



HYBRID VS MEMORY-TO-MEMORY COMMUNICATION IN MULTI-CORE PROCESSOR

S. ARUNA MASTANI^{#1}, K. KUSUMA^{*2}

Assistant professor, Dept. Of ECE, JNTUACEA, Anantapur

PG student (DSCE), Dept. Of ECE, JNTUACEA, Anantapur

Email: ¹aruna_mastani@yahoo.com, ²kusuma.mohith.9@gmail.com

Abstract: Now a day's multi-core architecture introduces new challenges for effective implementation of inter-core communication, as Inter-core communication plays an important role to balance the delay in a multi-core processor. The two mechanisms used for inter-core communication are shared-memory and message-passing communications. Shared-memory communication fails to provide sufficient scalability with the increasing number of processor, where as message-passing communication though have high scalability, but it doesn't have guaranteed Quality-of-service (QoS). To overcome the above drawbacks a combined mechanism named Hybrid inter-core communication [1] is prevalent to till date. Recently we proposed a new technique named Memory-to-Memory communication which provides the direct memory to memory communication by using DMA as memory interface [2]. This paper mainly concentrates on comparing our proposed method with the existing methods till date, with respect to delay. All the inter-core communication system mechanisms have been designed in 90nm CMOS using XILINX 12.2 version platform. Comparing the performance of all communication systems it is observed that the communication time is least for Memory-to-Memory communication.

Key words—Chip multiprocessor, Direct Memory Access, hybrid inter-core communication, inter-core communication, shared-memory, multi-core, message-passing,

network-on-chip (NoC), inter-core synchronization, Memory to Memory.

I. INTRODUCTION

In order to meet performance requirements single core designs were pushed to higher clock speeds, thereby the power requirement grew at a faster rate than the frequency. This power problem was exacerbated by designs that attempted to dynamically extract extra performance from the instruction stream, but it will be noted later that this led to designs that were complex, unmanageable and power hungry. To meet these requirements chip designers turned to multi-core processors. A multi-core processor is one which consists of multiple no. of processors on a single chip. All these processors work in parallel thereby the overall performance of the multi-core processor increases. To meet power budget many efforts are taken to optimise memory hierarchy and to increase parallelism concurrently.

When certain problem is given to an embedded processor the throughput depends on both computing capability and communication efficiency between cores. To enhance computing capability there are various technologies such as Very long instruction word (VLIW), Single instruction multiple data (SIMD), Super scalar, Reduced Instruction set computer (RISC) etc. But there are no matured solutions for inter-core communication, Hence the research focus on improving the efficiency of inter-core communication.

Shared-memory communication is most often used inter-core communication mechanism due to its simple programming model but it fails to

provide sufficient scalability with the increasing number of processors ([4]-[6]). Therefore the designers turned to message-passing communication mechanism which has high scalability even with the increase in number of resources ([7]-[11]). From key features of shared-memory and message-passing communication it is clear that different inter-core communication mechanisms are suitable for different scenarios. By integrating both the inter-core communication mechanisms high throughput and power efficiency can be obtained ([1]). So, a hybrid communication with both shared-memory and message-passing is effective compared to individual mechanisms. Recently a new mechanism called Memory-to-Memory communication is developed for inter-core communication. This paper implements both Hybrid and Memory-to-Memory communication on a multi-core processor and compares the performance parameters of both mechanisms.

This system mainly concentrates on implementation of effective inter-core communication mechanism rather than the processor type so, a simple processor with a router inside it is designed rather than complex processors. All the three inter-core communication mechanisms are implemented in a generalised 16-core processor which is connected in 3×6 2D Mesh NoC form that links 16 core processors (PCores) and 2 memory cores (MCores). Cluster-based architecture is employed with two clusters where each cluster contains eight PCores and one MCore. The PCores present in the cluster can able to access the MCore present in the same cluster ([3]) directly through hard wires.

This paper is organised as below. Section II describes the designing and implementation of existing methods. Section III describes the implementing work. Section IV presents the measured results. Section V concludes the paper. Section VI describes the future work.

