

# EASILY UPGRADE A 68030-BASED SYSTEM WITH A CLEVER CACHE DESIGN

BUILD A CACHE DAUGHTERBOARD THAT PLUGS INTO THE EXISTING PROCESSOR SOCKET TO EXPLOIT THE PROCESSOR'S SYNCHRONOUS MODE.

**C**ompanies spend lots of time and resources creating a microprocessor-based product. By the time the product is ready for the market, however, the industry has already introduced a faster or more advanced processor. Selling yesterday's technology is difficult, yet it's not profitable to sacrifice the existing system and develop another product that uses the updated technology. The ideal solution is to bring the updated technology to the existing system without major changes.

For instance, a 68030-based system that uses dynamic RAM (DRAM) runs at 3 clock cycles/access in the asynchronous mode. The system doesn't take advantage of the processor's synchronous operation, which runs at 2 clock cycles/access at 33 MHz. Adding cache memory to the existing system would convert a large number of the slow DRAM accesses to fast accesses in the cache static RAM (SRAM). A high ratio of SRAM accesses to DRAM accesses indicates better performance.

A cache can be built on a daughterboard that plugs into the system's existing processor socket. That way, designers can upgrade an outdated microprocessor-based system without changing its architecture. The cache system is a small subsystem that uses fast SRAM and cache technology. With that technology, the processor can run at its maximum speed. The cost of developing this small, add-in daughterboard is minimal compared to the cost and the time it takes to rebuild another system. This cache system can upgrade an application's speed up to 30%.

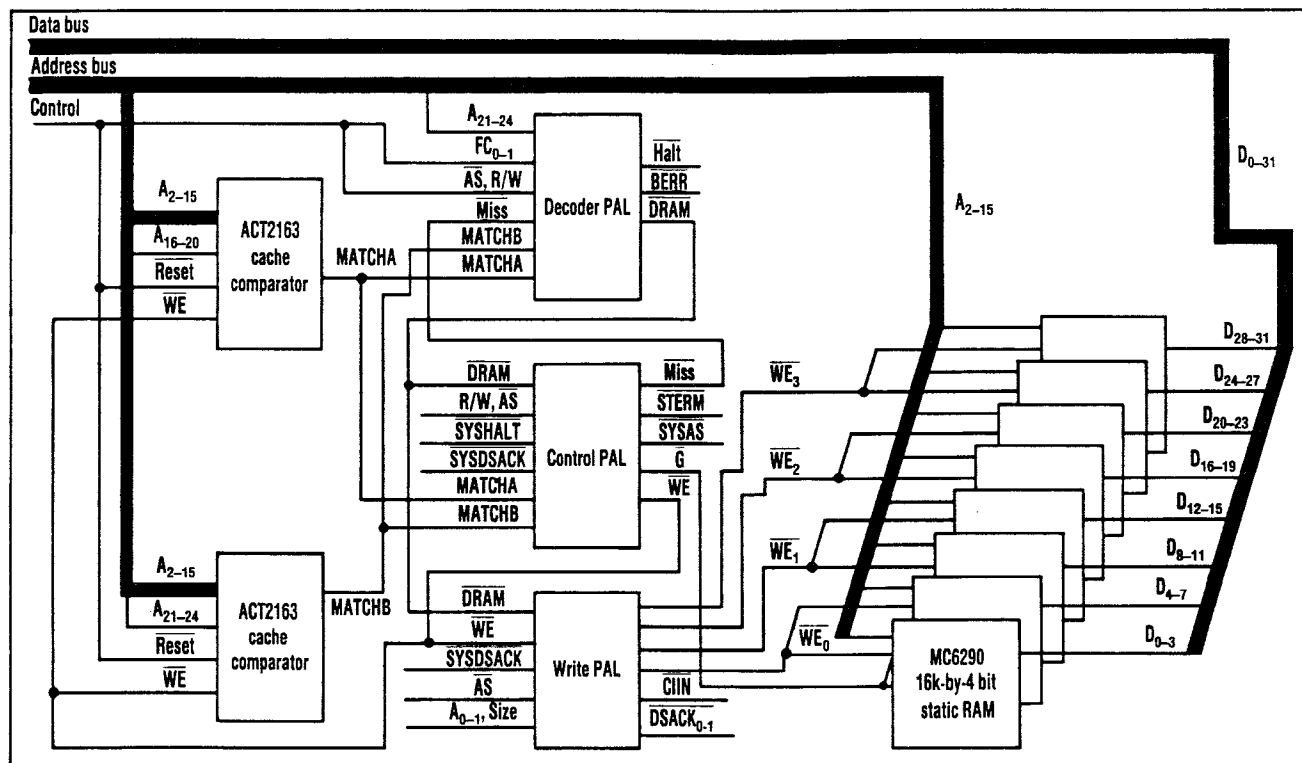
The operation of cache is based on the principle of program locality. Programs usually access the same memory blocks repeatedly. In addition, they spend much time executing instructions in a loop. DRAM is used for main memory and SRAM for the cache. Accesses to SRAM are fast and don't need wait states. When the program reads data or instructions from the slower DRAM memory, the processor writes the surrounding block of data into the cache. Subsequently, the next access to the same information comes from the cache memory with no wait states.

