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

NISTIR 5993

Information Technology Laboratory

Operating Principles of the PCI Bus MultiKron Interface Board

Alan Mink
Wayne Salamon

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

CMRF

COMPUTER MEASUREMENT
RESEARCH FACILITY
FOR HIGH PERFORMANCE
PARALLEL COMPUTATIONS

March 1997

Partially sponsored by the
Defense Advanced Research Projects Agency

NISTIR 5993

Operating Principles of the PCI Bus MultiKron Interface Board

Alan Mink
Wayne Salamon

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Information Technology Laboratory
Computer Measurement Research Facility
for High Performance Parallel Computations
Gaithersburg, MD 20899-0001

Partially Sponsored by the
Defense Advanced Research Projects Agency

March 1997

U.S. DEPARTMENT OF COMMERCE
William M. Daley, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Operating Principles of the PCI Bus MultiKron Interface Board

Alan Mink and Wayne Salamon
Scalable Parallel Systems Group
National Institute of Standards and Technology*†
amink@nist.gov

September 19, 1996

Abstract

The MultiKron[1] Experimenter's Toolkit contains a MultiKron interface printed circuit board (currently we have interface boards for the VME bus, SBus, and PCI bus), installation software, data logging software, and analysis software; all of the software supplied is written in C. The Toolkit allows users to take advantage of the NIST MultiKron performance measurement chips (MultiKron_II and MultiKron_vc) in systems that do not already have a MultiKron designed into them. The toolkit board is applicable to both multiprocessor systems and single-processor systems. The Experimenter's Toolkit allows researchers to obtain hands-on experience with the MultiKron performance measurement chips, without the engineering effort required to design and build a hardware interface between the MultiKron and their computer. Over 800,000 Trace Samples can be collected via the MultiKron_II during an experiment to the dedicated toolkit board memory; a practically-unlimited number of Samples can be collected if an optional external data-collection computer is used. The optional MultiKron_vc chip provides 8,192 virtual counters.

Key words: Computers; hardware support; MIMD; multiprocessor computers; performance characterization; printed circuit board; PCB; VLSI

INTRODUCTION

The MultiKron Experimenter's Toolkit contains a MultiKron interface printed circuit board (currently we have interface boards for the VME bus [MIN93], SBus

*This National Institute of Standards and Technology contribution is not subject to copyright in the United States. Certain commercial equipment, instruments, or materials may be identified in this paper to adequately specify experimental procedures. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that materials or equipment identified are necessarily the best available for the purpose.

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

[1]MultiKron is a Trademark of NIST

[MIN95A], and PCI bus), installation software, data logging software, and analysis software. The toolkit board is to be inserted into the corresponding I/O bus of the computer being measured, in this case the PCI bus. All of the software supplied with MultiKron Experimenter's Toolkit is written in C and distributed in source code. The Toolkit allows users to take advantage of the NIST MultiKron performance measurement chips (MultiKron_II [MIN94] and MultiKron_vc [MIN95B]) in systems that do not already have a MultiKron designed into the system. The toolkit board is applicable to both multiprocessor systems and single-processor systems. The Experimenter's Toolkit allows researchers to obtain hands-on experience with the MultiKron performance measurement chips, without the engineering effort required to design and build a hardware interface between the MultiKron and their computer. Up to 838,860 Trace Samples can be collected via the MultiKron_II during an experiment to the dedicated toolkit board memory; a practically-unlimited number of Samples can be collected if an optional external data-collection computer is used. The optional MultiKron_vc chip provides 8,192 virtual counters.

During execution of a program under test, performance measurement data (Samples) are acquired as directed by measurement probes. There are two types of measurement probes, hardware and software. A hardware measurement probe is a wire physically connected to an electrical signal in the system being measured. One of the options for the MultiKron Resource Counters is to count the occurrences of these external signals via a dedicated pin for each Resource Counter. A software measurement probe is the code that generates a Sample. This code is an assignment statement to a memory mapped MultiKron address. An experimenter wishing to obtain performance measurements via the toolkit board must insert measurement probes at points in their program. The probe code can be inserted into the source code, requiring re-compilation, or directly into the executable code via a binary patch. The Samples acquired can be processed "on-the-fly" using a separate data-logging computer or periodically reading the local MultiKron memory. Otherwise they are processed, after program execution, by reading the Samples out of the toolkit board local memory. A toolkit board/MultiKron initialization routine and an performance measurement data analysis program are supplied as part of the MultiKron Experimenter's Toolkit, but the experimenter can replace or modify them as desired.

1 FUNCTIONAL OVERVIEW

The toolkit board provides the necessary interfacing capabilities between a processor and the MultiKron_II/MultiKron_vc via a standard PCI bus, and provides MultiKron_II output data collection facilities.

1.1 Processor Interface

The toolkit board is memory mapped, so all interactions are memory reads and writes (i.e., assignment statements in high level languages) to addresses listed in Table 1. The PCI bus is an I/O bus and thus processor interactions are slower than if the MultiKron were directly interfaced to the processor memory bus. The toolkit

board provides facilities to control, access, and test the MultiKron from the processor being measured. The PCI bus-to-toolkit board data path dynamically and transparently configures to handle 32 or 64 bit data as indicated by the PCI protocol. Although the MultiKron_II/MultiKron_vc are 64-bit devices, they can accommodate 32-bit data transfers with the aid of an internal holding register. Thus the MultiKron_II/MultiKron_vc must be placed in their 32-bit mode when used in a 32 bit PCI bus. Both the MultiKron_II and MultiKron_vc have an internal holding register which is designed to hold the 32 high order MultiKron data bits (bits 32-63) during CPU interactions. This is a single register used for both input and output CPU interactions. For a 64-bit input (CPU write) operation, the experimenter should first load the MultiKron internal holding register with the high order 32 bits of data to be written (if any), and then write the low-order 32 bits directly to the MultiKron causing a full 64-bit input to the MultiKron. Similarly for output, the experimenter should read the low-order 32 bits of data directly from the MultiKron, which causes the high-order 32 bits of data to be loaded into the MultiKron internal holding register. Then the experimenter can read the high-order 32 bits of data from the MultiKron internal holding register.

