SEVENTH FRAMEWORK PROGRAMME

FP7-ICT-2013-10

DEEP-ER

DEEP Extended Reach

Grant Agreement Number: 610476

D3.4

Network Attached Memory (NAM)

Approved

Version: 2.0

Author(s): J. Schmidt (UHEI)

Contributor(s): U. Bruening (UHEI)

Date: 08.06.2016

Project and Deliverable Information Sheet

<table>
<tr><td rowspan="7">DEEP-ER Project</td><td colspan="2">Project Ref. №: 610476</td></tr>
<tr><td colspan="2">Project Title: DEEP Extended Reach</td></tr>
<tr><td colspan="2">Project Web Site: http://www.deep-er.eu</td></tr>
<tr><td colspan="2">Deliverable ID: D3.4</td></tr>
<tr><td colspan="2">Deliverable Nature: Report</td></tr>
<tr><td>Deliverable Level:
PU*</td><td>Contractual Date of Delivery:
31 / March / 2016</td></tr>
<tr><td>Actual Date of Delivery:
31 / March / 2016
(Updated with further results 17.05.2016)</td></tr>
<tr><td colspan="2">EC Project Officer: Panagiotis Tsarchopoulos</td></tr>
</table>

* - The dissemination level are indicated as follows: **PU** – Public, **PP** – Restricted to other participants (including the Commission Services), **RE** – Restricted to a group specified by the consortium (including the Commission Services). **CO** – Confidential, only for members of the consortium (including the Commission Services).

Document Control Sheet

<table>
<tr><td rowspan="5">Document</td><td colspan="2">Title: Network Attached Memory</td></tr>
<tr><td colspan="2">ID: D3.4</td></tr>
<tr><td>Version: 2.0</td><td>Status: Approved</td></tr>
<tr><td colspan="2">Available at: http://www.deep-er.eu</td></tr>
<tr><td colspan="2">Software Tool: Microsoft Word</td></tr>
<tr><td colspan="3">File(s):DEEP-ER_D3.4_Network_Attached_Memory_v2.0-ECapproved</td></tr>
<tr><td rowspan="4">Authorship</td><td>Written by:</td><td>J. Schmidt (UHEI)</td></tr>
<tr><td>Contributors:</td><td>U. Bruening (UHEI)</td></tr>
<tr><td>Reviewed by:</td><td>T. Wettig (UREG), E.Suarez (JUELICH)</td></tr>
<tr><td>Approved by:</td><td>BoP/PMT</td></tr>
</table>

Document Status Sheet

Version	Date	Status	Comments
1.0	31/March/2016	Final version	EC submission
1.1	17/May/2016	Update	Updated before review M33 to reflect additional results obtained after its initial submission.
2.0	08/June/2016	Approved	EC approved

Document Keywords

Keywords:	DEEP-ER, HPC, Exascale, Network Attached Memory, Hybrid Memory Cube, active memory

Table of Contents

List of Figures

List of Tables

Executive Summary

This document presents the current status of the Network Attached Memory (NAM). The NAM is a dedicated component that is directly connected to the EXTOLL network within the DEEP-ER system. It can either serve as a (shared) storage or application-specific compute node. For both, the NAM promises to lower the communication overhead on the network.

The development of the NAM encompasses the design of a hardware prototype board, FPGA logic that provides read/write capabilities and specialized functions, in particular targeting resiliency, executable from remote processes. The NAM with many of the desired functions is already operational and has been installed into the Software Development Vehicle (SDV) in Jülich. The SDV will later become the Cluster part of the final DEEP-ER Prototype, under development in WP8.

First application performance measurements have already been performed. The results show a remote process to NAM write bandwidth of 3.5 GByte/s and 3 GByte/s for reads respectively. As the DEEP-ER Prototype will integrate EXTOLL devices with twice the current bandwidth it is also expected that the NAM read/write performance will increase accordingly.

Preliminary estimations show that using the NAM will be beneficial for checkpoint/restart applications which target resiliency as one of the designated goals of the DEEP-ER project.

1 Introduction

The Network Attached Memory (NAM) addresses two main objectives of the DEEP-ER project: resiliency and I/O performance. It is a single card (PCB) that is directly connected to the EXTOLL interconnection network and provides access to shared memory for both, Cluster and Booster nodes in the DEEP-ER Prototype (Figure 1). The NAM does not replace any of the existing nodes. It complements the DEEP-ER system as an additional device with selected processing capabilities.

