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

Computer Systems Laboratory

CMRF

COMPUTER MEASUREMENT RESEARCH FACILITY FOR HIGH PERFORMANCE PARALLEL COMPUTATION

NISTIR 4737

Operating Principles of MULTIKRON Performance Instrumentation for MIMD Computers

Alan Mink
Robert J. Carpenter

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

March 1992

Partially sponsored by the
Defense Advanced Research Projects Agency

NISTIR 4737

Operating Principles of MULTIKRON Performance Instrumentation for MIMD Computers

Alan Mink and Robert J. Carpenter

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

Partially Supported by the
Defense Advanced Research Projects Agency
Arlington, VA

March 1992

TABLE OF CONTENTS

Operating Principles of MULTIKRON Performance Instrumentation for MIMD Computers

Alan Mink and Robert J. Carpenter

The single-chip MULTIKRON design replaces our earlier event trace (uTRAMS) and resource utilization (uREMS) performance instrumentation chips. It incorporates a longer timestamp, more bits of user-event and processor identification, and 16 counters for resource utilization measurements. The initial implementation uses a 64-bit processor bus, though the design allows simple modification to a 32-bit bus. The collection network output has a width of eleven bits (eight data, one parity, and two control), and can transfer up to 25 million data bytes per second.

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

INTRODUCTION

From our experience in performance characterization of MIMD computers and software [CAR88, LYO89A, LYO89B, LYO89C, LYO90A, LYO90B, LYSN89, MDRC87, MRDC86, NAMI89, RAM89, ROB86], we observed the need for a timed trace of user-defined events during software execution, and a need to quantify very-frequently-occurring events (such as memory accesses) related to utilization of the computer's hardware resources. Hardware support is needed to obtain these data with tolerable perturbation.

The MULTIKRON chip provides this hardware support for a combination of trace measurement and resource measurement. Trace measurement provides timestamped program event traces. For example, the duration of time spent in a state or routine can be calculated from the entry event and exit event timestamps. Alternatively, the set of sixteen resource counters can be used to count clock ticks (i.e., accumulate the time between events), external hardware events, or software-commanded events. The accumulated value of these counts can be sent to the collection point with any or all trace samples. Once we agreed to increase the number of "resource" counters, it was recognized that they would also have considerable utility as a very-low-overhead means of merely *counting* chosen software events [LYO90C, MIL90].

The single-chip MULTIKRON replaces our earlier event trace (uTRAMS) [CAR89, MCNR90] and resource utilization (uREMS) [MCNR, NAMI89] performance measurement instrumentation chips. It incorporates a much wider (56-bit) timestamp, many more bits of user-event and processor identification, and sixteen counters for resource utilization measurements. The initial implementation uses a 64-bit processor bus, though the design allows simple modification to a 32-bit bus. The byte-wide collection network output is now synchronous, with a simple two-way-handshake protocol. The network can collect about 1.5 million event trace samples or 0.3 million trace-with-resource samples per second. This discussion is based on Revision 7 (910703) of the MULTIKRON Preliminary Data Sheet. The reader should refer to Figure 1, the overall block diagram of the MULTIKRON, during the following discussion.

Event Tracing

The fundamental measurement function in event tracing is to determine the time of occurance of various key events in the execution of a computer's tasks with high precision. These times can be used later to calculate elapsed time, duty cycle, access latency, and the like. In theory, one could passively monitor program execution to determine when events occurred. In practice, passive monitoring requires a very large content-addressable memory to recognize the events [NAMI89]. Economical, large, fast content-addressable memories do not yet exist. In addition, many computers are based on microprocessors which contain an internal instruction cache or queue. External monitoring of instructions fetched is fruitless, since they do not closely correlate with actual program execution. Many fetched instructions are never executed. Yet another problem is that compilers produce code in terms of *virtual* addresses, and the translation into *physical* addresses is defered to run time, and may even change during execution.

Although passive monitoring does not perturb program execution, its severe drawbacks, just mentioned, have led us to employ a *hybrid* approach to event tracing. The tracing function relies on *embedded code* in the software running on the machine being measured to establish an *event* [MDRC87, MRDC86] . This very brief added code triggers hardware capture of the data about the event. Execution of this embedded code informs the measurement hardware of events by writing an event identification as data to a memory-mapped register. Our earlier measurement hardware allowed only 11 to 16 bits of user-written event identification [MRDC86, MCNR90]. This has proven inadequate, so that we now recommend 32 or more bits of arbitrary event identification.

As mentioned, current time is the most important data collected at an *event*. Again, we "saved bits" in earlier designs, resulting in an unacceptably-short epoch before the timestamp counters wrapped around. In order to have a reasonable epoch with the time resolution required by today's high processor clock rates, at least 40 bits and a 10 MHz timestamp timebase are recommended. The timestamp counter must have a private input pin, so that all timestamp counters in a system can be incremented by the same system-wide timebase signal. Provision must be made to synchronize all timestamp counters, at least at hardware reset time.

The event identification is a user-chosen value written to the measurement chip to initiate a trace sample which identifies *where* in program execution the event occurred. It does not identify which process may be using the (possibly-shared) code where the event occurred.

Process Identification. When a trace data sample is to be captured, the identity of the processor, the node (which may contain more than one processor), and the associated process must be determined and included in the data sample. The most practical solution to process identification seems to be the provision of a special Source Address register (containing process identification), which is updated by the operating system at each task change. 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 requires a set of Source Address registers; each processor is assigned its own Source Address register, in which process identification is kept.

Node Identification. 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 identification of the node where a process is located should be concatenated with the process identification stored in the Source Address registers.

A trace measurement sample consists of the concatenation of the user-written event identification, the hardware-determined identification of the processor that wrote the sample, a 32-bit field which identifies the process and node, and 40 bits of the timestamp counter. To allow correlation of samples taken by multiple MULTIKRON chips, the timestamp counters must all be driven by a common clock, and they are all reset to zero by the *hardware* reset signal [CAR88, CAR89, SNE91]. One can instrument subroutine entry and exit points, operating system calls, or processing "states" [LYO90B]. Many billions of individual event / processor / process combinations can be resolved, if needed.

Subset Data. Sixteen different classes or sets of user-defined trace data sampling commands are defined. The measurement chip can be configured by a Filter Register to accept or ignore any combination of these subsets at run-time. All trace data samples will belong to one of these classes. Once instrumented, embedded measurement code can remain undisturbed when different types of measurement reports are desired (or no report at all). Thus, software can be extensively instrumented and yet not burden the data-capture system in routine operation. Neither modification nor recompilation of the software on the measured computer is required to change the measurement set [MCNR90, MICA90]. The simple method we use to indicate trigger subset is to assign a block of address for triggering the taking of a data sample. The lowest four bits of the address then indicates the subset to which the trigger belongs.

Processor Identification. The tightly-coupled processors at a single node will often share code. Software techniques which could be used to identify between the processors have too much overhead to be acceptable. The solution is to include a few wires for *processor* identification in the interconnection network between each measurement chip and the small number of processors it measures. The processor identification is included in each trace measurement sample.