There are drawbacks, however. The data that the microprocessor needs may not always be in the cache memory. When the program attempts to find data in the cache, the system will indicate if the desired data is stored there. This is done with

**NAGI MEKHIEL**

Ryerson Polytechnical Institute, 350 Victoria St., Toronto, Ontario, Canada M5B 2K3; (416) 979-5000.

# CACHE MEMORY DESIGN



**1. THE CACHE SYSTEM** is made up of eight Motorola SRAM chips, three PAL devices, and two Texas Instruments tag comparators.

the cache tag comparator, which compares the current address being accessed by the processor to the addresses of data stored in SRAM and determines if they match.

A cache hit occurs when a match is found. In that case, the memory location needed by the program is already in the cache (SRAM), so the cache can give the required information to the processor without delay. A cache miss occurs when there's no match, and the processor has to fetch data from the main memory (DRAM) and update the cache.

## A TRICKY TASK

Designing the add-in cache system is tricky because it can't alter the existing system, but instead must work around it. This cache has unique features that optimize the overall system performance. For example, the design interleaves DRAM and SRAM accesses. Interleaving memory accesses reduces the penalty of a cache miss. In addition, the cache controller hides the dynamic RAM refresh in static RAM and peripheral accesses.

The daughterboard uses a simple direct-mapped cache that's made from 64 kbytes of fast static memory. Direct-mapped design offers simplicity and uses the least amount of tag memory. The main memory (DRAM on the system board) is divided into logical pages. Each page of DRAM is the same size as the cache.

The total number of the DRAM pages is calculated by dividing the DRAM size by the size of the SRAM. In this 68030-based system, 8 Mbytes of DRAM and 64 kbytes of SRAM exist. The number of pages is  $8000 \div 64 = 128$  pages.

Each page is directly mapped to the 64-kbyte SRAM space. A page consists of 16k 4-byte lines. The address space  $A_{2-15}$  covers the 16k lines needed for each page.

Tag memory, which is the fast SRAM part of the tag comparator, stores the tag of each line. The tag is the page number that's unique for each line. In this system, the tag takes a value of 0 to 127 for the total of 128 pages. When the processor stores any information into the

SRAM, the cache controller stores the line tag into the tag memory. The tag memory is 16k by 7 bits, and is mapped directly into the SRAM space  $A_{2-15}$ . Seven bits are needed to store 128 tags.

When the processor starts an access, the tag comparator reads the stored tag data for this location. It compares the tag data with the current accessed address ( $A_{16-22}$ ) and indicates a hit if they match. Address lines  $A_{16-22}$  select one page of the 128 pages.

The cache is made up of eight 15-ns, 16k-by-4-bit Motorola MC6290 SRAM chips (*Fig. 1*). Each page is organized as 16k 32-bit words. Address signals  $A_{0-1}$  are used to decode 4 bytes of the cache line, while  $A_{2-15}$  create the 16k words of address space.

This design uses two Texas Instruments ACT2163 16k-by-5-bit, 20-ns tag comparators. Address lines  $A_{2-15}$  are connected to both of the tag comparators and the SRAM address bus. Also,  $A_{16-20}$  and  $A_{21-24}$  are connected to the data inputs of the two cache tag comparators to store the page

# CACHE MEMORY DESIGN

number.

MATCHA and MATCHB are the output signals of the tag comparators. These signals go high on a compare cycle if the current processor address  $A_{2-24}$  matches a previously stored tag. Address tags are stored each time the processor writes to the cache.