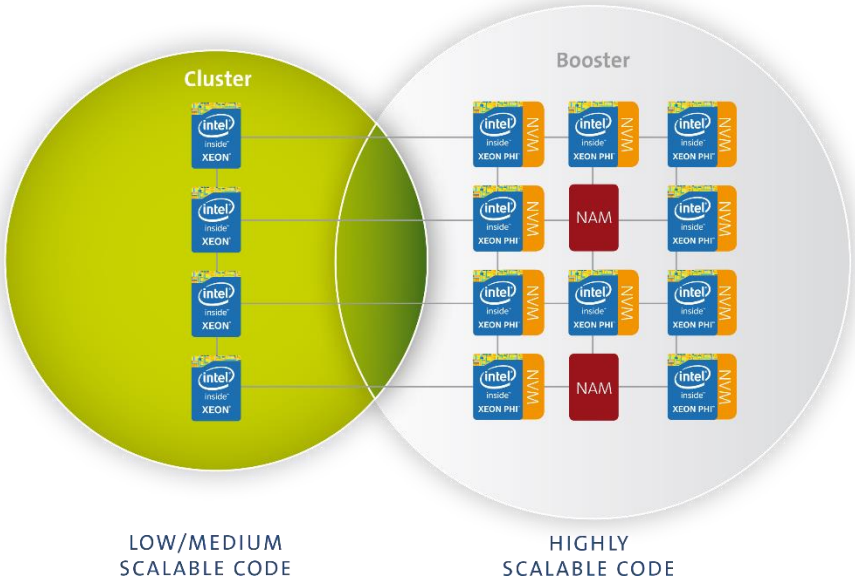


Figure 1: NAM in the DEEP-ER system

The NAM card basically consists of an FPGA and a Hybrid Memory Cube as the memory. It provides interfaces to connect EXTOLL, PCI-Express gen3 x8, and additional Hybrid Memory Cubes to increase the memory capacity. Details of the hardware are presented in section 3.

Use cases for the NAM range from general-purpose data processing, global operations execution, or shared memory to a checkpoint-restart (C/R) device. As an example, the C/R device is capable to calculate and store on-the-fly checkpoint-parity to decrease performance degradation in case of restart event. Interestingly, due to the nature of FPGAs, a NAM can be reconfigured, even at runtime, to adapt any of the mentioned use-cases.

The NAM can be easily integrated into the DEEP-ER system since it can be attached to any EXTOLL link in the system. In addition, existing software stacks require only minor modifications since communication with the NAM is based on the existing EXTOLL RMA engine and software, which is already supported by the DEEP software stack (the basis of the extensions done in DEEP-ER).

Performance-wise, the NAM provides a memory bandwidth of 40 GByte/s and 6 GByte/s on the EXTOLL link. In the future the bandwidths could increase to 50 GByte/s on the memory and 12 GByte/s on the EXTOLL link respectively. The memory capacity is currently limited to 2 GByte but the NAM also preserves the ability to attach additional HMCs and therefore increase the capacity. Evaluation of this feature is planned later on, once the currently planned NAM functionality has been completed.

2 Integration of the NAM with the DEEP-ER prototype

The NAM is directly connected to an EXTOLL NIC. It therefore remains in the EXTOLL network domain and allows data to be moved and processed without the need to enter a different processing node which would include additional delay through PCI-Express.

The figures below shall illustrate how the NAM is connected to an Extoll NIC (Figure 2) and how it can be integrated in the Extoll network (Figure 3).