Relating Events on Multiple Processors or Nodes. To correlate the activities reported by the the various measurement chips in a system for measurement purposes, it is often important simultaneously to know the current state of all the processors in a multiprocessor / multinode system to a very fine time resolution. This is particularly important at the time of some "major" events. In our earlier uTRAMS we synchronized the timestamp counters in all chips and provided a *global interrupt* to be used to cause all processors to drop their current activity and enter a routine to capture current status information [CAR88, MICA90] [RAM89, SNE91]. The first application of MULTIKRON is expected to be on the Touchstone SIGMA, which already provides global signaling. Other recent computer buses, such as IEEE 896 Futurebus+, also provide for global signaling or interrupt. Thus we did not incorporate a global interrupt function in the current MULTIKRON.

Hardware Resource Utilization

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 great reduction in the cost and data storage requirements. Of course some detail is lost.

Resource Counters. Resource 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 by hardware signals such as cache hits or misses, clock cycles during waits for shared-resource access, or events such as message transmissions. If the counters can also be incremented by software, some 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.

The content of all Resource Counters is concatenated with a Trace Sample to form a Resource Sample. Resource samples should be collected at key points in program execution to resolve resource utilization to specific activities. Collection network bandwidth and data collection storage could be conserved if means were provided to read only a subset of all the Resource Counters. For flexibility in application, a measurement chip should have *at least* 16 Resource Counters, and 32 would be much better. Since counters will be incremented at the processor clock rate during some intervals, a size of 32 bits seems to be a lower limit.

In our earlier uREMS chip, we implemented *pairs* of resource counters which automatically scaled to 13 bits of precision with a five-bit exponent common to both values. This reduced the amount of data storage required [NAMI89]. We now feel that was a poor decision. The size and complexity of the automatic-scaling hardware greatly reduced the number of signals we could measure, with little saving in the data collection and analysis system.

Counter Sources. In the normal mode of operation, it is expected that all resource counters will be disabled prior to a measurement experiment. Then the counting source for each counter will be selected by setting an source multiplexer selection register. During the measurement period, each counter can be selectively enabled and disabled as many times as desired, and can thus accumulate counts during multiple uses of a piece of code, for example. The enable register allows the resource registers to be individually enabled for counting. The source multiplexer selection register can configure the input of each of the resource registers to be incremented by:

> The processor clock frequency - for measuring short intervals of elapsed time with high precision.
> A source at 1/10 or 1/100 of the timestamp clock's frequency - for measuring much longer elapsed times with reduced precision.
> Positive-going transitions on its private external package pin, up to the processor clock frequency - for accumulating numbers of (positive going) transitions in any electrical signal in the system,
> A processor "write" to the address corresponding to the specific counter - for a low-overhead way of accumulating the number of software events.

Local Access to Measurement Registers

Many benefits accrue if processors closely-coupled to a measurement chip can directly read its registers. The *time dilation* technique for exploring the effects of proposed variations in loosely-coupled systems is an example [AS90]. Direct read access also allows many of the features of a measurement chip to be used even though there may be no measurement data collection network.

PROCESSOR INTERFACE

The data path between a processing node and the MULTIKRON is 64 bits wide. On a read by an node processor, any unused bit positions return a logical "0". On a write from the node, any unused bit positions are ignored. The MULTIKRON expects to receive seven bits of address, the least-significant of which corresponds to 32-bit data entities (four-byte increments). All higher address bits must be externally decoded to establish the base address of the MULTIKRON. The address format and mapping for the 7 low-order address bits to the MULTIKRON is:

Field Width	1 bit	2 bits	4 bits
Upper Bits of Base Address	Command = 0	Encoded "register" and control commands	
	Sample = 1	Sample Type Trace w or w/o Resource Data	Filter Value

Table 1
Use of MULTIKRON's seven low-order address bits

Thus addresses 0-63 are used for commands, such as read or write a register, and addresses 64-127 are used to trigger samples with various options. The Filter Value allows run-time selection of subsets of the possible data samples, as described in the discussion of the Filter Register. The external higher-order address decoder output must be externally ANDed with the processor's READ or WRITE pulse and so that the RD and WR signals to the MULTIKRON are only asserted (LOW) when the MULTIKRON is addressed. After the MULTIKRON has completed its commanded action, it will assert (LOW) the RDY line. Actual transfers are at positive-going transitions of the processor clock; ADDR(ess), DATA, RD, WR, and RDY signals must have adequate setup and hold times relative to these transitions. The processor interface timing is illustrated in Figure 2.

The data from the different sections of the MULTIKRON can be used in some combination of the following ways: read by the node, written by the node, and captured as part of the measurement sample (sample generation). For each of the MULTIKRON sections listed below, the offset from the MULTIKRON base address, the data field and its size are specified in parenthesis following the register name. Appendix A summarizes the addresses of all internal MULTIKRON registers.

EVENT TRACE MEASUREMENT SUPPORT

It is expected that one MULTIKRON 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 chip (Figure 1) is a memory mapped device requiring a block of addresses. In this discussion, addresses are those of 32-bit locations (or the less-significant part of a 64-bit location), and are given as **decimal** numbers. A sample is triggered by a memory write to a specified address within that block of addresses. A sample always contains trace data (Tables 5 and 6) which consists of the data from the memory write, along with a timestamp, error and class information, and a source address. The trace data is assembled and stored in a small internal FIFO. Optionally, a sample may also include the contents of the resource counters (Tables 5 and 7). The contents of the resource counters are stored in a set of shadow registers before transmission. Each sample is sent out over the collection network byte-serially, with a parity bit and end-of-message flag bit on each byte.

Software Reset

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

Timestamp

This 56-bit counter (Address=base+2; Read Only, 56 bits) tallies the Timestamp clock (10 MHz) and is reset to zero by a system reset on power-up. The MULTIKRON contains a 56-bit timestamp counter, with provisions for synchronizing this counter in all MULTIKRONs in a system by a system reset. Only the lower-order 40 bits of the counter are included in a data sample, although the processor to which the MULTIKRON is attached can read all 56 bits. This 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 in every node in the machine the occurance of this reset should be synchronized throughout the machine. Synchronization is then maintained by distribution of a common clock to each MULTIKRON chip. In the earliest runs of the MULTIKRON, the timestamp clock must not exceed 1/3 (one third) of the node clock. Thus for a node clock of 50 MHz, the timestamp clock can not exceed 16.6 MHz. The timestamp is readable in full 56-bit precision, but is not writable. The 40-bit time field of collection-network samples yields a wrap-around epoch of about 20 hours. Thus, as long as any node takes one sample within each 20 hour interval, time can be resolved for all nodes for an experiment of unlimited duration.

