

Design and Implementation Aspects for the SMiLE Hardware Monitor

Robert Hockauf, Jürgen Jeitner, Wolfgang Karl, Robert Lindhof, Martin Schulz *

Vicente González, Enrique Sanchis, and Gloria Torralba *

Abstract-- The SMiLE project (Shared Memory in a LAN-like Environment) at LRR-TUM investigates in high performance cluster computing using system area networks. In the context of this project, a hardware monitor is being developed to observe the SAN traffic. This hardware monitor is therefore capable of delivering detailed information about the run-time communication behaviour of applications run on SMiLE clusters. The central part of this monitor consists of a content-addressable counter array managing a small working set of the most recently referenced memory regions. This hardware monitor has been designed and is currently being implemented at LRR-TUM.

Index Terms--SCI, Cluster SMiLE, VHDL.

I. INTRODUCTION

The SMiLE project (Shared Memory in a LAN-like Environment)¹ [1] at LRR-TUM investigates in high performance cluster computing. It utilizes the Scalable Coherent Interface (SCI) [3] as the interconnection technology. This network fabric, which is standardized in the IEEE standard 1596-1992 [2], utilizes a state of the art split transaction protocol and currently offers link speeds of up to 400 MB/s. The base topology for SCI networks are small ringlets of up to 8 nodes, which hierarchically can be organized to configurations of up to 64K nodes. One of the fundamental features of SCI is the capability to directly access remote physical memory without software intervention resulting in a hardware DSM (Distributed Shared Memory) abstraction. This can be used to efficiently implement both message passing and shared memory programming models [1]. The latter one is implemented by merging the HW-DSM capabilities of SCI with software DSM techniques creating a hybrid DSM system that allows to directly exploit the communication benefits of the underlying network while still providing a true shared memory abstraction to the programmer [18].

In both programming paradigms, however, it is often the case that applications initially perform much worse than expected and require extensive tuning. While the

performance of message passing codes can easily be assessed and optimised using standard tools, the same task for shared memory codes in such a hybrid-DSM environment is much more difficult. The main source of inefficiencies are often excessive remote memory accesses across the network. Although these accesses via hardware-supported DSM deliver an extremely high communication performance, they are still an order of magnitude more expensive than local ones. To improve the efficiency of parallel applications a tool set consisting of both a run-time adaptive system with a locality optimiser and comprehensive visualization is needed. The goal of such a tool set is to focus on improving the locality of memory reference and to minimize remote memory accesses.

Growing complexity of hardware, however, makes this kind of performance evaluation and characterization more difficult, especially in a DSM environment on top of a NUMA architecture like PC-Cluster with HW-DSM support. The challenge is to design a powerful performance monitor, which is able to deliver detailed information about the run-time communication behaviour and to help the run-time adaptive system in making correct decisions concerning partitioning and re-distribution of data and threads. Such a monitoring system, the SMiLE hardware monitor, which has been designed and is currently under development at LRR-TUM, is presented in this work.

A hardware monitor alone, however, can not be the final stage. A comprehensive on-line monitoring infrastructure must be developed along with it [17] that allows the analysis and visualization of the observed behavior and the on-line optimisation of applications, either automatically though runtime intervention or by providing the programmer with precise performance hints. The main goal of all these tools implemented within this framework is to improve the locality of memory accesses, through a tight co-operation of monitoring hardware and tools [8].

The remainder of the paper is organized as follows: Section II describes the general design principles of the hardware monitor followed by the description of the interface to the SAN, the so called B-Link in Section III, the counter modules in Section IV, and the host interface in Section V. Section VI presents the current overall status and introduces some continuing work planned for the future. The paper is then rounded up by the discussion of related work in Section VII and some concluding remarks in Section VIII.

* Dpt. Computer Science, LRR, Technical University Munich, Germany.

* Dpt. Electronic Engineering, University of Valencia, Spain.

¹ More information about SMiLE is available at <http://smile.in.tum.de/>.

II. THE SMiLE HARDWARE MONITOR

The SMiLE hardware monitor as shown in Figure 1 consists of three modules: a B-Link interface, a counter module (or F.R.L. *Flexible Range Logic*), and a PCI interface in cooperation with the commercially available PCI bridge PCI 9060 [20]. The B-Link interface is the connection to the SCI network adapter, in this case the SMiLE adapter [10], and responsible for acquiring the network traffic. The data is then handed on to the counter module, which is responsible for keeping track of those events that relate to the parameters defined by the user. The data can then be transferred to the host through the PCI interface. In the next sections the individual parts will be described and the current status of their implementation will be highlighted.

