TRENDS IN MULTI-CORE DSP PLATFORMS

Lina J. Karam[*] Ismail AlKamal[*] Alan Gatherer[**] Gene A. Frantz[**] David V. Anderson[†] Brian L. Evans[‡]

[*] Electrical Engineering Dept., Arizona State University, Tempe, AZ 85287-5706, {ismail.alkamal, karam}@asu.edu
[**] Texas Instruments, 12500 TI Boulevard MS 8635, Dallas, TX 75243, {genf, gatherer}@ti.com
[†] School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA 30332-0250, dva@ece.gatech.edu
[‡] Electrical & Computer Engineering Dept., The University of Texas at Austin, Austin, TX 78712, bevans@ece.utexas.edu

1. INTRODUCTION

MULTI-CORE Digital Signal Processors (DSPs) have gained significant importance in recent years due to the emergence of data-intensive applications, such as video and high-speed Internet browsing on mobile devices, which demand increased computational performance but lower cost and power consumption. Multi-core platforms allow manufacturers to produce smaller boards while simplifying board layout and routing, lowering power consumption and cost, and maintaining programmability.

Embedded processing has been dealing with multi-core *on a board, or in a system,* for over a decade. Until recently, size limitations have kept the number of cores *per chip* to one, two, or four but, more recently, the shrink in feature size from new semiconductor processes has allowed single-chip DSPs to become multi-core with reasonable on-chip memory and I/O, while still keeping the die within the size range required for good yield. Power and yield constraints, as well as the need for large on-chip memory have further driven these multi-core DSPs to become systems-on-chip (SoCs). Beyond the power reduction, SoCs also lead to overall cost reduction because they simplify board design by minimizing the number of components required.

The move to multi-core systems in the embedded space is as much about integration of components to reduce cost and power as it is about the development of very high performance systems. While power limitations and the need for low-power devices may be obvious in mobile and hand-held devices, there are stringent constraints for non-battery powered systems as well. Cooling in such systems is generally restricted to forced air only, and there is a strong desire to avoid the mechanical liability of a fan if possible. This puts multi-core devices under a serious hotspot constraint. Although a fan cooled rack of boards may be able to dissipate hundreds of Watts (ATCA carrier card can dissipate up to 200W), the density of parts on the board will start to suffer when any individual chip power rises above roughly 10W. Hence, the cheapest solution at the board level is to restrict the power dissipation to around 10W per chip and then pack these chips densely on the board.

The introduction of multi-core DSP architectures presents several challenges in hardware architectures, memory organization and management, operating systems, platform software, compiler designs, and tooling for code development and debug. This article presents an overview of existing multi-core DSP architectures as well as programming models, software tools, emerging applications, challenges and future trends of multi-core DSPs.

2. HISTORICAL PRESPECTIVES: FROM SINGLE-CORE TO MULTI-CORE

The concept of a Digital Signal Processor came about in the middle of the 1970s. Its roots were nurtured in the soil of a growing number of university research centers creating a body of theory on how to solve real world problems using a digital computer. This research was academic in nature and was not considered practical as it required the use of state-of-the-art computers and was not possible to do in real time. It was a few years later that a Toy by the name of Speak N Spell™ was created using a single integrated circuit to synthesize speech. This device made two bold statements:
-Digital Signal Processing can be done in real time.
-Digital Signal Processors can be cost effective.
This began the era of the Digital Signal Processor. So, what made a Digital Signal Processor device different from other microprocessors? Simply put, *it was the DSP's attention to doing complex math while guaranteeing real-time processing.* Architectural details such as dual/multiple data buses, logic to prevent over/underflow, single cycle complex instructions, hardware multiplier, little or no capability to interrupt, and special instructions to handle signal processing constructs, gave the DSP its ability to do the required complex math in real time.

"If I can't do it with one DSP, why not use two of them?" That is the answer obtained from many customers after the introduction of DSPs with enough performance to change the designer's mind set from "how do I squeeze my algorithm into this device" to "guess what, when I divide the performance that I need to do this task by the performance of a DSP, the number is small." The first encounter with this was a year or so after TI introduced the TMS320C30 – the first floating-point DSP. It had significantly more performance than its fixed-point predecessors. TI took on the task of seeing what customers were doing with this new DSP that they weren't doing with previous ones. The significant finding was that none of the customers were using only one device in their system. They were using multiple DSPs working together to create their solutions.

As the performance of the DSPs increased, more sophisticated applications began to be handled in real time. So, it went from voice to audio to image to video processing. Fig. 1 depicts this evolution. The four lines in

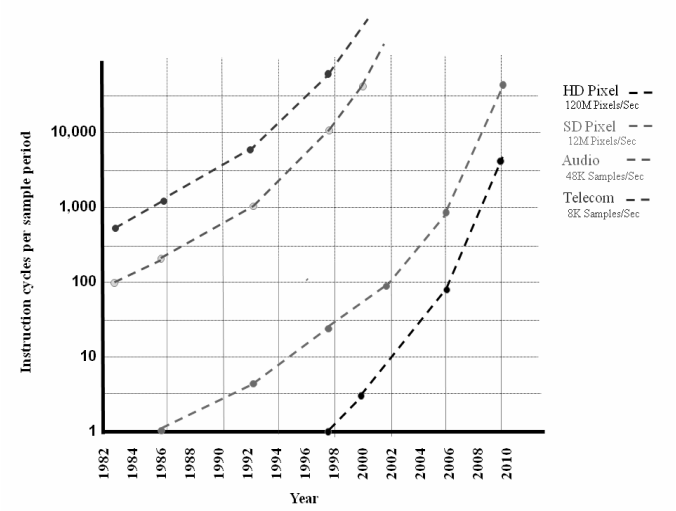

Fig. 1. Four examples of the increase of instruction cycles per sample period. It appears that the DSP becomes useful when it can perform a minimum of 100 instructions per sample period. Note that for a video system the pixel is used in place of a sample.

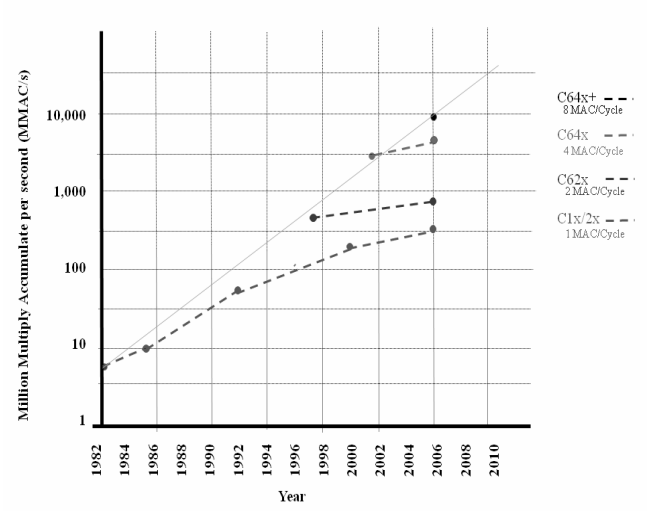

Fig. 2. Four generations of DSPs show how multi-processing has more effect on performance than clock rate. The dotted lines correspond to the increase in performance due to clock increases within an architecture. The solid line shows the increase due to both the clock increase and the parallel processing.