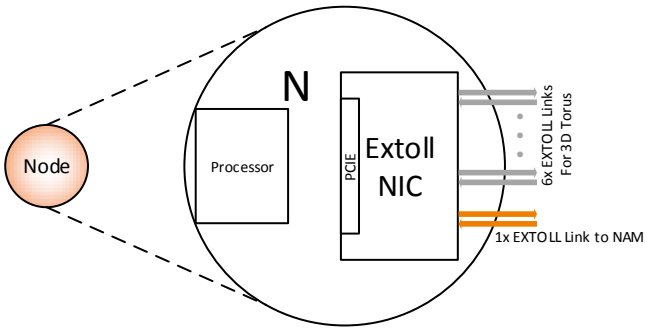


Figure 2: Schematic of a Node consisting of a processor and an EXTOLL NIC

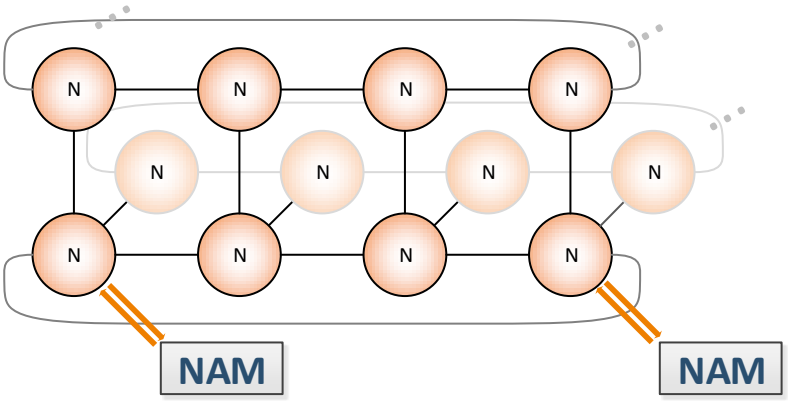


Figure 3: Illustration of an EXTOLL 3D Torus Network including two NAMs

3 NAM Hardware

UHEI has developed the NAM prototype board ‘Aspin v2’ as presented in Figure 4. The actual development process can be separated into two main work items: development and bring-up of the hardware, and the implementation of the NAM logic in the FPGA.

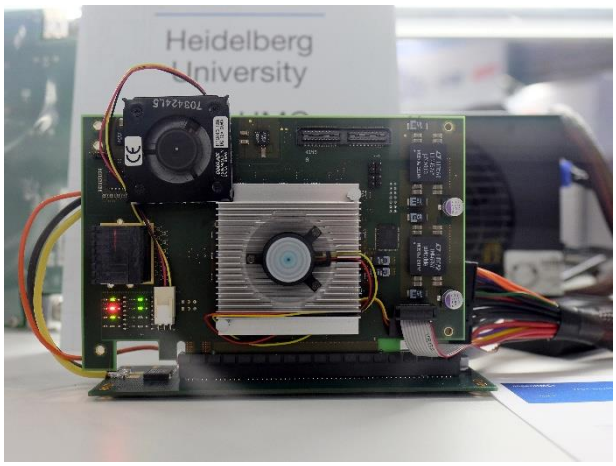


Figure 4: NAM Prototype Board Aspin v2

3.1 Hardware Development

The design of the NAM prototype board was finalized in September 2015. The first boards for assembly were available in October 2015. After several assembly issues caused by the fine-pitch ballout scheme of the FPGA the NAM was eventually operational early November. First results were already presented at the European Exascale Projects booth and the Emerging Technologies Track at the Supercomputing Conference (SC15) in Texas (USA), in November 2015. The presentation demonstrated the capability of the NAM to read and write to the HMC with up to 50 GByte/s.

The NAM prototype board provides two main interfaces for accessing the FPGA as shown in Figure 5:

- An HDI-6 connector for two 12x Extoll links.
- A 16x PCI-Express connector that can be used as either a proprietary, serial interface or to integrate the NAM as a regular PCI-Express device.

In addition, a surface mounted connector on the top of the card provides the ability to increase the memory capacity by adding (‘chaining’) additional HMCs to the local one.

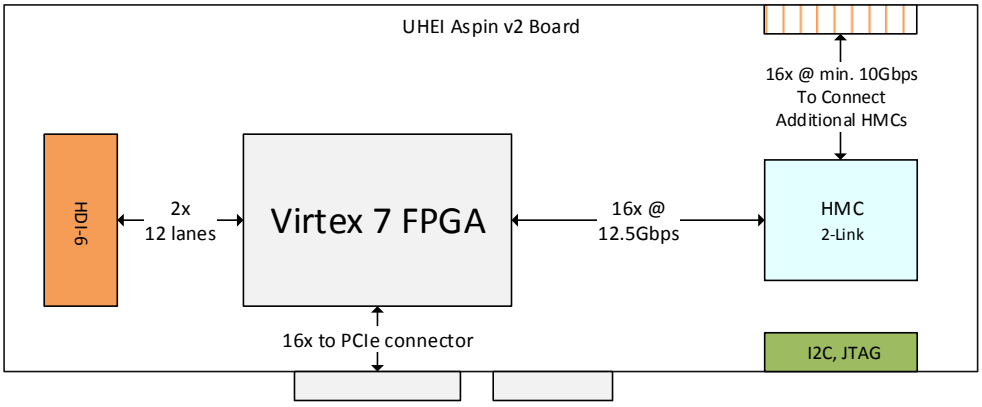


Figure 5: Aspin v2 Block Diagram

3.2 NAM Logic Development

The NAM logic is implemented in the FPGA and consists of three modules as shown in Figure 6: An HMC host controller to connect the Hybrid Memory Cube, an implementation of an Extoll Link suitable for FPGAs, and the actual NAM ‘intelligence’. These modules will be described in the following.