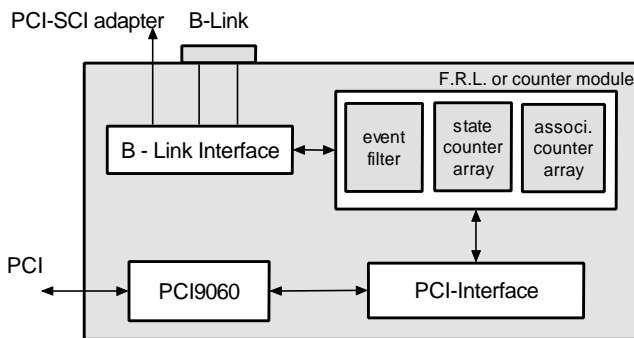


Figure 1. Architecture of the SMiLE hardware monitor

III. B-LINK INTERFACE

The B-Link, a 64-bit-wide synchronous bus, is a logical part of the PCI-SCI adapter [10] and serves as the carrier of all incoming and outgoing packets to and from the SCI link. It has been defined by Dolphin ICS [21] for their own SCI adapter cards and is the host-side interface of the LC-1 link chip [16] responsible for transmitting any SCI packet from and to the respective node.

The B-Link interface is therefore the central point of the hardware monitor on which all remote memory accesses can be monitored and acquired. The information includes transaction commands (read, write, lock, or unlock), source and destination IDs, physical memory addresses (page frame number and offset), and a description of the packet type (incoming, outgoing, response, or request).

In Figure 2 the B-Link packet format is shown. The information needed by the counter module can be found in the first and second 64 bits (*Blink_Data_1* and *Blink_Data_2*). The states when those two packets are present on the B-Link are in the following denoted as BD1 and BD2. Only during those states data will be acquired and transferred to the monitor (Figure 3). The actual payload data and the next fields are not relevant for monitoring and are therefore ignored.

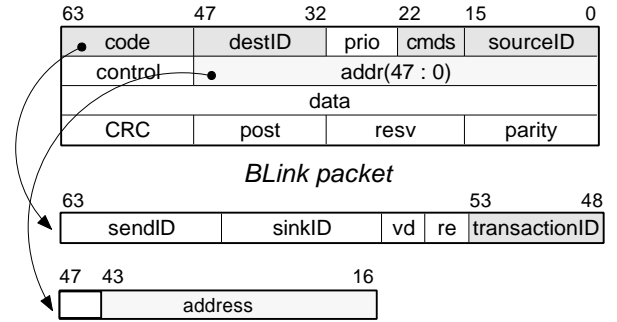


Figure 2. B-Link packet at the PCI-SCI card [10]

The design was simulated in VHDL. For the B-Link decoder entity the inputs of interest are:

- blink_data(64) : Data bus
- b_clk(1) : Signal clock
- bframe(1) : Beginning of relevant data
- reset(1), monitor_reset(1)

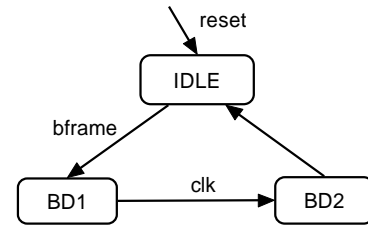


Figure 3. State machine for decoding the relevant B-Link information

The state machine of Figure 3 for the decoding of incoming B-Link blocks is designed in VHDL using three states: IDLE, BD1 (command and echo bits), and BD2 (address):

```

blink_frame: PROCESS (clk_b1, reset,
monitor_reset) -- handle frames
BEGIN
IF reset='1' OR monitor_reset = '1' THEN
--no valid SCI-command
decode_state<=IDLE;
--wait for frame start
ELSIF (clk_b1'event and clk_b1 = '1' ) THEN
--statemachine 1 (synchronous)
CASE decode_state IS
--first 64 bit (phase 1)
WHEN IDLE=> IF (bframe='0') THEN
--frame starts
decode_state<=BD1;
--get packet type and cmd -> Statemach. 2
ELSE
decode_state<=IDLE;
--nothing important happen (bframe='1')
END IF;
--second 64 bit (phase 2)
WHEN BD1=>
decode_state<=BD2; --get address if
REQUEST/RESPOND on next clock
WHEN OTHERS =>
decode_state<=IDLE; --block complete
END CASE;
END IF;
END PROCESS blink_frame;

```

In the first phase (BD1), the packet type information is extracted, more specifically the following fields:

- *Code*: only needed *trans_ID* = *blink_data*(48-53)
- *Command*: for decoding response, request, write, read, or lock packet, *blink_data*(16-22)
- *Dest_ID*: target node for packet, *blink_data*(32-47)
- *Source_ID*: source node for packet, *blink_data*(0-15)

The current monitor and card design is only intended for clusters with up to 32 nodes reducing the valid bit range of the last two fields to *blink_data*(32-36) and *blink_data*(0-4) respectively.

During the second phase (BD2), the address information is extracted from the B-Link packet. The outputs generated are:

- *Page number*: physical location of the data higher address bits / page *blink_data*(12-27)²
- *Page offset*: physical location of the data lower address bits / offset *blink_data*(2-11)

The necessary storage for the transaction identifiers and addresses is implemented in the decoder using a 64x27bit

SRAM. This memory is shown in Figure 4 and is basically used to look up and retrieve addresses for given *trans_ID*s (*page_number* and *page_offset*) from response packets. In the request case, the *trans_ID* and the address are stored in the table. Additionally, a column has been added to indicate actions that trigger responses (read or lock).