CSR Register

The Control and Status Register (Address=base+1; Read/Write, 32 bits) is used to configure the most basic operational modes of the MULTIKRON and to allow examination of these settings. A logical "1" in a bit position of the data written to the CSR commands the corresponding action. A logical "0" in a bit position of the data written to the CSR causes no effect on the corresponding action. All unused bit positions in the control and status register return a logical "0" on read. The current register contents are:

Status (CSR read)	Bit Position	Control (CSR write of "1" in this bit position)
Sampling Enabled	0	Enable Sampling
Logical "0"	1	Disable Sampling (default on reset)
Write Wait on Overrun	2	Wait for free FIFO/counters to avoid write overrun
Logical "0"	3	Discard data sample if write overrun would occur (default)
Read Wait for shadow regs.	4	Enable processor read wait if counters busy
Logical "0"	5	Cancel read wait if counters busy (Error bit returned) (default)
FIFO Full	6	Read Only, write data ignored.
Resource Shadow Registers Full	7	Read Only, write data ignored.
FIFO Overrun	8	Read Only, write data ignored.
Resource Shadow Register Overrun	9	Read Only, write data ignored.
129th FIFO output bit (w/Resource data)	10	Read Only, write data ignored.
Logical "0"	11	Read Only, write data ignored.
Number of Wait States	12-13	Read Only, write data ignored.
10 us slow clock enabled	14	Set slow clock to 10 us.
Logical "0"	15	Set slow clock to 1 us. (default on reset)
Logical "0"	16-31	Read Only, write data ignored.

Table 2
Control and Status Register Bit Assignments

The number of (processor clock) Wait States (bits 12-13) is set by the logic levels at package pins during hardware reset. The wait states allow use with processors that would otherwise allow very short times for address decoding by MULTIKRON.

Filter Register

The Filter Register (Address=base+4; Read/Write, 32 bits) controls whether or not a sample is taken when a trigger occurs. Sixteen filter levels (or subsets) of measurement data are defined. Any combination of subsets can be enabled by the pattern of bits in the Filter Register. Each sample specifies, as part of its address, a 4 bit encoded filter value which is decoded into one of 16 filter levels. If the corresponding bit in the Filter register is enabled (a "1"), the sample is taken. If the corresponding bit in the Filter register is disabled (a "0"), then the sample is discarded. This allows the program to be extensively instrumented with measurement triggers, while only taking data samples at the desired subset of these triggers. Thus the measurement instrumentation code could remain in the program, causing no difference in execution time, while only taking selective measurements or none at all.

Source Address Registers

The eight Source Address Registers (Addresses=base+16 to 23; Read/Write and Sample Generation, 32 bits) contain the identity ("node.process" - node number and process identification) of the process executing on the corresponding processor and should be updated at each context switch. The MULTIKRON contains a Source Address register for each of the eight processors that it can simultaneously support. The system's operating system should be instrumented to update the corresponding MULTIKRON Source Address register at each processor's context switches. That is, the identity of the process executing on processor 3 should be contained in source address register 3. Eight pins on the MULTIKRON chip are used to identify the processor writing the measurement sample and are used as an index (select) into this set of eight registers, causing the corresponding contents to be included in the trace sample. Thus, at most, only one of the eight lines can be active at any time. Each source address register is assigned a unique address, which is used when the contents must be updated or examined by a processor. Any processor in the node can read or write all of the source address registers.

RESOURCE UTILIZATION MEASUREMENT SUPPORT

The sixteen resource counters can be used to count clocks, external hardware events, or software-commanded events. The accumulated value of these counts can be sent over the collection network with any or all desired trace events.

Resource Counters and their Shadow Registers

The resource counters (Addresses=base+32 to 47; Read Only and Sample Generation, 32 Bits) accumulate counts of either the node clock (50 MHz), slow clock (either 1 MHz or 100 kHz), external signals (rising edge), or software generated triggers. In the current MULTIKRON, the slow clock frequency is the same for all resource counters and is prescaled from the Timestamp clock, either 1/10 or 1/100 of its frequency. Thus for a 10 MHz Timestamp clock, the slow clock is either 1 MHz or 100 kHz, selectable via the CSR register. Additional flexibility may be added in future versions. The external input signals must not exceed the processor clock frequency.

When a Resource sample is triggered (or one of the Resource counters is being read), the contents of all resource counters are copied into the corresponding shadow registers. This ensures that the counter contents used in the resource measurement sample all correspond to the same instant - the time of the sample. Then the contents of the shadow register can be used for data output and the resource registers can continue to be incremented. There is only one rank of shadow registers, in effect a one-level FIFO. The shadow registers will remain occupied for about 80 network clock periods (160 processor clock periods) during the normal capture of a resource data sample. The shadow registers cannot be reused during this period.

The resource counters will not count beyond their maximum value (i.e, no wrap around), but will stick at this maximum value until reset. The length of time it takes for a 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 1 MHz signal a 32-bit counter fills in about 1.2 hours; for a 100 kHz signal a 32-bit counter fills in about 12 hours.

Command	Conditions: Shadow Registers	Conditions: "Wait on Processor Read" Control bit	Actions
Read Resource Register	Empty	Don't Care	Resource Register data immediately returned; bit 63 = 0
	Full	Disabled = 0	Wrong data immediately returned; bit 63 = 1
		Enabled = 1	Wait until shadow register available, then return correct data; bit 63 = 0

Table 3A
Effects of "Wait on Processor Read" Control Bit

Command	Conditions: Shadow Registers	Conditions: "Wait on Overrun" Control Bit	Actions: Data Sampling	Actions: Overrun Flag
Take Resource Sample	Empty	Don't Care	Resource Data immediately put in shadow register for output	0
	Full	Disabled = 0	No new resource sample taken; no wait	1
		Enabled = 1	Wait until shadow register available, then take resource sample	0

Table 3B
Effects of "Wait on Overrun" Control Bit

Overruns and Waits on Resource Counter Use. There are two optional actions when an attempt is made to use the data in the resource counters when their shadow registers are still full from the previous Resource Sample. The shadow registers act as a one-level-deep FIFO for the resource counters. Table 3 indicates the situations in which program execution is temporarily blocked when the resource shadow registers are full and another sample with resource data is attempted, or a CPU attempts to read a resource register. The next sample with resource counter data may be blocked (but probably not Trace samples without resource counter data, since Trace samples have a deeper FIFO) while these shadow registers are being output to the collection network a byte at a time. For sixteen resource counters, this blockage could last 80 output network clocks, since only one byte is transferred on each network clock cycle. A Resource sample, containing both Trace data and Resource Counter contents, is eighty bytes. Attempts to read a resource register by a node CPU may suffer the same duration of blockage. When the shadow registers are empty, reading a resource counter by the CPU at the node will not cause a blockage to succeeding operations, since this is accomplished in a single CPU instruction.

Processor Reading of a Resource Counter. Each resource counter has a specific address and may be read by a processor. Data from the resource counters is read after transfer to the shadow register, just as in the case of a resource data sample. If the shadow registers are not already full (and in the process of outputting their contents to the network), then all the counters are copied to the shadow registers but only the selected counter's data is returned to the CPU. If the shadow registers are busy, there are two options controlled in the CSR:

(1) Read wait on Counters Busy asserted - The read will wait until the shadow registers are free, the new counter value will be transferred to the shadow registers, and then the requested data will be returned to the processor with bit 63 set to a logical "0".

(2) Read Wait on Counters Busy unasserted - The shadow register will not be modified, and incorrect data will be returned to the processor with bit 63 set to a logical "1".