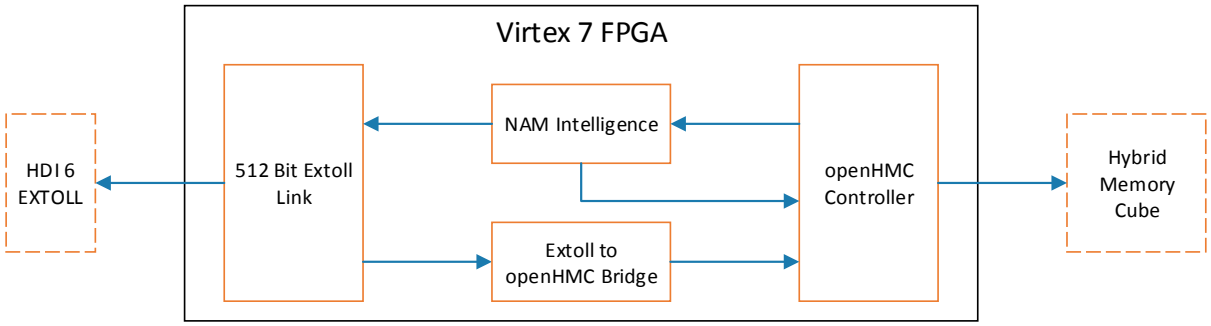


Figure 6: NAM logic block diagram

3.2.1 HMC Controller

UHEI has developed a fully functional and specification compliant HMC host controller. The controller has been released as an open-source core ‘openHMC’ to speed-up adoption of the HMC interface. It is meant to lower the barrier for others to experiment with the HMC, without the need to obtain expensive commercial solutions. For more information refer to [1] and [2].

The status of the openHMC module is 100% operational, verified, and mapped to the target FPGA technology. openHMC and its role in the DEEP-ER project were a topic at several conferences at the European Exascale booth in the past. Annex B provides a selection of a flyer and poster that were presented at the SC15 conference in Austin, Texas.

3.2.2 Extoll Link

In order to connect the NAM to the Extoll network, an Extoll link implementation derived from the original ASIC link had to be developed. In particular this is necessary as FPGAs cannot compete with ASICs with regard to the internal clocking frequency. This disparity requires FPGA implementations to use wider datapaths internally so that the actual throughput matches in both implementations. In addition, a functional unit that is able to communicate with the Extoll RMA engine was developed. Bring-up of the link in hardware was successfully performed. Host to HMC read and write capability is verified.

The status of this module is 100% verified in hardware.

3.2.3 NAM Intelligence

The ‚NAM intelligence‘ implements the actual NAM functionality. Besides application specific functions such as the checkpoint/restart engine or a Global Operation processor it must also provide a conversion from Extoll network packets to HMC packets and vice versa. Remote to HMC write capability has been simulated and successfully tested in-system connected to an Extoll Tourmalet.

Worth to mention is one property of the design and of FPGAs in general: the ability to partially reconfigure logic, i.e. a C/R engine can be replaced with another module while running in a system.

The current implementation of the NAM intelligence focuses on checkpoint/restart, 50% of which has already been simulated.

4 NAM Software

The developed software components where mainly derived from the existing Extoll RMA API which is used to grant read and write access to the NAM. Therefore, a new library called ‘libNAM’ was developed, operating as an additional layer on top of the libRMA. It uses slightly modified function calls provided by the libRMA. It furthermore offers extensions to manipulate the RMA software descriptor in a manner that the NAM is able to perform more sophisticated functions such as the checkpoint/restart mechanism. In addition, several logical and mathematical operations can be executed by the NAM. Annex A lists all currently available function calls.