Neither the MultiKron_II nor the MultiKron_vc provide the mechanism to handle their 32-bit mode as an indivisible operation. Thus, if an interrupt occurs between writing the high-order 32 bits of data into the MultiKron internal holding register and writing the low-order data to the MultiKron, another MultiKron write may occur that will overwrite the MultiKron internal holding register. A similar problem could happen between reading the low-order 32-bit data and the high-order 32-bit data. Since the MultiKron_II and the MultiKron_vc are separate devices, each with their own internal high-order 32 bit holding register, they must be managed separately. If the computer being measured is a multiprocessor the problem is even more pronounced. It is up to the experimenter to guarantee that register overwrite will not occur. Because of the anticipated processing overhead to handle this indivisibility correctly, it is recommended that only 32-bit data fields be transferred to the MultiKron_II in time critical Sampling operations over a 32 bit PCI bus. This reduces the range of values possible in Sampling data by wasting the upper bits of the data path. Less time critical operations, for example loading the Resource Counter control registers on either the MultiKron_II or the MultiKron_vc, should occur very infrequently and can endure the required overhead.

1.2 Configuration Register

A toolkit board Configuration register (Table 2) provides the means for an experimenter to control the MultiKron_II, the MultiKron_vc, and the toolkit board, and are defined as follows. On reset, the contents of bits 0-23 this register are initialized to zeros. Bits 24-31 are the read only board version number and never change.

Bits 0-7, CPU_ID, of the toolkit board Configuration Register represent the individual processor ID signals to the MultiKron_II. The CPU_ID signals are used in multiprocessor applications to identify which processor triggered each measurement Sample and select the corresponding MultiKron_II Source Address Register to be

included in that Sample. Assuming a single processor, one can permanently set CPU_ID = 01 (Hex), resulting in Source Address Register 0 being selected for each Sample.

Bit 8, TEST, of the toolkit board Configuration Register is used for testing and should always be in the operational state ("0") indicated in Table 2.

Bit 9, LOCAL, of the toolkit board Configuration Register selects the destination of MultiKron_II measurement Samples. The choice is between the toolkit board local memory ("1") or to another machine via an external cable ("0"). This is discussed in more detail below.

Bit 10, EXT_CPU, of the toolkit board Configuration Register selects whether the MultiKron_II CPU_ID inputs are connected to the external hardwired toolkit board input signals ("1") that the experimenter may configure, or to bits 0-7 of the toolkit board Configuration Register ("0").

Bit 11, WAIT_CPU, of the toolkit board Configuration Register enables a wait state for both the MultiKron_II and the MultiKron_vc on all processor interactions. This value is read by the MultiKron_II and the MultiKron_vc only upon a RESET, either a hardware reset or a software reset. For fastest operation no wait state ("0") is recommended, while a wait state ("1") will provide slower operations. Invoking a wait state will provide an additional clock cycle for the MultiKron input address lines setup time. This in not necessary on the toolkit boards.

Bit 12, TESTB, of the toolkit board Configuration Register is used for MultiKron testing and should always be in the operational state ("1") indicated in Table 2.

Bit 13, MK_OE, of the toolkit board Configuration Register is the output enable signal to both the MultiKron_II and the MultiKron_vc. This bit must be placed in the operational state ("1") when using either the MultiKron_II or the MultiKron_vc. If this bit is off ("0") then All MultiKron output signals are disabled. This will result in a program crash due to a bus time-out or bus operation abort, or if there is no time-out (as in some PCs) the system could hang or abort the operation with no indication!

Bit 14, WRAPB, of the toolkit board Configuration Register controls whether the toolkit board local memory operates as a simple buffer ("1") or a circular buffer ("0"). This is discussed in more detail below.

Bit 15, EN_EXT, of the toolkit board Configuration Register is used to select the source of the NODECLK and RESET signals which are shared by both the MultiKron_II and the MultiKron_vc. The choice is between a local 40 MHz oscillator on the toolkit board ("0") or an externally supplied clock signal ("1"). The toolkit board uses the PCI bus clock for all its processor interactions (33 MHz maximum). A toolkit board reset command, always resets the MultiKron_II and the MultiKron_vc. If the external clock option is selected, then in addition to the toolkit board reset command, an externally supplied signal will also cause a MultiKron reset.

Bit 16, DIS_STATE, of the toolkit board Configuration Register is used to enable ("0") or disable ("1") the state machine which controls the transfer of data out of the toolkit board FIFO. Disabling the FIFO state machine is mainly for testing purposes, normal operations should keep the state machine enabled. There are 3 FIFO

state machine sequences, one is transferring 4 data bytes to the memory, a second is transferring 2 data bytes to the external cable using the S16D protocol [EDT91], and the third is transferring 1 data byte to the CPU. Once a state machine sequence begins it cannot be disabled, it must continue to completion. Only a reset can clear a hung state machine sequence. When the DIS_STATE bit is active ("1") a new state machine sequence will not begin. If the MultiKron_II is generating samples, the FIFO state machine is active and switching modes (e.g., TEST or LOCAL) in the middle of a sequence can cause undetermined results or even hang the state machine. The function of the DIS_STATE bit is to place the FIFO state machine in its initial idle state so modes can be safely switched. A common test operation is to write to the toolkit board memory address pointer, which requires either TEST to be active or LOCAL to be inactive. Switching these modes can adversely effect the FIFO state machine, if it is not idle. Also, when LOCAL is inactive the S16D protocol is active and any FIFO data will be transferred out of the FIFO following that state machine sequence if the state machine is not disabled.

Bit 18, TS_RATIO, of the toolkit board Configuration Register is used to select the Timestamp clock rate as a function of the NODECLK rate. The Timestamp clock rate equal to 1/4 of the NODECLK rate ("0") is the default, or 1/3 of the NODECLK rate ("1") can optionally be selected. The Timestamp clock is a common signal used by both the MultiKron_II and the MultiKron_vc.

Bit 19, DIV2, of the toolkit board Configuration Register is used to select the MultiKron NODECLK clock frequency. The choices is either the local toolkit board 40 MHz oscillator ("0") or that oscillator frequency divided by two ("1"). The NODECLK clock is a common signal used by both the MultiKron_II and the MultiKron_vc. If the EN_EXT signal is active the NODECLK clock frequency is used directly from the external pin and DIV2 has no effect on it. The recommended settings are to use the local 40 MHz oscillator frequency for the NODECLK (DIV2="0"), and 1/4 of the NODECLK rate for the Timestamp clock (TS_RATIO="0"). This would yield NODECLK = 40 MHz and Timestamp clock = 10 MHz.