The DRAM controller used in the existing system supports three modes. A standard DRAM page-mode access requires a precharge time delay before the controller accesses a new row and column. This access takes 7 clock cycles at 33 MHz and is the slowest mode. Fast page mode is used if the controller accesses a new column in the same row. This is the fastest access, taking only 3 clock cycles. Interleaving mode is used when the controller finds an access in a different row and that row is precharged. It takes only 5 clock cycles/access in this mode.

## DIFFERENT MODES

The old non-cache system runs at an average speed of about 5 cycles/access with the DRAM memory. The three different modes that the controller uses with the DRAM in the old system will average 5 clock cycles/access. This average access time is calculated as  $(3 + 5 + 7) \div 3 = 5$ , assuming that each mode has an equal chance to occur.

In the new system, when the cache controller misses, the processor has to get the information from the DRAM main memory. On a cache miss, the cache controller adds two more clock cycles to the DRAM access because it enters the retry mode of the 68030. The retry mode is used because the 68030 samples the Synchronous Termination signal  $\overline{STERM}$  before a valid result of the miss or hit signal. The total average access time in a miss is  $5 + 2 = 7$  clock cycles/access.

With a cache hit, the system can run at 2 clock cycles/access. This design gives an average hit/miss ratio of about 85%. The average read access time becomes  $(2 \times 0.85) + (7 \times 0.15) = 2.75$  clock cycles/access assuming 85% SRAM accesses and 15% DRAM accesses.

The cache controller uses the write-through method in which the controller writes every time to the DRAM and to SRAM. This technique minimizes the memory write time. The average write time, 5 clock cycles/access, is the same as for the DRAM access because the retry operation isn't needed.

Overall read and write speed for the new system can be calculated from the read accesses and the write accesses. It averages about  $(2.75 + 5) \div 2 = 3.875$  clock cycles/access. The percentage performance increase over the non-cache system is  $5 \div 3.875 = 29\%$ .

The upgraded system board is the old system with the 68030 unplugged from its socket (Fig. 2). The cache daughterboard includes the static RAM, a control block containing the cache controller that arbitrates between the old system resources and the cache system memory, and the 68030 microprocessor. A 169-pin connector links the old system to the daughterboard through the existing processor socket.

The address bus, which is the 68030 address bus ( $A_{0-24}$ ), is shared by the processor, the cache system,

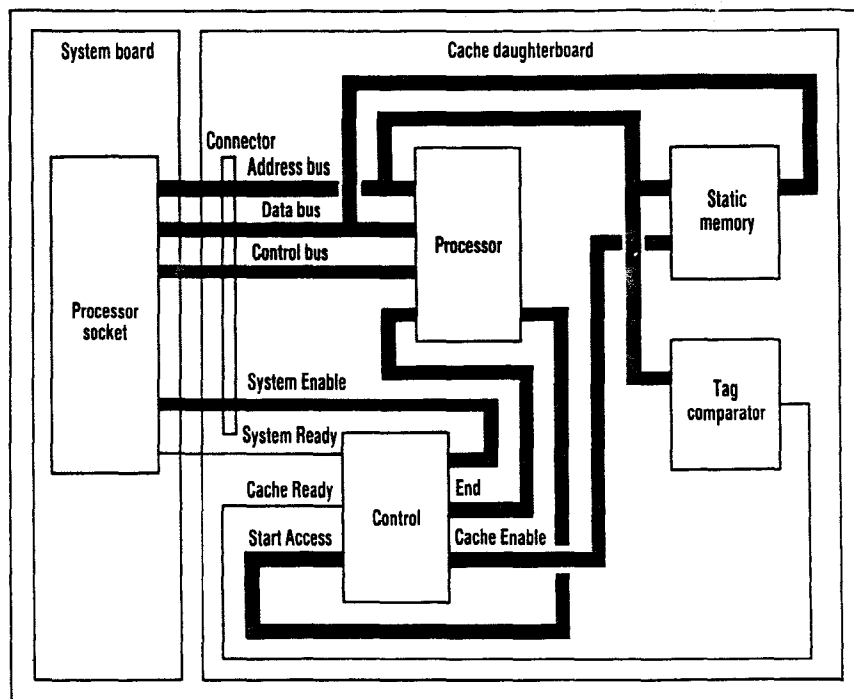
and the old system. The data bus is the 68030 data bus ( $D_{0-31}$ ) and is also shared by the processor, cache system, and the old system. The control bus consists of 68030 control signals that are shared by both the cache board and the system board. Those signals are Clock, Reset, Read/Write (R/W), and the dynamic bus-sizing signals.

