U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 5743

Computer Systems Laboratory

Operating Principles of MultiKron Virtual Counter Performance Instrumentation for MIMD Computers

Alan Mink

U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards and Technology
Gaithersburg, MD 20899

November 1995

CMRF

COMPUTER MEASUREMENT
RESEARCH FACILITY
FOR HIGH PERFORMANCE
PARALLEL COMPUTATION

Partially sponsored by the
Advanced Research Projects Agency

NISTIR 5743

Operating Principles of MultiKron Virtual Counter Performance Instrumentation for MIMD Computers

Alan Mink

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

November 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

TABLE OF CONTENTS

Operating Principles of MultiKron Virtual Counter Performance Instrumentation for MIMD Computers

Alan Mink

The MultiKron* and MultiKron II performance instrumentation provided both Trace sampling and Resource Counters, but required a separate measurement data collection facility for collecting sample data. Although providing a large amount of measurement detail, Trace sampling has the disadvantage of requiring additional investment in logic, wires and space to provide for the collection facilities. An alternative measurement approach that would eliminate the need for a collection facility, and its associated cost, is to eliminate Trace sampling and only provide for a very large number of Resource Counters, at the cost of some loss of measurement detail. The MultiKron virtual counter (MultiKron_vc) performance instrumentation chip provides such a feature. Similar in concept to virtual memory, thousands of virtual counters are available but only a small number are real counters that can be active at any one time. Unlike virtual memory, where swapping is transparent to the programmer, due to extra hardware and kernel software support, swapping of counter blocks must currently be handled by the programmer.

Key words: Computers; hardware support; MIMD; multiprocessor computers; performance characterization; VLSI.

INTRODUCTION

The MultiKron* [MIN92] and MultiKron II [MIN94] performance instrumentation provided both Trace sampling and Resource Counters, but required a separate measurement data collection facility for collecting sample data. Such data collection facilities have taken the form of local dedicated memory as well as a cable to a separate workstation. Although providing a large amount of measurement detail, Trace sampling has the disadvantage of requiring additional investment in logic, wires and space to provide for the collection facilities. The capital investment on a large massively parallel computer was estimated to be approximately 5% of the cost of the machine. An alternative measurement approach that would eliminate the need for a collection facility, and its associated cost, is to eliminate Trace sampling and only provide for a very large number of Resource Counters, at the cost of some loss of measurement detail. The MultiKron virtual counter (MultiKron_vc) performance instrumentation chip provides such a feature. Similar in concept to virtual memory, thousands of virtual counters are available but only a small number are real counters that can be active at any one time. Unlike virtual memory, where management is transparent to

---

* MultiKron is a trademark of NIST.

This work was partially sponsored by the Advanced Research Projects Agency.

the programmer, due to extra hardware and kernel software support, management of a "page" of counters must currently be handled explicitly by the programmer.

This evolution of performance measurement instrumentation is part of the parallel processor performance project at NIST [CAR88, CAR89, MIN90, MIN92, MIN93,MIN94,MIN95,ROB89]. The goal of the project is to characterize the performance of parallel computers as well as uniprocessors. The focus of our instrumentation work is to provide hardware support in obtaining performance measurement data with tolerable perturbation to both the processes executing and the architecture on which they are executing.

The body of this report provides a brief background discussion about performance measurement and then goes on to describe the features of the MultiKron_vc and it processor interface.

BACKGROUND

Tracing events and counting are two basic concepts of performance measurement. The concept of tracing events is to be able to follow different execution threads and to know when various events (in the execution of the code) occur and to be able to correlate their times of occurrence. The concept of counting provides the basic mechanism for clocks, stop watches, histograms, etc., and can also tally events at high frequency rates; a must for high speed hardware events. The exploration of instrumentation to provide these basic concepts in various ways while under the constraints of low perturbation, low cost, and similar technology has been the focus of our work. Perturbation refers to disturbances caused by the introduction of measurement to the process being measured, also known as the probe affect. If one disturbs the measured program by adding too much extra code, the measured performance could be different from that of the uninstrumented version of the program. Cost refers to the size (amount of real estate required) as well as the production cost of the measurement instrumentation. Integrated circuits (chips) are both small and inexpensive, in mass production, when compared to printed circuit boards. Similar technology refers to the circuit technology used in the measurement instrumentation compared to the circuit technology used in the architecture being measured. If you need faster circuit technology in your measurement instrumentation than in that used in the measured architecture, then there would be no such technology to use when your computer architecture is already using the fastest circuit technology!

Event Tracing

The fundamental measurement function in event tracing is to determine the time of occurrence of various key events in the execution of a computer's tasks (processes) with high precision. These times can be used later to calculate elapsed time, duty cycle, access latency, and the like. The measurement data necessary to specify a trace event we call a Trace sample. External monitoring of instruction addresses fetched is fruitless, since internal caches mask the actual instructions executed and external addresses are *physical* addresses while compilers produce code in terms of *virtual* addresses. Due to these problems, an internal view of program activity is necessary to determine when an event occurs so that measurement data for a Trace sample may be collected. Thus passive monitoring (an all hardware approach) is insufficient. For minimal perturbation of the executing program a *hybrid* approach (hardware assisted software vs. all

software) to event tracing is employed in which a small amount of *embedded code* in the program (known as a trigger or a probe) causes hardware capture of the Trace sample for each event.

Current time is important data to collect about an *event*. Thus a **Timestamp Counter** is needed with a time resolution on the same order as the instruction execution rate and enough precision to avoid counter wrap-around during any experiment. To allow correlation of samples taken by multiple measurement devices the Timestamp counters should all be synchronized.

Many modern computers are composed of a number of nodes. The coupling between nodes may be fairly loose, but each node may consist of a number of processors which are tightly-coupled together. A single measurement chip can efficiently serve all the tightly-coupled processors at a node. The Trace sample must include the identity of the node, the processor (if a node contains more than one processor), and the process.