Bit 20, WAIT_MEM, of the toolkit board Configuration Register is used to enable a wait state for the MultiKron_vc virtual counter SRAM memory. This value is read by the MultiKron_vc only upon a RESET, either a hardware reset or a software reset. For fastest operation no wait state ("0") is recommended, while a wait state ("1") will provide slower operations. Invoking a wait state will provide an additional clock cycle for storing and retrieving each virtual counter in a page on the dedicated SRAM. At the 40 MHz toolkit board frequency no wait state is necessary.

1.3 Output Data Collection

The toolkit board provides two ways in which to collect MultiKron_II output, selectable via the "LOCAL" option, bit 9 of the toolkit board Configuration Register. One can collect Samples directly in the toolkit board local memory, which is accessible immediately, without any additional devices or wires. The Samples can remain stored there until they are read out for processing or storage. The second method is to collect Samples on an external machine. In this case an external cable connects the

toolkit board to a hardware interface, an S16D commercially available SBus interface [EDT91], on another machine with an SBus. Information will be supplied by NIST to allow the experimenter to obtain the correct cable and connector. Using the S16D interface option on the toolkit board allows experimenters to perform "on-the-fly" analysis of the measurement data and store more Samples than will fit in the toolkit board local memory.

The toolkit board local memory contains 16 megabytes, and can store up to 838,860 Samples (20 bytes per Trace Sample). This memory can be read and written directly by the CPU as 32-bit words, even simultaneously while Samples are being taken. The toolkit board has a dedicated address pointer which it uses to contiguously place Samples into the toolkit board local memory. The CPU can read this address pointer to find out how many Samples are in the local memory and where the last Sample is located. The CPU can also write to this address pointer, but this is primarily for testing purposes and it is not expected to be used operationally.

The toolkit board local memory can be configured as a simple buffer, or as a circular buffer, via the "WRAPB" option, bit 14 of the toolkit board Configuration Register. As a simple buffer, loading starts at address 0 and ends at 3FFFFF. Once the memory address pointer reaches its maximum value, it stops incrementing, and new writes are disabled. Any Samples arriving after the local memory is full will be discarded. As a circular buffer, loading starts at address 0, but upon reaching 3FFFFF the memory address pointer "wraps around" to 0 and starts overwriting older data. Thus, when configured as a simple buffer the toolkit board memory will retain the oldest data, and when configured as a circular buffer it will retain the newest data (the cockpit voice recorder mode). The memory address pointer register is 24 bits wide, although only 22 bits are needed to fully address the entire 16 megabyte memory (as 4M x 32-bit addressable words) since the two least significant bits (bytes) are not stored. Thus when the memory address pointer is configured as a circular buffer and increments from 3FFFFF the results are actually 400000, and from 7FFFFF to 800000, ... , and from BFFFFF to C00000, and from FFFFFF to 0. These are all effectively address 0. The high order 2 bits provide an indication of how many times the memory has wrapped around.

Default Configuration

Initially, it is anticipated that the experimenter will not connect any external wires to the toolkit board. Therefore, the toolkit board Configuration Register should be set to 003201 (Hex). This configuration sets all the test controls inactive, causing storage of MultiKron_II Samples in the local memory, treating it as a circular buffer. All Samples are identified with CPU ID 0 (bits 0-7 of the toolkit board Configuration Register set = 01 (Hex)) and Source Address Register 0. For both the MultiKron_II and the MultiKron_vc, their outputs are enabled and no processor wait states are invoked by setting WAIT_CPU to 0. The local 40 MHz oscillator is selected and its frequency is used directly for the NODECLK clock. The NODECLK frequency is further divided by four to obtain the Timestamp frequency. This is the default setting used by the initialization routine supplied with the Experimenter's Toolkit.

2 HARDWARE ARCHITECTURE

The toolkit board is designed to provide control, access, and testing of both the MultiKron_II and the MultiKron_vc, and also to provide collection of the MultiKron_II output measurement Samples. The principal signals shared by the MultiKron_II and MultiKron_vc are the address and data lines (used for processor interaction), the resource counter external inputs (a selectable option used to count external signal occurrences), and control lines (used for testing and initialization). Specific to the MultiKron_II are signals for the output network lines (for output of measurement Samples to the dedicated 16 Mbyte DRAM memory or the external cable), and the processor ID input lines (used to identify the CPU triggering a Sample). Specific to the MultiKron_vc are signals for the dedicated 8 Kbyte SRAM memory used to transparently store the virtual counters. A block diagram of the PCI toolkit board is shown in Figure 1 and the printed circuit board layout is shown in Figure 2. Processor access to the toolkit board is provided via the PCI bus interface. The toolkit board and the MultiKron chips are memory mapped into the 32-bit memory address space. The PCI toolkit board address decoders recognize PCI bus accesses and convert them into the specified toolkit board operations. Configuration, access, and testing of the MultiKrons are supported through PCI bus access and the toolkit board Configuration register. Two MultiKron_II output Sample collection methods are supported by the toolkit board. One method is to the toolkit board local memory, which requires a memory address pointer. The other method is via an external cable connected to an S16D interface card [EDT91] on a separate machine with an SBus. The architecture of these facilities is discussed more fully below.

The toolkit board is based on a synchronous design using the PCI bus supplied clock. A local oscillator, or an optionally supplied external clock, is used to derive various clocks for both the MultiKron_II and the MultiKron_vc. All decoding and controls are implemented in programmable devices.

2.1 PCI Bus Interface

The toolkit board is a short PCI bus card and fits into a standard PCI bus slot. All toolkit board accesses are 32-bit or 64-bit data transfers aligned on 32-bit boundaries (the two least significant address bits are zero); Block transfer and DMA requests are not implemented. The PCI bus protocol implements a base register address mapping, where each PCI interface board recognizes its own base address and the offset address bits are used internally by the board. The offset address map for the toolkit board is listed in Table 1.