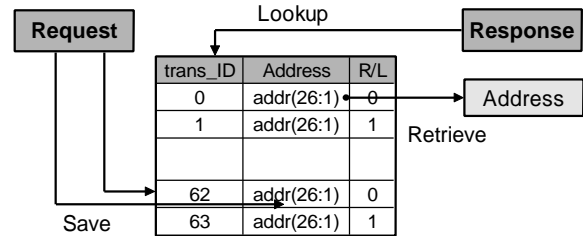


Figure 4. Table for storing transaction addresses

The design for the B-Link decoder has been done using VHDL and includes the SRAM component, which was generated with Xilinx's *LogiBLOX* [7]. For simulation and debugging the *Work View Office* tool from View Logic has been deployed. Figure 5 shows the waveforms from one of the tests with read responses and requests. Several test benches have been successfully simulated for debugging and design verification purposes.

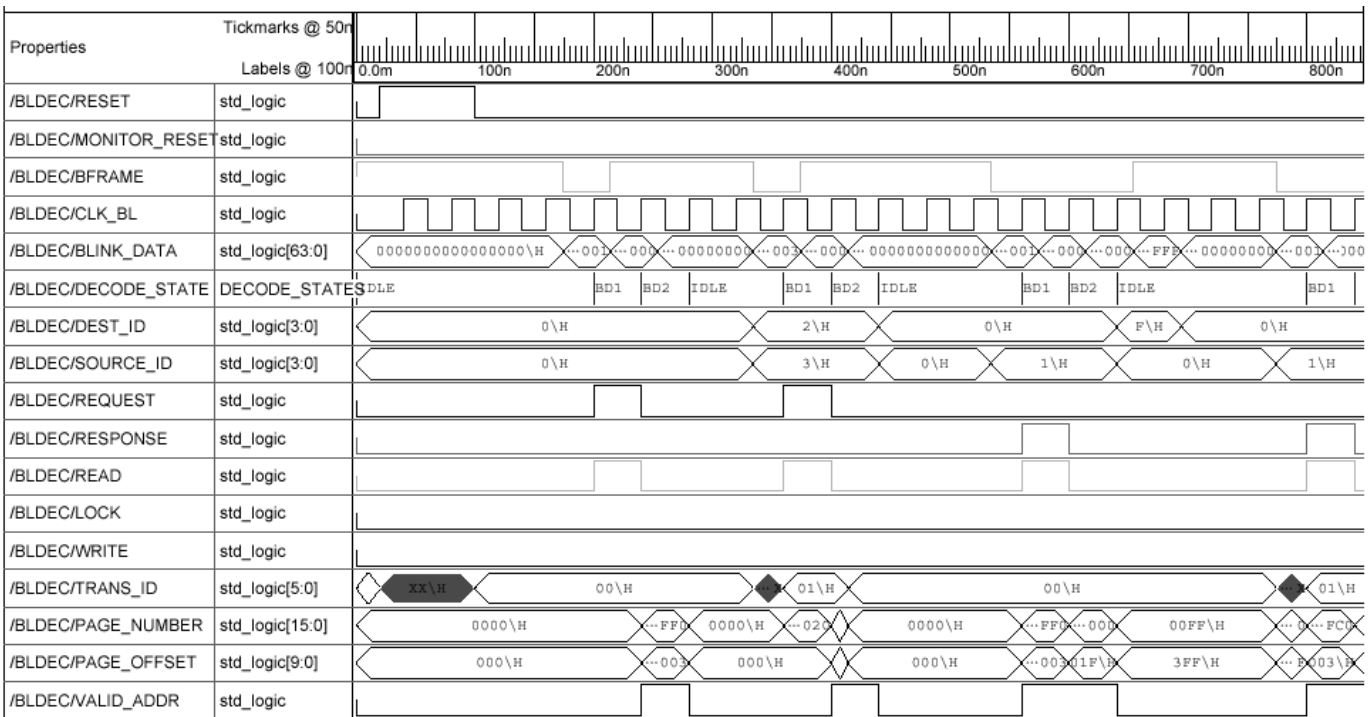


Figure 5. Waveforms taken from the simulation environment using a test bench with read responses and requests

² Allows the monitoring of 64K pages. The four higher address bits are not implemented in the SMiLE PCI-SCI adapter card.

IV. COUNTER MODULE OR F.R.L.

The primary part of the hardware monitor is the *counter module* or *Flexible Range Logic*. It receives the monitoring event information from the B-Link interface described above and then records them as specified by the user. Its three components, depicted in Figure 6, – the *event filter*, the *static counter array*, and the *associative counter array* – allow the programmer to utilize it in two different working modes for performance analysis: a static and a dynamic mode. The static mode, for which the *event filter* and the *static counter array* [4] are responsible, allows users to explicitly program the hardware for event triggering and the processing of actions on freely definable SCI regions. This can be used to monitor given data structures or parts of arrays and will likely be applied in combination with language specific profiling extensions. The dynamic working mode [5] is a histogram-driven monitoring, in which the counting is not controlled by events as in the static mode, but rather all packets through the B-Link are monitored in order to provide a fine-grained monitoring statistics across the complete application's working set. This mode is therefore suitable for gaining a first overview of an application's behavior without the need to annotate the given code or program the monitor explicitly to certain areas of interest.

