# Efficient Distributed Memory Management in a Multi-Core H.264 Decoder on FPGA

Jiajie Zhang[1], Zheng Yu[1], Zhiyi Yu[1], Kexin Zhang[2], Zhonghai Lu[3], Axel Jantsch[23]
[1]State Key Lab of ASIC and System, Fudan University, Shanghai, China
[2]Memcom.Soc Microelectronics co. ltd, Wuxi, China
[3]KTH-Royal Institute of Technology, Kista, Sweden
zhiyiyu@fudan.edu.cn {zhonghai,axel}@kth.se

*Abstract*—**Memory management is a challenging issue of multi-core architecture. With growing core numbers, Distributed Shared Memory (DSM) is becoming a general trend. In this paper, a DSM based multi-core architecture is explored and evaluated via an H.264 decoder application. The memory access and communication over Network-on-Chips is managed by the Data Management Engine (DME). Experimental results realized on an Altera Stratix VI show that 9-node distributed memory system increases performance by 1.5x compared to centralized memory. Moreover, the performance of proposed DSM architecture grows linearly with the number of cores deployed.**

***Keywords-Multi-core; DSM; H.264 decoder; DME; FPGA***

## I. INTRODUCTION

Multi-core platforms have demonstrated excellent performance for many applications. To enable multi-core platforms, scalable interconnect fabrics, such as Network-on-Chips (NoC), have been proposed over the last ten years. Serial buses are considered to be replaced by the parallel communication networks since buses do not scale well with the core numbers. On the other hand, the memory management represents another key issue in multi-core architecture. The traditional solution is a single shared memory connected with a bus, which scales badly with respect to performance, cost and power consumption. Distributed shared memory (DSM) is a promising approach which overcomes the limitations of centralized memory while retaining shared address space for programming and communication.

Cache coherency, memory consistency and synchronization are critical parts of distributed shared memory design. These memory and data management can be handled by a programmable controller called Data Management Engine (DME) [1]. The DME is a micro-programmable IP aiming at ASIC-like performance while maintaining the flexibility of SW. It hosts two mini-processors to concurrently queue and handle requests from the local core and remote cores via the communication network. The mini-processors also allow users to implement various functions such as virtual-to-physical address translation, memory access, and synchronization realized using microcode.

In this paper, a DSM based multi-core NoC architecture is explored. Each node hosts a DME connecting the core, the local memory and the network. Basic DSM functions are realized and an H.264 video decoder is mapping on the multi-core NoC to evaluate its performance. By partitioning and parallelizing the H.264 decoder program, the multi-core platform can achieve higher performance compared to a single core. The DSM scheme supports high memory bandwidth while the DME can implement efficient synchronization mechanisms. We implement the multi-core H.264 decoder on an Altera Stratix IV FPGA and a comparison of distributed memory with centralized memory system is presented.

The rest of the paper is organized as follows. Section II provides an overview of related work. Section III presents the FPGA demonstrator design and setup, including the hardware multi-core architecture and software H.264 decoder program. Section IV reports experimental results. Finally, section V draws conclusions.

## II. RELATED WORK

Distributed shared memory has attracted attention since the late 1970's. In [2], Monchiero explored a distributed shared memory architecture and focused on the energy/delay exploration. In [3], Hennessy reviewed the key developments that led to the creation of cache-coherent distributed shared memory and described the first implementation of hardware-supported scalable cache coherence. In [4], Chennareddy specified the weak consistency properties and verified the distributed shared memory weak consistency models. In [5], Singh proposed the notion of invariant consistency that allowed specification of inter-process synchronization constraints. Invariant consistency simplifies programming as it eliminates application-level synchronization code to enforce inter-process constraints. DME was first introduced to support flexible distributed shared memory management on multi-core NoC in [1]. It off-loads DSM management from the main-processor and has been used for a range of different applications and in different ways [6,7,8].

Regarding H.264 decoder mapping on multi-core, in [9], Nishihara proposed two parallelization methods to balance the

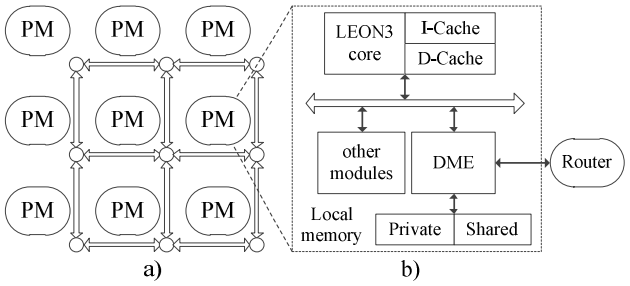


Figure 1. a) A mesh multi-core NoC, b) Processor-Memory node.