The MultiKron [MIN92] and MultiKron II [MIN94] performance measurement chips designed by NIST provide facilities to capture Trace samples. By designing the data capture hardware as a memory-mapped device, the probe code could be kept to as little as a single memory write per event. Execution of this probe code informs the measurement hardware of the occurrence of an event and causes capture of the **user specified data** from the write as part of the Trace sample. This user data, which is stored as part of the Trace sample, provides the means to communicate information about the internal program view. The user data will include event identification, which identifies *where* in program execution the event occurred, along with any information necessary to qualify the event. Event identification, as an example, can be specified as a single number which may only have meaning to the experimenter, such as "27" which could indicate a location in the code where a message was just received. Qualifying information could be the size and/or source of that message. The MultiKron and MultiKron II provide for a wire per processor, up to eight processors in a tightly-coupled node, to identify the processor requesting the sample. A **Source Address register** is provided which contains the node and the process identity packed together and should be updated by the operating system at each context switch. Since a single measurement chip may be used to measure a number of tightly-coupled processors, each working on a different process, process identification must be kept for each processor. This is done by a set of Source Address registers. Each processor is assigned its own Source Address register in which node and process identification is kept.

A MultiKron and MultiKron II **Trace sample** consists of the concatenation of the user-written data, the associated Source Address register (for node and process id), the identity of the processor within the node, the Timestamp counter, and the sample type.

Counters

Trace events occur relatively infrequently, typically hundreds or thousands of program instructions apart. Other types of events occur much more frequently, even on every processor clock cycle for short periods. The hybrid approach recommended for event trace support is unsuitable for measuring these very frequent events; they demand a fully-hardware measurement technique. As a compromise between resolution, cost, and data storage requirements, we have chosen to accumulate *counts* as a measure for these events. This amounts to a form of preprocessing of the measurement data, with a reduction in the cost and data storage requirements. Of course some detail is lost. This counter concept is the design basis for the

MultiKron_vc.

Counters can be used for a wide range of measurement if flexible input switching is provided on a per-counter basis. It must be possible to increment each counter by hardware signals such as cache hits or misses, clock cycles during waits for shared-resource access, or events such as message transmissions. A stop-watch capability can be realized by counting clock cycles between events. If the counters can also be incremented by software, they can be used to keep running tallies of items added to, or removed from, software-managed queues or buffers, average frequency-of-use of code, or other frequent software events. During the measurement period, each counter can be selectively enabled and disabled as many times as desired, and can thus, for example, accumulate counts during multiple uses of a piece of code.

The MultiKron and MultiKron II provide for counters via a set of **Resource Counters**. The contents of the Resource Counters can be either read directly or concatenated with a Trace sample to form a Resource sample.

MultiKron_vc FEATURES

To eliminate the need for a collection facility, and its associated cost, the MultiKron virtual counter (MultiKron_vc) performance instrumentation chip does not provide the capability to acquire Trace samples. Instead it provides for a very large number of Resource Counters, at the cost of some loss of measurement detail. The MultiKron_vc provides access for up to 64K virtual counters, although there are only 64 physical counters on the MultiKron_vc chip. The 64 physical counters are divided into 4 banks of 16 counters each, see Figure 1. As with the MultiKron and MultiKron II, flexible input switching is provided on a per-counter basis. To provide the thousands of virtual counters the MultiKron_vc requires a set of dedicated static RAM (SRAM) chips. These SRAM chips are tightly coupled to the MultiKron_vc chip and store all the inactive counters, while the active counters reside in the MultiKron_vc. This is similar to virtual memory where contiguous memory locations are grouped into pages (contiguous counters are grouped into banks), and active memory pages are kept in primary memory (active counter banks are kept in the MultiKron_vc), and inactive memory pages are kept in secondary memory (inactive counter banks are kept in the SRAM). Commands are implemented to SWAP, STORE, and LOAD a counter bank. A SWAP command stores the specified active counter bank into its home location in SRAM and then loads an inactive bank from SRAM, the location of which is specified in the command, into the same active bank in the MultiKron_vc. A STORE command stores the specified active counter bank into its home location in SRAM, leaving the active bank in the MultiKron_vc unchanged. A LOAD command loads an inactive bank from SRAM, from the location specified in the command, into the specified active bank in the MultiKron_vc, overwriting the previous contents of that active bank.

There are a set of bank registers, one register for each bank. These registers contain the home address of the associated bank, along with a valid bit. These registers are both readable and writable, although the valid bit is handled separately. When a bank register is written, the valid bit is automatically set as valid. A separate "invalidate bank" command is required to clear it. When a bank is marked invalid, a STORE command and the store portion of the SWAP command are ignored.

The counters can be treated either as separate 32-bit counters or an even/odd pair may be grouped to form a 64-bit counter. Each even/odd pair is configured independently. Thus one may configure a bank as 16 32-bit counters, 14 32-bit counters and 1 64-bit counter, ... , 2 32-bit counters and 7 64-bit counters , or 8 64-bit counters

As an operational example, one may consider allocating counter banks separately. Such as two banks for the OS kernel and the other two to a separate process. The kernel may use one bank for performance measurements of itself and this bank would always stay resident in the MultiKron_vc. The second kernel bank could be used for system level performance measurements of the current executing process. Thus as part of the context change when processes are switched, this second counter bank would be swapped for one that is associated with the next process to be executed. The other two counter banks could be allocated to the executing process and swapped as desired. On context change, the home address of the current process counter banks can be read by the kernel from the associated MultiKron_vc bank register and stored in its process table. Then these counter banks could be stored in their home locations in SRAM and new counter banks loaded for the new process from the addresses specified in its process table.

When a process is finished executing a part of the code it was measuring, it can store that counter bank in SRAM and leave it there until execution is completed. As the process progresses to other areas of code, it can use other counter banks and store them as needed or switch back to an earlier bank when re-entering previous code areas. All counter banks can be retrieved from SRAM at the end of execution for analysis.

PROCESSOR INTERFACE

The data path between a processing node and the MultiKron_vc is 64 bits wide. On a read by a processor, any unused bit positions return a logical "1". On a write from a processor, any unused bit positions are ignored. We divide the physical address issued by a processor into three fields, the byte field, the MultiKron field, and the device field. The byte field is the least-significant two bits of the address, which specifies a byte within a 4 byte word. Since MultiKron data is either 4 or 8 bytes aligned on 4 byte boundaries these two bits are always zeros for MultiKron access. The MultiKron field is the next 12 address bits which is used directly by the MultiKron_vc. The device field consists of the remaining higher order address bits which must be externally decoded to establish the base address of the MultiKron_vc.

