FPGA memory controller links embedded μP to cache-enhanced DRAM

JAMES JOSEPH, RAMTRON INTERNATIONAL CORP, AND CHARLES BROWN, INTEL CORP

DRAM chips often lack the performance that embedded systems require. Also, SRAM takes up too much space and is too expensive for any application using over 1 Mbyte of memory. But designers have another option: a memory built with DRAM enhanced with an on-chip cache.

Twice as fast as most standard DRAMs, an "enhanced DRAM" (EDRAM) combines as many as 4 Mbits of 35-nsec DRAM with a 2-kbit, 15-nsec SRAM cache on one chip (**Fig 1**). Because of the built-in cache, some varieties of this chip can complete zero-wait-state read cycles at clock rates as high as 50 MHz—without interleaving memory banks. Interleaving allows clock speeds as high as 100 MHz.

Their internal caches give the memory devices some unusual properties compared with conventional DRAM. First, as long as the system is reading or writing from the device's caches, refreshing the DRAM can proceed in the background without interrupting memory accesses. Second, and more subtly, the devices internally latch the RE signal. Thus, the memory controller can unassert the external RE signal and begin precharging the next memory access earlier than would be the case with conventional DRAM.

System sketch

To take advantage of EDRAMs, you need a controller tailored to the memory's particular operations. Fortunately, you can make such a controller from a single field-programmable gate array (FPGA). You must carefully select the FPGA to meet the memory subsystem's tight timing constraints. Specifically, you can build a 32-bit RISC system having as large as a 16-Mbyte memory using EDRAM SIMMs and one FPGA controller. Such a controller can access 4 to 16 Mbytes of EDRAM without incurring wait states.

To see how a one-chip memory controller can meet practical size, performance, and cost specifications, consider an embedded system built around a 33-MHz i960CF μP (**Fig 2**). This 33-MHz, 32-bit RISC μP incorporates a 4-kbyte instruction cache, 1-kbyte data cache, and 1-kbyte data memory. Its external nonmultiplexed 32-bit address and data buses allow burst-read and -write data transfers to external memory at rates as high as 132 Mbytes/sec.

In this design, as many as 16 Mbytes of EDRAM fit into two 72-pin SIMM sockets, taking 3 in.^{2} of board space. The 33-MHz i960 needs 15-nsec EDRAM chips. For lower cost, a 25-MHz μP works with less expensive 20-nsec chips.

Figs 3, **4**, and **5** show a worst-case timing analysis performed with Chronology's Timing Designer software using timing parameters entered from the data books for the design's components. **Figs 3** and **4** show the timing waveforms needed to describe the design's read and write cycles. The labels on the vertical axes of **Figs 3** and **4** and the third column of **Fig 5** match the nomenclature of each device's maker. The devices in this design are derated to allow them to drive added load capacitance.

(Ed Note: Although we tried to make the symbols in the article match the symbols in the devices' literature, one exception remains: the inversion operator—a leading slash ("/")—in the listing. Except for PALASM, programmable-device programming languages' inversion operators react to signals defined as active-low in eccentric and unwelcome ways. Consequently, signals defined as active-low by a leading slash or a trailing number sign in their literature (for example, /BLAST or RDY#) appear with a

BBS The listings this article mentions are available on the *EDN* Readers' BBS. Phone (617) 558-4241 with modem setting 1200/2400 8,N,1 (9600 baud=(617) 558-4580). From the Main System Menu, enter *ss/freeware*. Then, from the /freeware SIG menu, enter *rkms818*.

EMBEDDED SYSTEMS

trailing underscore (for example, BLAST_ or RDY_) in the listing. In the listing, a leading slash simply and unequivocally inverts the logic variable it precedes (for example, /BLAST_).

The timing analyses might be hard to interpret if you do not know that i960 μP does not distinguish between burst and single-cycle memory operations. Some μPs' status lines indicate one condition for burst memory operations and another for single-cycle memory operations. Instead, the i960 performs only burst memory operations but happily allows "bursts" of length one—a single-cycle operation, in effect. Thus, you need examine only burst memory operations that the **figures** present.

Critical timing

The most critical timing requirements that the system's memory controller must meet are a 6-nsec max clock-to-output delay and a 6.5-nsec data-setup time. Achieving this performance is difficult for most complex PLDs (CPLDs) and FPGAs. An Intel iFX780-10 FPGA can meet these requirements.