Processor Incrementing of a Resource Counter. Writing a resource counter's specific address (the same one used to read the individual register) causes the corresponding resource counter to be incremented, if the corresponding position in its MUX SEL (MUX select) register is set to the "software increment" option. This software command 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 a set of control registers: the MUX SEL; Reset; Enable; and Disable registers. Each of these control registers is configured to allow 32 resource counters, although only the lower sixteen fields are implemented in the current 16-counter MULTIKRON. It is expected that all the counters would be disabled prior to a measurement experiment, and the counting source selected for each counter via the MUX SEL (multiplexer selection) register. Each counter can then be selectively enabled and disabled as desired to accumulate counts during the program execution.

MUX SEL Register. The MUX SEL Register (Address=base+8; Read/Write, 64 bits) is used to select one of four counting inputs to each resource counter: a node clock (50 MHz); a slow clock of one-tenth or one-hundredth the timestamp clock frequency (1 MHz/100kHz); a software generated signal; or an external signal from a package pin private to each counter. These four selections are encoded into a two bit field.

Two Bit Encoding	Counting Input Selected
00	Slow Clock, 1 or 10 us
01	External Signal
10	Software Signal
11	Node Clock

Table 4
Resource Counter Input Multiplexer Choices

The MUX SEL register is configured to allow expansion to 32 resource counters. Currently only half of this register is used (bit 0-15 and 32-47); the other half is reserved for future expansion (bit 16-31 and 48-63). The two bit field is divided between the upper and lower half of the MUX SEL register. The least significant bit of the two bit field is allocated to the low order half of the MUX SEL Register, while the most significant bit is allocated to the corresponding bit in the high order half of the MUX SEL Register. The i-th field is assigned to the corresponding i-th bit in the low and high order halves of the MUX SEL register (i.e., the two bit field for counter 0 is assigned to bits 32, MSB, and 0, LSB). For example, to select all odd counters (1,3,...,15) to count 50 MHz node clocks (encoding=11) and all even counters (0,2,..., 14) to count external signals (encoding=01) requires the following 64 bit setting:

MUX SEL = 0000000000000000 1010101010101010 0000000000000000 1111111111111111

Reset Register. A "1" written into a bit position of the Reset Register (Address=base+10; Write Only, 32 bits) causes the corresponding resource counter to be cleared. A "0" written into a bit position is ignored. This provides a method by which disjoint subsets of resource counters can be independently controlled by different parts of a program or different programs without interfering with each other. This is a self-clearing register; the register is immediately cleared after the resetting action has been completed. Currently only half of this register is used (bit 0-15); the other half is reserved for future expansion (bit 16-31).

Enable Register. A "1" written into a bit position of the Enable Register (Address=base+11; Read/Write, 32 bits) enables the corresponding resource counter to accumulate counts of the occurrences of the currently selected counting signal. A "0" written into a bit position causes no change in the status of the corresponding counter (i.e., "0" inputs are ignored). This allows disjoint subsets of resource counters to be independently controlled by different parts of a program or different programs without interfering with each other. Currently only half of this register is used (bit 0-15); the other half is reserved for future expansion (bit 16-31).

Disable Register. A "1" written into a bit position of the Disable Register (Address=base+12; Write Only, 32 bits) disables the corresponding resource counter from counting. A "0" written into a bit position causes no change in the status of the corresponding counter (i.e., "0" inputs are ignored). This permits disjoint subsets of resource counters to be independently controlled without interfering with each other. Currently only half of this register is used (bit 0-15); the other half is reserved for future expansion (bit 16-31).

Wait Counter

The Wait Counter (Address=base+5; Read or Clear, 32 Bits) tallies the number of 50 MHz cycles in which the measurement chip has delayed the ready (RDY) signal to the CPUs, beyond the normal preset number (i.e., invoked additional wait states) because the FIFO or Shadow Registers of the Resource Counters are full. This counter is only activated when either the "Write (sample) Wait on Overrun" or "Read Wait on counters" options are enabled in the CSR register. In the unlikely event that the number of waits reaches 2^32, this counter wraps around to zero and continues counting.

Overrun Counter

The Overrun Counter (Address=base+6; Read or Clear, 32 Bits) tallies the number of times a sample has failed to be taken because either the Trace sample FIFO or the Shadow registers were already full when the sample required these resources. This counter is only activated when the "Write Wait on Overrun" option is disabled in the CSR register. In the unlikely event that the number of overruns reaches 2^32, this counter wraps around to zero and continues counting.

NETWORK OUTPUT

The data collection network output of the MULTIKRON chip provides a way for the data samples to reach a central collection point without using the computer's normal data paths. Output from the MULTIKRON consists of ten-bit elements: eight data bits, an odd-parity bit, and a sample-end flag bit. The MULTIKRON delivers a Network Clock output signal at one-half the processor clock frequency. Data output is synchronous with this signal. See Figure 3 for timing specifications of the network output. There are two control signals:

(1) External FIFO Free (input) . The network must assert (high) or de-assert (low) the External FIFO Free signal at least 16 nanoseconds before the positive-going transition of the Network Clock output of the MULTIKRON.

(2) Load External FIFO (output). The MULTIKRON asserts (low) the Load External FIFO signal at least 12 nanoseconds before the positive-going edge of the Network Clock if the network must accept the accompanying output data byte. This signal will not be asserted if the External FIFO Free signal is not asserted.

Data Sample Formats

The MULTIKRON data sample has the basic format of a header byte (Table 5), which is included in the Trace sample data (Table 6). The Resource information is appended to these to form the Resource sample (Table 7). The Trace data sample contains 128 bits (16 bytes). In a resource sample, each of the sixteen Resource counters adds 32 more bits, for a grand total of 640 bits (80 bytes).

CPU ID Encoded	Sample Type 10=Trace 11=Resource	FIFO Over-run Error	RSRC Over-run Error	RSRC Read Error
3 bits	2 bits	1 bit	1 bit	1 bit

Table 5
Header for both Trace and Resource Sample Types

Error & Class Field of Sample

(Sample Generation Only, 8 bits: 3 - CPU ID, 2 - sample class, 3 - error flags). This field in each data sample (Table 5) is used to identify the CPU within the node taking the sample, any error flags associated with the sample, and the sample class (type) identification. The three error flags are:
1) FIFO overrun - previous sample(s) lost (reset by next successful sample),
2) Shadow register overrun - previous sample(s) lost (reset by next successful sample),
3) No Resource Counter data returned to processor due to Shadow register overrun (this should always be a logical "0" since resource counter reads are indivisible commands and the flag is cleared at the end of the read).

Two bits are allocated for sample class, trace sample - with (encoding 11) or without (encoding 10) resource counter data. The CPU ID is a 3 bit encoding of the 8 CPU lines (external MULTIKRON pins) associated with the memory bus, only one of which is active at any time.

