Insight 64

Everything You Always Wanted to Know About HSA*

Explained by Nathan Brookwood

Research Fellow, Insight 64

October, 2013

* But Were Afraid To Ask

Abstract

For several years, AMD and its technology partners have tossed around terms like HSA, FSA, APU, heterogeneous computing, GP/GPU computing and the like, leaving many innocent observers confused and bewildered. In this white paper, sponsored by AMD, Insight 64 lifts the veil on this important technology, and explains why, even if HSA doesn't entirely change your life, it will change the way you use your desktop, laptop, tablet, smartphone and the cloud.

Table of Contents

1. What is Heterogeneous Computing?

Many in the industry have adopted the term “heterogeneous computing” to describe an approach that tightly integrates CPUs, GPUs, DSPs and other programmable accelerators into a single System on Chip (SoC).

Processor architects apply three broad templates to the chips they design, depending on the specific tasks the chip must handle. Their most general design, known as a Central Processing Unit (CPU), addresses the wide range of tasks required to run an operating system and support application programs such as web browsers, word processors and video games. CPUs struggle with the massive calculations needed to manipulate 3-D images on a computer display in real time, so chip architects devised specialized Graphics Processing Units (GPUs) to handle these types of parallel operations. Contemporary GPUs not only accelerate a broad range of parallel computations; they also help reduce the energy needed to perform those operations, a desirable feature in power-constrained environments. Conventional CPU designs also struggle when called upon to support high-speed streaming data in digital communications systems and audio applications, along with a variety of other low latency, compute-intensive tasks, so specialized units, known as Digital Signal Processors or DSPs, emerged to handle these workloads. DSPs embedded in your cell phone convert the analog signals in your speech into a stream of bits, move those bits to the nearest cell phone tower using a variety of radio protocols, and then convert any received digitized voice signals back into a form you can hear using the phone’s speaker. That same DSP also handles the data you send and receive via WiFi and any Bluetooth communications between your tablet and its keyboard.

Programmable devices optimized for these diverse workloads evolved independently for several decades, and designers often included all three flavors of chips in the systems they assembled. Increased transistor budgets now allow designers to place CPU, GPU and DSP elements onto a single System on Chip (SoC). This physical integration enables smaller devices, reduces cost and saves power, since on-chip communication uses far less energy than connecting chips via a motherboard or substrate. The co-location of these logic blocks on a single slab of silicon is just the beginning. To gain the full benefits of integration, separate functions need shared access to the data they process. Getting these diverse processor personalities to work as a team is no easy task; as with any team-building exercise, a manager needs to help the team members work together. The creation of a model that presents these features in a manner comprehensible to mainstream software developers, and supported by their development environments, is a more challenging task. We’ll examine moves the industry is taking to address these issues a little later in this document.

2. Why does Heterogeneous Computing matter?

Today’s computer users want to handle a wide variety of tasks and interact naturally with their systems. Instead of typing on a keyboard and pointing with a mouse, they want to aim their phones and tablets at a scene, pinch and zoom the image they see on the display, and capture a great photo. They expect their systems to recognize faces, track eye movements and respond to complex inputs (touch, voice and gestures) in real time. When playing games, they want the objects they blow up to explode realistically, with full visual and physical fidelity. They want batteries that last a long time on a single charge. These new user experiences place far greater computational demands on system resources than the usage models of just a few years ago. Conventional CPU designs cannot meet the response time expectations these new workloads demand, especially on limited power budgets. Fortunately, much of this work fits nicely with the talents GPUs and DSPs bring to the table. GPUs can easily extract meaning from the pixels captured by a camera, and transform those pixels into objects a CPU can recognize. DSPs have long played a key role in voice and image processing systems. These different processing elements have long been available, albeit packaged discretely. Clever programmers have been able to cobble together proofs of concept that demonstrate what this technology can accomplish. The challenge the industry now faces is to integrate these capabilities into standard environments that simplify the programming task for the broader community of software developers who haven’t had an opportunity to earn a PhD in computer science from Stanford or MIT.