This proposed hardware monitor has been simulated using a software simulator [8,11]. Several experiments were executed using various benchmarks, mostly from the SPLASH-II suite [19] in order to find the right tradeoff between counting resources and hardware complexity. These experiments have shown that the optimal number of counters keeping the necessary hardware resources at a workable level while providing enough details to the monitoring infrastructure is between 16 and 32 counters.

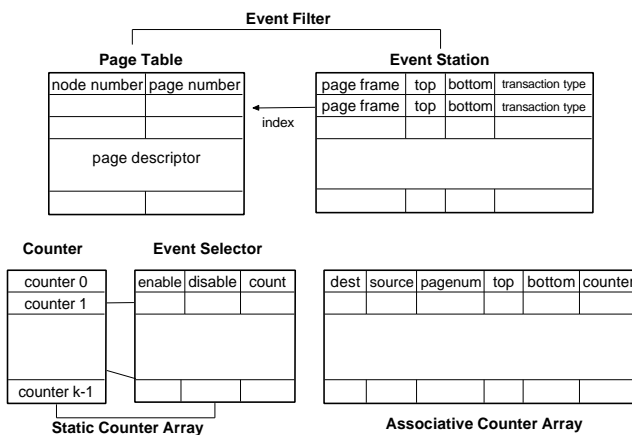


Figure 6. The hardware structure of the counter module

A. THE STATIC PART OF THE F.R.L

The main purpose of the so called static part [4] within the flexible range logic is to register and count well defined events. These events can be either transactions within the

SCI network or so called *global events*, including a *timer event*. Global events are driven by the monitoring software or a clock timer and can be used to implement some kind of interaction between the user and the monitoring hardware.

To define memory locations within the global memory the static part uses two tables as depicted in Figure 7: the *page table* and the *static event table*. Within the page table 32 entries can specify pages of 4KB by using their node ID and the higher 20 bit of their local address. The static event table defines the SCI events to be monitored. Each entry consists of a pointer to one entry of the page table, two registers defining a top and a bottom address within the 4KB page specified by the corresponding page table entry, and some flags to filter read, write, lock, request, and response transactions.

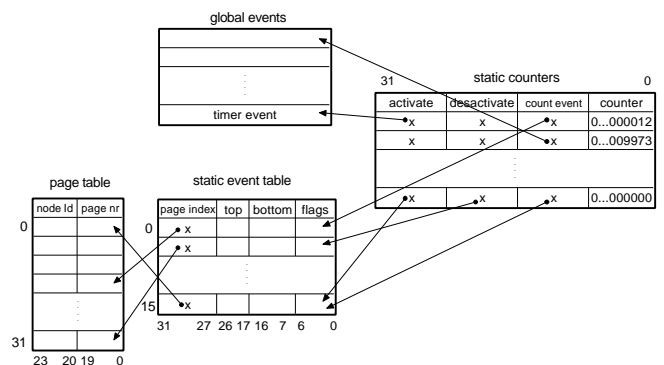


Figure 7. Static Part of the F.R.L.

The actual static counters are controlled by three registers holding event IDs: one for activating, one for deactivating, and one to define the events to be counted. In addition, each counter delivers a special output signal that allows to detect counter overflows.

The page table has 32 entries, which have to check the incoming address simultaneously. This requires an equal comparator of 24 bits (20 bits address and 4 bits node ID) for each entry. In order to lower the signal width, the outgoing result is encoded to a 5 bit page table index.

The static event table needs two magnitude comparators for each of its 16 entries, which have to be capable of checking whether the 10 bit page offset is well within the defined range. In order to also filter the transaction type and establish a correlation to the page segment (from the page table entry) each entry of the static event table also requires a 12 bit equal comparator. The outgoing signals are being encoded to a 4 bit event table index and are then combined with another 4 bit signal, which holds the information about global events, resulting in a signal of 5 bits representing the complete event index. The registers of the static counters each point to one of these 32 events. This requires an additional 48 (3 events for each of the 16 static counters) equal comparators of 5 bits. The necessary hardware

components are then completed by the 16 counters themselves, which hold a 16 bit value each.

In order to detect an overflow, each counter has a special overflow signal. These overflow signals are again encoded to a 4 bit overflow index signal for further processing. On the other side, an incoming 4 bit index signal needs to be decoded to 16 single signals (one for each counter) in order to select a counter for reading or resetting.

Besides those components listed above (comparators, en-/decoders and counter) the static part needs almost no additional logic.

B. DYNAMIC FLEXIBLE RANGE LOGIC

The data collected within the dynamic mode [5] of the FRL allows the creation of memory access histograms. The user is able to use the gathered information to gain an overview of the overall application runtime behavior without any specific knowledge about the application structure. This allows the easy detection of bottlenecks and communication hot spots and is very useful as a first step in the tuning process. It can be followed by a more precise evaluation of those points of interest using the static mode described above. The modes therefore complement each other and provide the user with a comprehensive base for application tuning.

For the following description of this dynamic part of the F.R.L. SCI-transactions will be referred to as events. An event consists of a 5 bit source ID, a 5 bit destination ID, a 26 bit SCI-Address, and a transaction type. In a first step, all incoming events are filtered based on their transaction types. Only events matching a user defined type stored in a global configuration register of the hardware monitor will be accepted and handed on to the counting logic.