Sent first TRACE SAMPLE (128 bits / 16 bytes)

Header	Timestamp	Source Identification	User-written Data
8 bits	40 bits	32 bits	48 bits

Table 6
Trace Sample

Sent first RESOURCE SAMPLE (640 bits / 80 bytes)

Header	Timestamp	Source Identification	User-written Data
8 bits	40 bits	32 bits	48 bits

Resource Counter 0 32 bits	Resource Counter 1 32 bits	Resource Counter 2 32 bits	Resource Counter 3 32 bits
Resource Counter 4 32 bits	Resource Counter 5 32 bits	Resource Counter 6 32 bits	Resource Counter 7 32 bits
Resource Counter 8 32 bits	Resource Counter 9 32 bits	Resource Counter 10 32 bits	Resource Counter 11 32 bits
Resource Counter 12 32 bits	Resource Counter 13 32 bits	Resource Counter 14 32 bits	Resource Counter 15 32 bits

Table 7
Resource Sample (includes Trace Sample)

STATUS

The MULTIKRON integrated circuit illustrates that powerful hardware support for multiprocessor computer performance characterization can be incorporated in a single chip. The chip has been fabricated in 1 micrometer CMOS using a scalable standard cell library. A 60 percent yield was achieved from the first fabrication run, based on 40 MHz testing. We are making slight modifications to the chip design to reduce the data access times to ensure operation up to 50 MHz. Further investigation may reveal that other changes are needed for correct operation at 50 MHz.

Appendix A - REGISTER ADDRESS ASSIGNMENTS

32-bit Address Offset	Write		Read
0	Reset	chip, Timestamp unchanged	N/A
1	Load	CSR Register	CSR Register
2		N/A	Timestamp Register (56 bits)
3		-	-
4	Load	Filter Register	Filter Register (16 bits)
5	Reset	Wait Counter to zero	Wait Counter
6	Reset	Overrun Counter to zero	Overrun Counter
7		-	-
8	Load	MUX SEL Register (64 bits)	MUX SEL Register (64 bits)
9		-	-
10	Pulse	Resource Reset Register	N/A
11	Load	Resource Enable Register	Enable Register
12	Load	Resource Disable Register	N/A
13		N/A	(32 bits of) FIFO Output (test mode ONLY)
14	Load	TEST Register (test mode ONLY)	TEST Register (test mode ONLY)
15		-	-
16-23	Load	Source Addr Register (0..7)	Source Addr Register (0..7)
24-31		-	-
32-47	Increment	Resource Counter (0..15)	Resource Counter (0..15)
48-63	Reserved	for more Resource Counters	Reserved for more Resource Counters
64-95		-	-
96-111	Trigger	a Trace sample w/o resource data, with filter level 0..15 and Source address index 0..7 from hardware pins.	N/A
112-127	Trigger	a Resource sample including Trace data, with filter level 0..15 and Source address index 0..7 from hardware pins.	N/A

Table A-1
MULTIKRON Address Assignments

A Trace sample w or w/o resource data, contains 48 bits of user data.
The source address index for commands 16-23 is computed from the 3 least significant address bits.
The source address index for sample triggering, commands 96-127, is the encoded value of the active CPU ID line of the MULTIKRON package. Only one of the eight CPU ID lines may be active at a time.
The filter level for sample triggering, commands 96-127, is computed from bits 0-3 (the least significant 4 bits) of the address.
The resource counter in commands 32-47 is computed from bits 0-3 (the least significant 4 bits) of the address.

Appendix B - PIN USAGE

Number of Pins	Input or Output	Use
64	I/O	CPU data bus
16	I	External inputs for resource counters
7	I	Address lines
8	I	CPU ID (NOT encoded)
1	I	Reset
1	I	50 MHz Clk
1	I	10 MHz Clk
1	I	Read strobe
1	I	Write Strobe
2	I	Number of CPU Wait States (encoded), valid only during reset
1	O	RDY (read/write ACK)
10	O	Network output (8 data; 1 odd parity; 1 EOM, high active)
1	I	Ext FIFO Free
1	O	Load Ext FIFO
1	O	clock to external synchronous FIFO
1	I	TEST Mode
1	I	Internal TEST Mode, GND in normal use
1	I	Place all output pins in high impedance mode
8	O	used for internal testing
127		TOTAL Data pins
52	I	PWR & GND
179		GRAND TOTAL pins

Table B-1
MULTIKRON Pin Usage for a 179 PGA package

Appendix C - TEST MODE Functions

The MULTIKRON chip can be placed in the test mode only by enabling the TEST Mode pin. This enables the TEST Register and any associated test functions. Test input data and test instructions are written into the TEST Register and then executed.

TEST Register

(Address=base+14; Read/Write - Operational Only in Test Mode, 48 bits) This register holds a test instruction (bits 32-47), and any associated data (bits 0-31). The set of TEST instructions, their associated data and functional description are listed in Table C-1. It is easily possible to encode conflicting commands into the TEST register. Detailed knowledge of the MULTIKRON's test architecture is needed to avoid error. For example, bits 0-3 of the TEST instruction must not be set so as to command setting **and** incrementing the same counter(s).

Set Counters Field. When in the test mode, setting bits 0-1 of the instruction portion of the test register (bits 32-33 of the test register) causes the specified counter to be set according to the associated test data in the data portion of the test register. No counters are writable in normal operational mode, but in test mode they can be written. All of the Resource Counters are written simultaneously with the same value. All 32 bits of associated test data are used to write these 32-bit counters. To save space in the timestamp counter and the error counters, groups of four bits are tied together for writing in the test mode. For the 56-bit timestamp counter we only need 14 (56/4=14) bits of associated test data and for the error counters (both the wait state and overrun counters are written simultaneously) we only need 8 (32/4=8) bits of associated test data to specify the write data pattern loaded. The least significant bit of data then sets the four least significant bits of the counter (i.e., data-bit[0]=0 sets counter-bits[0-3]=0000 and data-bit[0]=1 sets counter-bits[0-3]=1111), and so on for each group of four bits in the counters. This method does limit the bit pattern that can be loaded into the counters, but since its purpose is for testing, this limited ability is sufficient for testing all bits and carries for stuck-at faults and to verify that the counter is counting.

Increment Counters Field. When in the test mode, setting bits 2-3 of the instruction portion of the test register (bits 34-35 of the test register) causes the specified counters to be incremented. The normal incrementing of all counters is disabled in test mode and the only way to increment them is through the appropriate test instruction. To save space the error counters (both the wait state and overrun counters) are incremented simultaneously, and all the Resource Counters are also incremented simultaneously. This avoids the extra complexity of specifying which counter of a group to operate on, and also speeds up testing by doing these operations in parallel. There is no associated test data with this instruction.

Select FIFO Read Group Field. When in the test mode, the FIFO can be read by the CPU, but only in groups of 32 bits. Bits 4-5 of the instruction portion of the test register (bits 36-37 of the test register) selects which of these four groups will be selected. Once set, the selected group stays in effect until changed. There is no associated test data with this instruction.