The address mapping for the 12-bit MultiKron_vc address field is shown in Table 2. The four high order address bits of the MultiKron_vc field is the command subfield. Only 0 to 5, of the possible 0 to 15, are defined. Command 0 is a general command. Its low order 8 bits specify the operation and they are sparsely populated. Commands 1, 2, and 4 operate on a single Resource Counter and thus their low order 8 bits are divided into two 4-bit subfields one to specify the bank, b, and the other to specify the counter, n, within that bank. Commands 3 and 5 are bank level operations and their low order 8 bits are also divided into two 4-bit subfields one to specify the bank, b, and the other to specify the operation.

A MultiKron_vc processor interaction is initiated by asserting (LOW) either the READB or WRITEB signals. See Tables 5 and 6 for signal pin assignments and descriptions. The device field address decoder output must be combined with the processor's READ or WRITE signal to produce the corresponding READB or WRITEB and the STARTB signals. After the MultiKron_vc has completed its action, it will assert (LOW) the ACKB signal. The READB or WRITEB signals to the MultiKron_vc must be de-asserted before the MultiKron_vc asserts its ACKB signal. The STARTB signal to the MultiKron_vc is intended to augment the READB and WRITEB signals, so that the READB or WRITEB can be placed and left at their asserted logic level for longer than the MultiKron_vc requires and the STARTB can be asserted and de-asserted rather than READB or WRITEB. If STARTB is always asseted, it has no effect -- READB or WRITEB will control the MultiKron_vc processor interaction. If STARTB is always de-asserted, it will inhibit any MultiKron_vc processor interaction. Normally the ACKB signal will be asseted for one Node clock cycle. When ACKB is asseted any processor write is completed. When ACKB is

de-asserted the output data for any processor read is removed. If the processor requires more time to acquire the output data for a processor read, it should assert (LOW) the HOLDB signal any time before the ACKB signal is asserted. The data and the ACKB will remain active until the HOLDB signal is de-asserted. All signals are synchronized to the positive-going transitions of the Node clock; A(ddress), D(ata), STARTB, HOLDB, READB, WRITEB, and ACKB signals must have adequate setup and hold times relative to these transitions. The processor interface timing is illustrated in Figure 2.

The MultiKron_vc has two clock inputs, the Node clock and the Timestamp clock. The Node clock is the clock signal used internally by the MultiKron_vc for all synchronization. The MultiKron_vc will not function without a Node clock signal. The Timestamp clock is a slower clock (and should not exceed 1/3 of the Node clock frequency) and is used to increment the Timestamp counter and optionally any of the Resource Counters. The MultiKron_vc design targetted a Node clock frequency of 50 MHz and a Timestamp frequency of 10 MHz. Lower Node clock frequency presents no problem. For example, a Node clock frequency of 40 MHz and a and a Timestamp frequency of 13.3 MHz, or a Node clock frequency of 20 MHz and a Timestamp frequency of 6.66 MHz are acceptable operating conditions.

MEASUREMENT SUPPORT

It is expected that one MultiKron_vc chip will be used for each node of a multiprocessor, where a node is a shared-memory, tightly-coupled cluster of one or more processors. The MultiKron_vc chip (Figure 1) is a memory mapped device requiring a block of addresses, see Table 2 for a summary of the address mapping. In this discussion, addresses are those of 32-bit or 64-bit locations, and are given as **hexadecimal** numbers.

Software Reset

This pseudoregister (Address=base+0; Write Only, data is ignored) initializes most of the machine state of the MultiKron_vc. Unlike the hardware reset pin (RESETB), it does not affect the contents of the Timestamp Counter.

Timestamp

This 56-bit counter (Address=base+2; Read always but Write only in test mode, 56 bits) tallies the Timestamp clock and is reset to zero by a system reset (RESETB) on power-up. The hardware system reset signal is a low-active asynchronous signal and must be active for a minimum of five Node clock cycles. To synchronize the timestamp of multiple MultiKron_vc chips in the machine, the occurrence of this reset should be synchronized throughout the machine. Timestamp Synchronization is then maintained by

distribution of a common clock to each MultiKron_vc chip. The Timestamp clock should not exceed 1/3 (one third) of the MultiKron_vc Node clock frequency. The Timestamp is readable in full 56-bit precision, but is not writable except in test mode. The 56-bit Timestamp yields a wrap-around epoch of over a 100 years at 10 MHz.

CSR Register

The Control and Status Register (Address=base+1; Read/Write, 16 bits) is used to configure the basic operational modes of the MultiKron_vc and to allow examination of these settings. The current register format is shown in Table 1.

The enabling of a (Node clock) Wait State for CPU interactions (bit 12) and for SRAM interactions (bit 11) are set by the logic levels at the two package pins, WCPU and WMEM, during hardware and software reset. The CPU wait state provides additional setup time for processors that would otherwise provide very short times for address decoding by the MultiKron_vc. The SRAM wait state allows use of slower SRAMs or for a faster MultiKron_vc Node clock.

Bits 14 and 15 control the width of the processor data used by the MultiKron_vc, either 32 bits or 64 bits (default). Reading bit 14 provides the current status of this option, a 1 indicates 32-bit mode while a 0 indicates 64-bit mode. To set the desired option requires a 1 written to one of these bit positions. Writing a 0 has no effect and leaves the option unchanged.

High Order 32-Bit Register

The MultiKron_vc is designed for a 64-bit processor interface, but has a High Order 32 Bit Register (Address=base+7; Read/Write, 32 Bits) that can be used to implement a 32-bit processor interface. By placing the MultiKron_vc into its 32-bit mode via bit 14 of the CSR, input data bits 32-63 are taken from this High Order 32-bit register on processor writes rather than from the input pins. Output data bits 32-63 are always (whether or not in 32-bit mode) placed in that register, as well as on the corresponding output pins, on processor reads. Thus, in 32-bit mode two reads or two writes are required instead of one in 64-bit mode. In 32-bit mode the High Order 32-bit register should be written to first before the normal write, and read from after the normal read. Note, the MultiKron_vc makes no provision to keep these two reads or writes indivisible. Interrupts or multiple processors could overwrite data by interleaving their operations.