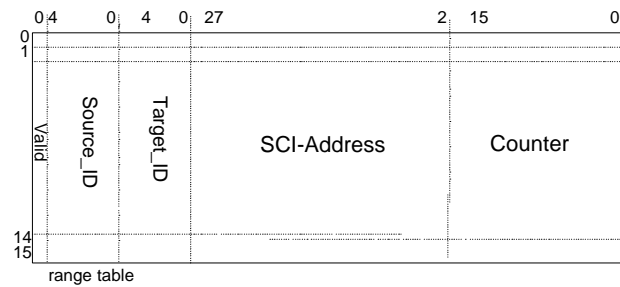


Figure 8. Range table structure

The core of the dynamic monitoring mode is the Range Table Structure shown in Figure 8. Source_ID, Destination_ID, and SCI-Address hold the corresponding data of an event, Valid indicates that a range table entry is filled with valid event attributes, and the remaining 16 bit are used as the actual Counter to keep track of the frequency of similar events. During the utilization of the monitor in this dynamic mode, every occurring relevant event is saved in this range table. If an event appears for the first time, a free range table entry is filled with the event attributes, the Valid bit of the entry is set, and the counter is

initialized with "1". If later a similar event is observed the corresponding counter is incremented. In order for two events to be considered similar, the Source_ID and Destination_ID of two events have to be equal. The SCI-Addresses, however, are allowed to vary, but need to be within a certain distance from each other. The maximal distance allowed, can be set by the user through a monitor configuration register.

To keep the necessary hardware resources at an acceptable level, only 16 entries are available on the monitor. In order to anyway allow a comprehensive monitoring overcoming this resource limitation, a simplified LRU-algorithm is used to swap range table entries to a buffer in the RAM of the host machine. The same mechanism is initiated in the case of a counter overflow.

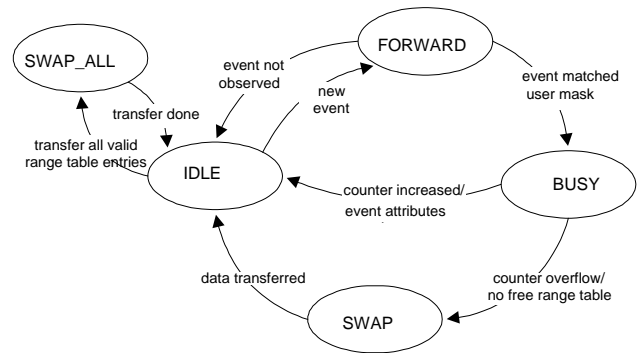


Figure 9. State machine for the F.R.L.

To implement the intended behavior described above, the state machine shown in Figure 9 was implemented with five states. After a reset or initialization of the monitor the machine stays in the Idle state, waiting for events. While Idle is active the user can demand data transfers or change monitor configuration registers to influence the measurements. If an event occurs the state Forward becomes active. During Forward the event's transaction type is compared with the user defined transaction type mask and the evaluation of the event attributes with valid range table entries is started. The state machine jumps back to Idle if the event doesn't pass the test, to Busy in the other case. Within Busy the determined action is executed, i.e. if the occurred event is similar to a range table entry, the corresponding counter is increased, else the event attributes are stored in a free range table entry. In the first case the state Swap is activated if there is a counter overflow, in the second case if all 16 table entries hold valid data. In any other case the execution starts back at the Idle state directly after Busy. Back at Idle, again user demands will be granted or new events processed. If the user signaled a data transfer the Swap_All state is activated. All valid range table entries are transferred to the local RAM before the state machine jumps back to Idle.

V. IMPLEMENTATION OF THE HARDWARE MONITOR SCI_HWM AND THE PCI TARGET INTERFACE.

The PCI interface connects the SMiLE hardware monitor with PC's I/O bus and is used to both configure the monitor from the host side and transfer the acquired data back to the host. It is implemented using the commercially available PCI bridge chip PCI 9060 [20]. In this section, the PCI interface will be described together with the evaluation and prototyping process that was undertaken to implement it.

A. PCI_TARGET INTERFACE

The PCI_Target Interface is the connection between host, which is used to run the monitor tool set, and the F.R.L.. It is used to set user parameters that influence the monitoring process as described in the previous section, as well as give the host software access to the acquired data in the counter registers. It is also used by the F.R.L. for the dynamic swapping mechanisms described above.

It includes the components SWAP_OUT, WRITE/READ, Pointer administration, and further components for the initialization and control of the static and dynamic F.R.L. Its core, however, is a Dual Ported RAM, which is organized as a ring buffer (4kx32bit).

The administration takes place at the PCI_Target interface within the component SWAP_OUT. It is also responsible for the management of the counter value pointing into the host ring buffer for both FRL_stat and FRL_dyn. It can be activated by two different events:

1. SWAP_OUT by user demand over the PCI interface: All 16 counter values for static or dynamic events and their identifier are stored in the ring buffer.
2. In the case of a counter overflow by the FRL_stat or FRL_dyn: The counter value or the index (counter number) together with the overflow identifier is stored in the ring buffer.