Fig. 1 represent the performance increases of Digital Signal Processors in terms of instruction cycles per sample period. For example, the sample rate for voice is 8 kHz. Initial DSPs allowed for about 625 instructions per sample period, barely enough for transcoding. As higher performance devices began to be available, more instruction cycles became available each sample period to do more sophisticated tasks. In the case of voice, algorithms such as noise cancellation, echo cancellation and voice band modems were able to be added as a result of the increased performance made available. Fig. 2 depicts how this increase in performance was more the result of multi-processing rather than higher performance single processing elements. Because Digital Signal Processing algorithms are Multiply-Accumulate (MAC) intensive, this chart shows how, by adding multipliers to the architecture, the performance followed an aggressive growth rate. Adding multiplier units is the simplest form of doing multiprocessing in a DSP device.

For TI, the obvious next step was to architect the next generation DSPs with the communications ports necessary to matrix multiple DSPs together in the same system. That device was created and introduced as the TMS320C40. And, as one might suspect, a follow up (fixed-point) device was created with multiple DSPs on one device under the management of a RISC processor, the TMS320C80.

The proliferation of computationally demanding applications drove the need to integrate multiple processing elements on the same piece of silicon. This lead to a whole new world of architectural options: homogeneous multi-processing, heterogeneous multi-processing, processors versus accelerators, programmable versus fixed function, a mix of general purpose processors and DSPs, or system in a package versus System on Chip integration. And then there is Amdahl's Law that must be introduced to the mix [1-2]. In addition, one needs to consider how the architecture differs for high performance applications versus long battery life portable applications.

3. ARCHITECTURES OF MULTI-CORE DSPs

In 2008, 68% of all shipped DSP processors were used in the wireless sector, especially in mobile handsets and base stations; so, naturally, development in wireless infrastructure and applications is the current driving force behind the evolution of DSP processors and their architectures [3]. The emergence of new applications such as mobile TV and high speed Internet browsing on mobile devices greatly increased the demand for more processing power while lowering cost and power consumption. Therefore, multi-core DSP architectures were established as a viable solution for high performance applications in packet telephony, 3G wireless infrastructure and WiMAX [4]. This shift to multi-core shows significant improvements in performance, power consumption and space requirements while lowering costs and clocking frequencies. Fig. 3 illustrates a typical multi-core DSP platform.

Current state-of-the-art multi-core DSP platforms can be defined by the type of cores available in the chip and include homogeneous and heterogeneous architectures. A homogeneous multi-core DSP architecture consists of cores that are from the same type, meaning that all cores in the die are DSP processors. In contrast, heterogeneous architectures contain different types of cores. This can be a collection of DSPs with general purpose processors (GPPs), graphics processing units (GPUs) or micro controller units (MCUs).

Another classification of multi-core DSP processors is by the type of interconnects between the cores.

More details on the types of interconnect being used in multi-core DSPs as well as the memory hierarchy of these multiple cores are presented below, followed by an overview of the latest multi-core chips. A brief discussion on performance analysis is also included.

3.1 Interconnect and Memory Organization

As shown in Fig. 4, multiple DSP cores can be connected together through a hierarchical or mesh topology. In hierarchical interconnected multi-core DSP platforms, data transfers between cores are performed through one or more switching units. In order to scale these architectures, a hierarchy of switches needs to be planned. CPUs that need to communicate with low latency and high bandwidth will be placed close together on a shared switch and will have low latency access to each others' memory. Switches will be connected together to allow more distant CPUs to communicate with longer latency. Communication is done by memory transfer between the memories associated with the CPUs. Memory can be shared between CPUs or be local to a CPU. The most prominent type of memory architecture makes use of Level 1 (L1) local memory dedicated to each core and Level 2 (L2) which can be dedicated or shared between the cores as well as Level 3 (L3) internal or external shared memory. If local, data is moved off that memory to another local memory using a non CPU block in charge of block memory transfers, usually called a DMA. The memory map of such a system can become quite complex and caches are often used to make the memory look "flat" to the programmer. L1, L2 and even L3 caches can be used to automatically move data around the memory hierarchy without explicit knowledge of this movement in the program. This simplifies and makes more portable the software written for such systems but comes at the price of uncertainty in the time a task needs to complete because of uncertainty in the number of cache misses [5].

In a mesh network [6-7], the DSP processors are organized in a 2D array of nodes. The nodes are connected through a network of buses and multiple simple switching units. The cores are locally connected with their "north", "south", "east" and "west" neighbors. Memory is generally local, though a single node might have a cache hierarchy. This architecture allows multi-core DSP processors to scale to large numbers without increasing the complexity of the buses or switching units. However, the programmer generally has to write code that is aware of the local nature of the CPU. Explicit message passing is often used to describe data movement.

Multi-core DSP platforms can also be categorized as Symmetric Multiprocessing (SMP) platforms and Asymmetric Multiprocessing (AMP) platforms. In an SMP platform, a given task can be assigned to any of the cores without affecting the performance in terms of latency. In an AMP platform, the placement of a task can affect the latency, giving an opportunity to optimize the performance by optimizing the placement of tasks. This optimization comes at the expense of an increased programming complexity since the programmer has to deal with both space (task assignment to multiple cores) and time (task scheduling). For example, the mesh network architecture of Fig. 4 is AMP since placing dependent tasks that need to heavily communicate in neighboring processors will significantly reduce the latency. In contrast, in a hierarchical interconnected architecture, in which the cores mostly communicate by means of a shared L2/L3 memory and have to cache data from the shared memory, the tasks can be assigned to any of the cores without significantly affecting the latency. SMP platforms are easy to program but can result in a much increased latency as compared to AMP platforms.

