Multicore Devices: A New Generation of Reconfigurable Architectures

Steven A. Guccione
Cmpware, Inc.
Austin, TX (USA)
Steven.Guccione@cmpware.com

Abstract

For two decades, reconfigurable computing systems have provided an attractive alternative to fixed hardware solutions. Reconfigurable computing systems have demonstrated the low cost and flexibility of a software solution combined with the high performance of fixed hardware. For a variety of practical reasons, much of the work in this area focused on commercial FPGA devices as the underlying hardware platform. Recently, several new designs have diverged from the bit-level, circuit-oriented architectures of FPGAs and produced a variety of architectures more suitable for computation and high level language programming. These new highly parallel architectures contain a relatively large number of programmable cores, each approaching the complexity of a traditional microprocessor. Today such devices can be found in popular consumer electronics including game consoles and desktop PC graphics controllers as well as a new generation of supercomputers. These new devices, often described using the generic term 'multicore' represent the latest phase in the evolution of reconfigurable systems. Like earlier reconfigurable systems they promise very high performance at relatively low power with high levels of programmability. These new systems also feature software development tools geared more toward traditional high level language programming than the hardware design orientation found in earlier generations of reconfigurable systems.

1. Introduction

Multicore devices have very recently emerged as a new architectural standard across a wide variety of traditional systems. Such multicore devices come from a variety of sources and are implemented in a variety of ways. They all tend to have the same motivations for opting for a multicore implementation. These are: *power*, *performance* and *price*.

Power: The initial motivation for the move to multicore in many systems is power consumption. High performance systems have reached the limits of heat dissipation technology and have opted to do more work by using a larger number of slower, but more efficient processor cores. On the other end of the spectrum, low power devices, typically for handheld or portable operation have opted for this same approach to further drive down already low power consumption levels.

Performance: Architectural techniques for accelerating traditional uniprocessor performance have, over time, produced diminishing returns. More and more transistors have been consumed to produce lower and lower performance gains. Additionally, clock speeds have plateaued somewhere near 4 GHz and do not appear to be going significantly higher. With the two major sources of performance improvement suddenly no longer available, other techniques have been required to achieve traditional levels of performance gains in new generations of devices. Using multicore techniques, a doubling the number of cores, for instance, potentially doubles the performance of a device. This has become the standard technique for increasing performance in modern processors.

Price: the cost of designing and verifying a complex processor architecture with one billion or more transistors has become too difficult and expensive. With a multicore approach, a repeated array of identical cells greatly reduces the design and verification effort.

For these reasons, multicore devices can now be found in a very wide variety of areas. In addition, these devices have all emerged over a relatively short period of time. This has led to a less inclusive view of the existing multicore landscape, with different application areas focusing on their own multicore variants. While multicore devices are found in a diversity of areas ranging from System-on-Chip (SoC) devices, Graphics Processing Units (GPUs) and desktop and server CPUs, all of these device architectures share a common architectural approach.

One definition of a multicore device is: a programmable, explicitly parallel hardware device. These three elements define the basic attributes of a multicore device. Taken singly they are:

Programmable: A multicore device must be programmable. Each unit in a multicore device should be configurable via software. Collections of fixed hardware units do not meet the criteria to be considered a multicore device.

Explicitly Parallel: While there are many conceivable hardware devices which operate in parallel, the parallelism must be explicit and exposed to the user. Systems which may operate at high levels of performance and contain high levels of internal parallelism are not necessarily considered multicore devices, unless the user has control over the parallel operation.

Hardware device: Finally, multicore devices are taken to be single, integrated, usually single die devices. Systems which may be explicitly parallel and programmable but are constructed of multiple devices would be classified as traditional multiprocessors, not multicore multiprocessors.

Given this definition a range of modern processing devices can claim to be multicore devices.

2. Trends in Multicore Devices

From this definition of multicore, a large number of processing devices covering a wide range of applications fir the definition of a multicore device. In many cases, such as servers and desktops CPUs, the term 'multicore' is widely used and these devices are popularly identified as multicore devices. In some other architectures such as Graphics Processing Units (GPUs), the term 'multicore' is found less frequently. These devices do, however, fit the general definition of multicore and are increasingly being identified as such.