This 132-pin FPGA is an 80-macrocell device that features a fixed 10-nsec propagation delay between all input (or I/O) and output pins. Internal registers' synchronous setup times are not more than 6.5 nsec, and clock-to-output delays are as short as 6 nsec.

For the μP to take advantage of the memory's high speeds, the memory controller must correspondingly swiftly address multiplexing and decoding and refresh cycles.

The controller must also perform functions for the μP and memory to communicate—including cache-hit/miss comparisons. The memory internally performs its own cache-hit/miss comparisons. But the memory does not externally signal a cache miss. Therefore, the controller registers the upper half of the address it read last. The controller then compares that address with the current read address to deduce if the memory will read from its cache or DRAM array. If the read operation accesses the memory's DRAM array, the controller commands the μP to wait.

In short, using a simple single-phase clock, the controller must supervise all of the μP's memory transactions and incur a minimum of memory wait states.

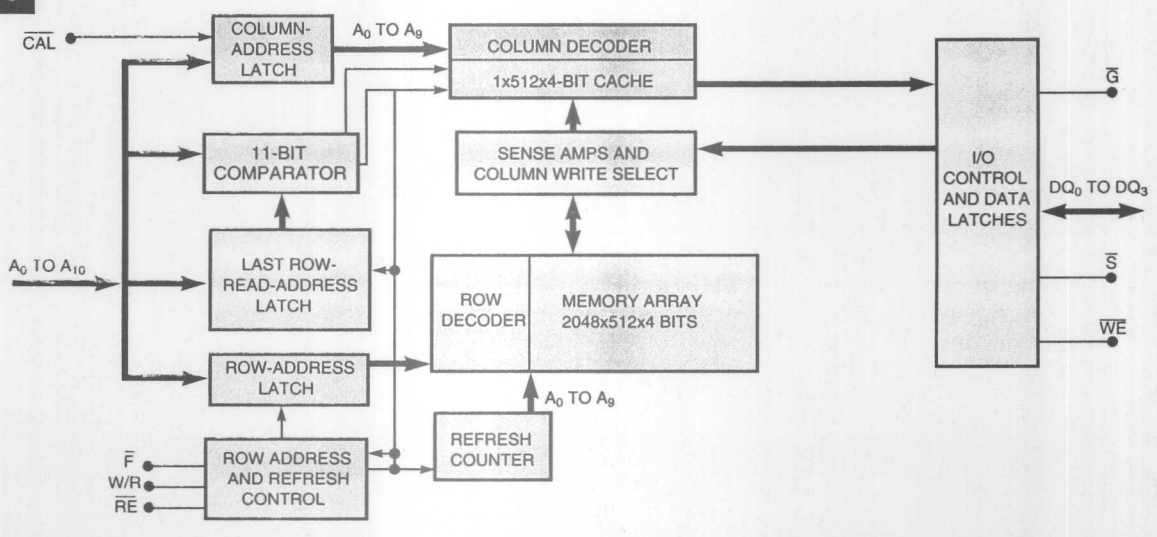
These memory transactions include one to four 32-bit-word read operations and 1- to 4-byte, word, and long-word write operations.

For each of these transactions, the FPGA's 6-nsec clock-

TABLE 2—CONTROLLER-TO-MEMORY SIGNALS

Controller	Memory
MAL0 to 9	Multiplexed address, Bank 0, 1
MAH0 to 9	Multiplexed address, Bank 2, 3
MALA10	Bank 0, MA10
MALB10	Bank 1, MA10
MAHA10	Bank 2, MA10
MAHB10	Bank 3, MA10
/RF0 to 3	Row enables for banks 0 to 3
/CAL0 to 3	Column-address latch inputs for bytes 0 to 3
W/R0, 1	Write/read-mode input for low/high banks
/F0, 1	Refresh-mode input for low/high banks
/S0 to 3	Chip selects for banks 0 to 3
/G0, 1	Output enable for low/high banks
/WE0, 1	Write enable for low/high banks

FIGURE 1



Twice as fast as most standard DRAMs, an "enhanced DRAM" (EDRAM) combines up to 4 Mbits of 35-nsec DRAM and a 2-kbit, 15-nsec SRAM cache on one chip. In addition, the device's DRAM-to-cache bus is 256 bytes wide.

to-output delay and 6.5-nsec data-set-up time allow zero-wait-state, noninterleaved operation.