TEST Instruction (TEST Reg) bits	Functional Description
bits 1-0 (33-32)	Set Counter from associated test data
Encodings	(Encoding selects counter)
00	NOP
01	Error Counters - both simultaneously (least significant 8 data bits)
10	Timestamp Counter (least significant 14 data bits)
11	Resource Counters - all simultaneously (all 32 data bits)
	(NOTE that the error counters are the Wait and Overrun counters)
bits 3-2 (35-34)	Increment Counter - no associated test data
Encodings	(Encoding selects counter)
00	NOP
01	Error Counters - both simultaneously
10	Timestamp Counter
11	Resource Counters - all simultaneously
bits 5-4 (37-36)	Select 32-bit Read field of FIFO output - no associated test data
Encodings	
00	FIFO bits 0-31
01	FIFO bits 32-63
10	FIFO bits 64-95
11	FIFO bits 96-127
	(NOTE that "bit 128" of the FIFO output is readable as CSR bit 10)
bit 6 (38)	Disable Network Output - no associated test data
Encoding	
0	Network output Enabled
1	Network output Disabled
bit 7 (39)	Select FIFO data source
Encoding	
0	Connect FIFO input to normal TRACE data sources, No write occurs
1	Load the data portion of the TEST Register and write it into the FIFO
bit 8 (40)	Shiftout (Advance) FIFO & free Shadow registers, no associated test data

Table C-1
TEST Instruction Register Bit Usage

Disable Network Output Field. When in the test mode, bit 6 of the test register (bit 38 of the test register) enables or disables the network output. Once set, the selection stays in effect until changed. When the network output is enabled and data is available at the output of the FIFO, the data will be transferred from the FIFO to the network (if it's operational) and will pass too quickly to be available for the processor to examine.

Select FIFO Data Source Field. When in the test mode, bit 7 of the instruction portion of the test register (bit 39 of the test register) selects the FIFO data source as either the normal Trace Sample data or the associated test data, which is placed in the data portion of the TEST Register. A "0" specifies that the FIFO should load its normal Trace Sample Data when the next Trace sample is requested, and ignore the associated test data. No data is immediately loaded into the FIFO for the "0" option. A "1" specifies that the data portion of the TEST Register, which is being loaded from the 32 bits of associated test data, is immediately written to the FIFO. The 32 bits of TEST Register data is replicated in each of the four 32-bit groups of the FIFO input for a total of 128 input bits (bits 0-127). The least significant bit of the TEST register data is also loaded into the 129th bit of the FIFO (bit 128). The 129th bit of the FIFO (bit 128) is always available from the CSR Register. Once set, the selected mode stays in effect until changed. Note that if the CPU requests a Trace Sample in the Test Mode (by writing to an address in the range 96-127), and if this Data Source instruction bit is set to "1", the sample data is loaded from the data portion of the TEST Register and not the normal Trace data.

Shiftout FIFO Field. When in the test mode, setting bit 8 of the instruction portion of the test register (bit 40 of the test register) shifts out the FIFO (i.e., advances the data in each level of the FIFO, and discards the current readable data in the last level) and frees the Resource shadow registers (i.e., clears their busy flag). Setting this bit of the test register sends a single advance pulse to the FIFO, and need not be reset. Additional writes, with this bit set, will each send an advance pulse to the FIFO.

FIFO Output

(Address = base + 13; Read Only - Operational Only in Test Mode, 32 bits). When in the test mode, the 128-bit output (bits 0-127) of the FIFO can be read by the processor in groups of 32 bits. This allows testing of the FIFO and Trace Sample Data path. The selection of which 32-bit group is specified by the last select FIFO group instruction written to the TEST Register. The 129th bit of the FIFO (bit 128) is a flag to indicate whether or not the Resource Counters are included in the sample.

REFERENCES

[AS90] Antonishek, J.K.; Snelick, R.D. Emulation through time dilation. Proc. 5th Distributed Memory Computing Conf. (DMCC5), Charleston, SC 1990; 8 pp. (1989).

[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.

[LYO89A] Lyon, G.E. Hybrid structures for simple computer performance estimates. Natl Inst of Standards and Technology. Gaithersburg, MD., NISTIR 89-4063; 1989 March, 24 p.

[LYO89B] Lyon, G.E. Capacity-and-use trees for estimating computer performance variations. Proc

Internat. Conf. on Computing and Information. Toronto, May 1989. 5 pp.

[LYO89C] Lyon, G.E. Processing rate sensitivities of a heterogeneous multiprocessor. Natl Inst of Standards and Technology, Gaithersburg, MD., NISTIR 89-4128; 1989 August. 11 p.

[LYO90A] Lyon, G.E. Workloads, observables, benchmarks, and instrumentation. Proc. Joint Internat. Conf. on Vector and Parallel Processing. Zurich: Springer-Verlag; 1990 September. pp. 86-97.

[LYO90B] Lyon, G.E. State occupancy information for performance comparisons. Natl Inst of Standards and Technology, Gaithersburg, MD., NISTIR 4418; 1990 October. 16 p.

[LYO90C] Lyon, G.E. Proposal for Hypercube Timing Registers. (internal communication) Natl Inst of Standards and Technology, Gaithersburg, MD., 26 Oct 1990, 1 p.

[LYSN89] Lyon, G.E.; Snelick, R.D. Architecturally-focused benchmarks with a communications example. Natl Inst of Standards and Technology, Gaithersburg, MD., NISTIR 89-4053; 1989 March. 38 p.

[MCNR90] Mink, A.; Carpenter, R.; Nacht, G.; Roberts, J. Multiprocessor performance-measurement instrumentation. IEEE Computer: 63-74; 1990 September.

[MDRC87] Mink, A.; Draper, J.M.; Roberts, J.W.; Carpenter, R.J. Hardware-assisted multiprocessor performance measurement. Courtois, Latouche (eds) Performance '87; 1988; North Holland; pp. 151-168. ISBN 0 444 70347 0.

[MICA90] Mink, A.; Carpenter, R.J. A VLSI chip set for a multiprocessor performance measurement system. Simmons, et al (eds). Performance Instrumentation and Visualization. 1990; Addison-Wesley; pp 213-134. ISBN 0-201-50937-7.

[MIL90] Miller, B.P.;*et al.* IPS2: The Second Generation of a Parallel Program Measurement System. IEEE Trans Par and Distr Sys, v 1 no 2, pp 206-217; 1990 April.

[MRDC86] Mink, A.; Roberts, J.W.; Draper, J.M.; Carpenter, R.J. Simple multiprocessor performance measurement techniques and examples of their use. Natl Bur of Standards, Gaithersburg, MD., NBSIR 86-3416; 1986 July, 19 p.