RESOURCE COUNTERS

The 64 Resource Counters, four banks of 16 counters, can be used to count clocks, external hardware events, or software-commanded events. The accumulated value of these counts can be read out directly from the counters by the processor or stored in the MultiKron_vc's dedicated SRAM for later retrieval by the processor.

Resource Counters and their Shadow Registers

The Resource Counters (Read/Write/Load/Store and Increment, 32 or 64 Bits) accumulate counts of either the Node clock, the Timestamp clock, 1/10 the Timestamp clock, 1/100 the Timestamp clock, external signals (rising edge), or software generated triggers. The external input signals must not exceed the Node clock frequency.

When one of the Resource Counters is being read, the contents of all Resource Counters in that bank are copied into the corresponding shadow registers, while the Resource Counters can continue to count. This ensures that the counter contents stored in the shadow registers all correspond to the same instant - the time of the read that initiated the copy. Then the contents of the shadow registers are used for data output.

The Resource Counters will count past their maximum value by wrapping around to zero. The length of time it takes for a 32-bit counter to reach its maximum value depends on the rate of the signal being tallied. For a 50 MHz signal a 32-bit counter fills in about 85 seconds; for a 5 MHz signal a 32-bit counter fills in about 14 minutes; for a 500 kHz signal a 32-bit counter fills in about 2.3 hours. Each even/odd address pair of Resource Counters can be configured as a single 64-bit Resource Counter, greatly increasing its maximum value. For a 50 MHz signal a 64-bit counter fills in about 12,000 years; for a 5 MHz signal a 64-bit counter fills in about 120,000 years; for a 500 kHz signal a 64-bit counter fills in about a million years.

Processor Reading and Writing of a Resource Counter. Each Resource Counter may be read or written by a processor. Writing to a Resource Counter will place the data written into that Resource Counter, the shadow registers are not affected. There are two types of reads for the Resource Counters, one is a read-with-copy, MultiKron_vc commands 1 and 2, and the other is a read-without-copy, MultiKron_vc command 4. All reads are done from the shadow registers, the copy refers to whether or not the contents of the Resource Counters of the specified bank are copied into the shadow registers before reading. Reading-with-copy any Resource Counter, causes all Resource Counters of that bank to be copied into their shadow register. Thus a read-with-copy yields the most current Resource Counter value, while a read-without-copy yields the value last copied (if any). Reading-with-copy all of the Resource Counters of a bank consecutively will yield data at 16 different instances. To obtain the data from all of the Resource Counters of that bank at the same instant, one read-with-copy should be followed by 15 reads-without-copy. This along with the ability to SWAP banks in SRAM provides a means to implement a virtual set of Resource Counters.

Processor Incrementing of a Resource Counter. A MultiKron_vc command 4 will cause the corresponding Resource Counter to be incremented, if the "software increment" option is selected in the bank configuration register field associated with this counter. This software increment allows the programmer to selectively use the Resource Counters to accumulate counts of software events.

Resource Counter Control Registers

The Resource Counters can be individually controlled via two registers per bank, the Enable register and the Configuration register. Each of these registers is divided into 16 4-bit fields, one for each of the 16 Resource Counters in that bank, see Tables 3 and 4. These control registers are designed so that a zero value written into any field leaves that field unchanged. Only a non-zero value will change a field. This allows subsets of the Resource Counters to be shared between different experimenters or different parts of one experiment, without the need to know the control settings of other fields. It is expected that the counters would be disabled prior to their use via the Enable register, and the counting source selected for each counter via the Configuration register. The Resource Counters can then be selectively enabled or disabled as desired via the Enable register to accumulate counts during the experiment. The Resource Counters can be configured to provide a stop-watch feature for either hardware or software events. For software events, set the configuration register to count one of the internal clocks. Then via the Enable register, enable the counter at a start event and disable it at an end event. For hardware events, set the configuration register to count one of the internal clocks only when gated by the external signal. Once enabled, via the Enable register, the counter will only count when the external signal is active. The external signal thus acts as an Enable/Disable to the counter.

Configuration Register. The Configuration Register (Read/Write, 64 bits) is used to select the counting sources for each Resource Counter, as well as determining if the odd/even counter pair should be a single 64-bit counter or separate 32-bit counters, see Table 3. The counting sources are internal clocks, a software generated signal, an external signal from a package pin private to each counter, or an internal clock only when the external signal is active (i.e., the external signal is used as an enable signal).

Enable Register. The Enable Register (Read/Write, 64 bits) is used to enable or disable the Resource Counters, see Table 4. There are two choices when enabling a counter. One choice is to reset the counter to zero before enabling it. The other choice is to enable the counter with its previous contents intact.

SRAM

The MultiKron_vc chip requires a dedicated static random access memory (SRAM) for storage of the virtual counter banks. The data path between the MultiKron_vc and the SRAM is 40 bits wide. This allows the contents of one 32-bit counter and its associated 4-bit fields in both the Enable and Configuration registers to be transferred together.

The SRAM is controlled by four signal lines from the MultiKron_vc, a read/write (MEM_R_WB), an output enable (MEM_OEB), and two chip enables (MEM_CE1B and MEM_CE2H). The SRAMs are not clocked devices and require only that the setup and hold times are met. To read from the SRAM requires enabling MEM_OEB (active low) and setting MEM_R_WB to read (high). After the required setup time the SRAM output is available. To write to the SRAM requires disabling MEM_OEB (high) and toggling MEM_R_WB (low and then back to high). As long as the input data setup time and the length of time MEM_R_WB is low is sufficient, then the data is latched into the SRAM on the low-to-high transition of MEM_R_WB. Such SRAMs are commercially available, [CYP91, IDT94] are two examples.