Dealing out the loads

A fast FPGA, however, is not enough. Achieving top performance requires that the design divide address and control loading between two address buses, each driving as many as two memory banks of 8 Mbytes each. Similarly, the design splits heavily loaded signals that drive both memory banks, such as W/R, /F, /WE, and /G, into two parts.

In contrast, the /CAL0 through /CAL3 signal lines are lightly loaded and can, therefore, drive all four memory banks from one pin. **Tables 1** and **2** show the μP's address and control signals that feed the memory controller as well as the memory signals the controller must generate.

TABLE 1—CONTROLLER-TO-PROCESSOR SIGNALS

Controller	Processor
A2 to 31	Address bus
BE#0 to 3	Byte enables
ADS#	Address strobe
W/R#	Write/read mode
BLAST#	Burst-last 2
RDY#	Non-burst-ready acknowledge
BTERM#	Burst-ready acknowledge
Reset#	Processor-reset input
PCLK	Processor's clock output

Around the blocks

The controller's main blocks (**Fig 6**) comprise a state machine (**Fig 7**), an optional refresh counter, an address decoder, an address multiplexer, and a burst-address generator. **Listing 1** (see pg 136) contains the equations for configuring the state-machine portion of the controller—the most complex section. Dittos indicate repetitious sections of the listing, which are cut to save space. You can find the full listing the *EDN* Readers' BBS, along with a version of the memory controller's program for an 80486 as MS818 on the /freeware Special Interest Group. Using the i960 and the 80486 versions for reference, you should be able to adapt the design to other μPs.

State machine's core

At the heart of the controller is the state machine, which executes the memory's control sequences (**Fig 7**). When the system issues a reset command, the controller initializes its logic. Then, for as long as the system holds its reset line low, the controller performs only refresh cycles.

Many designers provide a signal specifically for memory refreshing. If an external, 16-kHz refresh signal is not available, then the FPGA has enough spare logic to divide a clock signal down internally.

Listing 1 and **Fig 7** show that, when the state machine clears the memory's refresh request by entering the I2 state, the controller does not perform the next refresh until the memory again asserts REF_. In this way, the controller refreshes the memory every 62.5 μsec.

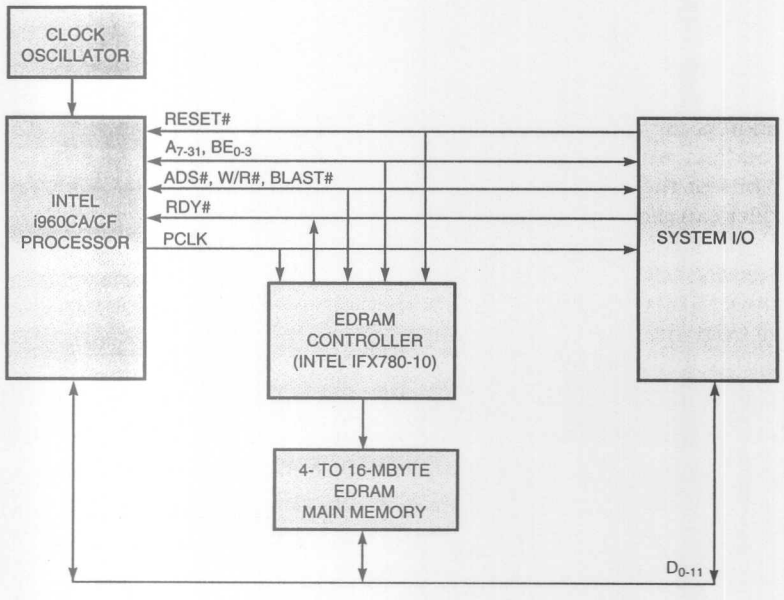
Initializing the memory is simple. Upon power-up, the memory requires only two read-miss cycles per bank for initial resetting. The start-up routine in the system's bootstrap ROM usually executes these cycles. Then, when the system releases the reset line, the controller enters its idle state and waits for memory transactions. Subsequently, a series of at least eight refresh cycles resets the memory's logic, but not its contents.

Memory sequences

When the controller's address decoder detects a valid memory address, the controller enables the memory for memory transactions. In this design, the memory uses only the lower 16 Mbytes of the address range. Normally, other memory and I/O devices would map into the remaining address space.