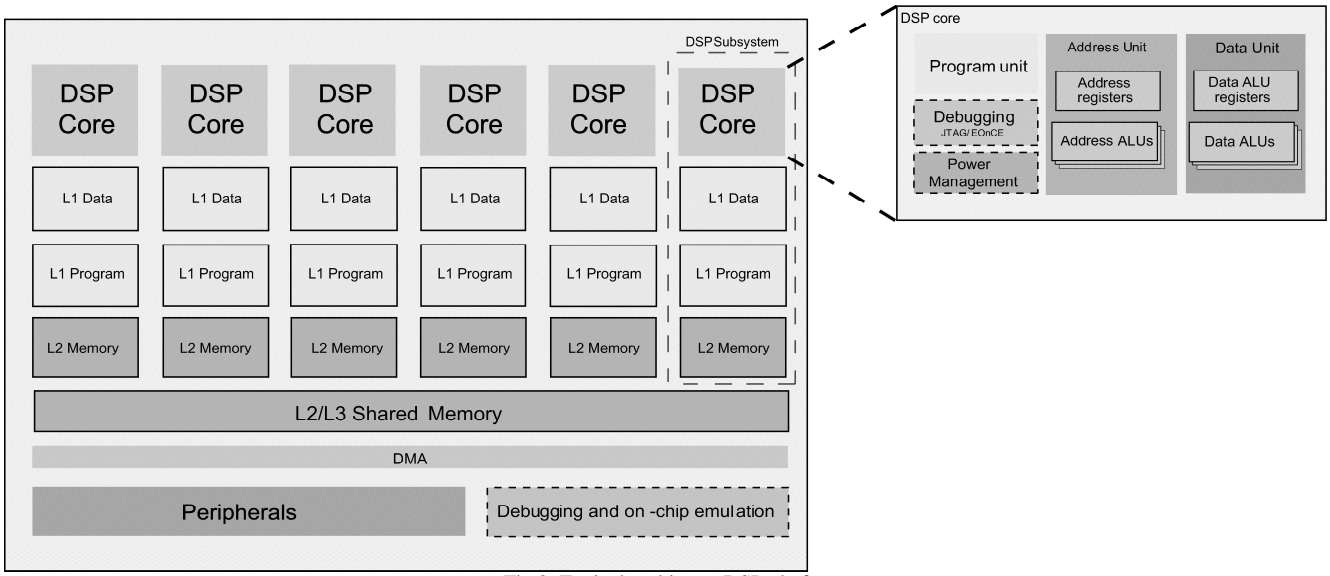

Fig.3. Typical multi-core DSP platform.

Table 1: Multi-core DSP platforms.

	TI [8]	*Freescale [9]*	*picoChip [10]*	*Tilera [11]*	*Sandbridge [12-13]*
Processor	TNETV3020	MSC8156	PC205	TILE64	SB3500
Architecture	Homogeneous	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous
No. of Cores	6 DSPs	6 DSPs	248 DSPs 1 GPP	64 DSPs	3 DSPs 1 GPP
Interconnect Topology	Hierarchical	Hierarchical	Mesh	Mesh	Hierarchical
Applications	Wireless Video VoIP	Wireless	Wireless	Wireless Networking Video	Wireless

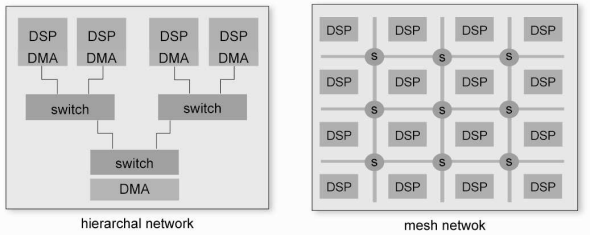

Fig.4. Interconnect types of multi-core DSP architectures.

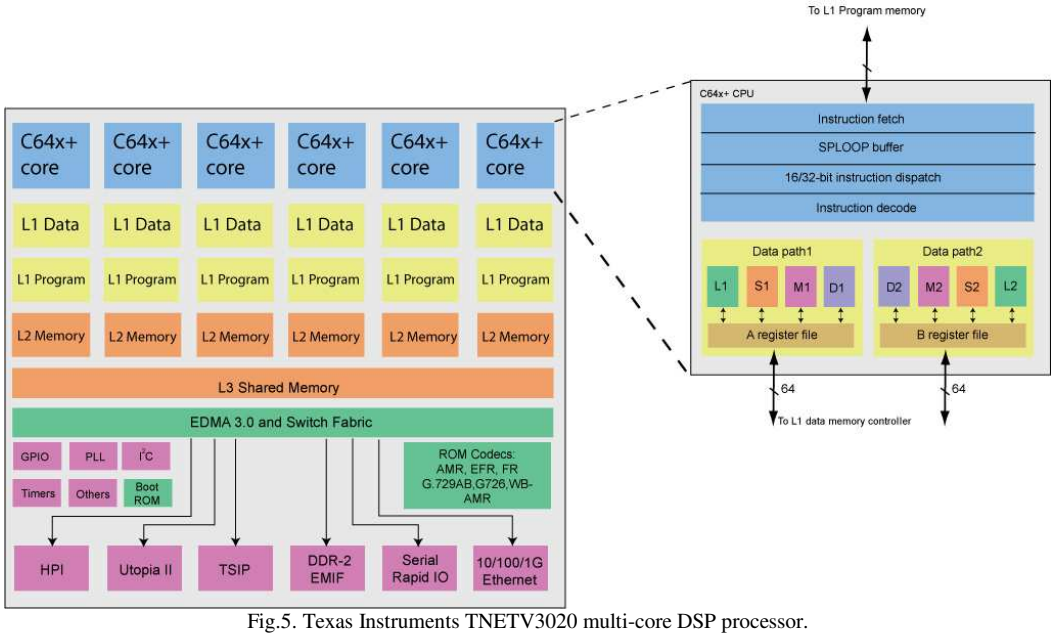

Fig.5. Texas Instruments TNETV3020 multi-core DSP processor.

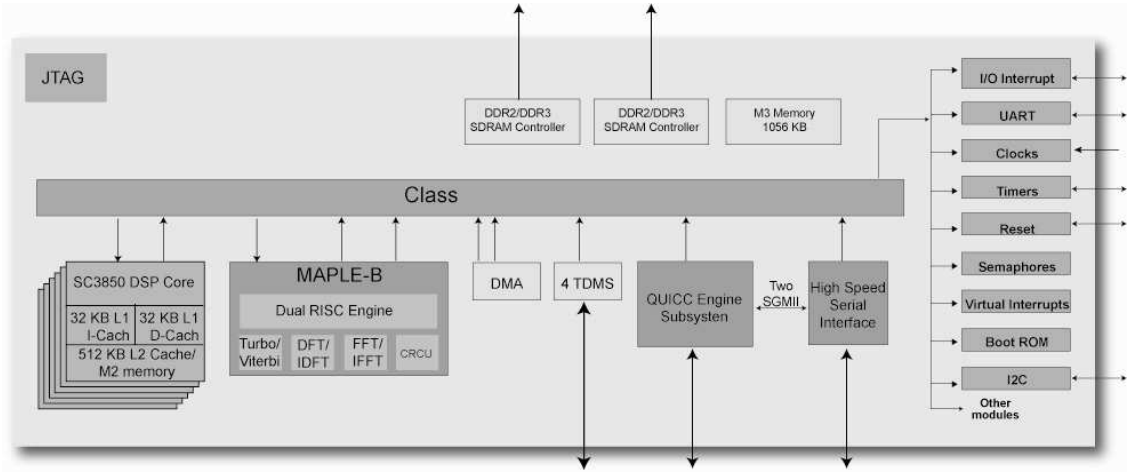

Fig.6. Freescale 8156 multi-core DSP processor.