The Start Access signal is the Address Strobe ( $\overline{AS}$ ) signal from the 68030 microprocessor that indicates a valid access.  $\overline{AS}$  indicates a valid address on the bus, and is connected to the cache controller to indicate a start of an access.

## A CACHE HIT

Cache Ready is an output of the tag comparator that indicates a cache hit if the information can be retrieved from the SRAM. The Cache Enable signal is an output of the cache controller to enable the SRAM so that the processor can get the information from it.

The cache controller drives the System Enable signal active if the processor needs information from the old system that has the DRAM and the peripherals. System Enable



**2. AN EXISTING SYSTEM** is upgraded by plugging a cache daughterboard into the system's microprocessor socket. The processor is relocated to the daughterboard.

# CACHE MEMORY DESIGN

is the Address Strobe signal to the old system. Called  $\overline{\text{SYSAS}}$ , it starts accesses on the old board.

System Ready is used on the old system board for the bus cycle termination signal  $\overline{\text{DSACK}}_{0-1}$ . On the cache board, it's called  $\overline{\text{SYSDSACK}}$  and tells the controller that the information is ready from the system board. The controller generates  $\overline{\text{STERM}}$  and  $\overline{\text{DSACK}}_{0-1}$  synchronous and asynchronous termination signals to end the processor access.

The cache controller has a special arbitrator that allows the 68030 to choose between the added system's SRAM memory and the old system's DRAM or peripherals. The arbitrator monitors the processor's Start Access signal ( $\overline{\text{AS}}$ ), Clock, and the address. It then generates a Cache Enable signal if the controller accesses the cache and generates System Enable if the controller accesses the old system memory.

The output of the cache tag comparator tells the arbitrator if the data can be found in the SRAM and gives a cache hit. In a cache hit, the cache controller ends the access by generating  $\overline{\text{STERM}}$  to the 68030.

In a cache miss or a peripheral access, the arbitrator generates another start signal for the system board called  $\overline{\text{SYSAS}}$ . This signal is connected to the system board's  $\overline{\text{AS}}$  signal in the interface socket. The system board sees  $\overline{\text{SYSAS}}$  as a Start Address strobe from the processor, and responds to it with the asynchronous ter-

mination signal  $\overline{\text{SYSDSACK}}$  when data is ready from the DRAM or a peripheral device.

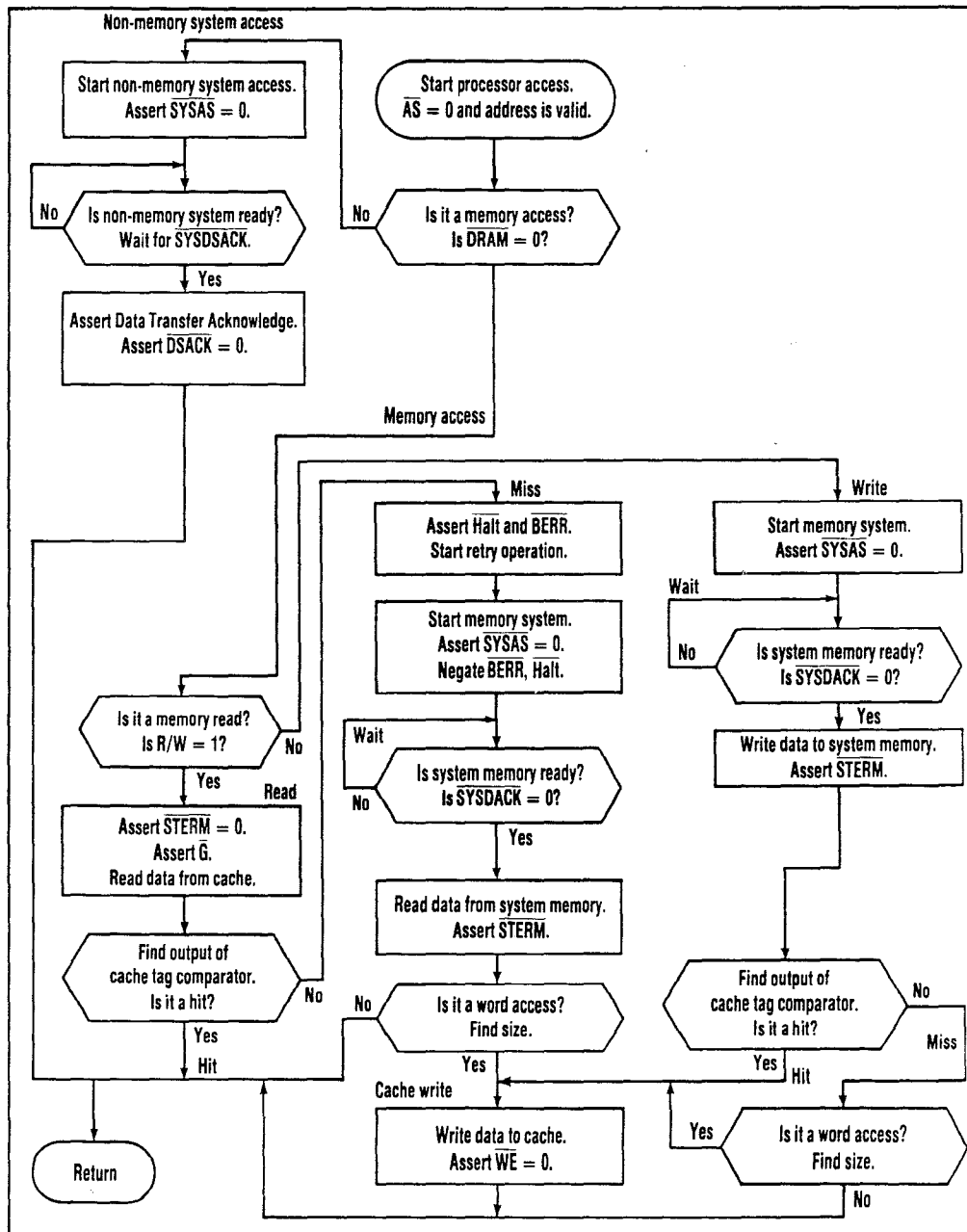
The cache controller regenerates  $\overline{\text{STERM}}$  after receiving  $\overline{\text{SYSDSACK}}$  on a DRAM access. It passes the  $\overline{\text{SYSDSACK}}$  signal directly to the processor as the  $\overline{\text{DSACK}}$  signal on a peripheral access.