II. EXISTING METHODS

Inter-core communication is defined as the communication between multiple no. of processors integrated on a single chip. Power and cost budgets limits high computability processors to be integrated on a chip thereby the overall performance of multi-core processor relies highly on inter-core communication. Thus the throughput is highly relevant to inter-core

communication. Till now only two types of inter-core communication mechanisms exist for an embedded processor. The first one is Shared-memory communication which is implemented by making use of a shared cache or memory units. Typical examples are cortexA9, UltraSPARC, HYDRA etc. The features of shared-memory communication are simple programming, used for transferring of large blocks of data. It faces several challenges which limits its use in future processors. First its low scalability, more than 8 cores are not allowed to share a single memory. In an 8 core processor the interconnections take area equivalent to 3cores and consume power equivalent to one core. Second cache coherence issues are very complex which results in more hardware overhead.

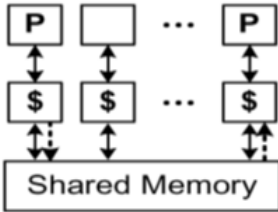
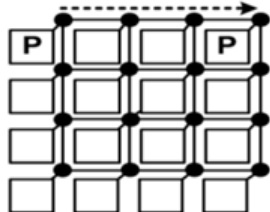
Because of its high scalability the second type of communication i.e., Message-passing communication attracts many designers. It is implemented by connecting the processors in a Network in certain topology like mesh, bus, ring etc. Typical examples are ASAP, Intel-80 tile etc. In spite of its strong scalability it has complex programming model, and the quality of service (QoS) is not guaranteed. Table-1 shows the features of shared-memory communication and message-passing communication.

1) Shared-Memory Communication:

The processors which are present in the same cluster can access the MCore with fixed priority order ([4]-[6]). The max no. of PCores in a cluster is limited to eight. The processor on the top left corner has highest priority and the PCore on the Bottom right corner has lowest priority. High inter-core synchronization efficiency has been achieved through hardware-aided mailbox mechanism. Shared-Memory communication involves mainly three steps, First the source PCore stores the data in to shared memory, next it sends a synchronization signal to the destination PCore, Finally the Destination PCore access the data from shared memory after the synchronization signal is received. Fig-1 shows the steps for shared-memory communication. And Fig-2 shows implementation of shared-memory communication within a cluster having 8PCores and 1MCore.

TABLE-1

Comparison of Shared-Memory and Message-Passing inter-core communications

Method	Shared Memory	Message passing
Usage	Large, unsplit data block	Frequent, scattered data
Pro	Simple programming	Better scalability
Con	Lower scalability	Uncertain channel
Medium	Shared cache/memory	Network-on-chip
Scenario	Computational data flow	Control data flow
Graphics		

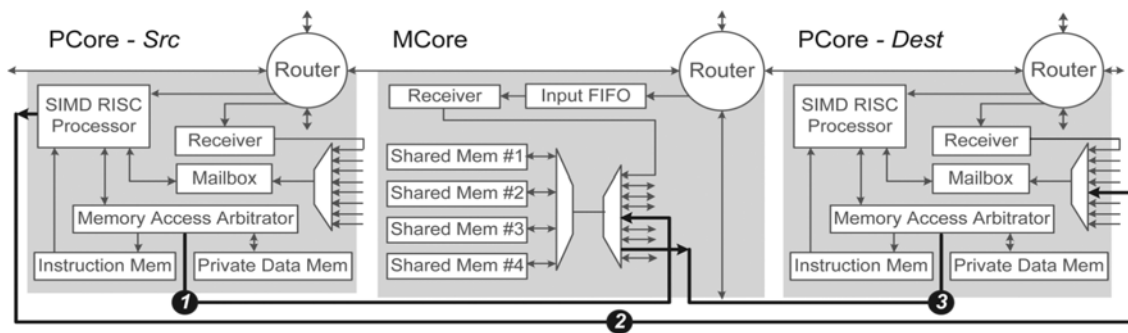


Fig-1. Three steps in a typical shared-memory communication: (1) Src PCore stores data to shared memory in MCore; (2) Src PCore sends synchronization signal to Dest PCore; (3) Dest PCore loads data from shared memory when synchronization signal is received.

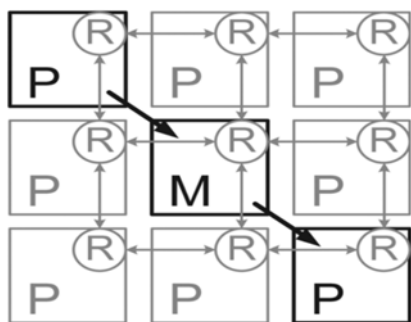


Fig-2. Shared-memory communication within a cluster.