A dedicated ‘NAM manager’ (NM) is a central instance within the HPC Cluster for managing memory of the connected NAMs. It tracks the status of any NAM regarding memory usage and network status. It also handles allocation requests and forwards them to the NAM returning valid allocations in case of success. Processes within a job will be able to communicate with NM directly via the API.

Listing 1 provides an example of a small application reading and writing to the NAM. It allocates address space from the NAM a writes a variable to it. After reading the variable from the NAM the allocated space is freed up.

```
int main(int argc, char **argv)
{
        nam_allocation_t *my_alloc;
        char hello[] = "Hello World!";
        char transferred[13];
        my_alloc =  nam_malloc(sizeof(hello));
        nam_put(hello, 0, sizeof(hello), my_alloc);
        nam_get(transferred, 0, sizeof(transferred), my_alloc);
        printf("Transferred from NAM: <%s>\n", transferred);
        nam_free(my_alloc);
        return 0;
}
```

Listing 1: libNAM code example

The status of the NAM software with regard to read/write from/to the HMC on the NAM is 90% implemented and simulated, yet to be fully verified in the DEEP-ER SDV.

About 50% of the NAM software for checkpoint/restart capability has been already implemented.

5 Co-design discussions

Close collaboration took place between WP3 and the application developers from WP6 to determine the functionality needed from the NAM. Discussions were triggered earlier in the project but until now it was hard for the application developers to give precise answers, due to the very new nature of the NAM concept and the at that time limited practical information about it. In October 2015, when the NAM design was mostly ready, WP3 contacted WP6 to discuss what functionalities the API for the NAM should provide. Afterwards, the application developers made a list of functionalities in the NAM API that their application would most likely benefit from. These functionalities include e.g. reduction or simple matrix operations.

WP3 and WP6 eventually agreed on a checkpoint/restart mechanism as first NAM use-case. This mechanism makes use of global reductions and fits well into the resiliency aspect of the DEEP-ER project. The decision is also based on the ability to re-use a great portion of the existing software stack, mitigating the risk of a delayed component in the project. In addition, the implementation of more complex functions in the FPGA hardware would have led to less time for hardware development and verification. A fully verified hardware base, however, is a mandatory element for any novel hardware component such as the NAM is.

6 Status of the Deliverable

The current status of the individual components, hardware and software development, and logic implementation have been assessed in the previous sections.

Assembly issues and difficulties in debugging the high-speed serial links of both links, HMC and Extoll, have delayed the overall progress of the component. At the time of writing, however, these issues have been overcome and the NAM has been integrated into the DEEP-ER SDV.

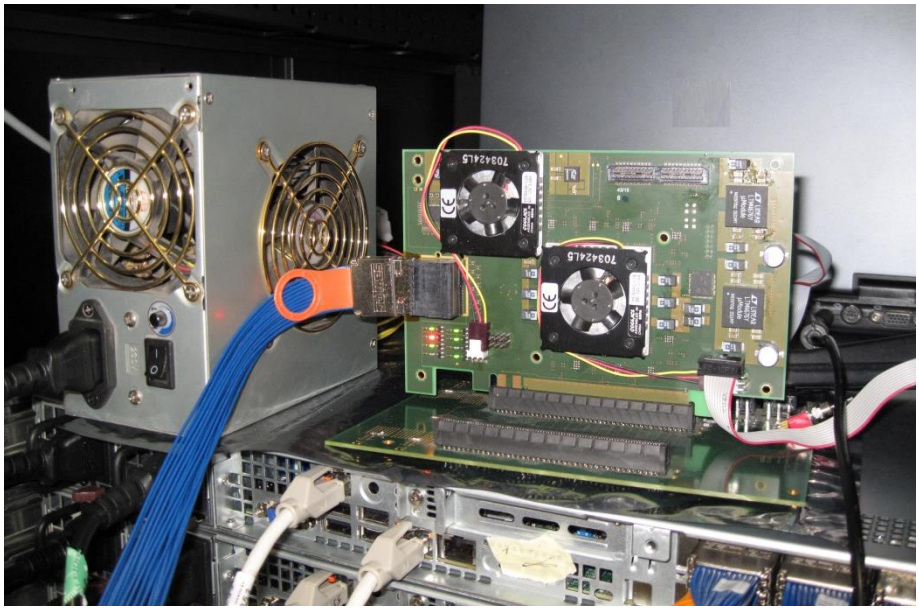


Figure 7: NAM connected to the SDV at Jülich

First performance assessments have already started and show a remote process to NAM write bandwidth of 3.5 GByte/s and read bandwidth of 3 GByte/s (both effective). The EXTOLL devices in the SDV, however, run only at 50% of their desired link bandwidth which

is currently 4 Gbit/s per lane. The final DEEP-ER prototype is expected to integrate EXTOLL NICs with 8 Gbit/s per lane. Consequently, the NAM read/write bandwidth will also double. In addition, practical experience and observations will allow to modify the individual parts of the NAM logic to deliver even better performance.