Under control of the state machine, the controller's address multiplexer selects the row, column, or burst address it will apply to the memory's multiplexed address inputs. Again, to limit the capacitive load that the controller must

FIGURE 2



This embedded system uses a 33-MHz Intel i960CF RISC μP. The design's 16 Mbytes of EDRAM fit into two 72-pin SIMM sockets, taking up 3 in.² of board space.

EMBEDDED SYSTEMS

drive, the design divides $MA_{0\ to\ 10}$ between two independent output buses. Using multiple outputs limits the clock-to-output delay to 8 nsec, allowing zero-wait-state operation.

The µP's burst-mode transfers can move one to four words. Consequently, the controller's burst generator toggles the lower two multiplexed address bits, $MA_{0,\ 1}$ appropriately. These bits, along with bits $MA_{2\ to\ 8}$, form the memory's column address.

A memory sequence starts when the address-strobe signal (ADS#) and a valid address are present. The exact control sequence that the state machine executes and the series of bus events that transpires depend on the state of the µP's read/write output; that is, whether the µP is requesting a memory-read or -write operation.

Read operations

In the case of a read, the controller must fetch one to four 32-bit words from the memory. These data can be in either the cache or the EDRAM's internal DRAM array. The controller registers the upper 16 bits of the previous read address. By comparing this registered address with the corresponding upper bits of the current address, the controller can detect read misses. That is, because of the size of the memory's cache relative to the size of its DRAM, if the upper 16 bits of the address do not change from one read to another, then the data are in the cache.

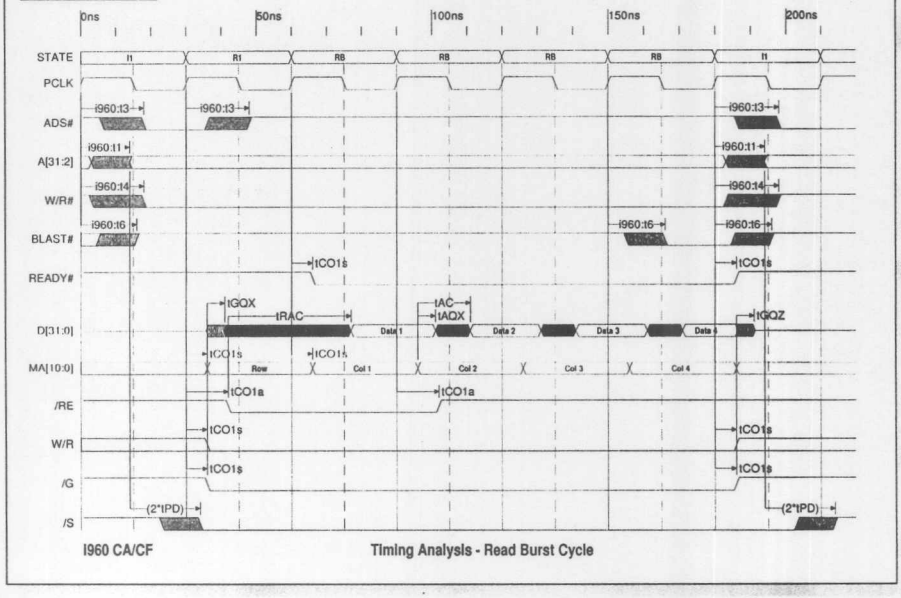
During a memory read, the state machine starts with a transition to state R1. There, the controller presents row address and chip-select signals to the memory and puts the memory in its read mode.

For the controller to access a row of DRAM within the EDRAM chip, the controller must clock the row-enable signal of the selected memory bank 13 to 18 nsec after the start of state R1. Once accessed, the DRAM row transfers its contents internally to SRAM cache.

In the next state, RB, the controller presents the column-address and output-enable signals to the cache memory and returns RDY# and BTERM# acknowledgment signals to the µP, confirming that the controller has accessed the data. All control signals are available within 6 to 8 nsec after the µP clock's rising edge.