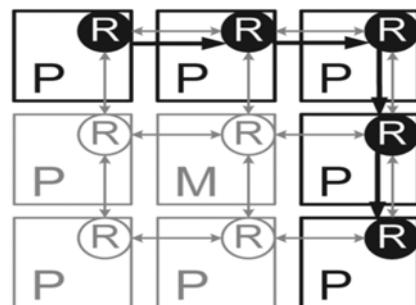


Fig-3. Message-passing communication within a cluster.

2) Message-Passing Communication:

Mesh NoC supports Message-Passing communication where an XY dimension ordered wormhole routing algorithm is implemented. Even the shared-memory communication is implemented only within the cluster, The Message-passing communication is implemented

between any two processors in the chip (i.e., with in the cluster or outside the cluster). It is more scalable and is mainly used for transferring of frequent and scattered data.

A part from its advantages it has 2 bottlenecks. The first one is the uncertainty in the communication channel. The network with

heavy traffic will block the data packets in the channel hence the latency gets increased. The second bottleneck lies in the data transferring between the processor and router. Fig-3 shows the data path in message-passing communication inside a cluster having 8 PCores and 1MCore.

III. IMPLEMENTING WORK

Initially Shared-memory communication is most often used mechanism, but it fails to provide sufficient scalability with the increasing no. of processors in a cluster. Therefore message-passing communication came in to existence with high scalability but the QoS is not guaranteed here. With the integration of both communication mechanisms the performance increases. So, a Hybrid inter-core communication is developed. Among the two based up on the type of data the system is processing, a particular communication scheme is decided. With the increasing number of processors the complexity of hybrid inter-core communication increases. In order to avoid these disadvantages a Memory-to-Memory communication is introduced which can directly transfer the data between all memory's of the clusters present in the processor through a memory interface. This paper implements both the Hybrid and Memory-to-Memory communications on a general 16 core processor and compares their performances.

A. Hybrid inter-core communication:

Shared-memory communication is low scalable with increasing number of processors and it is extremely complex

which results in high power consumption. Whereas the Message-passing communication is highly scalable but it suffers from 2bottlenecks. They are contention of data packets and how to differentiate data between processor and a router. To overcome the drawbacks of individual inter-core communication mechanisms a Hybrid communication is developed. On which the low

scalability of Shared-Memory scheme and contention of packets in message-passing scheme is solved because of existence of message-passing and shared-memory communication on the same multi-core processor.

And the second bottleneck is removed by using 2 input FIFO's in PCore on which the data coming from another PCore is stored in 1st FIFO and the data coming from MCore is stored in 2nd FIFO there by it can easily differentiate the data coming from router in to processor. To implement Hybrid inter-core communication in 16Core processor it needs to be arranged in a 2D manner which seems to be a 3×6 mesh Network-on-Chip supports message passing communication. A cluster based memory hierarchy is employed to support shared-memory communication i.e., Total 16Core processors and 2 memory processors are arranged in 2 clusters where each cluster consists of 8PCores and 1MCore. Shared-Memory communication is possible only within the cluster.

Fig-3 shows the architectural overview of hybrid inter-core communication in 16Core processor. The first processor is considered to be the source processor and the sixteenth processor is considered to be the destination processor. Here the path in which the data from source PCore to destination PCore is transferred is shown in Fig-3. First the data from source processor is transferred to MCore1 present in the same cluster then it is transferred to PCore3 which is present in the same cluster up to this shared memory communication is implemented to transfer the data. Next the data from PCore3 is moved to PCore9 present in second cluster using message passing communication and from PCore9 it is stored in MCore2 and finally the data present in the memory is loaded by the destination processor PCore16. Here the path that is considered is worst case path. Compared to individual shared-memory and Message-passing communications Hybrid communication is very effective and results in high performance.

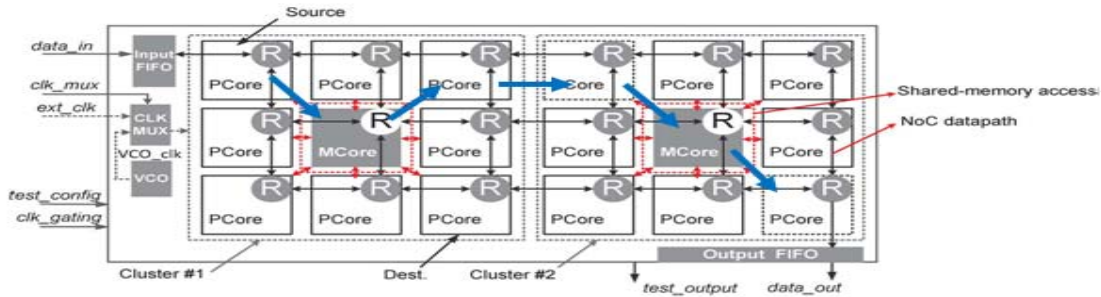


Fig-3. Architectural overview of Hybrid inter-core communication in 16Core processor.