3. What is Heterogeneous Systems Architecture (HSA)?

HSA is a new hardware architecture that integrates heterogeneous processing elements into a coherent processing environment. We’re using “coherent” both as Webster’s defines it (“logically or aesthetically ordered”) and as computer scientists define the term (“a technique that ensures that multiple processors see a consistent view of memory,” even when values in memory may be updated independently by any of those processors).[1, 2] Memory coherency has been taken for granted in homogeneous multiprocessor and multi-core systems for decades, but allowing heterogeneous processors (CPUs, GPUs and DSPs) to maintain coherency in a shared memory environment is a revolutionary concept. Ensuring this coherency poses difficult architectural and implementation challenges, but delivers huge payoffs in terms of software development (it enables standard programming models and simplifies lots of other issues), performance and power (the ability for CPUs, DSPs and GPUs to work on data *in situ* eliminates copy operations and saves both time and energy). This means programs running on a CPU can hand work off to a GPU or DSP as easily as to other programs on that same CPU; they just provide pointers to the data in the memory shared by all three processors and update a few queues. Without HSA, CPU-resident programs must bundle up data to be processed and make input-output (I/O) requests to transfer that data via device drivers that coordinate with the GPU or DSP hardware. Ugly!

The introduction of HSA represents the same kind of conceptual breakthrough that the introduction of virtual memory wrought in the 1970s. We take virtual memory (VM) architectures for granted now, but prior to their availability, software developers were forced to manage main memory within their own application programs. They developed complicated overlay schemes to load different portions of their code, depending on program flow. Since they could not fit all their data into their systems' small main memories (one megabyte was considered a lot of memory in those days), they had to call I/O drivers to move data back and forth to the hard disk. VM solved all those problems; programmers could assume they had vast amounts of physical memory at their disposal, and the operating system ensured the right code and data migrated into physical memory on demand. Analogously, HSA allows developers to write software without paying much attention to the processor hardware available on the target system. When a program needs to process voice inputs, it won't care whether those inputs are handled by a DSP or the CPU; likewise, when it calls a routine to find a face in a crowd, it won't care if the work is done by the CPU or GPU. (There may be some performance or power considerations here, but those are just "implementation details.") Virtual Memory was a big deal forty years ago (IBM charged $250,000 for the hardware to implement VM on its System/370 mainframes[3]), but it's a given today, even on $100 smartphones and tablets. HSA starts out at a much more affordable price point (it will most likely be in products selling for less than $600 within a couple of years), and it will get even more affordable as time goes by.

4. How can end-users take advantage of HSA?

Good news! The only thing end-users need do to benefit from HSA is buy hardware systems that are designed for HSA, and those will be available from many hardware suppliers in a wide range of configurations and form factors. The heavy lifting in the HSA world is done by the architects and engineers who design HSA SoCs. Software developers also need to optimize their applications, middleware and operating systems for HSA, but that work will be less complicated than otherwise because HSA solves most of the difficult aspects of coordinating heterogeneous processors in hardware. In short, aside from doing their homework when selecting a smartphone, tablet or notebook system, end users will be able to lean back, or forward, and enjoy the experiences HSA enables.

5. How does HSA affect system power consumption?

The need to reduce system power consumption looms large in all aspects of electronic component design these days, even for devices that get their power from the wall. Battery life ranks high in the purchase criteria for mobile devices, so all aspects of SoC design must be evaluated with regard to power use. Fortunately, two facets of the HSA design help tame the power budgets of even the most power-sensitive applications. First, HSA allows software packages to use the most power-efficient processor in an SoC for any given task. For example, software can farm out tasks with heavy parallel content to the GPU, which uses less power per calculation than the more general-purpose CPU cores need to do the same work. Or it can off-load the work to parse incoming streams of data to the DSP, which has been optimized for these types of chores. Second, it allows all processing elements to share data structures in system memory, and thus eliminates the need to copy data from a section of memory managed by the CPU to a different section managed by the GPU. Those

[1] Hennessy and Patterson, Computer Architecture, 3rd Edition, P. 605
[2] HSA gurus refer to this feature as "Cache Coherent Shared Virtual Memory (CC-SVM)"
[3] Davis and Richmond, What Price Virtual Memory?, Proceedings of the 2nd Symposium on Simulation of computer systems, 1974, Pp. 85-94

copy operations eat up both time and power, so eliminating them is a big win on both counts. [4]