If during any of the above operations, an overflow of the ring buffer is detected, the PCI_Target interface raises an interrupt and thereby triggers the complete save of all data by the host system software.

For the administration of the write/read operation 3 pointers are used: Besides a read and write pointer, indicating the current state of the ring puffer, an auxiliary pointer is used that always contains the storage address increased by one of the last written memory location. When this auxiliary pointer reaches the value of the reading pointer, the ring buffer is full and the host is notified as described above.

In addition, the following parameters are necessary for the initialization and control of the monitor and can also be accessed by the host software through the PCI interface: the own node ID to distinguish between inbound and outbound packets, the Max_Range area to determine the granularity of segments to be monitored by the FRL_dyn, and the

transaction type (READ, WRITE, LOCK etc.) filtered within the dynamic F.R.L., as described above.

B. H.O.T. II SYSTEM

For the evaluation of the PCI interface logic and the implementation of a prototype, the Hardware Object Technology system³ (H.O.T. II) is deployed [6]. H.O.T. II is a PCI based generally Reconfigurable Computing system containing both hardware and software components. One of its main features of this system is that the hardware components can be reconfigured dynamically by an executable program (so called Run Time Reconfiguration).

In the case of the prototype of the SMiLE hardware monitor, this can be used to dynamically load the dynamic or the static part of the F.R.L., as never both at the same time are required. This helps to save space on the prototype and reduces the complexity of the verification process.

C. EVALUATION SETUP USING H.O.T. II

The evaluation setup used for the SMiLE hardware monitor can be seen in Figure 10. The hardware is attached to a PCI card and a mezzanine daughter board. The PCI card contains two dual port RAMs (4kx32bit), a initialization Flash EEPROM (2Mbyte), a configuration RAM Cache (512kbyte), the Configurations Cache Manager CCM (EPLD XC95108), a Clock generator, and a FPGA XC4062XLA (approx. 60k logic gates).

The Dual Port RAM can be accessed through the PCI bus or the FPGA. The Flash EEPROM contains the initial configuration with the PCI core. On POWER ON or on RESET the initial configuration is being loaded into the FPGA. Two designs can be held within the RAM Cache and then loaded dynamically at run-time. Within the EPLD XC95108 the control logic (CCM) for the board is included.

The FPGA XC4062 contains the PCI core for the binding to the PCI interface, the H.O.T. interface for a comfortable use of the hardware resources, and the PCI_Target interface describe above in order to initialize and control the F.R.L.. The daughter board is connected through two 160 pin connector to the PCI card. For this prototype, it is equipped with a XC4085XLA with approximately 85k logic gates. This FPGA then holds the B-Link interface and either the FRL_static or the FRL_dynamic logic. These designs are loaded and verified over a serial cable (Xchecker™) from Xilinx [7]. In addition, two 50 pin connectors are used for the connection to the TUM PCI-SCI adapter [10], which in this experimental setup is installed in a separate machine.

Using this very efficient and easy-to-use prototype setup, the verification process for the SMiLE hardware monitor is close to be completed. Only some small detailed issues in the PCI interface still need to be resolved. Therefore a working system and first results are expected soon.

³ Hardware Object Technology is a trademark of Virtual Computer Corporation

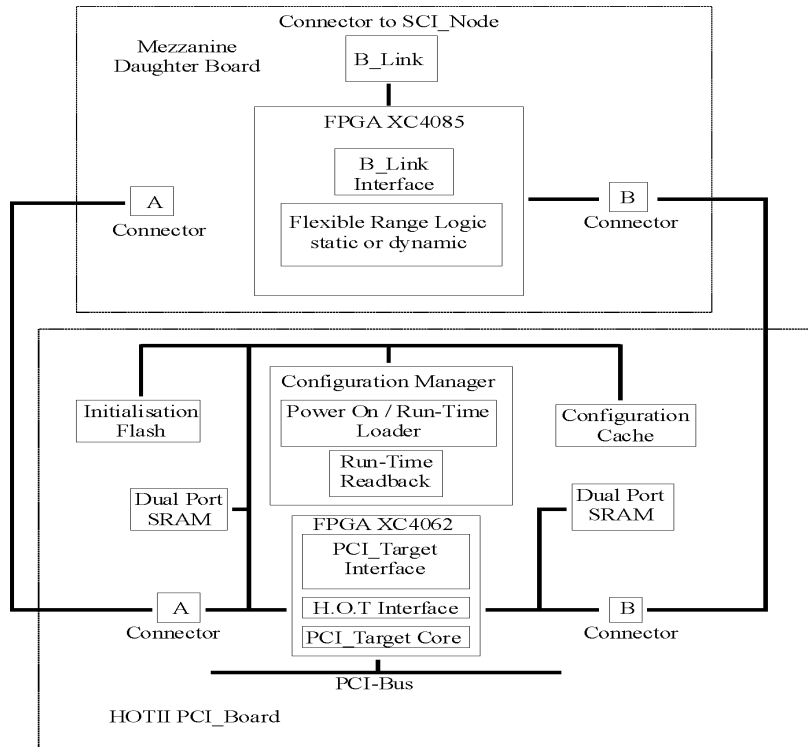


Figure 10. PCI_Target Interface

VI. CURRENT STATUS AND FUTURE WORK

Both, the F.R.L. and the PCI-Interface, have already been completely designed and tested at the LRR-TUM. The dynamic and static parts of the F.R.L. have been evaluated on one HOT2-Evaluation board. The PCI-Interface is currently implemented into an HOT2_FPGA as described in Section V.