Device	*Basic Cell*	*Cells*
FPGA	1-bit LUT	100,000s
ALU Array	Integer ALU	100s
GPU	FP ALU	100s
Server / Desktop	CPU	4-8
SoC	CPU	10s
Massively Multicore	CPU	100s

Table 1: Multicore architectures.

Table 1 gives a general listing of the modern processor architectures that fit this definition of a multicore device. The graph in Figure 1 demonstrates the trends from Table 1. The notable trend in the basic cell of the multicore devices is the increase in the size of the processing core, with a plateau being reached as the core has become a full CPU. While there is considerable variation in core sizes depending on architecture and implementation, the values used in Figure 1 are relative estimates of core sizes. At one extreme FPGAs are composed of single bit Look Up Tables (LUTs), on the other extreme full 32-bit of even 64-bit microprocessor cores are used as cells in the multicore architecture.

Between these two extremes, a number of devices have opted for cells that resemble the Arithmetic and Logic Units (ALUs) found inside of may processors. These ALUs are more dense than the FPGA LUTs, but do not contain the more complex control features of CPUs. For this reason, the cores in these devices tend to operate less independently than CPU based multicore architectures and more often resemble very coarse grained FPGAs or Single Instruction Multiple Data (SIMD) architectures.

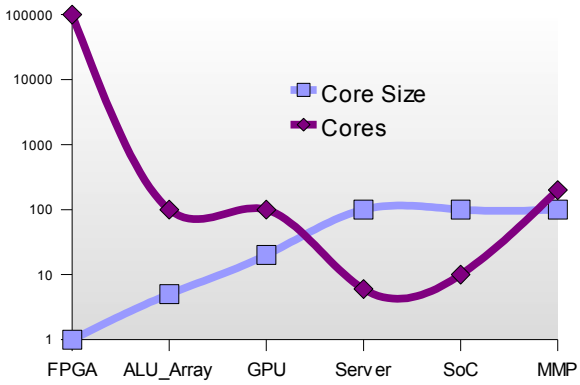

Figure 1: Granularity and parallelism in multicore devices.

The second trend in multicore devices is the total number of cores in a device. This defines the amount of parallelism and the potential overall performance of the device. While this number varies somewhat depending on the implementation technology, the general trend has tended to be the use of increasingly complex cores, but fewer in number. While FPGAs may have 100,000 LUTs, a multicore server device may have as few as two CPU cores.

The graph in Figure 1 indicates that this downward trend in the number of cores appears to have reached a low point and has begun to rise. As CPUs have been universally adopted as the core of choice, the trend is to continue to increase the number of cores. Of course, each of these types of architectures can still be found in active use, and the number of cores in each continues to

increase proportionally. While it can be expected that the number of CPU cores in a server will continue to increase, it may also be expected that the number of LUTs in an FPGA will also continue to increase, and at a similar rate.

2. Reconfigurable Computing

One feature of modern multicore devices is that they have emerged almost simultaneously in a variety of otherwise unrelated architectural areas. In many ways, each area has only begun to recognize the similarities with the other branches of the multicore family.

While it may be somewhat controversial to point to a single ancestor of each of these devices, it is useful to consider the FPGA device as used in reconfigurable computing to be the early ancestor of today's modern multicore devices. In particular, reconfigurable computing addressed the issues of software and performance in ways similar to those being confronted by the users of today's multicore architectures.

Architecturally, an FPGA can be viewed as a multicore device with its LUTs serving as the cores. While these cores are extremely simple, they are highly programmable and permit arbitrarily complex algorithms to be implemented using software. The levels of performance in reconfigurable computing systems also indicate the ability to exploit high levels of parallelism, at least for a certain range of applications.

While the hardware architecture of FPGAs fits the definition of a modern multicore device, its original purpose was to implement gate-level hardware designs. For this reason , the software tools for FPGAs are highly oriented toward circuit design. Most of the effort in reconfigurable computing has involved raising the level of abstraction of the software tools to permit high-level language programming of FPGAs.