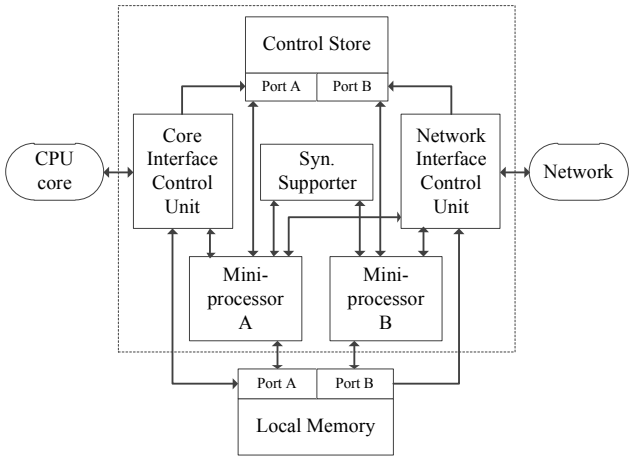


Figure 2. Architecture of the DME.

load across multiple cores and reduce memory access contention. In [10], Finchelstein proposed several techniques that enable multiple parallel decoders to process a single video sequence and also demonstrated several on-chip caching schemes.

In contrast to previous work we use a simple and scalable distributed shared memory architecture for a H.264 decoder and demonstrate its scalability when more cores are added. The DME simplifies the memory management and shields the application designer from the details of the memory and NoC architecture. Thus, the main contribution of this paper is a DME supported memory architecture that is scalable and simplifies application design.

## III. Demonstrator Design and Setup

The FPGA demonstrator is composed of a hardware multi-core NoC architecture and an H.264 decoder application mapping on it. The multi-core NoC platform is based on distributed memory while a comparing platform based on centralized memory is also presented in this section.

### *A. DSM Based Multi-core NoC Architecture*

Fig. 1 a) shows an example of the DSM based multi-core NoC platform. The system is composed of Processor-Memory

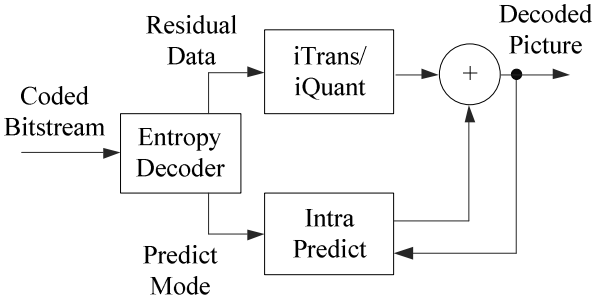


Figure 3. H.264 intra decoder diagram.

(PM) nodes interconnected via a packet-switched network. The network has a mesh topology and its size is configurable. Priority-based round-robin arbitration mechanism is used and it guarantees fairness (no starvation) among requests and allows any unused time slot to be allocated to a request whose round-robin turn is later but who is ready now. Each PM node has a LEON3 processor (with I-Cache and D-Cache), a DME plus a local memory. The AHB bus connects to the LEON3 and the DME, which connects to the local memory. As illustrated in Fig. 1 b), the local memory is partitioned into private and shared. The private memory speeds up frequent private access while all the shared memories maintain a single virtual space visible to all nodes.

The global memory access and communication over NoC is managed by the DME. As shown in Fig. 2, the DME contains six parts, Core Interface Control Unit (CICU), Network Interface Control Unit (NICU), Control Store, Mini-processor A, Mini-processor B, and Synchronization Supporter. As their names suggest, the CICU provides a hardware interface to the local core, and the NICU a hardware interface to the network. The Control Store dynamically uploads the microcode and feeds it to the two mini-processors during the program execution. The Synchronization Supporter coordinates the two mini-processors to avoid simultaneous accesses to the same memory address and guarantees atomic read-and-modify operations. The two mini-processors are the central processing engine which features a five-stage pipeline and four function units: Load/Store Unit, Adder Unit, Condition Unit and Message Passing Unit. The four function units can work simultaneously and independently under control of horizontal microinstruction. The instruction sets are identical and include general purpose instructions as well as a number of specialized instructions for support of synchronization, message generation and address manipulations. These features of the DME provide efficient and transparent communication for the programmers.

### *B. H.264 Intra Decoder*

H.264 standard facilitates high compression rate and requires large amounts of computation. The H.264 intra decoder functional diagram is shown in Fig. 3. Entropy decoder (DEC) parses the coded video bit stream, generating

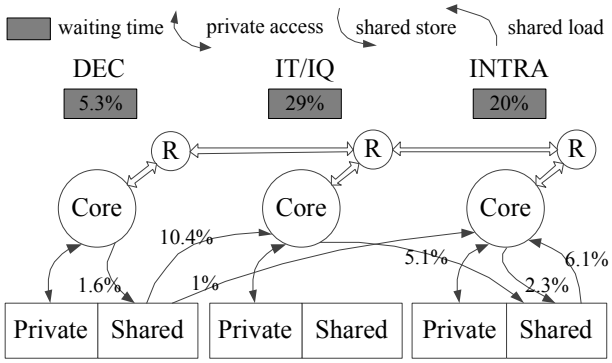


Figure 4. H.264 decoder communication diagram.