There is no direct interaction between the processor and the SRAM. The only interaction with the SRAM is through the SWAP, LOAD, and STORE commands of the MultiKron_vc. A SWAP command consists of first doing a STORE command followed by a LOAD command. The STORE command uses the 12 bits of the associated bank register concatenated with an internal 4-bit counter, as the low order bits, to form a 16-bit home (base) SRAM address. If the associated valid bit is disabled, indicating that the contents of this active counter bank is invalid, then the STORE command terminates with no further action (i.e., it does nothing). The upper 12 bits of the SRAM address, from the bank register, are held constant during the entire command. Only the the low order 4 bits from the internal counter change, to address the locations of each counter in the bank. The signals MEM_CE1B and MEM_CE2H are enabled to activate the SRAM, the signal MEM_OEB is set high to disable the SRAM output, and the internal 4-bit counter is initialized to all ones. Each SRAM write begins by setting the MEM_R_WB signal low, and terminates by setting it high, which causes the data to be latched into the SRAM. Then the MultiKron_vc steps through the entire 16 counter bank, one at a time, writing the contents of one 32-bit counter and its associated 4-bit fields in both the Enable and Configuration registers to the SRAM, and decrementing the internal 4-bit counter. When the internal 4-bit counter reaches zero the operation completes.

The LOAD command loads the low order 12 bits of the processor data into the specified bank register and sets the valid bit. The LOAD command then uses the 12 bits of the bank register concatenated with an internal 4-bit counter, as the low order bits, to form a 16-bit home SRAM address. The signals MEM_CE1B and MEM_CE2H are enabled to activate the SRAM, the signals MEM_OEB (low) and MEM_R_WB (high) are set to place the SRAM in read mode, and the internal 4-bit counter is initialized to all ones. The upper 12 bits of the SRAM address, from the bank register, are held constant during the entire command. Only the the low order 4 bits from the internal counter change, to address the locations of each counter in the bank. The SRAM control signals are held constant during the read operation, only the address changes which causes the SRAM output to produce the new data after the specified latency. The MultiKron_vc steps through the entire bank of 16 counters reading the contents of one 32-bit counter and its associated 4-bit fields in both the Enable and Configuration registers, from the SRAM, and writing that information into the specified bank of the MultiKron_vc, and decrementing the internal 4-bit counter. When the internal 4-bit counter reaches zero the operation completes.

TEST MODE

The MultiKron_vc chip can be placed in test mode only by enabling (LOW) the TESTB pin. When in test mode the Timestamp becomes writable and incrementable by the processor. These features could conflict with normal operations and so were placed in this special test mode.

STATUS

The MultiKron_vc integrated circuit illustrates that powerful hardware support for multiprocessor computer performance characterization can be incorporated in a single chip. The MultiKron_vc has been designed using a commercially available CAD toolset, and has been fabricated in 0.7 micrometer CMOS using a commercially available standard cell library and a commercially available fabrication line. These chips have been packaged in a 208 QFP (Quad Flat Pack), see Table 5. The MultiKron_vc chip is commercially available directly from the fabrication manufacturer without any NRE (non-reoccurring engineering) costs. Contact NIST for further information (Alan Mink (301) 975-5681 or amink@nist.gov). Preliminary testing indicates that these chips can successfully operate at 50 MHz.

The MultiKron_vc along with its required dedicated SRAM is a perfect candidate for an MCM (multi-chip module) implementation. We have designed such an MCM for a 140 QFP package, see Table 7, whose overall size is not much larger than the MultiKron_vc 208 QFP package. We have also designed an SBus printed circuit board capable of supporting either the separate MultiKron_vc and SRAM packaged chips or a single integrated MCM chip. This board will be part of an SBus experimenter's toolkit which will be available for distribution to collaborating researchers.

REFERENCES

[CAR88] Carpenter, R.J. Performance measurement instrumentation for multiprocessor computers. Gelenbe (ed)., High Performance Computer Systems. 1988; Paris; North Holland; pp 81-92; ISBN 0 444 70485 X.

[CAR89] Carpenter, R.J. Performance measurement instrumentation at NBS. Simmons, et al, eds.; Instrumentation for Future Parallel Computing Systems; Santa Fe; Addison-Wesley (1989) pp 159-183; ISBN 0 201 50390 5.

[CYP91] Cypress Semiconductor BiCMOS/CMOS Data Book, part# CY7B185 and CY7C185 (8K by 8-Bits SRAM), 1991.

[IDT94] Integrated Device Technology Data Book, part# IDT7164 (8K by 8-Bits SRAM), 1994.

[MIN90] Mink, A.; Carpenter, R.; Nacht, G.; Roberts, J. Multiprocessor performance-measurement instrumentation. IEEE Computer: 63-74; 1990 September.

[MIN92] Mink, A. and Carpenter, R.J. Operating Principles of MultiKron Performance Instrumentation for MIMD Computers. Natl Inst of Standards and Technology, Gaithersburg, MD., NISTIR 4737; 1992 Mar.

[MIN93] Mink, A., Roberts, J. W. and Antonishek, J., "Operating Principles of the VME MultiKron Interface Board," National Institute of Standards and Technology, NISTIR 5233, Aug. 1993.

[MIN94] Mink, A. "Operating Principles of MultiKron II Performance Instrumentation for MIMD Computers," National Institute of Standards and Technology, NISTIR 5571, Dec. 1994.

[MIN95] Mink, A., Nacht, G. and Antonishek, J., "Operating Principles of the SBus MultiKron Interface Board," National Institute of Standards and Technology, NISTIR 5652, May 1995.

[ROB89] Roberts, J.; Antonishek, J.; Mink, A. Hybrid performance measurement instrumentation for loosely-coupled MIMD architectures. Proc. 4th Distributed Memory Computer Conf. (DMCC4); 1989; Monterey, CA; 7 p.