While the hardware is parallel at the bit level, most of the algorithm implementations ignore this fine level of granularity. Groups of LUTs are used to produce larger blocks which are typically then used in parallel. For this reason even an FPGA with a potential bit level parallelism of many thousands of units will typically only achieve algorithms speedups of 10 to 100. Higher levels of parallelism can be extracted, but such algorithms and implementations were relatively rare.

A large body of work exists describing various approaches and implementations of reconfigurable computing systems. Major conferences on this subject contain more information on work in this area [1][2][3].

3. ALU Arrays

As the popularity of FPGA devices for reconfigurable computing emerged, many recognized that the architecture and programming tools were not well suited to the problems being solved with this system. For the most part, Digital Signal Processing (DSP) and related arithmetic operations were being implemented in bit level FPGA devices.

This led to a relatively large number of new, coarser grained architectures similar to traditional FPGA devices being designed and implemented. Table 2 gives a list of some of the more popular ALU array architectures offered by a variety of companies. What is perhaps most interesting is that all of these devices have very close roots to university research programs.

While hardware devices for FPGA-based reconfigurable computing was dominated by a few large, established commercial semiconductor companies, the ALU arrays represented a significant change. Nearly every ALU array offering was from a traditional small start-up venture.

Device	***ALU bits***	***ALUs***
PACT XPP	24	64
Rapport KiloCore	8	259
Elixent D-Fabrix	4	(IP core)
Chameleon CS2112	32	80
IPFlex DAPDNA	16	955
Systolix PulseDSP	16	144
MathStar FPOA	32	256
SPI Storm-1	32	960

Table 2: ALU arrays.

One goal of the new wave of ALU array architectures was to simplify the software tools used to perform computation. The larger grained devices promised to make mapping of computation to the hardware simpler. While there appeared to be some gains, it was perhaps the interconnection routing between the ALUs and its associated software that was a more significant challenge than the mapping of operations to ALUs. No significant improvement in software tools was recognized in these devices, while traditional FPGA devices continued to improve in density and cost, while their tools also continued to evolve.

While ALU Arrays represented the first break from LUT-based FPGA devices and the first step toward multicore, it was difficult for these devices to compete with existing FPGA devices. In particular, the high

sales volume and aggressive process technology of FPGA devices made up for their other limitations and made ALU arrays difficult to justify in most designs.

There was, however, one interesting exception. Some ALU arrays were able to demonstrate superior power consumption versus performance numbers when compared to FPGAs. While many of the ALU Array companies struggled, it appears that the ones which featured low power consumption fared better.

4. Early Multicore Devices

While reconfigurable computing devices were increasing their granularity and moving toward ALU array architectures, traditional DSP vendors began to explore a new approach of putting more than one processing unit on a device.

Device	*CPUs*	*DSPs*
T.I. OMAP	1	1
Xilinx Virtex-2 Pro	4	0
QuickSilver ACM	32	32
Cradle CT3400	8	6

Table 3: Early multicore devices.

The motivations here were typically for power reduction and performance improvement in a DSP application area. Perhaps the first popular multicore device was the Texas Instruments OMAP, which contained a single RISC CPU and a single DSP processor. This device was used in a very large number of mobile phones and made such an approach the standard in low power telecommunications equipment.

Also in this area, the QuickSilver Adaptive Computing Machine and the Cradle CT3400 were the next step in the evolution of multicore devices. These systems aggressively put on the order of a dozen high performance cores into a single device.

Lastly, the FPGA vendor Xilinx produced a device called the Virtex-2 Pro which contained four PowerPC cores embedded in their FPGA fabric. This unique architecture was aimed at the networking market and was for equipment such as telecommunications switches, were low power was not a crucial issue.

While the T.I. OMAP device was extremely popular, the other early multicore devices did not fare so well. Because the T.I. device targeted a very narrow application area, software libraries eased software development issues common to such new and unusual architectures. The OMAP was also the simplest possible multicore device, with only two cores, which also appears to have aided in its acceptance.