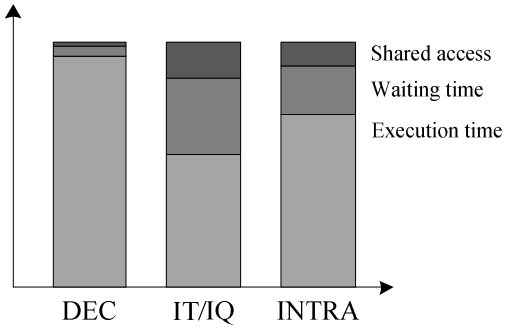


Figure 5. H.264 decoder timing profile.

transformed residual data for iTrans/iQuant (IT/IQ) and predictive mode for Intra Prediction (INTRA). IT/IQ demonstrates inverts-transform to restore residual, adding predictive pixels from INTRA to finally reconstruct the original frames.

Multi-core architectures emerge as good solutions to tackle with this complex media application by task partitioning and parallelizing. Based on above multi-core NoC structure, we firstly implement a 3-core H.264 intra decoder, using shared memory communication for data movement. Fig. 4 shows inter-core communication overhead of the H.264 decoder, mainly including shared memory access time. For example, DEC stores transformed residual data and predictive mode in local shared memory occupying 1.6% of the overall application processing time. Synchronization and data communication among these three nodes are supported by DME, which implements efficient distributed memory management in the multi-core H.264 decoder. According to the portion of waiting time, we allocate communication buffers on local shared memory of DEC and INTRA node respectively. A double-buffer scheme is also applied to the shared memory of each node to provide parallel data exchange between local and remote cores. For example, while a remote core is reading data from buffer one through network interface of DME, the local core can write new data to buffer two through core interface of

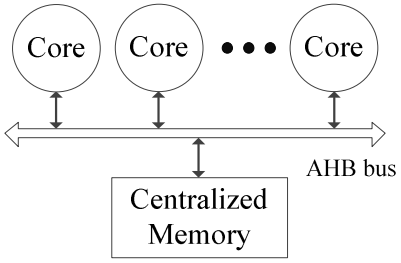


Figure 6. Multi-core platform with centralized memory.

DME, which implements parallelization of shared memory transactions. As depicted in Fig. 5, workloads of these three function nodes are well balanced.

### C. *Comparing Platforms*

In order to compare with distributed memory architecture, we build a multi-core platform with a single shared memory, interconnected with an AHB bus, as shown in Fig. 6. It is called symmetric multiprocessors (SMP), since all processors have an equal relationship with the centralized memory. The centralized memory architecture supports the traditional programming model, has lower communication costs since the cores communicate through bus rather than through NoC. However, memory access latency of a huge centralized main memory is much larger than small distributed memories maintaining the same address space together. At the same time, bus conflicts become so severe and result in lower efficiency with more cores integrated and frequent memory access.

We also map H.264 decoder on more cores to compare the performance of distributed memory to centralized memory based platforms, as Fig. 7 shows. The multi-core platform is partitioned into clusters each containing a 3-node subsystem. It needs to add a Task Distributor in hardware to distribute decoding tasks frame by frame to the clusters. In summary, we setup a demonstrator with frame-level parallel H.264 decoders mapping on distributed memory and centralized memory multi-core platforms.

## IV. EXPERIMENT AND RESULTS

We implement above multi-core platforms on an Altera Stratix IV FPGA, running 3-node, 6-node and 9-node H.264 decoder programs to process QCIF (176x144 pixels) and CIF (352x288 pixels) pictures. The 3-node distributed memory platform consumes 91K logic cells and takes 19M clock cycles to decode one QCIF frame. Fig. 8 depicts comparison between centralized memory and distributed memory multi- core platforms. Results show a sharp degradation in performance of the centralized memory platform while the performance of the distributed memory platform grows perfectly with the number of cores. The 3-node H.264 decoder based on distributed memory achieves throughput of 25fps for QCIF and 6fps for CIF when the FPGA runs at 125MHz, nearly the same as the