Once the design of the Blink_decode and F.R.L have been fully verified, the next step will be their integration into a common design, as depicted in Figure 11. As the target for synthetization the XILINX XC4085XLA has been chosen, due to the fact that the XC4000 Series chips [10] have generous routing resources to accommodate the most complex interconnect patterns.

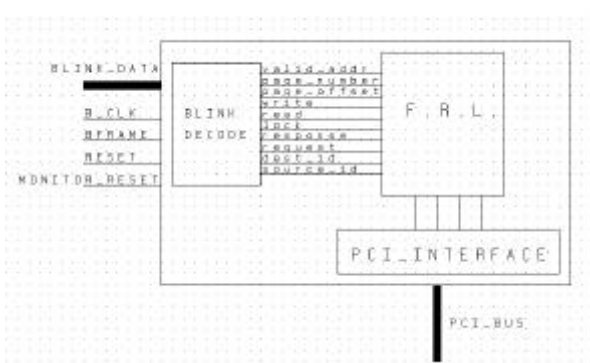


Figure 11. Hardware monitor top

For the evaluation of the monitor's design principles, a software simulator of the proposed hardware monitor already serves as the basis of a large number of experiments [9,11]. Those experiments will be continued with the goal of developing the comprehensive monitoring environment mentioned above. Once the hardware monitor is fully implemented in FPGAs and is available, they will then be moved to the hardware platform, resulting in the complete monitoring infrastructure as shown in Figure 12.

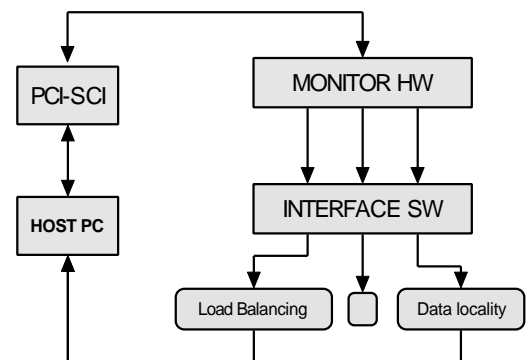


Figure 12. Hardware and software monitors

VII. RELATED WORK

Over the last years, basic monitoring support has become increasingly available on research as well as on commercial machines. For the CC-NUMA FLASH multiprocessor system the hardware cache coherence mechanism is complemented by components for the monitoring of fine-grained performance data (number and duration of misses, invalidation etc.) [12]. Also some modern CPU chips incorporate hardware counters which collect information about data accesses, cache misses, etc. For some multiprocessor systems, this acquired data can be exploited by performance analysis tools [13]. The information gained from those approaches, however, is not at that level of detail and granularity as with a direct monitoring of the NUMA interconnection fabric, as it is possible with the SMiLE monitoring approach.

Closer related to the SMiLE monitoring system is the monitoring approach proposed by Martonisi et. al. for the SHRIMP multiprocessor system. It is based on Myrinet [22] and is implemented as an additional Myrinet Control Program run by the LANai processor [14]. However, as the traffic over Myrinet is generally based on larger packets and intended for message passing systems, also here the granularity of information is less than in the SMiLE approach.

Another trace instrument for SCI-based systems has been designed at the Trinity College in Dublin [15]. In contrast to the SMiLE monitor it allows full traces of the complete network traffic and stores the data in a large memory space on the monitoring hardware itself. While this increases the accuracy of the monitoring, it requires, due to the large amount of data acquired, an offline evaluation and is therefore not suited to support adaptive runtime systems or on-line visualization tools.

VIII. CONCLUSIONS

Precise and low-level performance monitoring is essential for the development of efficient parallel applications. This is especially true for distributed shared memory systems that rely on hardware support for communication in a hybrid fashion. In those systems, it is impossible to observe any communication in software without incurring a substantial overhead, as all communication is performed implicitly at in advance unknown locations. Only additional hardware support in the form of a hardware network monitor is capable of performing this task.

The design and the initial implementation work of such a monitor for the SMiLE system has been presented and discussed in detail in this work. Attached to an SCI adapter card through the B-Link, the main data path between the PCI bus and the SCI link for all common SCI cards, this monitor is capable of observing the complete traffic from and to a node. With its flexible counter logic, which is capable of both the static monitoring of given memory

segments as well as the general observation of a priori unknown access hotspots, it is capable of providing the user with a complete assessment of the application's behaviour and potential performance bottlenecks.

In order to reach an efficient implementation of this hardware design in the forms of FPGAs, several complexity reducing concepts have been applied. Most importantly, the counter registers, responsible for a great percentage of the total complexity have been limited to only a small width. To compensate for this, the monitor includes within its own PCI host interface several mechanisms that enable the swapping of full counters into main memory. This allows, in combination with an appropriate driver infrastructure on the main host node, to transparently use the monitor beyond the limited count width without increased hardware complexity or limited functionality.