The other devices, while more sophisticated and higher performance, typically suffered from inadequate software development tools, or at least the perception of inadequate software development tools. Perhaps the lesson of these early multicore devices was the difficulty of writing software for such architectures.

5. Multicore in Networking

While the early Virtex-2 Pro multicore device from Xilinx found limited success in the networking area, the idea of focusing on a single narrow application area became the approach for the next wave of multicore devices. Perhaps most notable, the area of networking was one of the first to see a large number of multicore devices emerge.

Unlike the early DSP multicore devices, the networking applications targeted multicore devices typically by using many copies of a single popular CPU architecture for the core. It is worth noting that the multicore networking devices came from a wide variety of sources. This includes large and small companies, including start-ups as well as traditional telecommunications equipment manufacturers.

Table 4 gives a list of some of the prominent multicore devices used in networking. This continues to be a popular approach to implementing networking hardware and this list continues to expand rapidly.

Device	*Core Type*	*Cores*
Cisco CRS-1	Tensilica	192
PA Semi	PowerPC	2
Raza XLR700	MIPS64	8
Freescale MPC8572	PowerPC	2
Broadcomm BMC1250	MIPS64	4

Table 4: Networking multicore devices.

It is also worth noting that these devices tended to compete directly with the earlier generation of multicore architectures, in particular, FPGA devices. While these devices were still somewhat more difficult to program than traditional single core devices, their uniform core structure and the similarly uniform application area made software tools less of a problem. In fact, most of these devices offered no particular software support for multicore devices. The compilers tended to be identical to the single core compilers and tools for earlier single core systems.

6. Game Consoles

Game consoles occupy a somewhat unique niche in consumer electronics. While these tend to be full computing systems similar to a workstation or personal

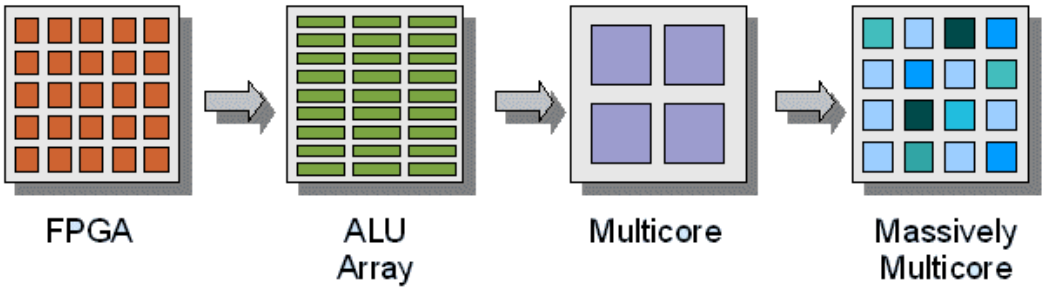

Figure 2: The evolution of multicore architectures.

computer in hardware structure, they are used for a much narrower application area: game playing.

Because the application area is so narrow, these systems have tended to focus on performance, in particular graphics performance. And because these systems resemble consumer appliances, other features such as low power and reliability are very important. For these reasons, major game consoles have shifted, somewhat quietly, to multicore processors.

Perhaps the most prominent multicore game console is the Sony Playstation3. It is powered by the 'cell' processor, which was jointly developed by Sony, Toshiba and IBM ('STI') in a multi year project that was estimated to have cost hundreds of millions of dollars.

The Cell processor contains a PowerPC core and eight floating point cores. Each of these eight floating point cores are 4-way SIMD, giving a total of 32 floating point processors operating in parallel.

All of this floating point performance was intended to accelerate video game play, in particular graphics. While the Cell processor is capable of performing many graphics chores at very high speed, the final Sony Playstation3 design opted to add a traditional graphics controller, allowing the Cell processor resources to be freed for other tasks.

As with other multicore systems, the Cell was more difficult to program than other traditional game consoles. This led to an unexpectedly slow release of software titles, at least initially. But like other niche multicore solutions, the relatively low number of game programmers and the relatively narrow area of application seems to have helped make the Cell a powerful contender in game consoles.