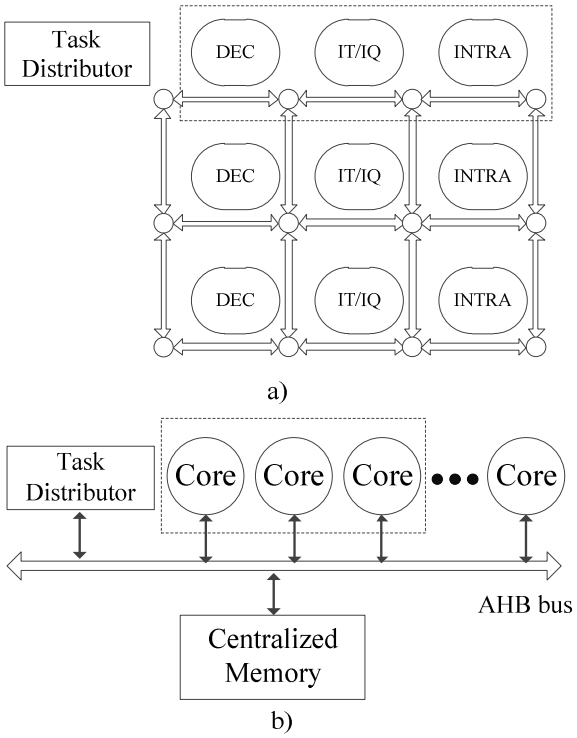


Figure 7. H.264 decoder 9-node implementation on a) distributed memory platform, and b) centralized memeory platform.

centralized memory platform. The Task Distributor and efficient distributed memory enable the 6-node and 9-node H.264 decoder to achieve almost 1 and 2 times performance improvement over the 3-node solution while that of the bus-based centralized memory saturates quickly.

## V. CONCLUSION

An FPGA demonstrator of a multi-core H.264 decoder supporting distributed memory has been presented. Efficient distributed memory management is realized by the DME, which supports synchronization, message generation, address manipulations and parallelization of shared memory transactions, etc. The results are compared to a centralized memory platform. With H.264 decoder mapping on more cores, the performance of distributed memory platform scales linearly.

## ACKNOWLEDGMENT

This work was supported in part by the grant from The State Key Laboratory of ASIC & Systems (Fudan University) Senior Visiting Scholarship.

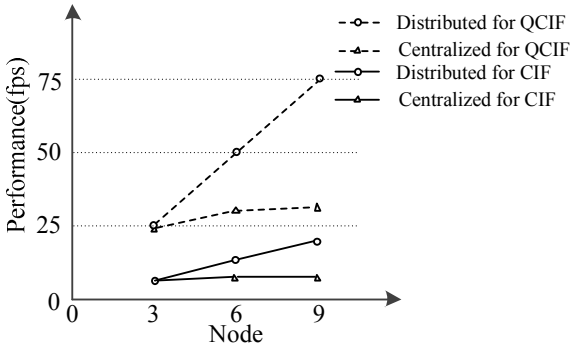


Figure 8. Comparison of distributed memory with centralized memory platforms.

## REFERENCES

[1] X. Chen, Z. Lu, A. Jantsch and S. Chen, "Supporting distributed shared memory on multi-core Network-on-Chips using a dual microcoded controller," Design, Automation & Test in Europe Conference & Exhibition, pp. 39-44, March 2010.

[2] M. Monchiero, G. Palermo, C. Silvano and O. Villa, "Exploration of distributed shared memory architectures for NoC-based multiprocessors," International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, pp. 144-151, July 2006.

[3] J. Hennessy, M. Heinrich and A. Gupta, "Cache-coherent distributed shared memory: perspectives on its development and future challenge," Proceeding of the IEEE, vol. 87, no. 3, pp. 418-429, September 1997.

[4] V. Chennareddy and J. K. Deka, "Formally verifying the distributed shared memory weak consistency models," International Conference on Advanced Computing and Communications, pp. 455-460, December 2006.

[5] G. Singh, "Specifying synchronization in distributed shared memory programs," International Conference on Parallel Processing Workshops, pp. 375-382, August 2002.

[6] X. Chen, Z. Lu, A. Jantsch, S. Chen and H. Liu, "Cooperative communication based barrier synchronization in on-chip mesh architectures," IEICE Electronics Express, vol. 8, no. 22, pp. 1856-1862, 2011.

[7] X. Chen, Z. Lu, A. Jantsch and S. Chen, "Reducing virtual-to-physical address translation overhead in distributed shared memory based multi-core Network-on-Chips according to data property," Computers and Electrical Engineering, May 2012.

[8] A. Jantsch, X. Chen, A. Naeem, Y. Zhang, S. Penolazzi and Z. Lu, "Memory architecture and management in an NoC platform," In A. Jantsch and D. Soudris, editors, Scalable Multi-core Architectures: Design Methodologies and Tools, Springer, 2011.

[9] K. Nishihara, A. Hatabu and T. Moriyoshi, "Parallelization of H.264 video decoder for embedded multicore processor," IEEE International Conference on Multimedia and Expo, pp. 329-332, April 2008.

[10] D. F. Finchelstein and V. Sze, "Multicore processing and efficient on-chip caching for H.264 and future video decoders," IEEE Transactions on Circuits and Systems for Video Technology, vol. 19, no. 11, pp. 1704-1713, 2009.