Table 1. MultiKron_vc Control and Status Register (CSR) Format

Status (read)	Bit Position	Control (write)
Logical "0"	0-11	Read Only, write data ignored.
SRAM Wait State Enabled	12	Read Only, write data ignored (set at RESET)
CPU Wait State Enabled	13	Read Only, write data ignored (set at RESET)
Logical "0"	14	Read Only, write data ignored
32-bit mode Enabled	14	Enable High Order Reg for upper 32 bits of input data
Logical "0"	15	Enable pins 32-63 for upper 32 bits of input data (64-bit mode)

Table 2. MultiKron_vc Address Assignments

Address Hex	Description	
	Read	Write
x0	-	Reset chip, Timestamp unchanged
x1	CSR Register	CSR Register
x2	Timestamp Register (56 bits)	* Timestamp Register (56 bits)
x3	-	* Increment Timestamp Register
x4-x6	-	-
x7	High Order 32-bit Reg	High Order 32-bit Reg
x8-x0FF	-	-
x1bn	** 32-bit Cntr n of Bank b	32-bit Cntr n of Bank b
x2bn	** 32/64-bit Cntr n/n+1 of Bank b	** 32/64-bit Cntr n/n+1 of Bank b
x3b0	-	Clear all Cntrs of Bank b
x3b1	-	Invalidate Bank b
x3b2	Bank register b	Bank register b
x3b3	Bank b 64-bit Enable register	Bank b 64-bit Enable register
x3b4	Bank b 64-bit Config register	Bank b 64-bit Config register
x3b5-x3bF	-	-
x4bn	Read-without-copy Cntr n of Bank b	Software Incr Cntr n of Bank b
x5b0	-	-
x5b1	-	Store Bank b (if valid) in SRAM
x5b2	-	Load Bank b from SRAM block ddd
x5b3	-	Swap Bank b with SRAM block ddd
x5b4-x5bF	-	-
x600-xFFF	-	-

* Available ONLY in TEST mode

** EVEN addresses 64 bits, ODD addresses 32 bits

ddd is the 12 least-significant bits of the data written

Bank b and Counter n are each 4-bit fields of the address.

There are 16 Counters per Bank, but only 4 Banks out of the possible 16 are implemented.

Table 3. Configuration Register encoding (4-bit field/Resource Cntr)

four Bit Encoding	Description
0000	NOP
0001	External Signal (Count external signals on low to high transition)
0010	"
0011	"
0100	Software Increment (Increment counter by software)
0101	"
0110	"
0111	Double Precision, for odd Cntrs only; Software Increment for even Cntrs (This setting applies only to ODD Resource Counters, which will then be tied to the counting source of its EVEN Counter pair)
1000	TSclk gated by EXT (count Timestamp clks gated by EXT signal -- high)
1001	Nodeclk gated by EXT (count Nodeclks gated by EXT signal -- high)
1010	1/10 TSclk gated by EXT (count TSclks divided by 10, gated by EXT signal -- high)
1011	1/100 TSclk gated by EXT (count TSclks divided by 100, gated by EXT signal -- high)
1100	TSclk (count Timestamp clks)
1101	Nodeclk (count Nodeclks)
1110	1/10 TSclk (count TSclks divided by 10)
** 1111	1/100 TSclk (count TSclks divided by 100)

** default setting

Table 4. ENABLE Register encoding (4-bit field/Resource Cntr)

four Bit Encoding	Description
xx00	NOP
** xx01	Disable counter
xx10	Enable counter
xx11	Reset & Enable counter

** default setting

Table 5. MultiKron_vc - 208 QFP Pin Assignments (941220).

1	D0	53	D42	105	P-8	157	AM4
2	D1	54	D43	106	M3	158	AM5
3	D2	55	P-5	107	M4	159	G-14
4	P-1	56	D44	108	M5	160	AM6
5	D3	57	D45	109	M6	161	AM7
6	D4	58	D46	110	G-9	162	AM8
7	D5	59	D47	111	M7	163	P-14
8	D6	60	G-6	112	M8	164	AM9
9	G-1	61	D48	113	M9	165	AM10
10	D7	62	D49	114	M10	166	AM11
11	D8	63	D50	115	P-9	167	AM12
12	D9	64	D51	116	M11	168	G-15
13	D10	65	P-6	117	M12	169	AM13
14	P-1	66	D52	118	M13	170	AM14
15	D11	67	D53	119	M14	171	AM15
16	D12	68	D54	120	G-10	172	P-15
17	D13	69	D55	121	M15	173	M_R_WB
18	D14	70	G-7	122	M16	174	M_CE2H
19	G-2	71	D56	123	M17	175	M_CE1B
20	D15	72	D57	124	M18	176	M-OEB
21	D16	73	D58	125	P-10	177	G-17
22	D17	74	D59	126	M19	178	AC11
23	D18	75	P-7	127	M20	179	AC10
24	P-2	76	D60	128	M21	180	AC9
25	D19	77	D61	129	G-11	181	AC8
26	D20	78	D62	130	M22	182	P-CO
27	D21	79	D63	131	M23	183	AC7
28	D22	80	G-16	132	M24	184	AC6
29	D23	81	X0	133	M25	185	AC5
30	G-3	82	X1	134	P-11	186	AC4
31	D24	83	X2	135	M26	187	G-CO
32	D25	84	G-CO	136	M27	188	AC3
33	D26	85	X3	137	M28	189	AC2
34	D27	86	X4	138	M29	190	AC1
35	P-3	87	X5	139	G-12	191	AC0
36	D28	88	X6	140	M30	192	NODECLK
37	D29	89	P-CO	141	M31	193	G-CLK
38	D30	90	X7	142	M32	194	P-CLK
39	D31	91	X8	143	M33	195	P-CO
40	G-4	92	X9	144	P-12	196	RESETB
41	D32	93	G-C0	145	M34	197	OE
42	D33	94	X10	146	M35	198	HOLDB
43	D34	95	X11	147	M36	199	WMEM
44	D35	96	X12	148	M37	200	TSCLK
45	P-4	97	P-CO	149	G-13	201	G-CO
46	D36	98	X13	150	M38	202	STARTB
47	D37	99	X14	151	M39	203	WCPU
48	D38	100	X15	152	AM0	204	READB
49	D39	101	G-8	153	AM1	205	WRITEB
50	G-5	102	M0	154	P-13	206	TESTB
51	D40	103	M1	155	AM2	207	G-0
52	D41	104	M2	156	AM3	208	ACKB

Signals names ending with "B" indicate active low, all others active high. The "CO" next to a power or ground indicates its use is for the core logic vs. the output pads.

Table 6. MultiKron_vc - description of pin names.