Operations on the NAM itself that only involve the HMC are not subject to the EXTOLL link bandwidth. These operations will be able to utilize the full HMC link bandwidth of 40 GByte/s.

For checkpoint and restart it is expected that the NAM will speed up checkpoint (re-)storing by reducing the amount of network communication required to generate a parity checkpoint. Furthermore a dedicated, hardware-enabled device to perform the required logical operations will free-up participating processors from these tasks.

7 Summary and Outlook

The biggest hurdle in developing the NAM component was the design and bring-up of the NAM prototype board which is now considered to be final. The design and implementation of the actual NAM logic in the FPGA showed good progress. Read and write capability from a remote process to the NAM was already tested and verified in hardware.

The decision for a checkpoint/restart mechanism as the first application for NAM fits well into the DEEP-ER project, where resiliency optimization is one of the top goals. It is expected that a NAM-based checkpoint/restart scheme will speed up writing and restoring of application checkpoints. Network traffic and read/write latencies will be reduced due to application-specific hardware functions of the ‘NAM intelligence’.

The capacity of the Hybrid Memory Cube device on the NAM is currently limited to 2 GByte but should not affect software development progress and exploration of this novel architecture. The NAM furthermore provides the option to increase the capacity by attaching additional HMCs to a dedicated connector. The NAM component in its current setup, however, should be completed first before the exploration of increasing the capacity takes place. In addition, it is expected that the capacity of single HMC will grow to 4 or 8 GByte in the near future.

8 Bibliography

[1] H. University, "openHMC home," [Online]. Available: http://ra.ziti.uni-heidelberg.de/cag/research/recent-research-projects/openhmc. [Accessed 28 2 2016].

[2] J. Schmidt und U. Bruening, „openHMC - a configurable open-source hybrid memory cube controller,“ *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig),* pp. 1-6, 7 12 2015.

Annex A: libNAM Function Calls

A.1 Basic Function Calls

Table 1: libNAM basic functions

nam_allocation_t *nam_malloc(size_t size)	size: request bytes return: new NAM allocation
nam_allocation_t *nam_realloc (nam_allocation_t *alloc, size_t size)	alloc: existing allocation size: bytes of re-allocation return: new NAM allocation
nam_allocation_t *nam_calloc (size_t nmemb, size_t size)	nmemb: number of elements size: size of one element return new NAM allocation initialised with 0
int nam_free(nam_allocation_t *alloc)	alloc: allocation to free return: if the allocation could be freed
ssize_t nam_put(void *buf, size_t offset, size_t bytes, nam_allocation_t *alloc)	buf: pointer to data to be put offset: offset within allocation bytes: bytes of data in buf alloc: allocation where to put return: number of bytes put on the NAM
ssize_t nam_get(void *buf, size_t offset, size_t bytes, nam_allocation_t *alloc)	buf: pointer to data to be get offset: offset within allocation bytes: bytes of data to get alloc: allocation to get from return: number of bytes got from the NAM

A.2 Advanced Function Calls

Table 2: libNAM advanced functions

int nam_vector_op(size_t first_offset, size_t second_offset, size_t result_offset, size_t n, void *result, nam_datatype type, nam_operation op, nam_result mode, nam_allocation_t *alloc);	first_offset: start address of first operand second_offset: start address of second operand result_offset : start address to store result n: number of elements to process result: return value if requested type: INTEGER/DOUBLE/FLOAT/... op: ADD/SUB/MULT mode: combination of NAM_RETURN/NAM_STORE

	alloc: NAM allocation return: 0 if operation was successful, -1 if not + errno
int nam_reduce_op(size_t offset, size_t result_offset, size_t n, void *result, nam_datatype type, nam_operation op, void *pattern, nam_result mode, nam_allocation_t *alloc);	offset: start address of range result_offset: start address to store result n: number of elements to process result: return value if requested type: INTEGER/DOUBLE/FLOAT/... pattern: to search for when op=BS/AS op: MIN/MAX/SUM/PRODUCT/AND/OR/XOR/BS/AS mode: combination of NAM_RETURN/NAM_STORE alloc: NAM allocation return: 0 if operation was successful, -1 if not + errno