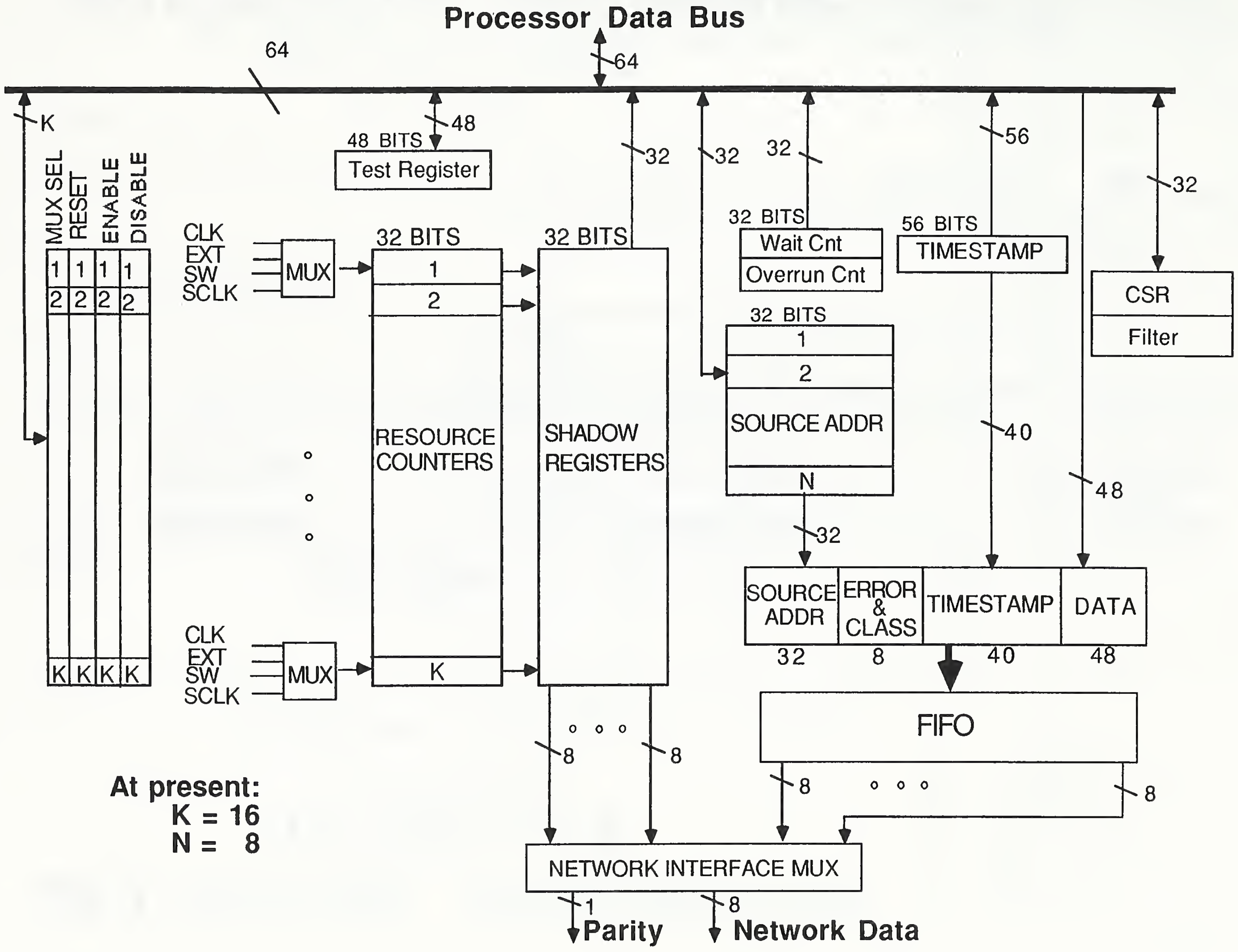
[NAMI89] Nacht, G.; Mink, A. A hardware instrumentation approach for performance measurement of a shared-memory multiprocessor. Puigjaner, Potier (eds) Modeling Techniques and Tools for Computer Performance Evaluation; 1989; Plenum; pp 249-264. ISBN 0 306 43368 0.

[RAM89] 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.

[ROB86] Roberts, J.W. Performance measurement techniques for multiprocessor computers. Natl. Bur. of Standards, Gaithersburg, MD., NBSIR-85-3296; 1986 February. 51 p.

[SNE91] Snelick, R.D. Performance evaluation of hypercube applications: using a global clock and time dilation. Natl Inst of Standards and Tech., Gaithersburg, MD., NISTIR 4630; 1991 July. 26 p.

Figure 1. Block diagram of the **MULTIKRON** Chip



MULTIKRON PROCESSOR INTERFACE

PRELIMINARY

5 JULY 1991 REVISION

Tu	5 ns min, 15 ns max
Tsu	5 ns minimum
Th	approx 2 ns minimum
Tasu	5 ns minimum
Tah	3 ns min, 10 ns max
Tdsu	5 ns minimum
Tdh	Th minimum, 13 ns max
Tbf	15 ns maximum