Pin Name	Direction	Description
ACKB	(Output)	This low active signal indicates that the MultiKron_vc chip has completed a CPU Read or Write cycle.
AC0-AC11	(Input)	CPU Address Lines to the MultiKron_vc chip.
AM0-AM15	(Output)	SRAM Address Lines from the MultiKron_vc chip.
D0-D63	(Bidirectional)	CPU Data Bus to the MultiKron_vc chip.
G-x G-CO G-CLK	(Input) (Input) (Input)	GROUND for PADS GROUND for Core logic GROUND for clock distribution pad
HOLDB	(Input)	This low active signal is used to extend a processor read by keeping the data and ACKB signal active until it is de-asserted.
M0-M39	(Bidirectional)	SRAM Data Bus from the MultiKron_vc chip.
Node_clk	(Input)	The Node clock to the MultiKron_vc chip. It is used to synchronize all internal chip operations.
OE	(Input)	When low, this signal disables ALL normal output pins; when high all output pins are enabled.
P-x P-CO P-CLK	(Input) (Input) (Input)	POWER for PADS POWER for Core logic POWER for clock distribution pad
READB	(Input)	This low active signal from CPU starts a read cycle on the MultiKron_vc chip.
RESETB	(Input)	This low active asynchronous signal resets the MultiKron_vc chip to its initial state. This signal MUST be active for a minimum of five node clocks.
STARTB	(Input)	This low active signal is used in conjunction with the READB and WRITEB to initiate a processor interaction, if always kept active this signal has no effect.
TESTB	(Input)	This low active signal places the MultiKron_vc chip in a general test mode, which disables all counters and allows them to be written and incremented by the CPU. This pin MUST be disabled for normal operation.
Timestamp_clk	(Input)	The Timestamp clock to the MultiKron_vc chip. It MUST NOT Exceed 1/3 the Node clock frequency.
WMEM	(Input)	Wait State for an SRAM Read/Write request, only active during a MultiKron_vc chip RESET.
WCPU	(Input)	Wait State to respond to a CPU Read/Write request, only active during a MultiKron_vc chip RESET.
WRITEB	(Input)	This low active signal from CPU starts a write cycle on the MultiKron_vc chip.
X0-X15	(Input)	External Signal for Resource Counters, counts on positive edge.

Table 7. MultiKron_vc MCM - 140 QFP Pin Assignments (950316).

1	GND-0	36	VDD-4	71	GND-9	106	VDD-13
2	D0	37	D26	72	D52	107	X14
3	D1	38	D27	73	D53	108	X15
4	D2	39	D28	74	D54	109	AC11
5	VDD-0	40	GND-5	75	VDD-9	110	GND-14
6	D3	41	D29	76	D55	111	AC10
7	D4	42	D30	77	D56	112	AC9
8	D5	43	D31	78	D57	113	AC8
9	GND-1	44	VDD-5	79	GND-10	114	VDD-14
10	D6	45	D32	80	D58	115	AC7
11	D7	46	D33	81	D59	116	AC6
12	D8	47	D34	82	D60	117	AC5
13	VDD-1	48	GND-6	83	VDD-10	118	GND-15
14	D9	49	D35	84	D61	119	AC4
15	D10	50	D36	85	D62	120	AC3
16	D11	51	D37	86	D63	121	AC2
17	GND-2	52	VDD-6	87	GND-11	122	VDD-15
18	D12	53	D38	88	X0	123	AC1
19	D13	54	D39	89	X1	124	AC0
20	D14	55	D40	90	X2	125	NODECLK
21	VDD-2	56	GND-7	91	VDD-11	126	GND-16
22	D15	57	D41	92	X3	127	RESETB
23	D16	58	D42	93	X4	128	OE
24	D17	59	D43	94	X5	129	HOLDB
25	GND-3	60	VDD-7	95	GND-12	130	VDD-16
26	D18	61	D44	96	X6	131	Wmem
27	D19	62	D45	97	X7	132	TSCLK
28	D20	63	D46	98	X8	133	STARTB
29	VDD-3	64	GND-8	99	VDD-12	134	GND-17
30	D21	65	D47	100	X9	135	Wcpu
31	D22	66	D48	101	X10	136	READB
32	D23	67	D49	102	X11	137	WRITEB
33	GND-4	68	VDD-8	103	GND-13	138	VDD-17
34	D24	69	D50	104	X12	139	TESTB
35	D25	70	D51	105	X13	140	ACKB

Signals names ending with "B" indicate active low, all others active high.