Annex B: openHMC Flyer and Poster

Use Case DEEP-ER project: Applications increasingly suffer from diverging growth rates of the computational capabilities of the processors versus the memory and network bandwidth. In view of Exascale the integration management of yet another storage class memory serving as a cache-hierarchy for the storage subsystems becomes mandatory. Network Attached Memory (NAM) is one promising technology the European exascale project DEEP-ER experiments with in respect to parallel I/O and checkpoint / restart.

Figure: NAM in the DEEP-ER Cluster-Booster Architecture

	Parallel I/O	Checkpoint / Restart
Applications	– Directly store configuration data and metadata – Shared I/O buffers – Staging areas – Distribute or concentrate data	– Serve as targets to hold checkpoint data
Benefits	– Asynchronous I/O – Reduction of disk traffic – Reduction of network congestion	– Reduce time taken to write a checkpoint – Reduces time taken for restart – Exploiting locality reduces load of network

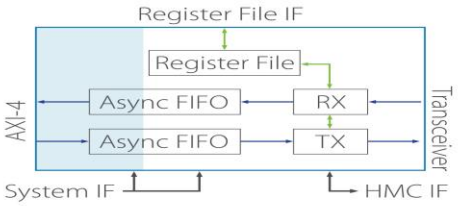
For more info on HMC and NAM in the DEEP-ER project, please visit
www.deep-er.eu

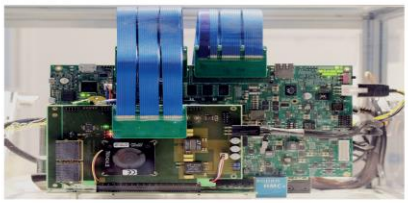
»openHMC«
contact: openhmc@ziti.uni-heidelberg.de

»openHMC«

An Open-Source Hybrid Memory Cube Controller

»openHMC« Controller





Features

» Link-training, sleep mode and link retraining
» All types of packets and sizes
» Full packet flow control
» Packet integrity checks
» Full link retry
» Configurable by a set of parameters

Available user interfaces (8x or 16x HMC Link)

» 256 bit
» 512 bit
» 768 bit
» 1024 bit

» Free & FPGA Proven
» Test Environment included

Use Case DEEP-ER

In the EU-funded DEEP-ER project, openHMC is used in the Network Attached Memory (NAM).
The NAM introduces an additional level of memory hierarchy with processing capabilities – directly connected to the EXTOLL high-performance network.

For more information on the NAM and DEEP-ER visit www.deep-er.eu

Licensing

Contact us for specific solutions and licensing opportunities

Evaluation Hardware Available

Xilinx Virtex 7 + Micron HMC on a PCIe form factor card

Website | Contact Email | LinkedIn

»openHMC« in cooperation with EXTOLL latency matters. and Micron Foundation

List of Acronyms and Abbreviations

A

API: Application Programming Interface

ASIC: Application Specific Integrated Circuit, Integrated circuit customised for a particular use

E

EXTOLL: High speed interconnect technology for cluster computers developed by University of Heidelberg

F

FPGA: Field-Programmable Gate Array, Integrated circuit to be configured by the customer or designer after manufacturing

H

HMC: Hybrid Memory Cube

Hybrid Memory Cube: Novel type of computer RAM that uses 3D packaging of multiple memory dies to increase memory capacity and number of data banks per device area. Technology is being developed by Micron Technology and backed by the Hybrid Memory Cube Consortium.

N

NAM: Network Attached Memory, nodes connected by the DEEP-ER network to the DEEP-ER BN and CN providing shared memory buffers/caches, one of the extensions to the DEEP Architecture proposed by DEEP-ER

NIC: Network Interface Card, Hardware component that connects a computer to a computer network

S

SC: International Conference for High Performance Computing, Networking, Storage, and Analysis, organised in the USA by the Association for Computing Machinery (ACM) and the IEEE Computer Society

SDV: Software Development Vehicle: a HW system to develop software in the time frame where the DEEP-ER Prototype is not yet available.

U

UHEI: University of Heidelberg, Germany

UREG: University of Regensburg, Germany