The 68030 processor supports synchronous bus cycles that are terminated with the Synchronous Termination signal  $\overline{\text{STERM}}$ . These cycle

are for 32-bit ports and take a minimum of two clock cycles. The bus cycle is synchronous if:

- Data Transfer Acknowledge ( $\overline{\text{DSACK}}$ ) is not asserted.
- The port size is 32 bits.
- Synchronous-input setup and hold time for  $\overline{\text{STERM}}$  is met.

Synchronous mode is used to ru-



**3. THE DIFFERENT PATHS** taken by the cache-based system for memory and non-memory accesses are shown in a flow chart.

# CACHE MEMORY DESIGN

the processor at two cycles in the SRAM. To avoid wait states, the system must assert the  $\overline{STERM}$  signal before the rising edge of the second clock cycle. Data must be valid 0 ns before the falling edge of the second clock so that the processor latches valid data. The cache controller asserts the Synchronous Termination ( $\overline{STERM}$ ) signal early and doesn't wait for Address Strobe ( $\overline{AS}$ ).

Asynchronous operation is used for data transfer to peripherals on the main system board. The system uses  $\overline{AS}$ ,  $\overline{DS}$ , and  $\overline{DSACK}$  to control data transfers.

$\overline{AS}$  signals the start of a bus cycle, and  $\overline{DS}$  is used as a condition for valid data on a write operation. Decoding the 68030 size output signals and  $A_{0-1}$  provides strobes that select the active portion of data bus. The peripheral then responds by placing the requested data on the correct portion of the data bus and asserting the  $\overline{DSACK}$  signals to terminate the cycle.