36 PWR&GND pins
104 signal pins
140 Total Pins
64 I/O pins
39 In pins
1 Out pins

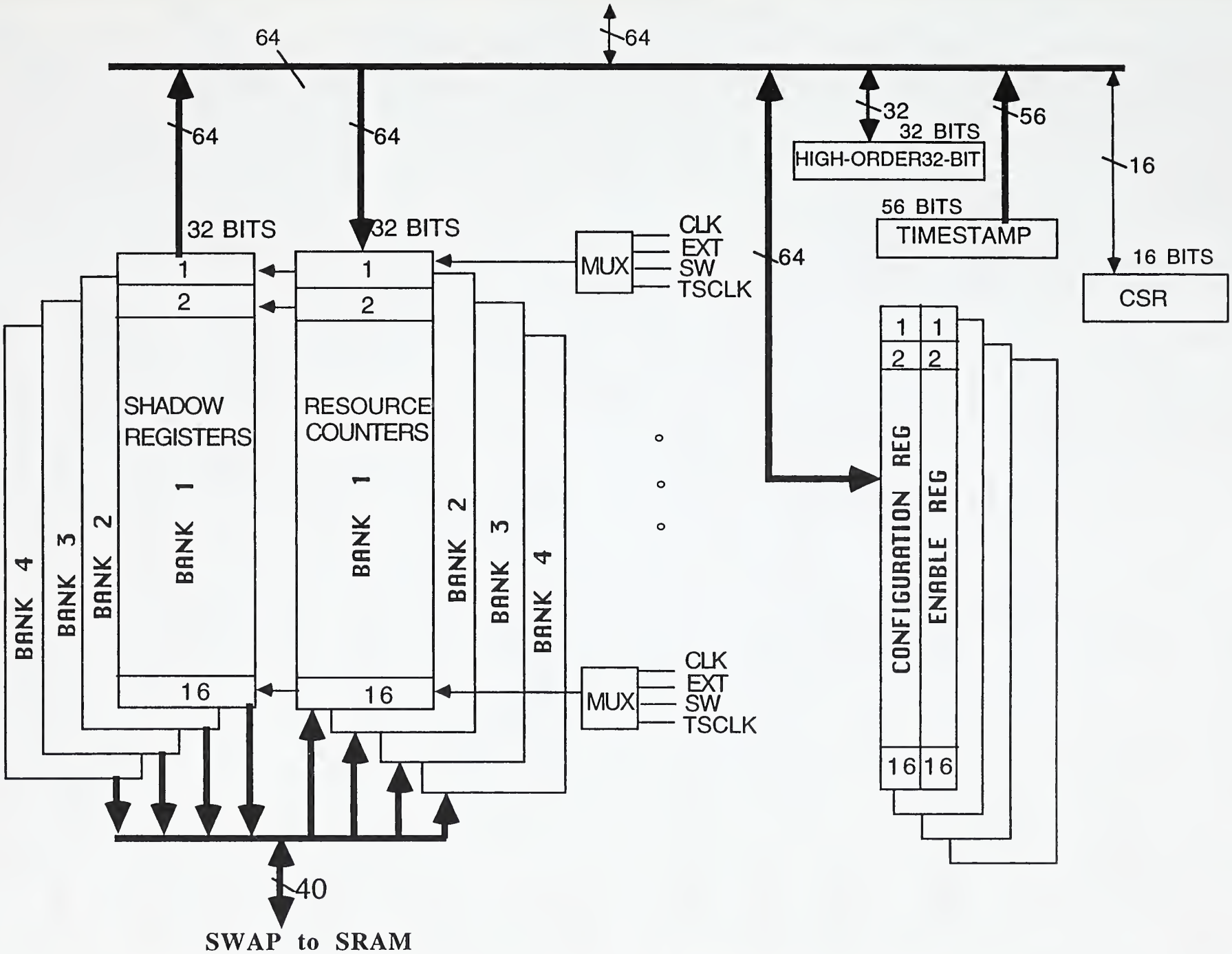

Figure 1. Block diagram of the MultiKron_vc Chip

950428

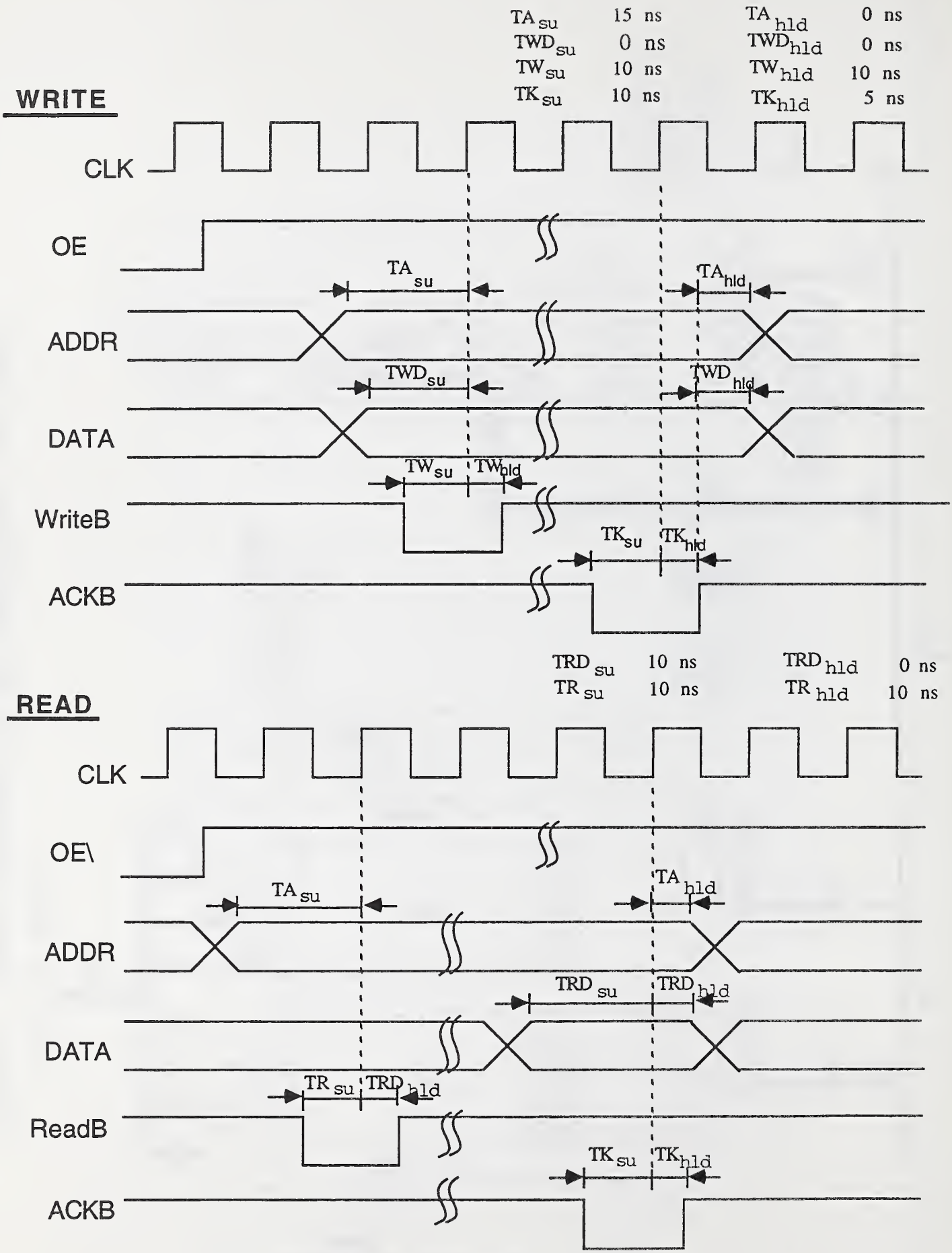

Figure 2. MultiKron_vc I/O Cycle