2.2 MultiKron_II CPU ID Signals

The eight, unencoded CPU ID lines are used to allow hardware identification of individual processors in multiprocessor environments. In its intended use, this feature has two functions: (1) it is encoded in a three-bit field in the Sample header to identify the processor taking the Sample, and (2) it selects the contents of one of the eight 32-bit Source Address registers to be placed in the Sample. The MultiKron_II Source

Address registers are written before Sampling starts and should be updated, by the operating system, on context switches. They are intended to contain the node number (if applicable) and the process identification of the process writing the Sample. The CPU ID consists of eight input lines–one per processor, so only one line may be asserted at any time. The toolkit board provides for two options to drive these lines: (1) an toolkit board register – bits 0-7 of the Configuration Register, or (2) external signal lines, via the toolkit board connector, for hardware identification of the active processor. EXT_CPU, bit 10 of the Configuration Register (see Table 2) controls this selection. The connector option will require custom wiring to the connector and consultation with NIST to obtain the connector specification. For single processor machines, the toolkit board register option should be selected and initialized once. The effect is to select a single MultiKron_II Source Address Register for all Samples.

2.3 MultiKron External Counter Signals

The external counter inputs are one of the selectable counting sources shared by both the MultiKron_II and MultiKron_vc Resource Counters. A toolkit board connector option to provide these external signals will require custom wiring to the connector and consultation with NIST to obtain the connector specification.

One of the PCI bus signals, DEVSEL#, is a device active signal. To measure the utilization of the PCI bus we hardwired two versions of the DEVSEL# signal to the external input pin of the MultiKron Resource Counters, numbers 0 and 2. One version is the direct signal. The other version is the signal gated only when the toolkit board is active. These signals can be used to determine the overall PCI bus utilization, and the fraction used by the toolkit board. By properly configuring counters number 0 and 2 to use these input signals as enables, they will tally the total number of clocks that the PCI bus is busy and busy only with the toolkit board, respectively. These counters provide the numerator for our utilization fraction. Configuring counter 4 to tally the total number of elapsed clocks, will provide the denominator for our utilization fraction. These clocks are operating at a rate faster than the PCI bus clock, thus yielding a fairly accurate measure. Each MultiKron Resource Counter is a 32 bit counter, but "even" numbered counters can be configured as 64 bit counters by concatenating them with their neighboring "odd" numbered counter. When counting high speed clocks, a 32 bit counter will overflow quickly (e.g., in less than 90 sec at 50 MHz).

2.4 Toolkit Board FIFO

The MultiKron_II output network sends measurement Sample data to the toolkit board FIFO, which is 16 bits wide by 1024 ranks deep. Only 10 of the 16 bits are used. Each transfer consists of eight data bits along with an "end of Sample" flag and an odd parity bit. The "end of Sample" flag and an odd parity bit transferred from the MultiKron_II are discarded, only the eight data bits are saved. The toolkit board FIFO provides buffer storage for up to 51 20-byte Trace Samples to improve peak Sample rate performance and reduce the risk of Samples being lost due to

collection delays. For diagnostic purposes, there are toolkit board commands to manually control the flow of data from the toolkit board FIFO.

The input and output of the FIFO are independent of one another. A toolkit board FIFO write clock is produced by the MultiKron_II (at half the rate of the MultiKron_II input NodeClk). The FIFO read clock is the PCI bus clock. The FIFO full flag generated by the FIFO input logic is fed into the MultiKron_II, where it can stop further MultiKron_II output when the toolkit board FIFO is full. The MultiKron_II can continue to generate Samples until its small internal FIFO is filled, at which point it either forces the processor to wait until room is available or discards new Samples and sets an error flag. The course of action to take is determined by bits in the MultiKron_II CSR, which are written prior to Sampling. The FIFO empty flag generated by the FIFO output logic is used to determine whether there is data available to read. This data will either be sent over the external cable (16 bits at a time) to an S16D interface on another computer or written into the toolkit board local memory (32 bits at a time), depending on the toolkit board configuration option selected.

2.5 FIFO Testing

A test mode can be activated by bit 8 of the toolkit board Configuration register. Enabling this test mode allows processor control of the toolkit board FIFO output. A read to the toolkit board TEST address (see Table 1) will remove one entry from the FIFO and send it to the processor via the PCI bus. A write to the toolkit board TEST address will remove four entries from the FIFO and send it to the toolkit board local memory at the location pointed to by the toolkit board memory address pointer. The toolkit board memory address pointer will then be incremented.

2.6 External Cable Interface

The External Cable Interface provides the logic needed to extract two entries from the toolkit board FIFO, combine them into a single 16-bit value and send it over an external cable to an S16D interface [EDT91], a commercial SBus I/O interface board, installed on another machine. The S16D board plugs into an SBus slot of the external data collection computer. Disabling the LOCAL option ("0"), bit 9 in the toolkit board Configuration register, allows experimenters to perform "on-the-fly" analysis of measurement Samples and also store more Samples than the toolkit board local memory. The average Sample collection rate is slower via the non-LOCAL option than the toolkit board local memory.

2.7 Toolkit Board Local Memory

The 16 Mbyte toolkit board memory is configured as four banks of 1M x 32 bits, built from 4M x 16 bit DRAM chips. Four entries are extracted from the FIFO and combined into a 32-bit word and then written into the local memory via a pipeline holding register. Although a FIFO entry is 16 bits, only 8 bits are Sample data. The

toolkit board local memory requires 22 address bits to access any 32-bit word, 2 bits for bank select and 20 bits for word select. Since these are word accesses, aligned on word boundaries, the byte level address is always 00. The toolkit board local memory may be accessed via the PCI bus at any time, since toolkit board memory arbitration will interleave PCI bus access with Sample storage. Writing to the toolkit board local memory by the processor is provided mainly for testing purposes.

