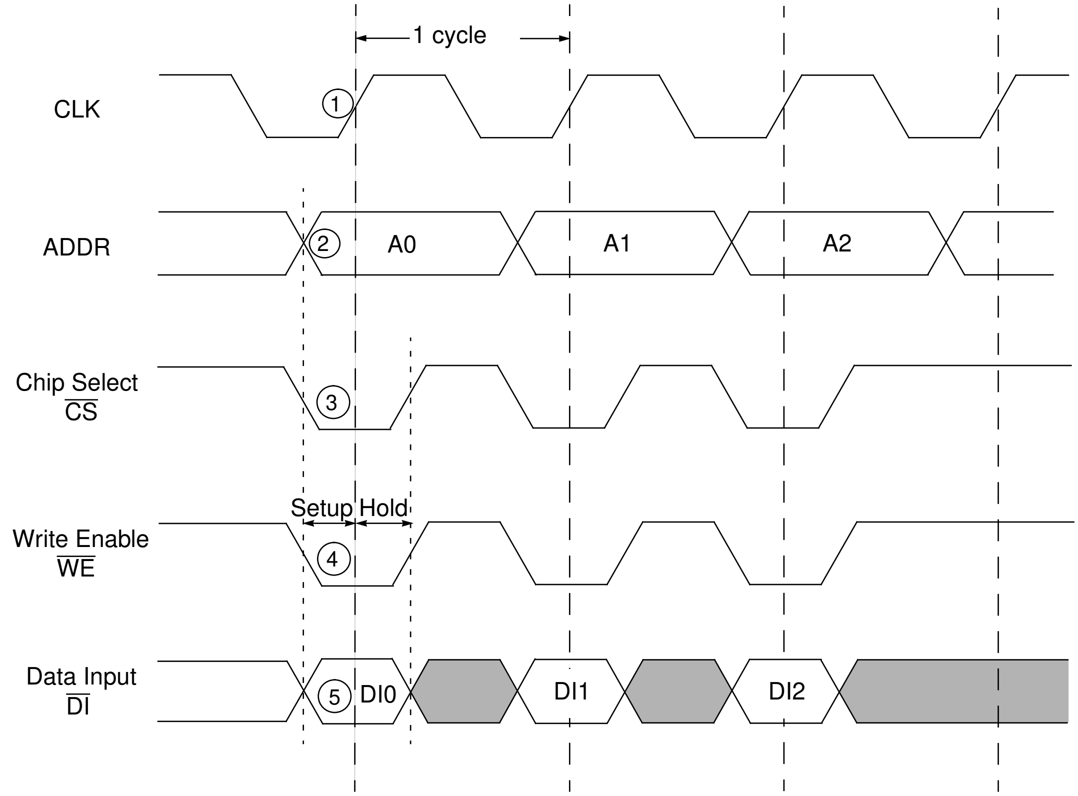
1. Before the clock transition (low to high) that initiates the write operation (1), the row and column addresses must be applied to the address input pins (ADDR) (2), the chip must be selected (3), the Write Enable must be low (4) and the data to be written must be applied to the data input pins (5). If the SRAM has Byte Write Enables, they must be low as well.

   Note that each of these signals must be present and valid a specified amount of time (the set up time) before the clock switches from low to high, and must remain valid for a specified amount of time (the hold time) after the clock switches.

   When the chip select ($\overline{CS}$) is low, the chip is selected. When it is high (inactive), the chip cannot accept any input signals. The Write Enable is used to choose between reading and writing. When it is low, a write operation occurs; when it is high, a read operation occurs. When the Byte Write Enables are high, no data may be written to the memory. When they are low, data may be written to the associated data inputs.

2. On the rising edge of the clock (CLK) (1), the address and input data are latched and the write operation begins. The data is stored in the selected memory cell.

Figure 6. Writing to Memory (Standard Write in Flow Thru mode)



Note: DI0 is written to the memory location with address 0 (A0).

Standard vs. Late Write

Because it is more commonly used, a standard write part was used to explain the write operation in the example above. In reality, two distinct types of SRAMs are available — those that use the standard write feature and those that offer late write. Standard write parts are generally used on PC-compatible applications, while late write parts are favored for RISC-based designs such as high performance workstations.

In a standard write SRAM, switching from a read to a write operation results in two dead cycles. In a late write SRAM, only one cycle is lost when switching from a read to a write operation. The following sections (along with Figure 7 on page 11) explain the differences between standard and late write.

Standard Write

In order to compare standard write with late write, you must look at what happens when you switch from a read operation to a write operation and back again. In the example, a pipelined part is used. In the example shown in Figure 7, the write operation occurs in the fourth clock cycle.

In the standard write part, address A0 is applied at the beginning of the first cycle (1). When the clock switches, the address is stored in the input register and the read cycle begins.

In time for the next cycle, address A1 is applied (2), and the second read cycle begins. Meanwhile, the data for the first address (DQ0) appears at the outputs (3) (in pipelined parts, the data always appears at the outputs on the cycle after the addresses are selected.)

In the third clock cycle, address A2 is selected (4). The data corresponding to address A1 (DQ1) appears at the outputs (5). Because the data must be tri-stated before data can be applied during the write operation, DQ1 may not be usable since it may not be valid for a sufficient period of time. In most cases, the second cycle, and DQ1 will be lost. This is called a “dead cycle.”