3.2 Existing Vendor-Specific Multi-Core DSP Platforms

Several vendors manufacture multi-core DSP platforms such as Texas Instruments (TI) [8], Freescale [9], picoChip [10], Tilera [11], and Sandbridge [12-13]. Table 1 provides an overview of a number of these multi-core DSP chips.

Texas Instruments has a number of homogeneous and heterogeneous multi-core DSP platforms all of which are based on the hierarchal-interconnect architecture. One of the latest of these platforms is the TNETV3020 (Fig. 5) which is optimized for high performance voice and video applications in wireless communications infrastructure [8]. The platform contains six TMS320C64x+ DSP cores each capable of running at 500 MHz and consumes 3.8 W of power. TI also has a number of other homogeneous multi-core DSPs such as the TMS320TCI6488 which has three

1 GHz C64x+ cores and the older TNETV3010 which contains six TMS320C55x cores, as well as the TMS320VC5420/21/41 DSP platforms with dual and quad TMS320VC54x DSP cores.

Freescale's multi-core DSP devices are based on the StarCore 140, 3400 and 3850 DSP subsystems which are included in the MSC8112 (two SC140 DSP cores), MSC8144E (four SC3400 DSP cores) and its latest MSC8156 DSP chip (Fig. 6) which contains six SC3850 DSP cores targeted for 3G-LTE, WiMAX, 3GPP/3GPP2 and TD-SCDMA applications [9]. The device is based on a homogeneous hierarchical interconnect architecture with chip level arbitration and switching system (CLASS).

PicoChip manufactures high performance multi-core DSP devices that are based on both heterogeneous (PC205) and homogeneous (PC203) mesh interconnect architectures. The PC205 (Fig. 7) was taken as an example of these multi-core DSPs [10]. The two building blocks of the PC205 device are an ARM926EJ-S microprocessor and the picoArray. The picoArray consists of 248 VLIW DSP processors connected together in a 2D array as shown in Fig. 8. Each processor has dedicated instruction and data memory as well as access to on-chip and external memory. The ARM926EJ-S used for control functions is a 32-bit RISC processor. Some of the PC205 applications are in high-speed wireless data communication standards for metropolitan area networks (WiMAX) and cellular networks (HSDPA and WCDMA), as well as in the implementation of advanced wireless protocols.

Tilera manufactures the TILE64, TILEPro36 and TILEPro64 multi-core DSP processors [11]. These are based on a highly scalable homogeneous mesh interconnect architecture.

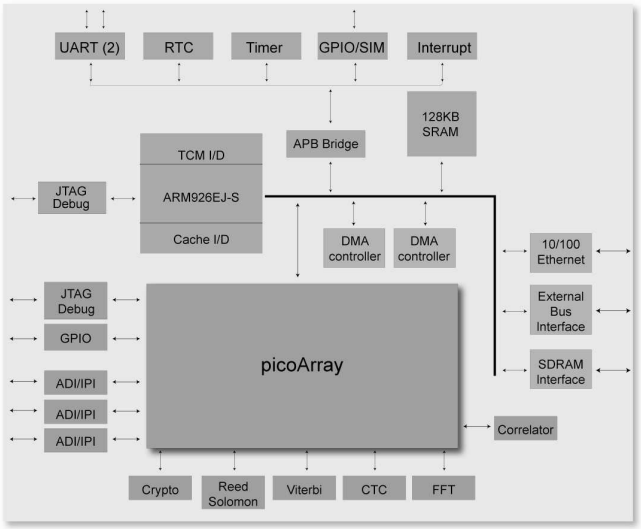

Fig.7. picoChip PC205 multi-core DSP processor.

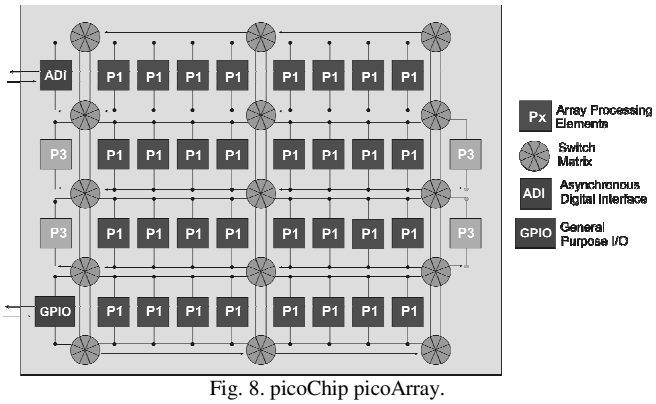

Fig. 8. picoChip picoArray.

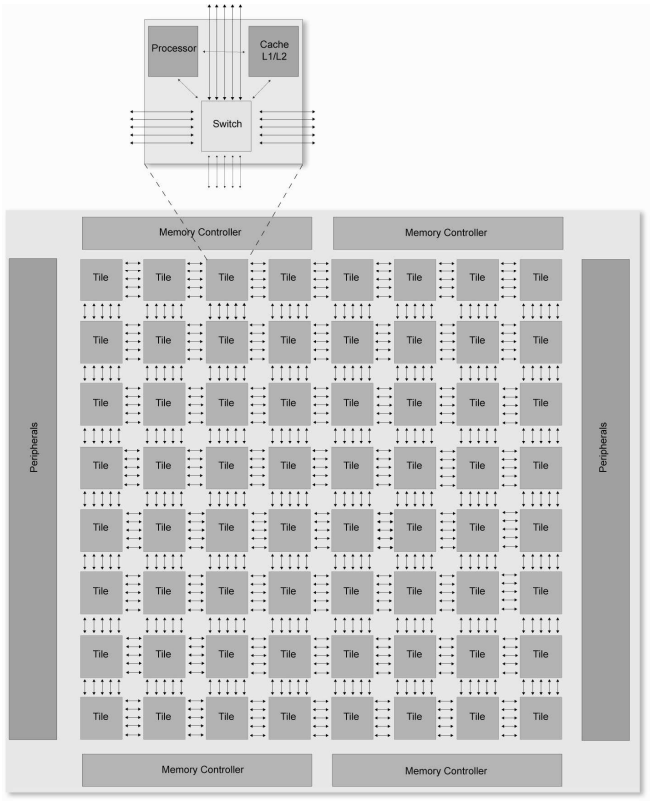

Fig. 9. Tilera TILE64 multi-core DSP processor.