This comprehensive hardware approach alone, however, is only the first, but necessary step towards efficient and precise performance monitoring. In addition, an online monitoring software infrastructure is required to allow an evaluation, visualization, and dynamic application optimization based on the acquired data. Such an environment closing the gap between low-level monitoring data and application level abstractions is currently being designed and implemented at LRR-TUM. Together with previous research within SMiLE, this will result in a complete DSM programming tool set intended to help the programmer during the whole application development, debugging, and optimization process.

REFERENCES

- [1] W. Karl, M. Leberecht, M. Schulz. Supporting Shared Memory and Message Passing on Clusters of PCs with a SMiLE, In A. Sivasubramaniam and M. Lauria, editors, *Proceedings of CANPC'99*, volume 1602 of *Lecture Notes in Computer Science*, Springer Verlag 1999, Berlin
- [2] IEEE Computer Society. *IEEE Std 1596-1992: IEEE Standard for the Scalable Coherent Interface*. The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA, August 1993
- [3] Hermann Hellwagner and Alexander Reinefeld, editors. *SCI: Scalable Coherent Interface: Architecture and Software for High Performance Computer Clusters*, volume 1734 of *Lecture Notes in Computer Science*. Springer Verlag, 1999.
- [4] Hockauf, Robert: Ein in programmierbarer Logik realisierter Hardware-Monitor zur Beobachtung von Kommunikationsereignissen in einem SCI-basierten PC-Cluster Diploma thesis, Technical University Munich, Department of Computer Science, LRR, August 1999.
- [5] Jeitner, Jürgen: Dynamische Laufzeitanalyse in einem SCI-basierten Cluster. Diploma thesis, Technical University Munich, Department of Computer Science, LRR, February 2000.
- [6] Virtual Computer Corporation: <http://www.vcc.com/Hot2sys>.
- [7] Xilinx: <http://www.xilinx.com/>

- [8] W. Karl, M. Schulz, J. Tao, Using the SMiLE Monitoring Infrastructure to Detect and Lower the Inefficiency of Parallel Applications, In *Proceedings of HPCN-Europe*, volume 1823 of *Lecture Notes in Computer Science*, Springer Verlag, 2000. Berlin.
- [9] XILINX: The Programmable Logic Data Book, San Jose 1996.
- [10] Acher, G.; Karl W.; Leberecht, M.: The TUM PCI/SCI Adapter. In: Hellwagner, H., Reinefeld, A. (Eds.): *SCI Scalable Coherent Interface Architecture and Software for High-Performance Compute Clusters*, Springer-Verlag, LNCS State-of-the-Arte Survey, Vol. 1734, Berlin, (1999), pp. 89-101.
- [11] W. Karl, M. Leberecht, M. Schulz. Optimizing data locality for SCI-based PC-clusters with the SMiLE monitoring approach. In *PACT '99*, Newport Beach, USA, October 1999
- [12] M. Martonosi, D. Ofelt and M. Heinrich. Integrating Performance Monitoring and Communication in Parallel Computers. In *SIGMETRICS'96 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computers Systems*, 1996.
- [13] M. Zagha, B. Larson, S. Turner, and M. Itzkowitz. Performance Analysis Using the MIPS R10000 Performance Counters. In *Supercomputing SC'96*, 1996.
- [14] C. Liao, M. Martonosi, and D. W. Clark. Performance Monitoring in a Myrinet Connected SHRIMP Cluster. In *Proceedings of SIGMETRICS Symposium on Parallel and Distributed Tools*. ACM, 1998.
- [15] M. Manzke, B. Coghlan. Non-intrusive deep tracing of SCI interconnect traffic. In *Proceedings of SCI Europe '99*, 1999.
- [16] Dolphin Interconnect Solutions AS. *Link Controller LC-1 Specification*, 1995.
- [17] W. Karl, M. Schulz, J. Trinitis, Multilayer Online Monitoring for Hybrid-DSM systems on top of PC clusters with a SMiLE., In *Proceedings of the 11th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, volume 1786 of *Lecture Notes in Computer Science*, Springer Verlag, 2000, Berlin.
- [18] W. Karl and M. Schulz. Hybrid-DSM: An Efficient Alternative to Pure Software DSM Systems on NUMA Architectures, In *Proceedings of the 2nd International Workshop on Software DSM*, May 1995.
- [19] S. Woo, M. Ohara, E. Torrie, J. Singh, A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24-36, June 1995.
- [20] PLX Technology., 625 Clyde Avenue, Mountain View, CA. *PCI9060 PCI Master Bus Interface Chip for Adapters and Embedded Systems*, Apr. 1995. Data Sheet.
- [21] Dolphin Interconnect Solutions AS. *A Backside Link (B-Link) for Scalable Coherent Interface (SCI) Nodes*, 1996.
- [22] N. Boden, D. Cohen, R. Felderman, J. Seizovic, A. Kulawik, C. Seitz, and Wen-King Su. Myrinet: A Gigabit-per-Second Local Area Network, *IEEE Micro*, 15(1): 29-36, February 1995.