6. Will HSA make the smartphone, tablet or laptop I just bought run better?

Odds are that system you bought in 2013 includes an SoC that integrates CPU and GPU technology and perhaps a DSP as well. The concept of co-locating all these processors on a single piece of silicon has been around for a few years, and most hardware vendors have adopted it in one form or another. As we've noted elsewhere, HSA takes this integration to the next level, by allowing these processors to work harmoniously in a shared memory environment. This aspect of HSA is so new it won't be available in any systems planned for shipment in 2013. The systems you can buy today deliver better battery life and performance in thinner and lighter form factors than the ones you could buy a few years ago, but as the song goes, you ain't seen nothing yet. Between now and then, enjoy your new device, and be sure to download the latest apps that take advantage of the integrated features in today's hardware.

7. What workloads benefit the most from HSA?

HSA makes it easier to deploy CPU and GPU resources harmoniously to execute application programs, so it makes sense that the applications that benefit the most include those that spend a lot of time performing calculations that can be done in parallel on a GPU. (The CPU side of a current quad-core SoC can deliver around 100 GFLOPS, while the GPU side offers peak rates approaching 700 GFLOPS[5]). Although CPU resources are more than adequate for most traditional scalar workloads, they struggle when called on to analyze images, sort through vast databases, or interact with users via "natural user interfaces." Fortunately, these latter tasks are just the ones that benefit the most from the GPU on the SoC; it's just a matter of adapting applications to become "HSA-aware."

8. How does HSA compare to FSA, the previous acronym describing your efforts to combine CPU and GPU into one SoC?

For several years, AMD dubbed its efforts to combine CPU and GPU technologies into integrated SoCs, "AMD FSA." Last year, AMD formed an alliance with other silicon suppliers sharing AMD's vision of diverse processors working harmoniously in a shared memory environment, and rechristened the concept "Heterogeneous Systems Architecture." (See "HSA Foundation" below)

At Insight 64, we like the name change. *Heterogeneous* definitely implies the inclusion of different elements, which more accurately characterizes what HSA is all about.

9. What heterogeneous devices does HSA support?

Although AMD's original efforts regarding HSA focused on uniting the CPU and GPU technologies that dominate PC workloads, the HSA concept applies to other programmable elements, including DSPs, FPGAs, video decoders, codecs and image processors. AMD's architects concluded that once they accomplished the difficult task of integrating CPUs and GPUs, adding additional processor types would be relatively straightforward. As future HSA specifications are developed, HSA Foundation members plan to extend the system architecture beyond CPUs and GPUs, and accommodate a wide variety of programmable accelerators.

10. How does HSA compare to GP/GPU computing?

The use of discrete GPUs to accelerate compute-intensive applications has gained popularity ever since it was first introduced by NVidia in 2006. Almost one hundred supercomputers in the Top500 list rely on AMD and NVidia GPUs to deliver breathtaking performance.[6] The peak performance of high-end discrete GPUs exceeds that of the GPUs integrated in SoCs, since discrete GPUs contain substantially more floating point hardware, and utilize dedicated high speed memory to store parallel programs and data. The same semiconductor technology that allows the GPUs in SoCs to get faster every year also allows discrete GPUs to get faster, so

[4] The HSA team at AMD analyzed the performance of a commonly used multi-stage video analysis algorithm used to identify faces in a video stream. They compared a CPU/GPU implementation in OpenCL against an HSA implementation that seamlessly shared data between CPU and GPU, eliminating memory copy operations. The net result was a 2.3x relative performance gain and a 2.4x relative power reduction on the same hardware configuration. (http://bit.ly/VDDmyO)

[5] AMD A10 APU at 4.4 (CPU Frequency) * 4 (Cores) * 8 (FLOPS) + .844 (GPU Core Frequency in GHz) * 384 (Shader count) * 2 (FLOPS) = 140.8 (CPU GFLOPS) + 648.192 (GPU GFLOPS) = 788.992 Total GFLOPS

[6] The Top500 list gets updated every six months. The latest version can be found at http://www.top500.org/list/2013/06/ .

the discrete units will likely always have a raw performance advantage. The discrete approach has to move data back and forth over the system's PCIe bus between the system's main memory and the GPU's dedicated memory. The bandwidth of that PCIe connection is a fraction of the system's memory bandwidth, so these transfers take time, consume power, and even worse, complicate the software programming model. This added complexity has had little impact on GP/GPU acceptance within the high performance computing community, where there's no shortage of very smart graduate students willing to traverse technical hurdles in pursuit of their PhD's, but it has limited the GP/GPU concept's ability to go mainstream.