The TILE64 family features 64 identical processor cores (tiles) interconnected using a mesh network of buses (Fig. 9). Each tile contains a processor, L1 and L2 cache memory and a non-blocking switch that connects each tile to the mesh. The tiles are organized in an 8 x 8 grid of identical general processor cores and the device contains 5 MB of on-chip cache. The operating frequencies of the chip range from 500 MHz to 866 MHz and its power consumption ranges from 15 – 22 W. Its main target applications are advanced networking, digital video and telecom.

SandBridge manufactures multi-core heterogeneous DSP chips intended for software defined radio applications. The SB3011 includes four DSPs each running at a minimum of 600 MHz at 0.9V. It can execute up to 32 independent

Table 2: BTDI OFDM benchmark results on various processors for the maximum number of simultaneous OFDM channels processed in real time. The specific number of simultaneous OFDM channels is given in [17].

	Clock (MHz)	***DSP cores***	***OFDM channels***
TI TMS320C6455	1200	1	Lowest
Freescale MSC8144	1000	4	Low
Sandbridge SB3500	500	3	Medium
picoChip PC102	160	344	High
Tilera TILE64	866	64	Highest

instruction streams while issuing vector operations for each stream using an SIMD datapath. An ARM926EJ-S processor with speeds up to 300 MHz implements all necessary I/O devices in a smart phone and runs Linux OS. The kernel has been designed to use the POSIX pthreads open standard [14] thus providing a cross platform library compatible with a number of operating systems (Unix, Linux and Windows). The platform can be programmed in a number of high-level languages including C, C++ or Java [12-13].

3.3 Multi-Core DSP Platform Performance Analysis

Benchmark suites have been typically used to analyze the performance among architectures [15]. In practice, benchmarking of multicore architectures has proven to be significantly more complicated than benchmarking of single core devices because multicore performance is affected not only by the choice of CPU but also very heavily by the CPU interconnect and the connection to memory. There is no single agreed-upon programming language for multicore programming and, hence, there is no equivalent of the "out of the box" benchmark, commonly used in single core benchmarks. Benchmark performance is heavily dependent on the amount of tweaking and optimization applied as well as the suitability of the benchmark for the particular architecture being evaluated. As a result, it can be seen that single core benchmarking was already a complicated task when done well, and multicore benchmarking is proving to be exponentially more challenging. The topic of benchmark suites for multicore remains an active field of study [16]. Currently available benchmarks are mainly simplified benchmarks that were mainly developed for single-core systems.

One such a benchmark is the Berkeley Design Technology, Inc (BTDI) OFDM benchmark [17] which was used to evaluate and compare the performance of some single- and multi-core DSPs in addition to other processing engines. The BTDI OFDM benchmark is a simplified digital signal processing path for an FFT-based orthogonal frequency division multiplexing (OFDM) receiver [17]. The path consists of a cascade of a demodulator, finite impulse response (FIR) filter, FFT, slicer, and Viterbi decoder. The benchmark does not include interleaving, carrier recovery, symbol synchronization, and frequency-domain equalization.

Table 2 shows relative results for maximizing the number of simultaneous non-overlapping OFDM channels that can be processed in real time, as would be needed for an access point or a base station. These results show that the four considered multi-core DSPs can process in real time a higher number of OFDM channels as compared to the considered single-core processor using this specific simplified benchmark.

However, it should be noted that this application benchmark does not necessarily fit the use cases for which the candidate processors were designed. In other words, different results can be produced using different benchmarks since single and multi-core embedded processors are generally developed to solve a particular class of functions which may or may not match the benchmark in use. At the end, what matters most is the actual performance achieved when the chips are tested for the desired customer's end solution.

4. SOFTWARE TOOLS FOR MULTI-CORE DSPs

Due to the hard real-time nature of DSP programming, one of the main requirements that DSP programmers insist on having is the ability to view low level code, to step through their programs instruction by instruction, and evaluate their algorithms and "see" what is happening at every processor clock cycle. Visibility is one of the main impediments to multi-core DSP programming and to real-time debugging as the ability to "see" in real time decreases significantly with the integration of multiple cores on a single chip. Improved chip-level debug techniques and hardware-supported visualization tools are needed for multi-core DSPs. The use of caches and multiple cores has complicated matters and forced programmers to speculate about their algorithms based on worst-case scenarios. Thus, their reluctance to move to multi-core programming approaches. For programmers to feel confident about their code, timing behavior should be predictable and repeatable [5]. Hardware tracing with Embedded Trace Buffers (ETB) [18] can be used to partially alleviate the decreased visibility issue by storing traces that provide a detailed account of code execution, timing, and data accesses. These traces are collected internally in real-time and are usually retrieved at a later time when a program failure occurs or for collecting useful statistics. Virtual multi-core platforms and simulators, such as Simics by Virtutech [19] can help programmers in developing, debugging, and testing their code before porting it to the real multi-core DSP device.

Operating Systems (OS) provide abstraction layers that allow tasks on different cores to communicate. Examples of OS include SMP Linux [20-21], TI's DSP BIOS [22], Enea's OSEck [23]. One main difference between these OS is in how the communication is performed between tasks running on different cores. In SMP Linux, a common set of

tables that reflect the current global state of the system are shared by the tasks running on different cores. This allows the processes to share the same global view of the system state. On the other hand, TI's DSP/BIOS and Enea's OSEck supports a message passing programming model. In this model, the cores can be viewed as "islands with bridges" as contrasted with the "global view" that is provided by SMP Linux. Control and management middleware platforms, such as Enea's dSpeed [23], extend the capabilities of the OS to allow enhanced monitoring, error handling, trace, diagnostics, and inter-process communications.

As in memory organization, programming models in multi-core processors include Symmetric Multiprocessing (SMP) models and Asymmetric Multiprocessing (AMP) models [24]. In an SMP model, the cores form a shared set of resources that can be accessed by the OS.

The OS is responsible for assigning processes to different cores while balancing the load between all the cores. An example of such OS is SMP Linux [18-19] which boasts a huge community of developers and lots of inexpensive software and mature tools. Although SMP Linux has been used on AMP architectures such as the mesh interconnected Tilera architecture, SMP Linux is more suitable for SMP architectures (Section 3.1) because it provides a shared symmetric view. In comparison, TI's DSP/BIOS and Enea's OSE can better support AMP architectures since they allow the programmer to have more control over task assignments and execution. The AMP approach does not balance processes evenly between the cores and so can restrict which processes get executed on what cores. This model of multi-core processing includes classic AMP, processor affinity and virtualization [23].