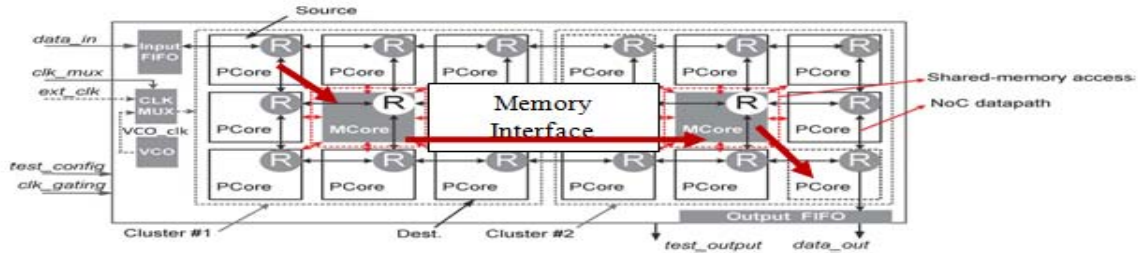


Fig-4. Architecture overview of Memory-to-Memory communication in 16core processor.

B. Memory-to-Memory communication:

Even though Hybrid communication is very effective for providing the inter-core communication in multi-core processors it has a disadvantage of very complex to maintain with increasing number of processors and clusters. To avoid this new technique called Memory-to-Memory communication is implemented [2]. A Memory-to-Memory communication is one which directly transfers the data between the memory cores (MCores) of clusters with in a multi-core processor through a memory interface. Direct Memory Access (DMA) can be used as a memory interface. DMA is one of several methods for coordinating the timing of data transfers between an input/output (I/O) device and the processing unit or memory in a computer. DMA is one of the faster types of synchronization mechanism which provides improvement in terms of both latency and throughput. DMA allows the I/O device to access the memory directly, without using the core.

The memory-to-memory communication between clusters in 16Core processor is implemented simply in five steps first the data is loaded in to the source processor (PCore 1) through an input FIFO. Second the data from source processor is loaded directly in to memory core present in the same cluster. Next by using a memory interface the data is loaded in to the interfacing component and this data is transferred

from interfacing component to memory core present in the second

cluster. Next the data from second MCore is directly loaded by destination Processor core (PCore 16). Lastly the data from destination processor is collected through an output FIFO. Fig-4 shows the architecture overview of Memory-to-Memory communication in 16Core processor. It is observed that among the two communication mechanisms the communication path is greatly reduced from cluster to cluster in case of Inter-memory communication i.e., The path from source to the destination requires only 3 intermediate components hence it is very efficient to use memory to memory communication in multi-core processors compared to shared-memory, message-passing and Hybrid inter-core communications.

IV. RESULTS AND DISCUSSION

In VLSI the performance parameters are Area, Delay and power. For any system performance is expressed in these three parameters only. Inter-core communication mainly concentrates on reduction of delay from source to destination processor, so this system mainly focuses on delay parameters of communication mechanisms rather than area and power. The simulation results are obtained by testing the communication mechanisms on a 16 core processor

1) Delay:

In multi-core processors the throughput mainly depends on computing capability and communication efficiency between cores. This paper mainly concentrates on inter-core communication rather than computing capability so the throughputs of two communication mechanisms are compared. Fig-5 shows the simulation results of Hybrid inter-core communication in which the input data is given as 0F0F0F0F to the input FIFO and it is collected at the output FIFO after 370ns. If the same input is allowed to pass using shared-memory and message-passing communication mechanisms then it takes 610 and 470ns respectively. So compared to Shared-memory and Message-passing, Hybrid inter-core communication is more effective. But compared to Hybrid communication Memory-to-Memory communication is more efficient. Fig-6 shows the simulation results of Memory-to-Memory communication in which the input data is given as 0F0F0F0F to the input FIFO and it is collected at the output FIFO after 330ns. These two communication delays are represented with yellow lines in their simulation results. Hence

from these results it is observed that the Memory-to-Memory communication is very much effective and results in high performance compared to shared-memory and message-passing communications.