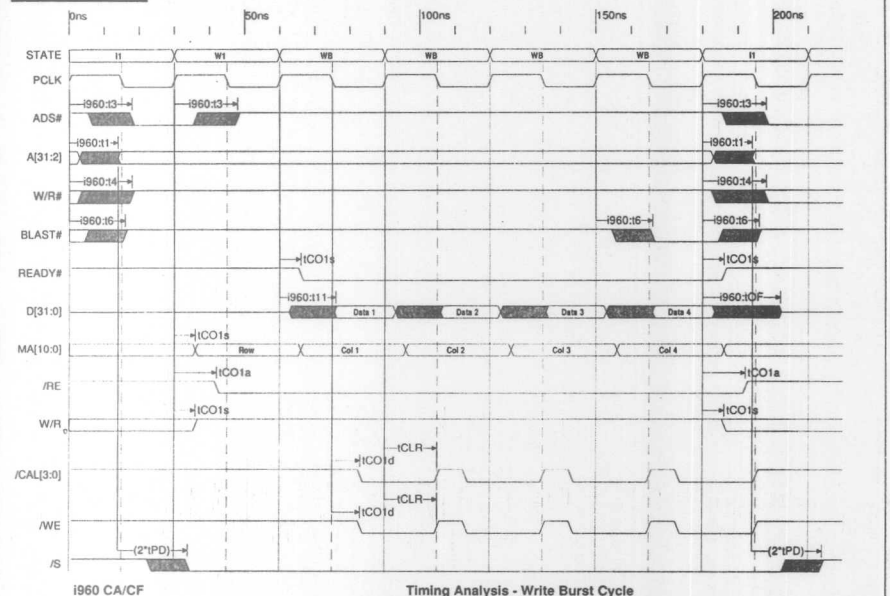
The state machine cycles through state RB one to four times, depending on how many words the µP wants to transfer in a burst. During each new RB state, the controller sends

FIGURE 3



As this timing analysis shows, the system µP can read successive 32-bit words in a burst memory without incurring wait states.

FIGURE 4



Somewhat similar to the read-burst sequence in Fig 3, write-burst sequences transfer all but the first of a burst of 32-bit words with no wait states.

a new burst address to the memory. During this time, output-enable, chip-select, and BTERM# signals all remain valid.

Background refresh cycle

If a refresh request is pending when a burst sequence begins, the state machine lets the memory execute an internal refresh operation during cache-read cycles, taking advantage of the EDRAM's architecture. Such a background refresh operation starts at the second RB state and continues through the third and fourth RB states. During the second and third RB states, the controller brings the /F control signal low as the state machine clocks the memory's row-enable lines.

The memory can complete a background refresh operation during a four-word burst-mode transfer. However, for shorter burst-mode transfers, the background refresh operation will still be in progress when the transfer ends. In this case, the state machine jumps to state F2. This move allows the refresh operation to finish before the controller returns to its idle state.

Like read operations, write operations can happen in one- to four-word bursts. Further, the µP can specify which byte to overwrite in a 32-bit word. The µP's three byte-enable lines determine which bytes to overwrite in memory.

The state machine first advances to state W1. To enable the requested memory locations, the state machine activates the appropriate column-address-latch inputs in the memory bank. Next, the state machine selects the corresponding row-address and chip-select signals for that bank and activates the memory's write mode. The state machine also sends RDY# and BTERM# acknowledgments to the µP. The controller issues a row-enable signal between 13 and 18 nsec after the start of state W1.

Advancing to the next state, WB, the state machine issues the column-address and, to latch the forthcoming data, a write-enable signal. The column-address-latch signals leave the controller from 13 to 18 nsec after the start of state WB.