Classic AMP is the oldest multi-core programming approach. A separate OS is installed on each core and is responsible for handling resources on that core only. This significantly simplifies the programming approach but makes it extremely difficult to manage shared resources and I/O. The developer is responsible for ensuring that different cores do not access the same shared resource as well as be able to communicate with each other.

In processor affinity, the SMP OS scheduler is modified to allow programmers to assign a certain process to a specific core. All other processes are then assigned by the OS. SMP Linux has features to allow such modifications. A number of programming languages following this approach have appeared to extend or replace C in order to better allow programmers to express parallelism. These include OpenMP [25], MPI [26], X10 [27], MCAPI [28], GlobalArrays [29], and Uniform Parallel C [30]. In addition, functional languages such as Erlang [31] and Haskell [32] as well as stream languages such as ACOTES [33] and StreamIT [34] have been introduced. Several of these languages have been ported to multi-core DSPs. OpenMP is an example of that. It is a widely-adopted shared memory parallel programming interface providing high level programming constructs that enable the user to easily expose an application's task and loop level parallelism in an incremental fashion. Its range of applicability was significantly extended by the addition of explicit tasking features. The user specifies the parallelization strategy for a program at a high level by annotating the program code; the implementation works out the detailed mapping of the computation to the machine. It is the user's responsibility to perform any code modifications needed prior to the insertion of OpenMP constructs. In particular, OpenMP requires that dependencies that might inhibit parallelization are detected and where possible, removed from the code. The major features are directives that specify that a well-structured region of code should be executed by a team of threads, who share in the work. Such regions may be nested. Work sharing directives are provided to effect a distribution of work among the participating threads [35].

Virtualization partitions the software and hardware into a set of virtual machines (VM) that are assigned to the cores using a Virtual Machine Manager (VMM). This allows multiple operating systems to run on single or multiple cores. Virtualization works as a level of abstraction between the OS and the hardware. VirtualLogix employs virtualization technology using its VLX for embedded systems [36]. VLX announced support for TI single and multi-core DSPs. It allows TI's real-time OS (DSP/BIOS) to run concurrently with Linux. Therefore, DSP/BIOS is left to run critical tasks while other applications run on Linux.

5. APPLICATIONS OF MULTI-CORE DSPs

5.1 Multi-core for mobile application processors

The earliest SoC multi-core in the embedded space was the two-core heterogeneous DSP+ARM combination introduced by TI in 1997. These have evolved into the complex OMAP line of SoC for handset applications. Note that the latest in the OMAP line has both multi-core ARM (symmetric multiprocessing) and DSP (for heterogeneous multiprocessing). The choice and number of cores is based on the best solution for the problem at hand and many combinations are possible. The OMAP line of processors is optimized for portable multimedia applications. The ARM cores tend to be used for control, user interaction and protocol processing, whereas the DSPs tend to be signal processing slaves to the ARMs, performing compute intensive tasks such as video codecs. Both CPUs have associated hardware accelerators to help them with these tasks and a wide array of specialized peripherals allows glueless connectivity to other devices.

This multi-core is an integration play to reduce cost and power in the wireless handset. Each core had its own unique function and the amount of interaction between the cores was limited. However, the development of a communications bridge between the cores and a master/slave programming paradigm were important developments that allowed this model of processing to become the most highly used multi-core in the embedded space today [37].

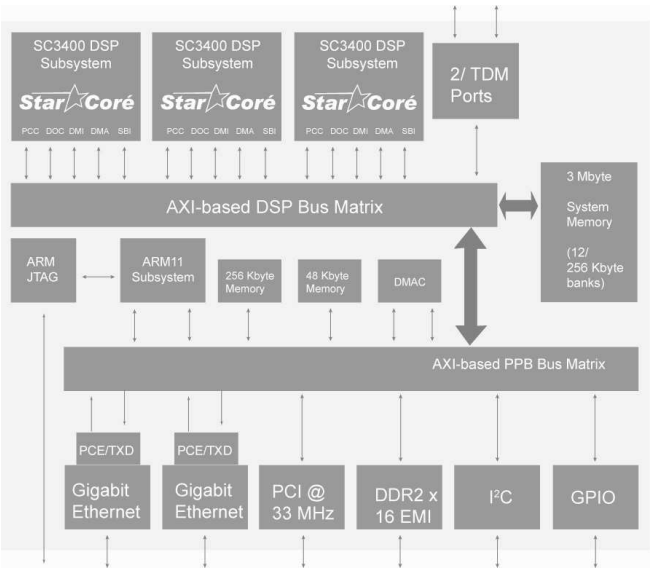

Fig.10. The Agere SP2603.

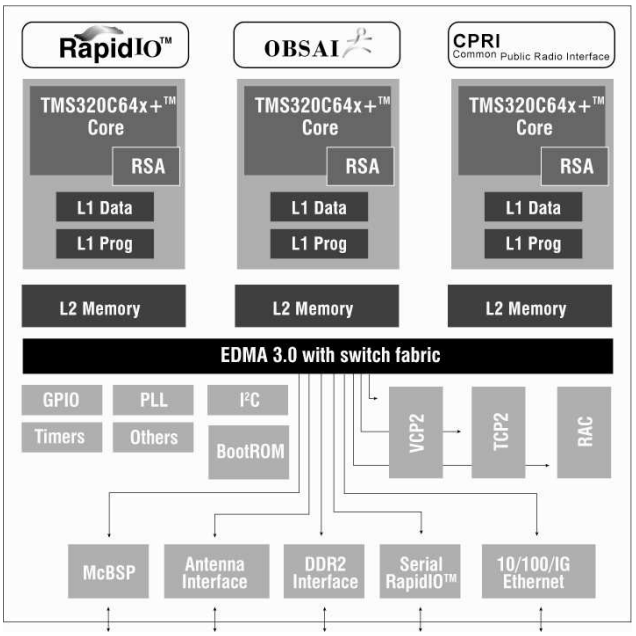

Fig. 11. Texas Instruments TCI6487.

5.2 Multi-core for Core network Transcoding

The next integration play was in the transcoding space. In this space, the master/slave approach is again taken, with a host processor, usually servicing multiple DSPs, that is in charge of load balancing many tasks onto the multi-core DSP. Each task is independent of the others (except for sharing program and some static tables) and can run on a single DSP CPU. Fig. 10 shows the Agere SP2603, a multi-core device used in transcoding applications.

Therefore, the challenge in this type of multi-core SoC is not in the partitioning of a program into multiple threads or the coordination of processing between CPUs, but in the coordination of CPUs in the access of shared, non CPU, resources, such as DDR memory, Ethernet ports, shared L2 on chip memory, bus resources, and so on. Heterogeneous variants also exist with an ARM on chip to control the array of DSP cores.