2) Area and power:

The proposed cache-free architecture can significantly reduce chip area. Moreover, embedded applications usually require limited memory resources, so only 256 KB on-chip memory units are implemented. The no of LUT slices required to implement Hybrid communication and memory-to-memory communication are represented in Table-2. Table-2 shows the comparison of performance parameters of Hybrid and Memory-to-Memory communication. Two key features contribute to the low power consumption. First, cache is discarded in the proposed cluster-based memory hierarchy therefore related hardware overhead is also reduced. Second the data locality is improved by using extended register file and with separation of private and shared memory.

TABLE-2
Comparison of Hybrid and Memory-to-Memory communication

Parameter	Shared-Memory Communication	Message-Passing communication	Hybrid inter-core communication	Memory-to-Memory communication
No. of sliced Registers	770	868	681	573
No. of sliced LUT's	1279	1363	1015	922
No. of fully used LUT's	607	644	475	445
No. of bonded IOB's	73	73	73	73
Transmission Delay	610ns	470ns	370ns	330ns

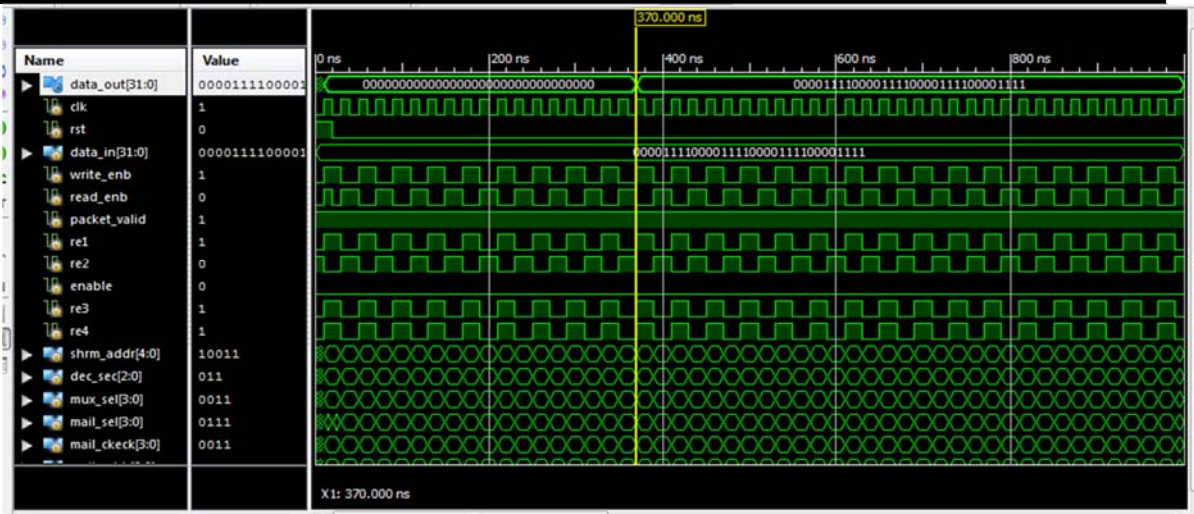


Fig-5. Simulation results of Hybrid inter-core communication.