FIGURE 5

C:\TIMING\PARMS\TABLE_2.TD

Row		Name	Formula	Min	Max	Margin	Comment
1	V	tSUs	[6.50;]	6.50			Input or I/O Setup Time to CLK (Synchronous)
2	V	tSUa	[2,]	2			Input or I/O Setup Time to CLK (Asynchronous)
3	V	tHs	[0,]	0			Input or I/O Hold Time form CLK (Synchronous)
4	V	tHa	[5,]	5			Input or I/O Hold Time form CLK (Asynchronous)
5	V	tCO1s	[,6]		6		CLK to Output Valid (Synchronous)
6	V	tCO1a	[,12]		12		CLK to Output Valid (Asynchronous)
7	V	tCO2s	[,16]		16		CLK to Output Valid Fed Through Combinational Macrocell (Synchronous)
8	V	tCO2a	[,22]		22		CLK to Output Valid Fed Through Combinational Macrocell (Asynchronous)
9	V	tAC	[,15]		15		Column Address Access Time
10	V	tACH	[15,]	15			Column Address Valid to CAL High (Write Cycle)
11	V	tAQX	[5,]	5			Column Address Valid to Output Turn-On
12	V	tASC	[5,]	5			Columnn Address Setup Time
13	V	tASR	[5,]	5			Row Address Setup Time
14	V	tPD	[,10]		10		Input or I/O to Output Valid
15	V	tGQX	[0,5]	0	5		Output Enable to Output Drive Time
16	V	tGQZ	[0,5]	0	5		Output Turn-Off Delay From Output Disabled
17	V	i960:t1	[3,14]	3	14		PCLK to A[31:2] to Output Valid Delay
18	V	i960:t2	[3,16]	3	16		PCLK to BE#[3:0] Output Valid Delay
19	V	i960:t3	[6,18]	6	18		PCLK to ADS# Output Valid Delay
20	V	i960:t4	[3,18]	3	18		PCLK to W/R# Output Valid Delay
21	V	i960:t5	[4,16]	4	16		PCLK to D/C#, SUP# and DMA# Output Valid Delay
22	V	i960:t6	[5,16]	5	16		PCLK to BLAST# and WAIT# Output Valid Delay
23	V	i960:t7	[3,16]	3	16		PCLK to DEN# Output Valid Delay
24	V	i960:t8	[4,16]	4	16		PCLK to HOLDA and BREQ Output Valid Delay
25	V	i960:t9	[4,16]	4	16		PCLK to LOCK# Output Valid Delay
26	V	i960:t10	[4,18]	4	18		PCLK to DACK#[3:0] Output Valid Delay
27	V	i960:t11	[3,16]	3	16		PCLK to D[31:0] Output Valid Delay
28	V	i960:t12	[18,29]	18	29		PCLK to DT/R# Output Valid Delay
29	V	i960:t13	[2,14]	2	14		PCLK to FAIL# Output Valid Delay
30	V	i960:t14	[3,18]	3	18		PCLK to EOP#[3:0] and TC#[3:0] Output Valid Delay
31	V	tIS1	[3,]	3			D[31:0] to PCLK Setup Constraint
32	V	tIS2	[17,]	17			BOFF# to PCLK Setup Constraint
33	V	tIS3	[7,]	7			BTERM# and READY# to PCLK Setup Constraint
34	V	tIS4	[7,]	7			HOLD to PCLK Setup Constraint
35	V	tIH1	[5,]	5			PCLK to D[31:0] Hold Constraint
36	V	tIH2	[5,]	5			PCLK to BOFF# Hold Constraint
37	V	tIH3	[2,]	2			PCLK to BTERM# and READY# Hold Constraint
38	V	tIH4	[3,]	3			PCLK to HOLD Hold Constraint
39	V	tRAC	[,35]		35		Row Enable Access Time, On a Cache Miss
40	V	i960:tOF	[3,22]	3	22		PCLK to ALL SIGNALS Output Float Delay
41	V	tCO1d	[,8]		8		CLK to Output Valid (Delayed)
42	V	tCLR	[,15]		15		Input or I/O to Asynchronous Clear/Preset

Along with Figs 3 and 4, this worst-case timing analysis performed with Chronology Timing Designer software uses timing parameters for the design components. The third column matches the nomenclature of each device's maker.

EMBEDDED SYSTEMS

As long as the μP keeps its burst-write signal, BLAST# high, the μP will continue to issue data in bursts until it releases BLAST#. Depending on the state of BLAST#, the state machine will repeat WB as many as three more times. During each pass through the WB state, the controller generates a new column address. It also activates the memory's write-enable and column-address-latch inputs to write the incoming data to memory.

The last word

After writing the last word, the controller ends the burst-write operation by deactivating the memory's write-enable, column-address-latch, and row-enable inputs.