Such multi-core chips have reduced the power per channel and cost per channel by an order of magnitude over the last decade.

5.3 Multi-core for Base Station Modems

Finally, the last five years have seen many multi-core entrants into the base station modem business for cellular infrastructure. The most successful have been DSP based with a modest number of CPUs and significant shared resources in memory, acceleration and I/O. An example of such a device is the Texas Instruments TCI6487 shown in Fig. 11.

Applications that use these multi-core devices require very tight latency constraints, and each core often has a unique functionality on the chip. For instance, one core might do only transmit while another does receive and another does symbol rate processing. Again, this is not a generic programming problem. Each core has a specific and very well timed set of tasks to perform. The trick is to make sure that timing and performance issues do not occur due to the sharing of non CPU resources [38].

However, the base station market also attracted new multi-core architectures in a way that neither handset (where the cost constraints and volume tended to favor hardwired solutions beyond the ARM/DSP platform) nor transcoding (where the complexity of the software has kept “standard” DSP multi-core in the forefront) have experienced. Examples of these new paradigm companies include Chameleon, PACT, BOPS, Picochip, Morpho, Morphics and Quicksilver. These companies arose in the late 90s and mostly died in the fallout of the tech bubble burst. They suffered from a lack of production quality tooling and no clear programming model. In general, they came in two types; arrays of ALUs with a central controller and arrays of small CPUs, tightly connected and generally intended to communicate in a very synchronized manner. Fig. 8 shows the picoArray used by picoChip, a proponent of regular, meshed arrays of processors. Serious programming challenges remain with this kind of architecture because it requires two distinct modes of programming, one for the CPUs themselves and one for the interconnect between the CPUs. A single programming language would have to be able to not only partition the workload, but also comprehend the memory locality, which is severe in a mesh-based architecture.

5.4 Next Generation Multi-Core DSP Processors

Current and emerging mobile communications and networking standards are providing even more challenges to DSP. The high data-rates for the physical layer processing,

as well as the requirements for very low power have driven designers to use ASIC designs. However, these are becoming increasingly complex with the proliferation of protocols, driving the need for software solutions.

Software defined radio (SDR) holds the promise of allowing a single piece of silicon to alternate between different modem standards. Originally motivated by the military as a way to allow multinational forces to communicate [39], it has made its way into the commercial arena due to a proliferation of different standards on a single cell phone (for instance GSM, EDGE, WCDMA, Bluetooth, 802.11, FM radio, DVB).

SODA [40] is one multi-core DSP architecture designed specifically for software-defined radio (SDR) applications. Some key features of SODA are the lack of cache with multiple DMA and scratchpad memories used instead for explicit memory control. Each of the processors has a 32x16bit SIMD datapath and a coupled scalar datapath designed to handle the basic DSP operations performed on large frames of data in communication systems.

Another example is the AsAP architecture [41] which relies on the dataflow nature of DSP algorithms to obtain power and performance efficiency. Shown in Fig. 12, it is similar to the Tilera architecture at a superficial glance, but also takes the mesh network principal to its logical conclusion, with very small cores ($0.17mm^2$) and only a minimal amount of memory per core (128 word program and 128 word data).The cores communicate asynchronously by doubly clocked FIFO buffers and each core has its own clock generator so that the device is essentially clockless. When a FIFO is either empty or full, the associated cores will go into a low power state until they have more data to process. These and other power savings techniques are used in a design that is heavily focused on low power computation. There is also an emphasis on local communication, with each chip connected to its neighbors, in a similar manner to the Tilera multi-core. Even within the core, the connectivity is focused on allowing the core to absorb data rather than reroute it to other cores. The overall goal is to optimize for data flow programming with mostly local interconnect. Data can travel a distance of more than one core but will require more latency to do so. The AsAP chip is interesting as a "pure" example of a tiled array of processors with each processor performing a simple computation. The programming model for this kind of chip is however, still a topic of research. Ambric produced an architecturally similar chip [42] and showed that, for simple data flow problems, software tooling could be developed.

An example of this data flow approach to multi-core DSP design can be found in [43], where the concept of Bulk-Synchronous Processing (BSP), a model of computation where data is shared between threads mostly at synchronization barriers, is introduced. This deterministic approach to the mapping of algorithms to multi-core is in line with the recommendations made in [44] where it is argued that adding parallelism in a non deterministic manner (such as is commonly done with POSIX threads [14]) leads to systems that are unreasonably hard to test and debug. Fortunately, the parallelization of DSP algorithms can often be done in a deterministic manner using data flow diagrams. Hence, DSP may be a more fruitful space for the development of multi-core than the general purpose programming space.

Sandbridge (see Section 3.2) has also been producing DSPs designed for the SDR space for several years.

6. CONCLUSIONS AND FUTURE TRENDS

In the last 2 years, the embedded DSP market has been swept up by the general increase in interest in multi-core that has been driven by companies such as Intel and Sun.

One of the reasons for this is that there is now a lot of focus on tooling in academia and also a willingness on the part of users to accept new programming paradigms. This industry wide effort will have an effect on the way multi-core DSPs are programmed and perhaps architected. But it is too early to say in what way this will occur. Programming multi-core DSPs remains very challenging. The problem of how to take a piece of sequential code and optimally partition it across multiple cores remains unsolved. Hence, there will naturally be a lot of variations in the approaches taken. Equally important is the issue of debug and visibility. Developing effective and easy-to-use code development and real-time debug tools is tremendously important as the opportunity for bugs goes up significantly when one starts to deal with both time and space.

The markets that DSP plays in have unique features in their desire for low power, low cost and hard real-time processing, with an emphasis on mathematical computation. How well the multi-core research being performed presently in academia will address these concerns remains to be seen.

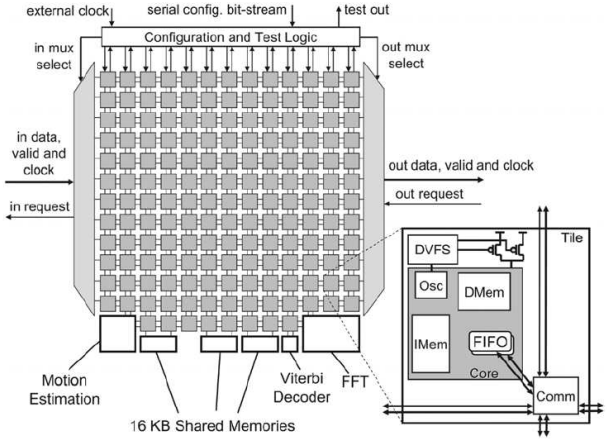

Fig.12. The AsAP processor architecture.