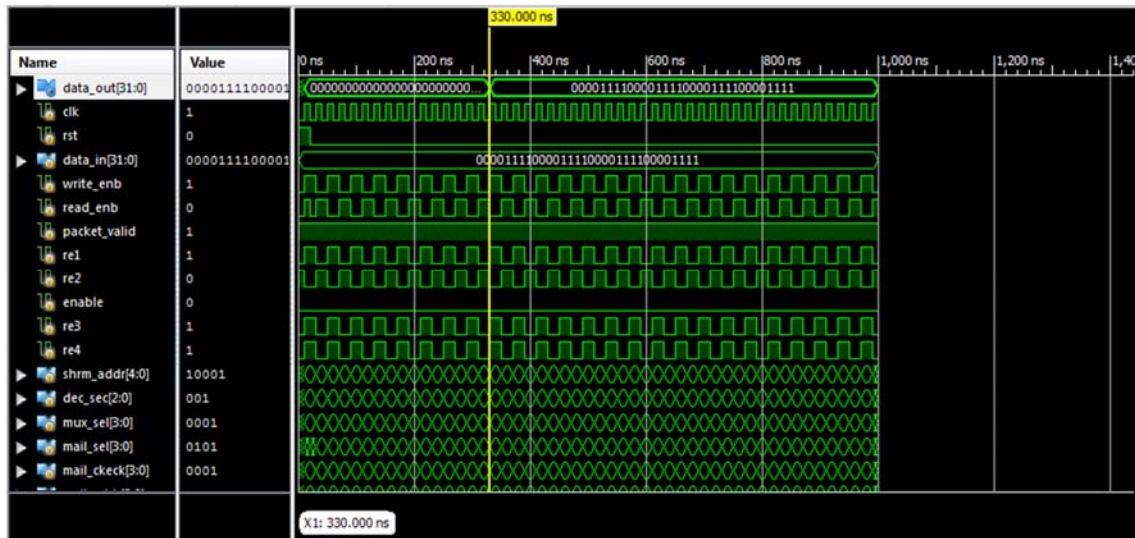


Fig-6. Simulation results of memory-to-memory communication.

V. CONCLUSION

The drawbacks in shared-memory and message-passing communication mechanisms are overcome by using Hybrid inter-core communication. This paper implements the Hybrid inter-core communication and Memory-to-Memory communication in a generalised 16 core processor which has 16 processor cores and 2 memory cores which are arranged in a 3x6 2D mesh NoC. Both these communication mechanisms are compared with each other in all performance parameters. A Hybrid inter-core communication is implemented by a combination of shared-memory and message-passing communication. A memory-to-memory

communication is implemented using a memory interface called DMA. The DMA used here is a prototype one which performs only specific functions. Compared to Shared-Memory and Message-Passing communications, Hybrid inter-core communication produces high through put but compared to Hybrid inter-core communication the newly proposed Memory-to-Memory communication results in high performance. The system is implemented in 90nm CMOS using XILINX 12.2 version software.

VI. FUTURE WORK

The performance of the processor gets still increased by using pipelined and highly computable processors

REFERENCES

- [1] Zhiyi Yu, and Ruijin Xiao , "A 16-Core Processor With Shared-Memory and Message-Passing Communications." *IEEE Trans. circuit syst. vol.61, No.4, April 2014*.
- [2] S. Aruna Mastani and K. Kusuma, "A Novel Technique for inter-core communication in Multi-core processor" *International conference on advanced signal processing and communications, Narasaraopeta engg.college., pp.114-120, July 13th 2015*.
- [3] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore processors: A review of their common attributes," *IEEE Signal Process. Mag.*, pp. 26–37, Nov. 2009.
- [4] R. Kumar, V. Zyuban, and D. Tullsen, "Interconnections in multi-core architecture: Understanding mechanisms, overheads and scaling," in *Proc. 32nd Int. Symp. Computer Architecture (ISCA'05)*, 2005, pp. 408–419.
- [5] H.-Y. Kim, Y.-J. Kim, J.-H. Oh, and L.-S. Kim, "A reconfigurable SIMT processor for mobile ray tracing with contention reduction in shared memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, no. 60, pt. 4, pp. 938–950, Apr. 2013.
- [6] L. Hammond, B.-A. Hubbert, M. Siu, M.-K. Prabhu, M. Chen, and K. Olukolun, "The stanford Hydra CMP," *IEEE Micro*, vol. 20, no. 2, pp. 71–84, 2000.
- [7] A. S. Leon, B. Langley, and L. S. Jinuk, "The UltraSPARC T1 processor: CMT reliability," in *Proc. Custom Integrated Circuits Conf. (CICC'06) Dig. Tech. Papers*, 2006, pp. 555–562.
- [8] M.-B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrati, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Stumpfen, M. Frank, S. Amarasinghe, and A. Agarwal, "The Raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar/Apr. 2002.
- [9] Tiler Corp., Tilepro64 Processor Tiler Product Brief, 2008 [Online]. Available: http://www.tiler.com/pdf/Product-Brief_TILEPro64_Web_v2.pdf.
- [10] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-WteraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuit*, vol. 43, no. 1, pp. 29–41, Jan 2008.
- [11] Z. Yu, M. J. Meeuwsen, R. W. Apperson, O. Sattari, M. Lai, J. W. Webb, E. W. Work, D. Truong, T. Mohsenin, and B. M. Baas, "AsAP: An asynchronous array of simple processors," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 695–705, Mar 2008.