A 24-bit counter serves as the (32-bit word) memory address pointer for Sample storage; the PCI bus (byte) address is used for processor access. In principle, this counter may be read or written by the processor at any time. However, since a write to the counter while it is storing MultiKron_II Samples could cause the stored Samples to be scrambled, a safety interlock has been programmed into the controlling logic which ignores any write to the memory address pointer while in LOCAL mode. If there is a need to write to the counter during the course of an experiment, the experimenter should disable the FIFO state machine via the DIS_STATE control bit, and wait for the current state machine sequence to complete, then disable LOCAL mode in the toolkit board Configuration register and write the new value into the memory address pointer, then re-enable LOCAL mode and enable the FIFO state machine. Enabling or disabling LOCAL while Samples are being taken has an undetermined effect on the state machine handling that transfer and, therefore, care should be taken that the FIFO state machine is idle.

Before sampling begins, the user should initialize the toolkit board memory address pointer. The initialization code distributed with the Experimenter's Toolkit provides for this initialization. The user can read the memory address pointer during the experiment to determine how much of the memory space has been written. The memory address pointer indicates the next available memory location.

In the simple buffer mode, Sample writing is halted when the memory is full. If new Samples continue to be generated, the toolkit board will discard all new Samples arriving from the MultiKron_II.

In the circular buffer mode, when the memory is full, Sample writing continues from the beginning of memory by wrapping the memory address pointer around to zero, overwriting the oldest Samples and thus retaining the most recent Samples generated.

3 SOFTWARE FOR PCI/LINUX

The software included with the MultiKron toolkit consists of a device driver to map the MultiKron Interface Board into user memory, some test programs, sample applications, and basic programs to retrieve the data from the MultiKron interface board. Analysis software is also included to produce time interval reports based on the MultiKron data. The software is distributed in source form.

The requirements for using the software are: A PCI bus personal computer with the MultiKron Interface Board installed; Linux kernel version 1.3.84 or higher installed. (The software will work with earlier 1.3.x kernels with minor modification). The kernel must be compiled with module support. The Gnu C compiler, version

2.7.0 or later must also be installed. (Other C compilers may work, but have not been tested because Gnu C is the standard Linux compiler.)

3.1 Obtaining the Software

The software may be downloaded from `cmr.ncsl.nist.gov` as follows:

```
ftp cmr.ncsl.nist.gov
Name: anonymous
 Guest login ok, send your complete e-mail address as password
Password: e-mail address
ftp> cd pub/multikron
ftp> bin
ftp> get mk_Linux.tar.Z
ftp> quit
```

3.2 Installing the Software

The software is distributed as one Unix tape archive file called `mk_linux.tar.Z`. The file must be uncompressed with the Unix `uncompress` utility. When the tape archive is extracted using the `tar` command, an install subtree `mk_toolkit_linux` is created.

In the root directory of the install subtree, there are two scripts: `prepare.mk` and `install.mk`. `prepare.mk` will compile the device driver module, data retrieval and reduction code, and the test programs. Run `prepare.mk` before `install.mk`

The script `install.mk` will:

- copy the "man" pages to the specified directory from the install subtree.
- copy the header file `mk.h` to the specified directory.
- install the library `libmk.a`.
- copy the driver module and load/unload scripts to `/dev`.
- run `/dev/mk.LOAD` to load the MultiKron driver module.

The `install.mk` script *must* be run with root permission. The target install directories can be changed by modifying the shell scripts.

A typical installation would be as follows, assuming the downloaded tar file is in directory /tmp:

```
cd /usr/local/
uncompress /tmp/mk_linux.tar.Z
tar -xvf /tmp/mk_linux.tar
cd mk_toolkit_linux
```

The tar command will create the following directories:

```
/usr/local/mk_toolkit_linux
/usr/local/mk_toolkit_linux/docs
/usr/local/mk_toolkit_linux/install
/usr/local/mk_toolkit_linux/src
```

3.3 Installing the Device Driver

Two shell scripts are provided to assist in loading the MultiKron device driver. The shell script `mk.LOAD` will install the device driver module and create the special file `/dev/mk0` used to access the device. The shell script `mk.UNLOAD` will unload the driver and remove the special file. Both of these scripts are installed in the `/dev` directory by the `install.mk` script, and require root access to run. The script `install.mk` will run `mk.LOAD` to load the driver module.

To manually load the kernel module, use the `insmod` command:

```
insmod mk.o
```

See the "man" page for the `insmod` command for more information.

To unload the module, use the `rmmmod` command:

```
rmmmod mk
```

The device driver module only needs to be loaded once per system boot, or after it has been unloaded. A call to `/dev/mk.LOAD` could be placed in the local startup script to ensure that the module is loaded automatically at boot time.

To list the currently loaded modules, use the `lsmod` command.

3.4 Running the Test Programs

There are several programs provided with the toolkit that are used for testing the operation of the MultiKron chip and the MultiKron interface board. Examination of these programs will show how to access various parts of the MultiKron interface board. The programs are located in `src/tests`. A make file is provided for compiling all or part of the test suite.

The program `mk_mem_test` tests the operation of the MultiKron interface board memory. To run the program, enter:

```
mk_mem_test <start> <count>
```

where `start` is the memory word location to start the test, and `count` is the number of memory words to test. This test program writes a pattern into each memory word, then reads the pattern back. If the value read does not match that written, then an error is reported.

The program `test_mk` is used to test the basic data flow to and from the MultiKron chip. Reads and writes are performed to one source address register in order to verify the data transfer occurs correctly. A resource sample is taken to verify the capture of

the resource data from the user. Next, the first 2048 locations of memory are written then read. Last, the MultiKron timestamp register (low 32-bits) are displayed before and after a 10 second wait. This test verifies that the timestamp register is counting at the correct rate.

The program `mk_sample_test` tests the operation of the trace and resource sampling functions. This program shows how to use the source address registers and how to take trace and resource samples. The data from the MultiKron is displayed after the samples are taken. Trace samples are captured with the CPU ID set to 0, then CPU ID set to 1. The 20 bytes of data from the trace sample are then displayed, along with the value of the memory address pointer. For resource samples, the 84 bytes for each sample are displayed. Again, the samples are taken for CPU ID set to 0, then CPU ID set to 1.

The program `mk_nite_test` performs continuous, extensive testing of the MultiKron, and is designed to be run for several hours. The tests exercise the memory, sampling, and register usage of the MultiKron chip. The program checks the MultiKron operation against expected results, and reports any errors found. The program is designed to be run with no user interaction except for the final check of the results. To stop the tests, press CTRL-C at any time.