Although the integrated GPUs in HSA SoCs possess less raw floating point processor performance than their high-end discrete counterparts, (and always will for the same size die, since they have to include CPU, GPU, DSP and other accelerators), integrated GPUs can operate on data wherever it resides within the system's memory; HSA eliminates the need to shuttle data back and forth. Better still, instead of needing to manage the CPU/GPU interface with a traditional driver model, HSA allows the GPU to be treated as just another processor. GPU-specific code gets invoked the same way CPU code gets invoked, via a set of standard library interfaces that are bound at installation time and dispatched via task queues. This approach simplifies software development and makes it easier for programmers who may not have had the time to pursue a PhD to take advantage of the capabilities of traditional GP/GPU approaches.

11. How does HSA compare to NVidia's CUDA programming environment?

CUDA is a representative example of the GP/GPU computing model, which has clearly attained a high degree of acceptance within the high-performance computing community. But CUDA suffers from all the GP/GPU constraints we discussed earlier, along with one other that transcends the technical issues (good and bad) that come with that model. CUDA is a single-vendor, proprietary solution that works only with NVidia hardware. HSA follows a multi-vendor, open standards business model. Through the HSA Foundation (that we'll discuss later), AMD has proliferated its technology to other SoC suppliers who design chips based on a variety of CPU and GPU architectures. Many of these suppliers serve the smartphone market, which dwarfs the PC and workstation markets in size, and is growing far more rapidly. CUDA may remain relevant to ISVs serving niche markets where the economics of the proprietary GP/GPU model work, and to educational institutions seeking low cost teraflops computing resources, but HSA allows ISVs to use this technology to reach high volume end-user markets around the globe.

12. Can HSA help me if I design hardware?

Absolutely! The hardware you design won't be very useful without software, and HSA helps software developers take full advantage of the features in the hardware you design. HSA also includes a secret sauce that lets your hardware conform to HSA's cache coherent shared virtual memory architecture, which in turn makes it easier for software developers to create software for your hardware. HSA also includes a virtual instruction set architecture known as HSAIL (HSA Intermediate Language) that simplifies software distribution, by allowing a common set of binaries to operate on a range of otherwise incompatible hardware systems. We'll say more about how HSAIL accomplishes this later in this paper.

13. Do developers have to learn a new programming language to use HSA?

Absolutely not! The HSA platform has been designed to support a variety of high-level parallel programming languages, including C++, C++ AMP, C#, OpenCL, OpenMP, Java and Python. The companies promoting HSA have been working with their ecosystem partners to jump start the creation of these development tools, although it's a little early in the process for those partners to announce their plans without an NDA. Most of the development for early HSA systems will be done in OpenCL along with C or C++, but Java work has started, and the HSA developer tools pipeline is filled with adaptations of other popular programing environments.

14. How will developers create and distribute HSA-based applications?

The development process for HSA software proceeds along the same lines as development for non-HSA systems, but developers need to use HSA-aware compilers and libraries to create their code. These compilers typically include features that identify portions of the code that can be accelerated via parallel execution, and then create data structures that enable those parallel operations to be invoked as the program executes. Instead of emitting binary code that is bound to a specific GPU architecture, HSA compilers compile to a virtual instruction set known as HSAIL (HSA Intermediate Language). HSAIL also serves as a distribution mechanism for HSA software. HSA's runtime environment contains an element known as the "HSAIL finalizer" that maps the intermediate code to the target system's GPU-specific instruction set as it installs or executes that code. The use of HSAIL allows the underlying hardware to evolve in a manner transparent to the software developer and the tools used to create HSA code.