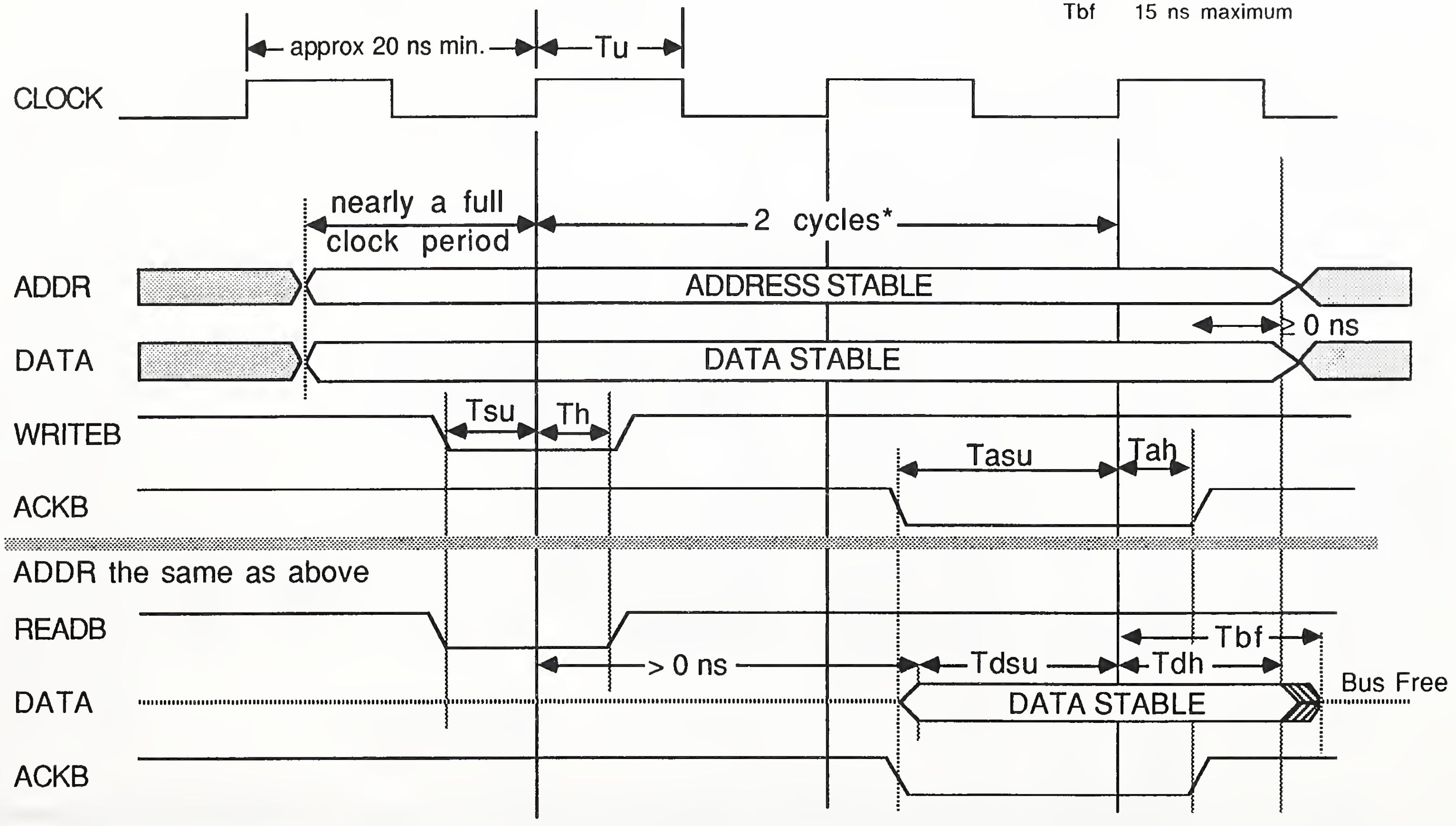


Figure 2

*** One extra CLOCK cycle for "Read RSRC REG"; "Wait for FIFO" or "Wait for RSRCRD" may cause additional waits.**

RJC NIST 910111
910405 910617
910705

MULTIKRON NETWORK INTERFACE

PRELIMINARY
17 JUNE 1991

Tu	10 ns min, 30 ns max
Tsu1	10 ns minimum
Th	3 ns minimum
Tsu2	12 ns minimum
Twh	Tfh + 4 ns = 23 ns max
Tfsu	Tsu2 + 4 ns min = 16 ns min
Tfh	5 ns min, 19 ns max

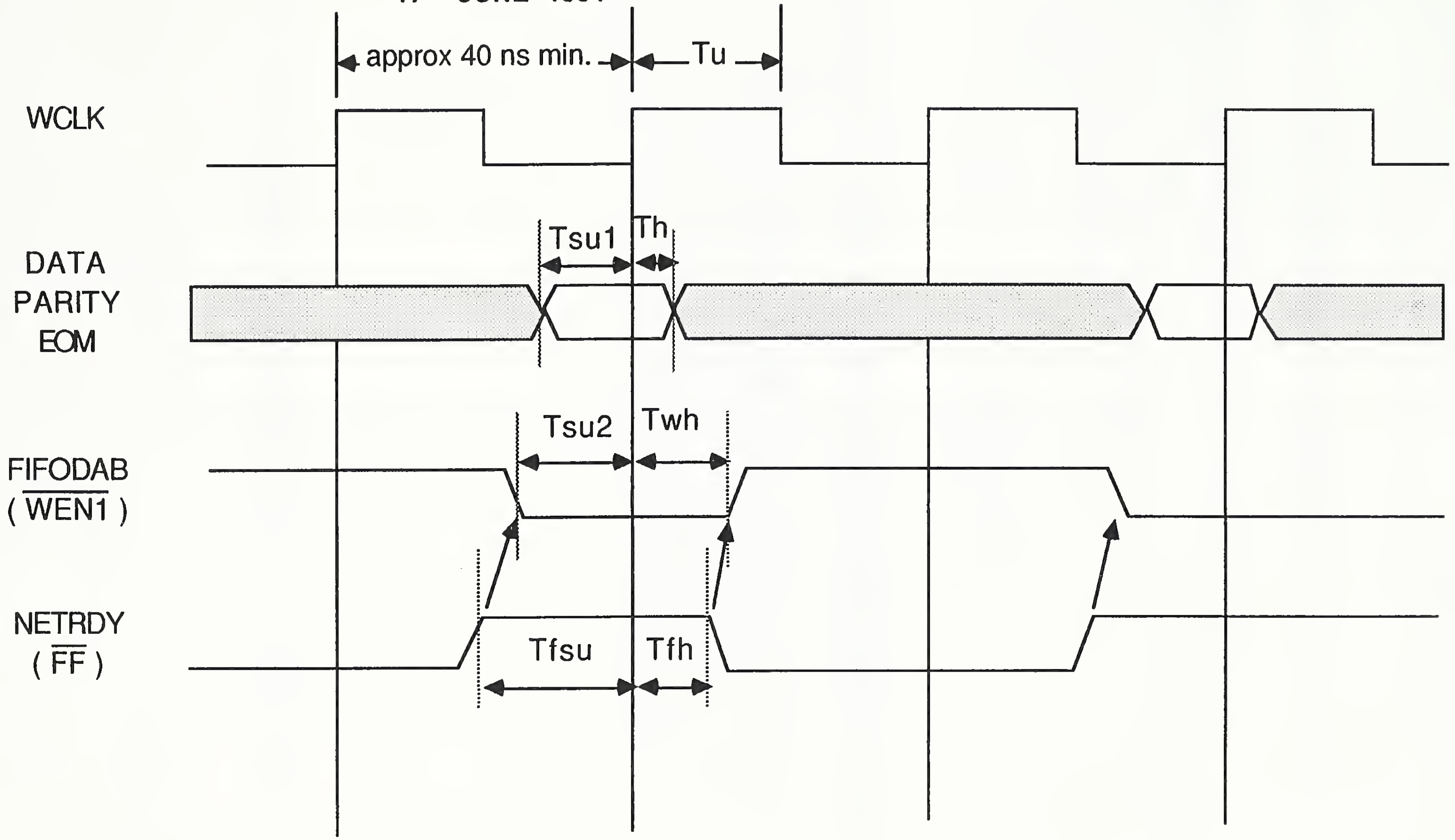


Figure 3

RJC NIST 910429
rev 910617

NIST-114A (REV. 3-90) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY **BIBLIOGRAPHIC DATA SHEET**	1. PUBLICATION OR REPORT NUMBER NISTIR 4737
	2. PERFORMING ORGANIZATION REPORT NUMBER
	3. PUBLICATION DATE MARCH 1992

4. TITLE AND SUBTITLE

Operating Principles of MULTIKRON Performance Instrumentation for MIMD Computers

5. AUTHOR(S)

Alan Mink and Robert J. Carpenter

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY GAITHERSBURG, MD 20899	7. CONTRACT/GRANT NUMBER
	8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Defense Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The single-chip MULTIKRON design replaces our earlier event trace (uTRAMS) and resource utilization (uREMS) performance instrumentation chips. It incorporates a longer timestamp, more bits of user-event and processor identification, and sixteen counters for resource utilization measurements. The initial implementation uses a 64-bit processor bus, though the design allows simple modification to a 32-bit bus. The collection network output has a width of eleven bits (eight data, one parity, and two control), and can transfer up to 25 million data bytes per second.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

computers; hardware support; multiprocessor computers; performance characterization; VLSI

13. AVAILABILITY		14. NUMBER OF PRINTED PAGES 28
X	UNLIMITED	
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).	
	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.	15. PRICE A03
X	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.	

ELECTRONIC FORM