3.5 Data Retrieval and Reduction

In the directory `src/misc` are several programs used to retrieve the data from the MultiKron interface board and reduce the data.

`longlong.c` A set of utility functions to handle 64-bit math

`mk_rdsample.c` retrieves the data from the MultiKron interface board and writes the data into the binary file `mk.dat`, which is used by `mk_expand` and `mk_rpt`

`mk_expand.c` prints the MultiKron data from the file `mk.dat` in human readable form

`mk_rpt.c` analyzes the data from file `mk.dat`, producing a condensed report with several statistics based on the data

`mk_status` reports the status of the MultiKron interface board and the MultiKron chip

Using `mk_rdsample`:

```
mk_rdsample [-f filename] [-s start_addr] [-e end_addr] [-v]
```

where options are:

`-f filename` write data to file `filename` instead of default file `mk.dat`

`-s start_addr` retrieve data starting from memory location `start_addr`; default location is `0`

`-e end_addr` stop retrieving data at memory location `end_addr`; default is to stop at the current interface board memory address pointer location

`v` verbose flag; print some progress messages

Using `mk_expand`:

```
mk_expand [-c] [-e] [-h] [-r[dox]] [-s[dox]] [-t[dox]] [-u[dox]] [-x]
```

where options are:

`c` output counters for resource samples; default is to print the header, timestamp,source ID, and user data for resource samples

`e` output elapsed time; default is actual timestamp value for sample

`h` print a heading at top of data output

`r` select radix for resource counter output data format; see below

`s` select radix for source address output data format; see below

`t` select radix for timestamp output data format; see below

`u` select radix for user data output data format; see below

`x` expand header data by separating the bits in the header

For options `r`, `s`, `t` and `u`, the user must also specify `x` for hexadecimal output, `d` for decimal output, or `o` for octal output. The default radix for these values is hexadecimal.

Using `mk_rpt`:

```
mk_rpt [-v] [-f file] [-h] [-e value] [-s] [-n] [-d]
```

where options are:

`v` verbose output

`f file` use `file` as the interval description file instead of the default `interval.info` file

`h` create histogram data

`e` change event size from default (see below)

`s` create summary histogram across all processors

`n` don't sort the histogram, list in chronological order

`d` use double (64 bit) math for timestamp values

`mk_rpt` is an analysis program which provides simple statistics from the MultiKron data. The binary input data file `mk.dat`, which contains the MultiKron measurement data, must be present in the current directory. This file is normally created by `mk_rdsample`. One other input file is needed: `interval.info`, although this file can be of a different name if the `-f` option is used. The file `interval.info` contains the user's description of the MultiKron measurement data. It is an ASCII file, and can be created by the user with any editor.

`mk_rpt` does not process resource counter information. If a resource sample is encountered, the resource counters are discarded and the sample is processed in the same manner as a trace sample.

In the following description below, *sequence numbers* are the numbers assigned to arbitrary events by the experimenter and are written to the MultiKron by the probe code in the program being measured. See the sample program `src/sample/sample1.c` for an example on how to do this. Another example can be found in `src/tests/mk_sample_test.c`.

The ASCII file `interval.info` contains lines broken into fields. The number of fields is dependent on the class of the interval. The are currently four classes supported by `mk_rpt`:

Class 1 a single interval delineated by 2 events:

Event 1 the beginning of the interval

Event 2 the end of the interval

For example, the line in `interval.info` might be:

```
1 10 20 "whole program"
```

where `1` is the class, `10` is the sequence number for the beginning event, `20` is the sequence number for the end event, and `''whole program''` is the description of the interval.

Class 2 a single interval with 2 possible outcomes, e.g., a successful end or a failed end, or an `if-then-else` clause, consisting of a sequence of 2 out of 3 events:

Event 1 the beginning of the interval

Event 2 the first possible end of the interval

Event 3 the second possible end of the interval

For example, the line in `interval.info` might be:

```
2 81 82 83 "if then" "if else"
```

to show the path of an `if` check, where `2` is the class, `81 82 and 83` are the user selected sequence numbers for the three events, and `"if then" "if else"` are the text descriptions of the intervals.

Class 3 an interval consisting of two consecutive sub-intervals, where the end of the first sub-interval indicates the beginning of the second sub-interval

Event 1 the beginning of the interval which also indicates the beginning of the first sub-interval

Event 2 the end of the first sub-interval which is also the beginning of the second sub-interval

Event 3 the end of the interval which is also the end of the second sub-interval

So, the line in `interval.info` might be:

```
3 8 17 12 "process input" "write output"
```

where 3 is the class, `8 17 and 12` are the user selected sequence numbers for the three events, and `"process input" "write output"` are the text descriptions of the intervals. The report also displays the total interval with the name "*sub-interval-1 sub-interval-2*", which in this example would be `"process input process output"`.

Class 4 an interval delineated by two events, where the the start and end events may be out of sequence, but are always ordered chronologically, (e.g. nested or overlapped):

```
start1 start2 ... end1 end2
```

or

```
start1 start2 end1 start3 end2 end3
```

The software can only handle one event of this type, for now. The line in `interval.info` might be:

```
4 27 36 "message timer"
```

where 4 is the class, 27 is the send message event, and 36 is the acknowledgment of receipt event, but multiple sends may go out before a message is received, and `"message timer"` is the text description of the interval.

3.5.1 Putting it all together

What follows is a brief overview of the steps to take to use the MultiKron board and software provided in this toolkit. First, the program to be measured must be modified by inserting software measurement probes. The example below shows a measurement probe in both the sender and receiver code. Next, create an `interval.info` file with the class, event numbers, and text descriptions of the intervals that are to be traced. Run the modified programs to be measured, then execute `mk_rdsample` to retrieve

the measurement data from the MultiKron, creating the file `mk.dat`. One can use `mk_expand` to inspect the raw measurement data in file `mk.dat`. Finally, use `mk_rpt` to produce the condensed interval reports with their associated statistics from the data file `mk.dat`.

3.6 Running the Sample Programs

In the directory `src/sample` are programs that show some basic application use of the MultiKron interface board. `src/sample/sample1.c` shows how to use interval classes 1, 2, and 3. The program does some system calls inside of a loop, and the overall time spent in various parts of the program can be calculated from the data captured by the MultiKron. Note that the time values are wall clock time, and not process time.