There has been an attempt to use the Cell processor for other applications, but these have been slow to materialize. Some users of the Playstation3 have begun to use the game console itself as a general purpose development workstation. Linux has been available on the Playstation3 for several years and impressive results of the Folding At Home project indicate that very large amounts of computing power are available from processors such as the Cell.

Another major game console, the Microsoft Xbox 360, also uses multicore technology, but somewhat less aggressively. The Xbox 360 uses three PowerPC cores with some vector processing enhancements. This architecture does not appear to have had the same barriers to adoption as the Sony Playstation3, but this is perhaps due to the simple architecture of the Xbox 360.

7. Desktop and Server Processors

Perhaps the most prominent and widely discussed move to multicore has been in the traditional desktop and server devices. All of the major manufacturers of desktop and server CPUs have shifted their high end designs to multicore.

These devices have taken a relatively conservative approach and have gone multicore somewhat reluctantly. Desktop and server processors are almost exclusively concerned with performance. Historically, performance has been increased by the doubling of the number of transistors in a design every 18 months (popularly referred to as "Moore's Law") by producing smaller transistors, and the increasing of the clock speed of these smaller transistors.

Device	***Core Type***	***Cores***
AMD Phenom X4	x86	4
Intel Core 2 Duo	x86	4
Sun UltraSPARC T2	SPARC	8

Table 5: Sever and desktop multicore devices.

Sometime near the turn of the new millennium, the power consumption of traditional single core processors began to limit the increases in performance. A this time all major manufacturers of desktop and server processors began to build 'dual core' processors

containing two identical processor cores. The process continues with four and eight cores now standard for high performance devices.

While these server and desktop processors contain multiple cores, much of the software which runs on these devices cannot take advantage of these multiple cores. While operating system support can help distribute task level parallelism across cores, this type of parallelism is difficult to find on many desktop systems.

Server systems, on the other hand, are quite capable of making use of multiple cores running multiple tasks. Their relatively low power considering their high performance have made these devices a favorite of data centers and installations which require large amounts of server hardware.

Unfortunately, one of the primary bottlenecks of traditional single core devices was the memory - CPU interface, often referred to as the 'von Neumann bottleneck'. While the number of cores and the work they perform continues to increase dramatically, the bandwidth from main memory to the multicore CPU device has remained essentially unchanged. In addition new problems such as 'cache pollution' in multicore devices that share caches has become recognized as a problem with these new architectures.

8. Multicore SoC

Yet another place where multicore has taken hold quietly is in System On Chip (SoC) devices. These are custom silicon devices, typically for some commercial embedded application. A common example would be the processor in an MP3 player.

The move to multicore by SoC devices has been motivated by some slightly different factors than other multicore devices. As with desktop and server devices, power is a prime concern for many SoC devices. In a technique similar to the one used on these larger machines, SoC devices have used multicore as a way to reduce power consumption.

In a typical CPU, the clock speed is proportional to the voltage. Increasingly higher voltages are required to increase the clock speed, and hence, the processor performance. But increasing the voltage also increases the power consumption, and in a non-linear fashion. A doubling in clock speed results in significantly more than doubling the power consumption in most CPUs. So if the clock speed can be cut in half and two processors effectively used to share the processing duties, a significant power savings can be realized. Many SoC devices have begun to take this approach.

Additionally, many SoC devices use multicore as a way to simplify hardware design and reduce risk. It may be possible to implement some part of an algorithm, for instance, using a dedicated processor as opposed to a hard-wired Intellectual Property (IP) core. This will not only save the effort of designing, implementing and verifying this IP core, it will also produce a programmable solution in place of a fixed one. This permits bugs to be fixed, even in fielded systems.

Additionally, this styler of design can provide some obsolesce-proofing, since new features, including performance and power consumption enhancements can be added on to the system via a software upgrade long after the systems has been deployed. A fixed hardware solution would require the replacement of hardware, which may be difficult or impossible.

Again, it is notable that this benefit of emerging multicore SoCs covers many of the same themes of earlier reconfigurable computing systems.