7. REFERENCES

[1] G.M. Amdahl, "Validity of the single-processor approach to achieving large scale computing capabilities," in *AFIPS Conference Proceedings*, vol. 30, pp. 483-485, Apr. 1967.

[2] M.D. Hill and M.R. Marty, "Amdahl's Law in the multicore era," *IEEE Computer Magazine*, vol.41, no.7, pp.33-38, July 2008.

[3] W. Strauss, "Wireless/DSP market bulletin," Forward Concepts, Feb 2009. [Online] http://www.fwdconcepts.com/dsp2209.htm.

[4] I. Scheiwe, "The shift to multi-core DSP solutions," *DSP-FPGA*, Nov. 2005 [Online] http://www.dsp-fpga.com/articles/id/?21.

[5] S. Bhattacharyya, J. Bier, W. Gass, R. Krishnamurthy, E. Lee, and K. Konstantinides, "Advances in hardware design and implementation of signal processing systems [DSP Forum]," *IEEE Signal Processing Magazine* , vol. 25, no. 6, pp. 175-180, Nov. 2008.

[6] Practical Programmable Multi-Core DSP, picoChip, Apr. 2007, [Online] http://www.picochip.com/.

[7] Tile Processor Architecture Technology Brief, Tilera, Aug. 2008, [Online] http://www.tilera.com.

[8] TNETV3020 Carrier Infrastructure Platform, Texas Instruments, Jan. 2007, [Online] http://focus.ti.com/lit/ml/spat174a/spat174a.pdf

[9] MSC8156 Product Brief, Freescale, Dec. 2008, [Online] http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MSC8156&nodeId=0127950E5F5699

[10] PC205 Product Brief, picoChip, Apr. 2008, [Online] http://www.picochip.com/.

[11] Tile64 Processor Product Brief, Tilera, Aug. 2008, [Online] http://www.tilera.com.

[12] J. Glossner, D. Iancu, M. Moudgill, G. Nacer, S. Jinturkar, M. Schulte, "The Sandbridge SB3011 SDR platform," *Joint IST Workshop on Mobile Future and the Symposium on Trends in Communications (SympoTIC)*, pp.ii-v, June 2006.

[13] J. Glossner, M. Moudgill, D. Iancu, G. Nacer, S. Jintukar, S. Stanley, M. Samori, T. Raja, M. Schulte, "The Sandbridge Sandblaster Convergence platform," Sandbridge Technologies Inc, 2005. [Online] http://www.sandbridgetech.com/

[14] POSIX, IEEE Std 1003.1, 2004 Edition. [Online] http://www.unix.org/version3/ieee_std.html

[15] G. Frantz and L. Adams, "The three P's of value in selecting DSPs," *Embedded Systems Programming*, pp. 37-46, Nov. 2004.

[16] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, K. A. Yelick, "The landscape of parallel computing research: a view from Berkeley," Technical Report No. UCB/EECS-2006-183, http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf, Dec. 2006.

[17] BDTI, [Online] http://www.bdti.com/bdtimark/ofdm.htm

[18] Embedded Trace Buffer, Texas Instruments eXpressDSP Software Wiki, [Online] http://tiexpressdsp.com/index.php?title=Embedded_Trace_Buffer

[19] VirtuTech, [Online] http://www.virtutech.com/datasheets/simics_mpc8641d.html

[20] H. Dietz, "Linux parallel processing using SMP," July 1996. [Online] http://cobweb.ecn.purdue.edu/~pplinux/ppsmp.html

[21] M. T. Jones, "Linux and symmetric multiprocessing: unblocking the power of Linux SMP systems," [Online] http://www.ibm.com/developerworks/library/l-linux-smp/

[22] TI DSP/BIOS, [Online] http://focus.ti.com/docs/toolsw/folders/print/dspbios.html

[23] Enea, [Online] http://www.enea.com/

[24] K. Williston, "Multi-core software: strategies for success," *Embedded Innovator*, pp. 10-12, Fall 2008.

[25] OpenMP, [Online] http://openmp.org/wp/

[26] MPI, [Online] http://www.mcs.anl.gov/research/projects/mpi/

[27] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, and V. Sarkar, "X10: an object-oriented approach to non-uniform cluster computing," in ACM *OOPSLA*, pp. 519-538, Oct. 2005.

[28] MCAPI, [Online] http://www.multicore-association.org/workgroup/comapi.php

[29] Global Arrays [Online] http://www.emsl.pnl.gov/docs/global/

[30] Unified Parallel C, [Online] http://upc.lbl.gov/

[31] Erlang, [Online] http://erlang.org/

[32] Haskell [Online] http://www.haskell.org/

[33] ACOTES, [Online] http://www.hitech-projects.com/euprojects/ACOTES/

[34] StreamIT, [Online] http://www.cag.lcs.mit.edu/streamit/

[35] B. Chapman, L. Huang, E. Biscondi, E. Stotzer, A. Shrivastava, and A. Gatherer, "Implementing OpenMP on a high performance embedded multi-core MPSoC," accepted and to appear the *Proceedings of IEEE International Parallel & Distributed Processing Symposium*, 2009.

[36] VirtualLogix [Online] http://www.virtuallogix.com/products/vlx-for-embedded-systems/vlx-for-es-supporting-ti-dsp-processors.html

[37] E. Heikkila and E. Gulliksen, "Embedded processors 2009 global market demand analysis," VDC Research.

[38] A. Gatherer, "Base station modems: Why multi-core? Why now?," *ECN Magazine*, Aug. 2008. [Online] http://www.ecnmag.com/supplements-Base-Station-Modems-Why_Multicore.aspx?menuid=580

[39] Software Communications Architecture, [Online] http://sca.jpeojtrs.mil/

[40] Y. Lin, H. Lee, M. Who, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, K. Flautner, "SODA: A high-performance DSP architecture for Software-Defined Radio," *IEEE Micro*, vol.27, no.1, pp.114-123, Jan.-Feb. 2007

[41] D. N. Truong et al., "A 167-Processor computational platform in 65nm," *IEEE JSSC*, vol. 44, No. 4, Apr. 2009.

[42] M. Butts, "Addressing software development challenges for multicore and massively parallel embedded systems," *Multicore Expo*, 2008.

[43] J. H. Kelm, D. R. Johnson, A. Mahesri, S. S. Lumetta, M. Frank, and S. Patel, "SChISM: Scalable Cache Incoherent Shared Memory," Tech. Rep. UILU-ENG-08-2212, Univ. of Illinois at Urbana-Champaign, Aug. 2008. [Online] http://www.crhc.illinois.edu/TechReports/2008reports/08-2212-kelm-tr-with-acks.pdf

[44] E. A. Lee, "The problem with threads," UCB Technical Report, Jan. 2006 [Online] http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.pdf