## RETRY OPERATION

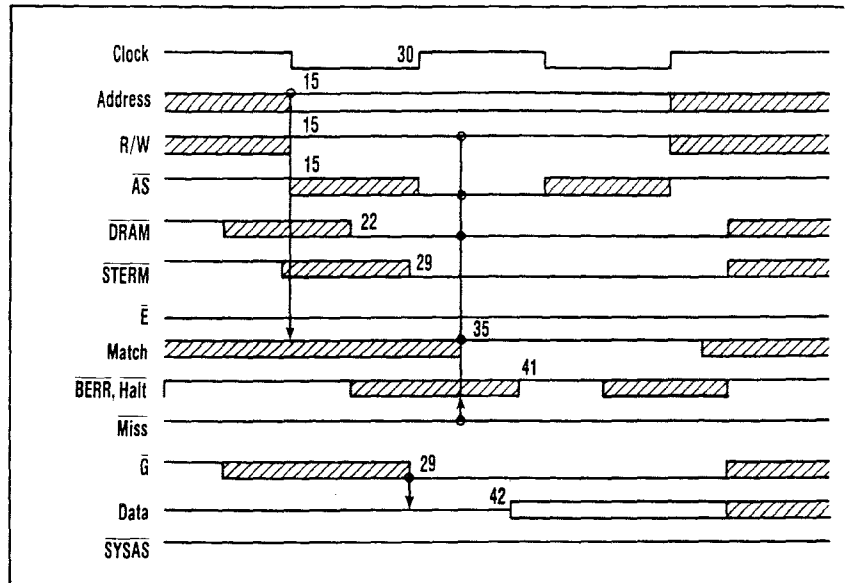
On a cache miss, the cache controller forces a retry operation to prevent the processor from latching bad data. The system asserts  $\overline{STERM}$  and doesn't wait for the output of the tag comparator to become valid, avoiding a wait state.

The cache controller asserts  $\overline{BERR}$  and  $\overline{Halt}$  during a bus cycle so that the processor enters the retry sequence. The processor samples the  $\overline{BERR}$  and  $\overline{Halt}$  signals on the falling edge of the second clock cycle. It also terminates the bus cycle and places the control signals in their inactive states. The processor retries the previous access when  $\overline{BERR}$  and  $\overline{Halt}$  are negated.

The decoder PAL device asserts  $\overline{DRAM}$  if the accesses are to the memory (Fig. 3). The PAL equation to generate the memory decode signal is:

$$\overline{DRAM} = \overline{A_{24}} * \overline{A_{23}}$$

If  $\overline{DRAM}$  is false, the access is to non-memory devices. On the start of processor access, the control PAL device monitors  $\overline{AS}$  and  $\overline{DRAM}$ . It asserts  $\overline{SYSAS}$  if the access is to the



4. A CACHE HIT occurs when the  $\overline{Miss}$  signal is false. The processor can read data located in the SRAM when the  $\overline{G}$  signal is asserted.

non-memory devices. The PAL equation is:

$$\overline{SYSAS} = \overline{DRAM} * \overline{AS}$$

$\overline{SYSAS}$  is the Start Access signal, previously  $\overline{AS}$  in the old system. Assertion of  $\overline{SYSAS}$  causes the system to start the non-memory access.

The write PAL device then waits for  $\overline{SYSDSACK}$ , which comes from the old system when it has valid data. The PAL device asserts the  $\overline{DSACK}$  signals to end the processor cycle. The PAL equation is:

$$\overline{DSACK} = \overline{DRAM} * \overline{SYSDSACK}$$

The control PAL device asserts  $\overline{STERM}$  when the processor starts a memory read.  $\overline{STERM}$  comes early enough to be sampled by the processor's second clock rising edge for a synchronous access with zero wait states. The PAL device examines the results of the tag comparator's output and generates  $\overline{Miss}$  if it's false. It also asserts a read enable signal called  $\overline{G}$  to the SRAM. The PAL equations are:

$$\overline{STERM} = \overline{DRAM} * \overline{R/W} * \overline{Miss}$$

$$\overline{G} = \overline{DRAM} * \overline{AS} * \overline{Miss}$$

$\overline{DRAM}$  is generated by the address, and is valid at: time = address valid time + PAL delay = 15 + 7 = 22

ns (Fig. 4).  $\overline{DRAM}$  and  $\overline{R/W}$  generate the  $\overline{STERM}$  signal, which is valid at: time =  $\overline{DRAM}$  valid time + PAL delay = 22 + 7 = 29 ns. The microprocessor samples  $\overline{STERM}$  at the rising edge of  $S_2$  at 30 ns for zero-wait-state access.

During an SRAM or DRAM read cycle, the cache controller asserts  $\overline{STERM}$  and  $\overline{G}$ . It then generates  $\overline{Miss}$  state signal if  $\overline{MATCHA}$ ,  $\overline{MATCHB}$ , or both are false. The  $\overline{Miss}$  state is generated from a PAL register clocked by the falling edge of the clock.

The decoder PAL device examines  $\overline{MATCHA}$  and  $\overline{MATCHB}$ , and asserts  $\overline{BERR}$  and  $\overline{Halt}$  if a miss occurs. The processor samples  $\overline{BERR}$  and  $\overline{Halt}$  on the falling edge of the second clock, and enters the retry operation. It stays idle for two cycles.  $\overline{Miss}$  becomes active and negates  $\overline{BERR}$  and  $\overline{Halt}$ , therefore the retry time is minimum.