9. Massively Multicore

While multicore SoC devices are aimed at a particular narrow application area and server and workstation devices are aimed at general purpose processing, an emerging multicore device architecture attempts to cover the area between these two approaches. These devices typically feature dozens or even hundreds of cores and may soon contain thousands. While no standard term exists for this group of devices they will be referred to here as *massively multicore* devices.

Device	***Core Type***	***Cores***
Azul Vega2	Java	48
Ambric Am2000	Custom	336
picoChip cp200	16-bit DSP	248
Tilera Tile64	MIPS	64
Boston Circuits gCore	ARC	16
Intellasys SEAforth	Custom	24
Plurality HAL-256	?	256
Parallax Propeller	Custom	8
ElementCXI	?	?
Coherent Logix	?	?

Table 6: Massively multicore devices.

Today there are many such systems being developed, primarily by venture capital backed start-up companies. Table 6 gives a list of this generation of massively multicore devices. This list is not complete and it is expected that many similar companies will produce similar devices in the near future.

Unlike multicore SoC devices, massively multicore architectures do not target a specific application or even application area. The goal of most of these devices is to provide high performance combined with low power for a wide range of applications. While some of these devices may focus on a specific market for business reasons, their architectures are all general purpose and fully programmable.

While general purpose, these massively multicore devices are not aimed at traditional desktop and server style general purpose computing. They will typically be used to provide high levels of performance in applications that could otherwise not use a programmable solution. This ability to implement a variety of programmable applications using a single device is very attractive in that it eliminates the need to design and implement a custom solution. Implementing such a custom solution is not only very expensive, but can delay a product's time to market substantially.

Massively multicore architectures hope to provide a superior solution over custom Application Specific Integrated Circuits (ASICs) and even FPGAs for many applications. Clearly the performance of these devices can be as much as one to four orders of magnitude increase in raw performance over traditional single core programmable solutions. The ability to purchase such devices commercially should provide a significant cost and time to market advantage for designers of embedded high performance systems.

10. Graphics Processing Units

Graphics Processing Units (GPUs) were originally designed to accelerate graphics applications on workstations and personal computers, and increasingly on other devices such as mobile phones. GPUs have been commercially available for several years, but have only recently begun to evolve into systems which clearly fit the definition of multicore. While many modern GPU devices fit the definition of massively multicore devices, they are treated as a separate category because of their long history and the tendency to use these devices in a very narrow application area.

While there have been a large number of makers of graphics acceleration hardware in recent years, there are currently two major manufacturers of these devices: Nvidia and ATI. Both have relatively similar products and both are making inroads into more general purpose processing.

Table 7 gives a brief overview of the current offerings from GPU vendors. While earlier architectures were very focused on performing standard graphics operations, newer devices have provided architectural enhancements and even software support to perform more general purpose computations.

These systems tend to be some of the highest performing in the multicore landscape with all supporting floating point arithmetic operations. These devices tend to consume relatively large amounts of power, which also separates them from many of the current massively multicore devices.

In addition to the GPU devices, similar architectures not specifically oriented toward graphics are also emerging. These devices are aimed at the high performance and scientific computing areas and also feature a large number of floating point cores on a single device. Two examples of these devices are the ClearSpeed CSX600 and the GRAPE-DR device.

Device	***Core Type***	***Cores***
Nvidia GeForce	32-bit FP	112
ATI Firestream	64-bit FP	320
ClearSpeed CSX600	64-bit FP	96
Grape-DR	64-bit FP	512

Table 7: GPU and GPU-like devices.

The ClearSpeed device is commercially available, while the GRAPE-DR is a component used in a supercomputer project sponsored by the Japanese government. Because of the high rates of computation, it is expected that similar devices will emerge to address the needs of the high performance computing market.

11. Soft Multicore

The most recent and still emerging multicore category is soft multicore. This approach uses an FPGA device configured as a multicore processor. While certainly much less efficient that custom ASIC multicore implementations, this technique is attractive for a variety of reasons.