What follows is some pseudo-code showing how class 4 would be used. The code consists of two separate processes running on separate machines. Each process takes trace samples in its own MultiKron. The data is then read from each machine by `mk_rdsample`, and combined into one `mk_dat` file by a simple concatenation.

Sender process

```
#define SEND_EVENT 10

{
  /* init MultiKron device      */
  MK_DEFAULT_MAP();

  for(;;) {

    mk_ts1 = SEND_EVENT;

    send_msg();

  }
```

Receiver process

```
#define RECEIVE_EVENT 20

{
  /* init MultiKron device      */
  MK_DEFAULT_MAP();

  for(;;) {

    mk_ts1 = RECEIVE_EVENT;

    send_msg();

  }
```

The `interval.info` file would look like this:

```
4 10 20 "message send/receive latency"
```

References

[EDT91] Engineering Design Team, Inc., "S16D High Speed 16-bit I/O Interface for the SUN Sparc station–User's Guide," 1100 NW Compton Dr, Beaverton, OR 97006, July 1991.

[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.

[MIN95A] Mink, A., Nacht, G. G., and Antonishek, J., "Operating Principles of the SBus MultiKron Interface Board", National Institute of Standards and Technology, NISTIR 5652, May 1995.

[MIN95B] Mink, A., "Operating Principles of MultiKron Virtual Counter Performance Instrumentation for MIMD Computers," National Institute of Standards and Technology, NISTIR 5743, Nov. 1995.

TABLE 1: PCI Toolkit Board Offset Byte Address Map

Address (Hex)	Description
0,000,000 - 0,000,FFC	N/A
0,001,000 - 0,001,1FC	MultiKron_II
0,001,200 - 0,001,20C	toolkit board
0,001,200	toolkit board Software Reset
0,001,204	toolkit board Configuration Register
0,001,208	toolkit board Memory Address Pointer
0,001,20C	toolkit board TEST Operation Read - read 1 data byte from the FIFO Write - assemble 4 data bytes from the FIFO and write it into memory and incr the MAP
0,001,210 - 0,003,FFC	N/A
0,004,000 - 0,007,FFC	MultiKron_vc (if installed)
0,008,000 - 0,FFF,FFC	N/A
1,000,000 - 1,FFF,FFC	toolkit board Local Memory

TABLE 2: PCI Toolkit Board Configuration Register Format

This register is located at relative (byte) address 0000,1204 (Hex)
Its recommended default value is 003201 (Hex)

Bits	Name	Description
0-7	CPU_ID[0..7]	CPU_ID to MultiKron (multiplexed with XCPU[0..7] via EXT_CPU)
8	*TEST	Used for toolkit board testing–operationally set to 0 1 = "manually" control the output of the toolkit board FIFO
9	LOCAL	1 = MultiKron_II Samples are stored in local toolkit board memory, 0 = MultiKron_II Samples are sent to the External Cable Interface
10	EXT_CPUID	Select the source of the MultiKron_II CPU ID inputs 1 = use toolkit board external inputs XCPU[7..0] 0 = use CPU_ID[7..0] from the toolkit board Configuration register
11	WAIT_CPU	Enable a MultiKron CPU Wait State (1 active)
12	*TESTB	Used for MultiKron testing–operationally set to 1 0 = places MultiKron into TEST mode
13	MK_OE	1 = enable MultiKron outputs (if Low ALL outputs are disabled)
14	WRAPB	1 = toolkit board memory configured as a simple buffer 0 = toolkit board memory configured as a circular buffer
15	EN_EXT	Enables the use of an external source for the toolkit board clock and reset 1 = use external source for the toolkit board clock and reset 0 = use local clock & reset, ignore external clock & reset signals
16	DIS_STATE	1 = Disable board FIFO state machine 0 = Enable board FIFO state machine
17	N/A	
18	TS_RATIO	1 = 3-1 ratio of MultiKron NODECLK clock to Timestamp clock 0 = 4-1 ratio of MultiKron NODECLK clock to Timestamp clock
19	DIV2	1 = Divide local Osc by 2 for MultiKron Node clock 0 = Use local Osc directly for MultiKron Node clock
20	WAIT_MEM	Enable a MultiKron_vc memory wait state (1 = active)
21-23	N/A	
24-31	V_NUM	Toolkit Board Version number = 0x90 - Read Only