In the fourth clock cycle, the user chooses to write data. Address A3 is selected (6), the write enable is activated (7), and data is applied at the data inputs (8) (also the outputs). Because of the timing of the data inputs, the data read from address A2 is lost; it does not appear at all. Thus, the third cycle is also a dead cycle.

In the fifth cycle, address A4 is applied (9). If, after the write operation, the next cycle is a read (10), then expect to see the data from A3 at the outputs on that cycle (11). The contents of address A4 will appear at the outputs in the sixth cycle (not shown).

Late Write

In the late write part, the data associated with A1 is not lost since it is valid for the entire cycle. See (13) in Figure 7.

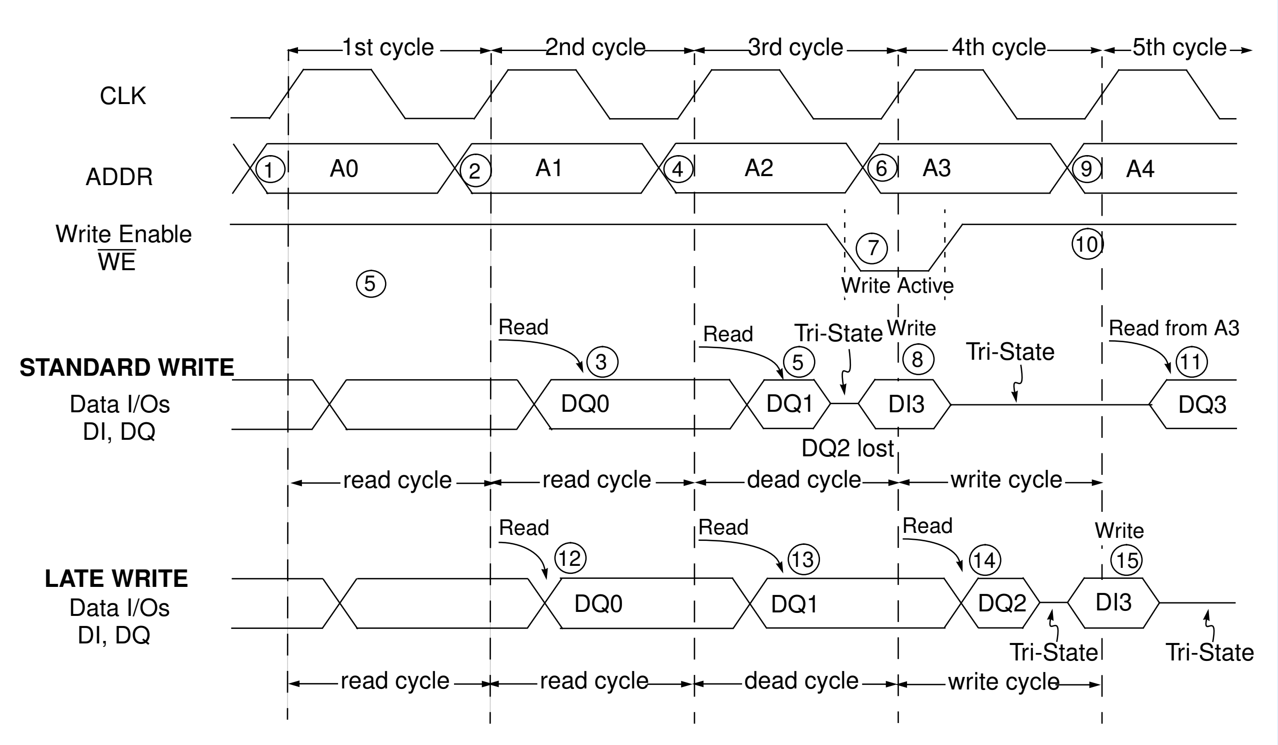
As in the standard write part, the address A0 is applied in the first cycle (1), and the data DQ0 appears at the output in the second cycle (12).

In the second cycle, address A1 is applied (2), and the data DQ1 appears in the third cycle (13). In the third cycle, address A2 is applied (4), and the data, DQ2, appears in the fourth cycle (14). Because the data must be tri-stated before data can be applied during the write operation, DQ2 may not be usable since it may not be valid for a sufficient period of time. This cycle is a dead cycle. DQ2 will be lost.

In the fourth cycle, address A3 is selected (6), along with the Write Enable (7). Unlike the standard write part, though, the input data for the write operation is not needed until the following cycle (15).

In the fifth cycle, address A4 is applied (9), and a read operation is selected (10). The contents of address A4 will appear at the outputs in the sixth cycle (not shown).

Figure 7. Comparing Standard Write and Late Write (Pipeline mode)



IBM ®

This document may contain preliminary information and is subject to change by IBM without notice. IBM assumes no responsibility or liability for any use of the information contained herein. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. The products described in this document are not intended for use in implantation or other direct life support applications where malfunction may result in direct physical harm or injury to persons. **NO WARRANTIES OF ANY KIND, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT**.

For more information contact your IBM Microelectronics sales representative or
visit us on World Wide Web at http://www.chips.ibm.com

IBM Microelectronics manufacturing is ISO 9000 compliant.