$\overline{SYSAS}$  is asserted by the control PAL device as soon as  $\overline{Miss}$  is generated. It doesn't wait for the processor to finish the retry mode.  $\overline{SYSAS}$  starts the DRAM access from the system memory, thereby interleaving it with the cache accesses. The PAL equations are:

$$\overline{Miss} = \overline{DRAM} * \overline{AS} * \overline{R/W}$$

# CACHE MEMORY DESIGN

$$\frac{(\text{MATCHA} * \text{MATCHB})}{\text{MATCHB}}$$

$$\text{Halt} = \text{BERR} = \text{DRAM} * \text{AS} * \text{R/W} * \frac{(\text{MATCHA} * \text{MATCHB})}{\text{MATCHB}} * \text{Miss}$$

$$\text{SYSAS} = \text{DRAM} * \text{Miss}$$

The old system asserts  $\overline{\text{SYSDSACK}}$  when it completes the DRAM access. The control PAL device waits until  $\overline{\text{SYSDSACK}}$  comes before asserting  $\overline{\text{STERM}}$  to end the access. The PAL equation is:

$$\overline{\text{STERM}} = \text{DRAM} * \text{Miss} * \overline{\text{SYSDSACK}}$$

Byte read or write from the SRAM is possible because the cache controller writes the data that comes from the system DRAM to the SRAM if it's a word. The PAL equation is:

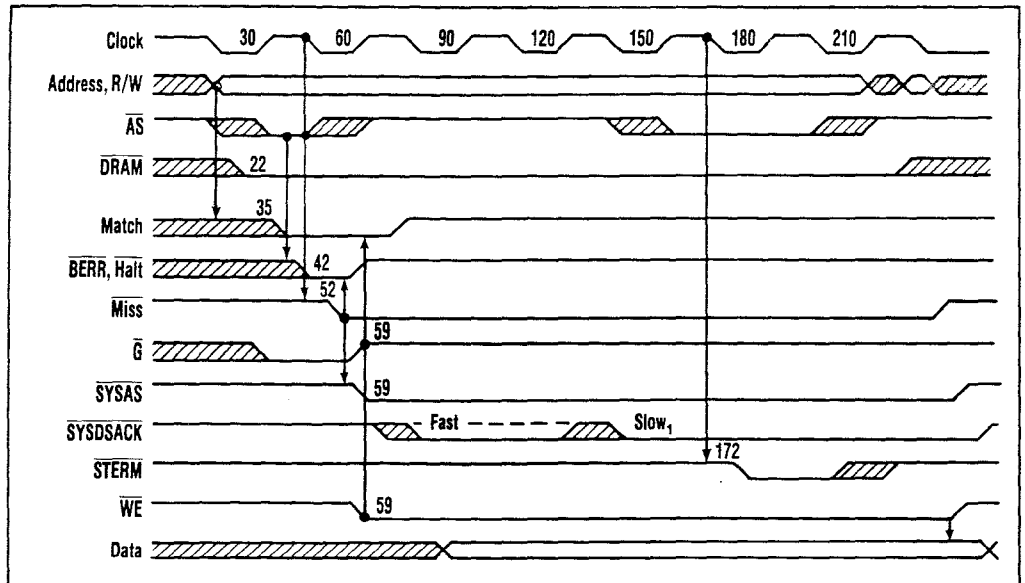
$$\text{WE} = \text{Miss} * (\overline{\text{A}_0} * \overline{\text{A}_1} * \overline{\text{Size}_0} * \overline{\text{Size}_1})$$

The rising edge of  $\overline{\text{Miss}}$  makes  $\overline{\text{WE}}$  go from low to high and writes the data into the SRAM.

$\overline{\text{MATCHA}}$ ,  $\overline{\text{MATCHB}}$  comes from the address tag comparators, and is valid at: time = address valid + access time = 15 + 20 = 35 ns (Fig. 5).  $\overline{\text{BERR}}$ ,  $\overline{\text{Halt}}$  comes from  $\overline{\text{MATCHA}}$ ,  $\overline{\text{MATCHB}}$ ,  $\overline{\text{DRAM}}$ ,  $\text{R/W}$ , and  $\overline{\text{AS}}$  at: time = match valid + PAL delay = 35 + 7 = 42 ns. The processor samples  $\overline{\text{BERR}}$ ,  $\overline{\text{Halt}}$  at the falling edge of  $\text{S}_2$ , which occurs at 45 ns.