*used for testing only

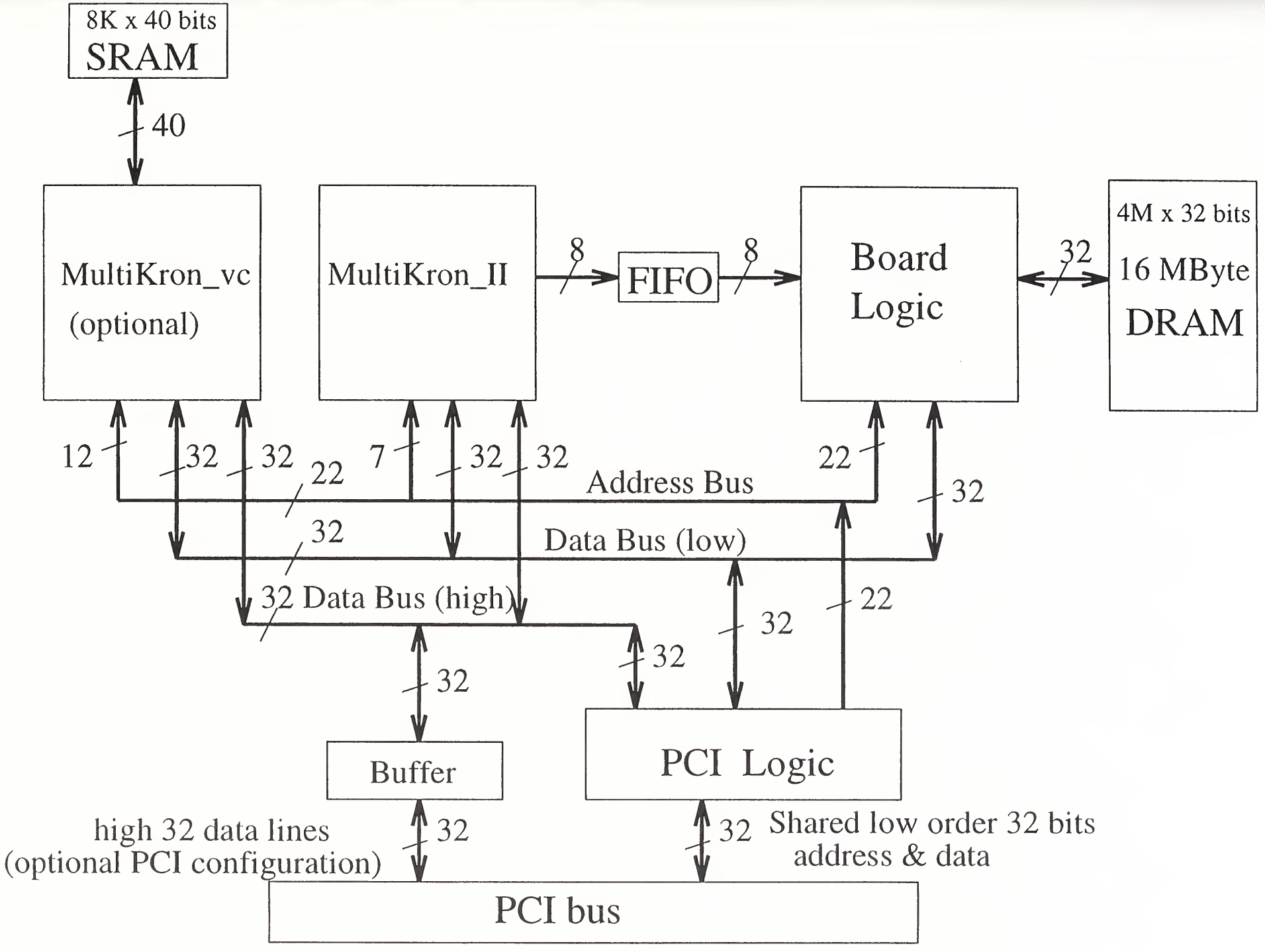

Figure 1. Block diagram of PCI Toolkit Board.

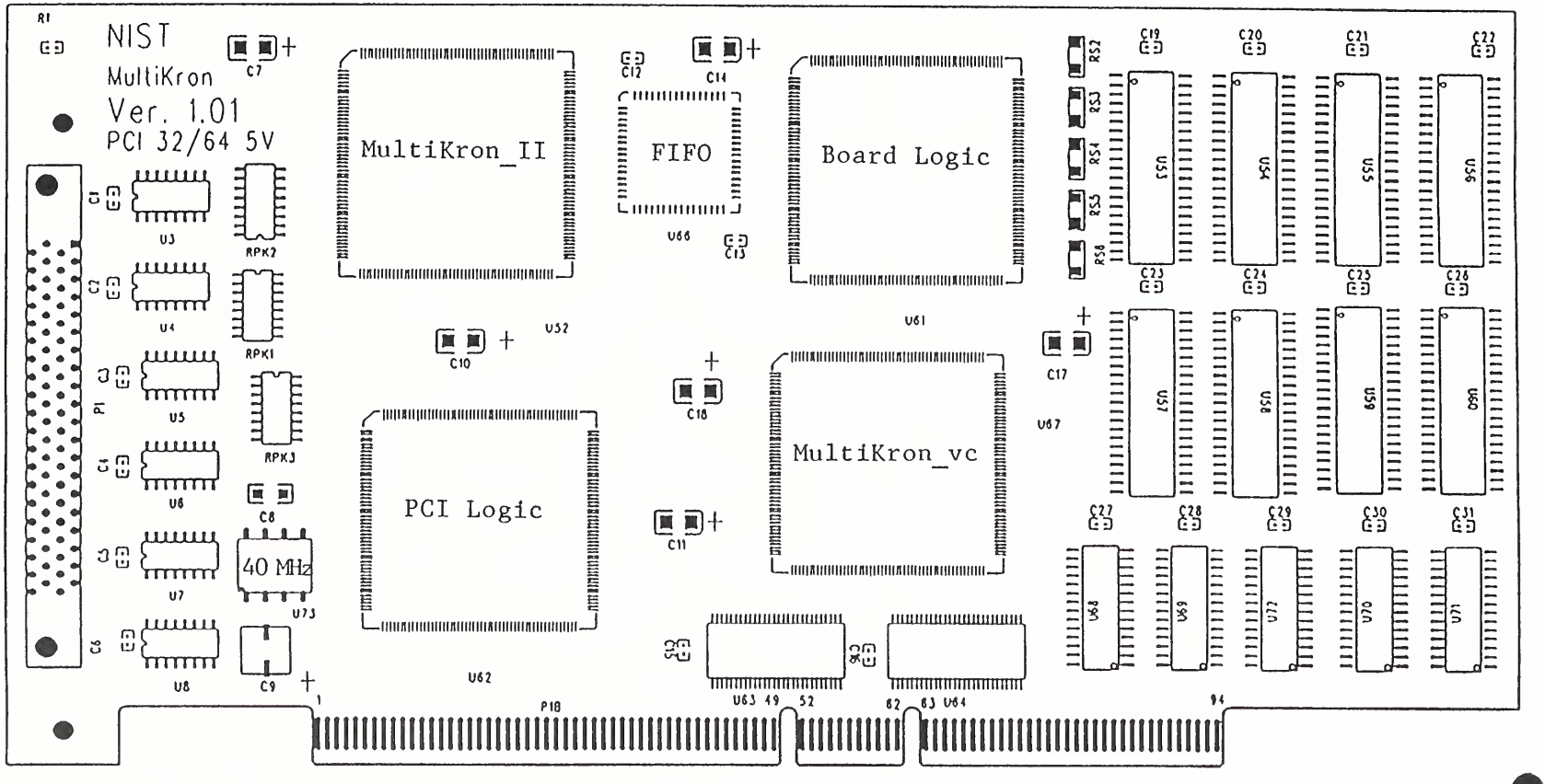

Figure 2. PCI MultiKron toolkit printed circuit board layout; surface mounted, single-sided.