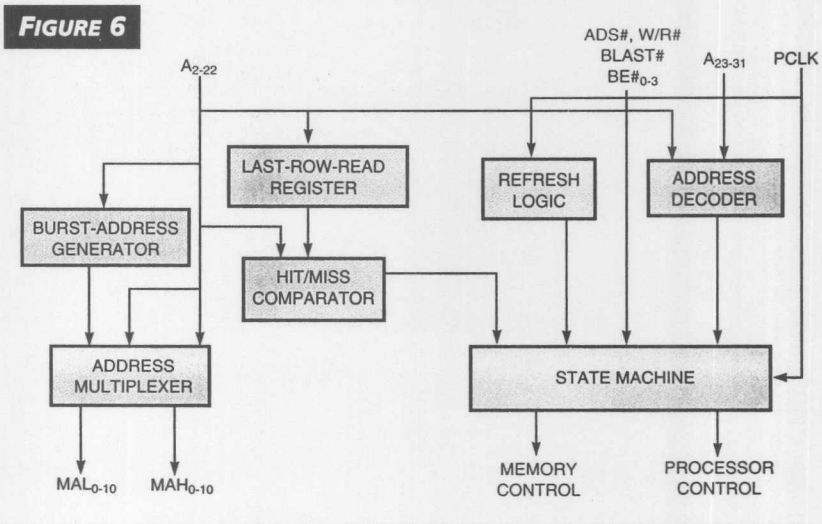
Applications for a fast, 32-bit embedded system such as this one include network bridges and routers, disk-array controllers, laser printers, telecommunication switches, video servers, and image processors. EDN

Authors' biographies

James Joseph is the chief engineer for Ramtron International Corp's specialty RAM operation in Colorado Springs, CO. Joseph has been with the company for four years and is currently specializing in applying the EDRAM. His past efforts include memory controllers for the 80486 and Pentium μPs. Joseph obtained a BS in engineering physics and an MS in physics from Ohio State University, Columbus, OH, and an MSEE and a PhD from the California Institute of Technology, Pasadena, CA.

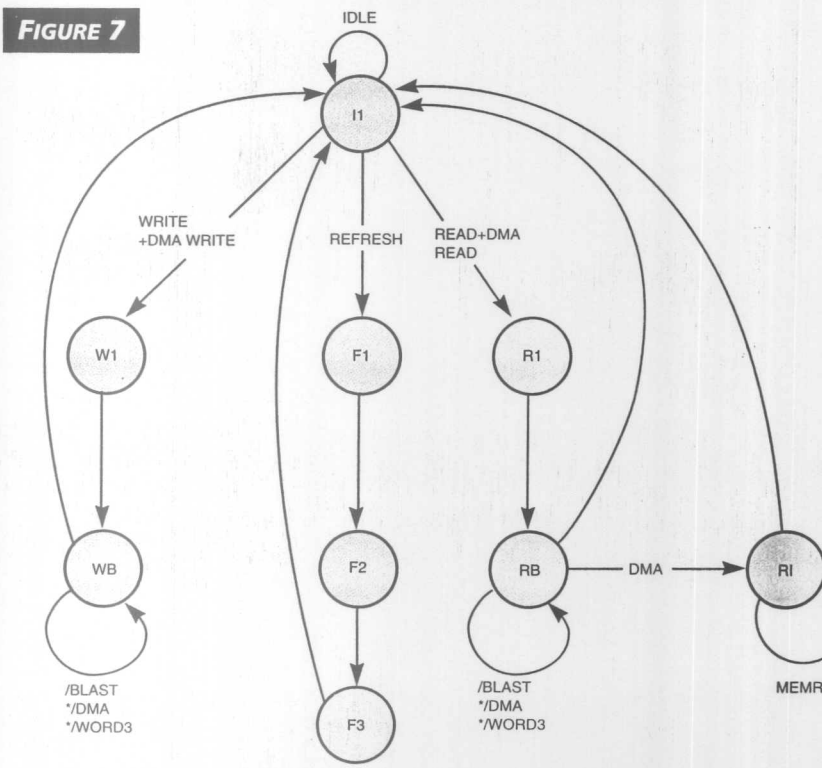
Charles Brown is a senior application engineer for Intel's Programmable Logic Division in Folsom, CA, where he has worked for one year. Brown helped develop the iFX8160 FPGA. He obtained a BSEE from the University of California—Davis, Davis, CA, and an MSEE from Stanford University, Stanford, CA. In his spare time he enjoys golf.

FIGURE 6



A memory controller tailored to EDRAM's operations needs a refresh counter, an address multiplexer, a burst-address generator, an address decoder, and a read/write/refresh state machine. The controller must accept and acknowledge certain commands from the system μP as well as provide properly partitioned commands and data to the memory.

FIGURE 7



This state machine is the heart of the memory-controller FPGA. Starting from reset, the state machine supervises all the memory's control sequences.