First, existing FPGA technology permits literally hundreds of simple RISC CPU cores to be implemented in a single FPGA device. This provides thousands of MIPS of raw performance in a single device.

In addition, the configuration of the FPGA as a multiprocessor provides a new level of abstraction. It is no longer necessary to view the device as a collection of LUTs. In fact, it may no longer be necessary to use the hardware design tools supplied by the FPGA vendor. Once the multiprocessor is implemented, standard software development tools such as high level language compilers may be used in much the same way as with any other multicore device.

Finally, the reprogrammability of the FPGA device opens the door for a variety of enhancements unavailable in fixed multicore architectures. Density may be improved and power consumption reduced, for instance, by replacing one or more soft CPU cores with a hard-wired functional core. Similarly, the soft CPUs in the FPGA can be enhanced by adding application specific instructions, for instance, to increase performance.

It is interesting to note that such soft multicore devices bear a close resemblance to earlier reconfigurable computing applications. A number of cores are used to implement some application in a parallel fashion. The only difference with the soft multicore approach is that the cores themselves may be programmable CPUs. This two-level programmability adds a new dimension of flexibility and potential performance to FPGA-based systems.

Soft multiprocessing also expands the sorts of applications that may be suitable for FPGA implementation. In spite of may theoretical claims of generality, reconfigurable computing systems have tended to be used in fairly narrow application areas involving highly regular and highly parallel algorithms. The use of soft CPUs relaxes this restriction somewhat and should permit the implementation of algorithms that may not have so regular a structure.

Soft multiprocessing is only now emerging, but its advantages may make it a popular design methodology for high performance computing with FPGA devices. It is also notable that while emerging multicore devices appear to be quickly encroaching on the traditional FPGA high performance application space, soft multiprocessing may be able to reclaim these applications to FPGA hardware platforms.

11. Multicore Tools

While the focus of this paper is to survey the current landscape in multicore architectures, some mention of software tools is in order. As has often been the case in high performance computing, hardware has led software. While a wide variety of multicore architectures has been launched in the market, the offerings for software tools have been lacking.

Most of the effort seems to be in the area of GPUs. Both Nvidia and ATI have tool sets to assist in implementing high performance computing applications for their GPUs. In addition, the software company *RapidMind* has commercially offered similar GPU oriented tools. *RapidMind* also appears to be extending its offering to popular multicore devices, in particular in the server and desktop area.

Each of the many small companies offering multicore devices offer some programming environment. None have caused any particular excitement in the wider programming community. After several years of deployment of multicore hardware it is becoming accepted that there will be no 'magic bullet' in the form of an automatic compiler or parallelizing tool. The emphasis at this stage seems to be on minimizing the learning curve and beginning training of programmers in the use of multicore. In general this means the explicit expression of parallelism in the software.

While many are pressing for new languages that aid in the expression of parallelism, there is also much resistance to attempts to require the use of new languages. It is notable that the multiprocessor world has faced similar issues for decades and has not moved toward new languages.

But some lessons have been learned from the multiprocessor experience. Libraries to produce and control parallel threads or objects and to explicitly manage communication across processors is gaining some attention. But it is not clear if the needs of system-level multiprocessors overlap sufficiently with those of multicore. In particular, the core interconnection bandwidth for single multicore device is so much higher than that of traditional multiprocessors and the memory and operating system environments so different, that it is uncertain if this approach will be productive.

Finally, even the notion of coarse grained parallelism has come into question. While the most popular model for parallelism today is the 'threads' model, it is not clear that it will translate well into large numbers of cores with very high bandwidth communication channels. In particular, the reliability of such an approach has been called into question [16].

12. Conclusions

Multicore devices are emerging from a variety of areas and are targeting a wide variety of applications. There are some notable trends, in particular the increasing numbers of cores. Somewhat troubling, however, it the lack of investment in multicore software as compared to the apparent investment in multicore hardware [18][23].

Unlike other technologies, multicore performance is highly sensitive to software implementation. Writing software one particular way as opposed to another can have a dramatic impact on performance. While many wait for tools to fill this gap, others are opting for programmer education. Some combination of both is likely to be the solution that drives multicore forward. The progress is expected to be evolutionary, rather than revolutionary.