$\overline{\text{Miss}}$  is generated from the clock's falling edge and  $\overline{\text{AS}}$  at: time = falling edge of  $\text{S}_2$  + PAL delay = 45 + 7 = 52 ns.  $\overline{\text{BERR}}$ ,  $\overline{\text{Halt}}$  becomes false from  $\overline{\text{Miss}}$  at: time =  $\overline{\text{Miss}}$  valid + PAL delay = 52 + 7 = 59 ns. The processor samples  $\overline{\text{BERR}}$ ,  $\overline{\text{Halt}}$  by the falling edge of the clock at 75 ns. In addition,  $\overline{\text{SYSAS}}$  is generated by  $\overline{\text{Miss}}$  at: time =  $\overline{\text{Miss}}$  valid + PAL delay = 52 + 7 = 59 ns.  $\overline{\text{SYSDSACK}}$  comes from the system DRAM controller before the clock's falling edge.

The cache system supports byte access from the SRAM to increase



**5. IF THE TAG COMPARATORS** don't find a match, a cache miss occurs;  $\overline{\text{BERR}}$ ,  $\overline{\text{Halt}}$  is asserted; and the processor reads data from the main memory.

system performance. To support the byte read from the cache memory, the controller uses byte writes to the static RAM in a write access with a hit. In a memory write with a miss, the cache controller writes the data to the SRAM if it's a word access. The PAL equations for a write operation are:

$$\text{SYSAS} = \text{DRAM} * (\text{R/W}) * \text{AS} * \overline{\text{SYSHALT}}$$

$$\overline{\text{STERM}} = \text{DRAM} * (\text{R/W}) * \text{AS} * \overline{\text{SYSDSACK}}$$

$$\text{WE} = \text{DRAM} * (\text{R/W}) * \text{AS} * \frac{(\text{MATCHA} * \text{MATCHB}) + \text{DRAM} * (\text{R/W}) * \text{AS} * (\overline{\text{A}_0} * \overline{\text{A}_1} * \overline{\text{Size}_0} * \overline{\text{Size}_1})}{\text{Size}_0 * \text{Size}_1}$$

On a write to memory, the control PAL asserts  $\overline{\text{SYSAS}}$  to start the DRAM access. Control PAL waits for the  $\overline{\text{SYSDSACK}}$  (DRAM is ready) signal, then it asserts  $\overline{\text{STERM}}$  to finish the cycle. The control PAL generates the  $\overline{\text{WE}}$  signal to do byte write to the SRAM if the accesses are a write with a hit. In a write with a miss, the controller asserts  $\overline{\text{WE}}$  to do only word write if  $(\overline{\text{A}_0} * \overline{\text{A}_1} * \overline{\text{Size}_0} * \overline{\text{Size}_1})$  is true.

$\overline{\text{SYSDSACK}}$  generates  $\overline{\text{STERM}}$  on DRAM accesses and when  $\overline{\text{STERM}} = \overline{\text{SYSDSACK}}$  valid + PAL delay = 45 + 7 = 52 ns. The processor samples  $\overline{\text{STERM}}$  at the clock's rising

edge at 60 ns.

The cache controller improves the system refresh time by hiding the refresh in memory or peripheral accesses. The old system uses the  $\overline{\text{Halt}}$  signal to refresh the DRAM. With the new design, the cache controller can monitor  $\overline{\text{SYSHALT}}$  (the old  $\overline{\text{Halt}}$  that goes to the processor socket in the old system board) and doesn't assert a  $\overline{\text{Halt}}$  signal to the processor. The controller, instead of halting SRAM or peripheral accesses, allows them to continue.

The cache controller uses the  $\overline{\text{SYSAS}}$  and  $\overline{\text{SYSHALT}}$  signals to stretch a system DRAM access if the refresh is occurring during a memory access. If a refresh has already started, the controller delays the start of a system DRAM access until the refresh is finished. □

*Nagi Mekhiel, a professor in the Electrical Engineering Dept. at Ryerson Polytechnical Institute in Toronto, has a BSEE from the University of Assiut, Egypt, and an MSEE from the University of Toronto.*

HOW VALUABLE?	CIRCLE
HIGHLY	541
MODERATELY	542
SLIGHTLY	543