Other aspects of multicore are just beginning to be explored. Defect and fault tolerance in particular may be game-changers. The ability to dramatically improve silicon yields for large and complex devices is surely attractive to all makers of multicore hardware. This is expected to be an active area of research in the very near future.

11. References

[1] IEEE Symposium on Field Programmable Custom Computing Machines (FCCM), http://www.fccm.org/, 2008.

[2] International Conference on Field Programmable Logic and Application (FPL), http://www.fpl.org/, 2008.

[3] International Conference on Field Programmable Technology (FPT), http://www.icfpt.org/, 2008.

[4] "Xilinx enhances FPGAs with embedded PowerPCs", http://www.eet.com/semi/news/OEG20020304S0017, March 4, 2002.

[5] "The Future of Microprocessors", David Patterson, U. California at Berkeley, June 2001. http://www.cs.berkeley.edu/~pattrsn/talks/NAE.ppt

[6] Chip Multiprocessing Resources: http://www.princeton.edu/~jdonald/research/cmp/

[7] The Cmpware Multiprocessor Report, http://www.MultiprocessorReport.com/, 2008.

[8] Programmability Issues for Multicore Computers (MULTIPROG), http://www.valimar.it/multiprog/.

[9] Forum on Application Specific Multiprocessor SoC (MPSOC), http://www.mpsoc-forum.org/index.html

[10] Harry Goldstein, "Cure for the Multicore Blues", IEEE Spectrum, Volume 44, Number 1, January 2007, pages 40-43.

[11] Richard McDougall, "Extreme Software Scaling", ACM Queue, Volume 3, Number 7, September 2005, pages 36-46.

[12] Herb Sutter and James Larus, "Software and the Concurrency Revolution", ACM Queue, Volume 3, Number 7, September 2005, pages 54-63.

[13] Kunle Olukotoun and Lance Hammond, "The Future of Microprocessors", ACM Queue, Volume 3, Number 7, September 2005, pages 26-34.

[14] Luiz Andre Barroso, "The Price of Performance", ACM Queue, Volume 3, Number 7, September 2005, pages 48-53.

[15] Mache Creeger, "Multicore CPUs for the Masses", ACM Queue volume 3, number 7, September 2005, pages 63-64.

[16] Edward A. Lee, The Problem with Threads, IEEE Computer, volume 39, Number 5, May 2006, pages 33-42.

[17] Seth Copen Goldstein, Herman Schmidt, Mihai Budiu, Srihari Cadambi, Matt Moe and R. Reed Taylor, "Piperench: A Reconfigurable Architecture and Compiler", IEEE Computer, Volume 33, Number 4, April 2000, pages 70-77.

[18] Agam Shah, "Intel Exec: Programming for Multicore Chips a Challenge", Washington Post, April 2, 2008, http://www.washingtonpost.com/wp-dyn/content/article/2008/04/02/AR2008040201520.html

[19] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, Katherine 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.html, December 18, 2006.

[20] Krste Asanovic, Ras Bodik, Jim Demmel, John Kubiatowicz, Kurt Keutzer, Edward Lee, George Necula, Dave Patterson, Koushik Sen, John Shalf, John Wawrzynek, and Kathy Yelick, "The Landscape of Parallel Computing Research: The View from Berkeley 2.0", Manycore Computing Workshop, June 2007, http://science.officeisp.net/ManycoreComputingWorkshop07/Presentations/David%20Patterson.pdf

[21] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Google Labs, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004, http://labs.google.com/papers/mapreduce.html

[22] Brian Hayes, "Computing in a Parallel Universe", American Scientist, volume 95, 2007, pages 476-480, http://www.americanscientist.org/content/AMSCI/AMSCI/ArticleAltFormat/2007102151724_866.pdf

[23] Richard Goering, "Dearth of Tools Could Stall Multicore Onslaught", EE Times, April 2., 2007, http://www.eetimes.com/showArticle.jhtml?articleID=198701494