15. Will developers have to update their code every time HSA changes?

The companies promoting HSA have stated that as HSA and HSAIL evolve, they will retain the ability to run binary code compiled to an earlier HSAIL standard. They view this issue the same way CPU suppliers view maintaining backward compatibility for their hardware instruction sets; it generally makes good business sense to preserve the installed base of software as they move the architecture forward. But, just as with the evolution of hardware ISAs, there's no practical way to ensure forward compatibility, so developers will need to recompile if and when they want to take advantage of features added to the architecture over time.

16. Do I need HSA if I already use OpenCL?

If you are already using OpenCL, you deserve to be congratulated for your technological astuteness and ability. Although it's been around for several years, OpenCL has gained acceptance only with a relatively small cadre of elite programmers. Rest assured, HSA preserves your investments in OpenCL training and code. The applications you already created may immediately run faster, as they benefit from a larger virtual memory space and elimination of copies that occurred in the legacy model. You may want to revisit your code, however, since the HSA runtime and compilers will let you take advantage of new HSA-specific APIs that further improve the performance of your software.

17. Can HSA accelerate my Java-based applications?

There's a lot of work going on in the community to extend Java's "write once run anywhere" (WORA) philosophy into the realm of heterogeneous computing, where they seek the ability to "write once run anywhere with performance" (WORAP). To date, most of this work has followed the traditional GP/GPU programming model, but developers are now beginning to focus on what they can do when freed of the need to move data between main memory and dedicated GPU memory. The Aparapi project (https://code.google.com/p/aparapi/) provides a tool that converts Java bytecode into OpenCL at runtime, and then executes that code on the GPU. Project Sumatra (http://openjdk.java.net/projects/sumatra/) takes advantage of GPU resources to accelerate the Java Virtual Machine (JVM) runtime environment. By allowing the CPU and GPU to share in-memory data structures, HSA improves the way both of these Java enhancements work. And yes, Java programmers will keep writing Java. Pure Java.

18. Will HSA work if my system doesn't include an AMD APU?

Applications adapted to take advantage of HSA should work on all HSA-enabled systems, including (but not limited to) those that contain AMD processors. In 2012, seven companies that shared AMD's vision of heterogeneous computing -- AMD, ARM, Imagination Technologies, MediaTek, Qualcomm, Samsung Electronics and Texas Instruments -- came together to form the HSA Foundation (HSAF). These companies agreed to develop products compatible with HSA, so that developers can target software for one HSA implementation with the expectation their software will run on the other suppliers' HSA devices as well. (Since then this movement has gathered momentum and the Foundation now has over 30 of the leading companies in the world as members). This compatibility promise relies on the HSAIL secret sauce we discussed earlier. HSAIL uses a common intermediate format to mask the different characteristics of each supplier's GPU architecture. Each supplier provides HSA runtime and finalizer software optimized for its own processors to handle the task of mapping the common intermediate language onto its own unique devices when the application is installed on a specific target system. The usual caveats about testing on specific platforms still apply. HSA will help a great deal, but it won't eliminate all problems.

19. Can anyone join the HSA Foundation?

The HSAF is a not-for-profit consortium for SoC vendors, SoC IP vendors, OEMs, OSVs, ISVs and Academia whose goal is to make it easier to program for parallel computing. Organizations can participate at various membership levels, including Founder, Promoter, Supporter, Contributor, Associate and Academic, with annual dues that range from $125,000/year (Founder) to $500/year (Academic). Organizations typically join the HSAF in order to drive the future direction of HSA, to get rights to make HSA part of their processor products or gain early access to HSA specifications. You can get more information regarding the HSA Foundation at http://hsafoundation.com/.

20. **Do I have to join the HSA Foundation to use or develop applications with HSA capabilities?**
No, the HSAF works on an open IP model for specifications and tools and runtimes. All the HSA specifications software developers need to write applications are freely downloadable and may be used on a royalty-free basis. This makes sense, given that the primary goal of the HSAF is to make it easier to program heterogeneous parallel devices and drive the proliferation of this technology.

21. **Will the HSAF certify HSA-based applications conform to HSA standards?**
The HSAF provides conformance tests to make sure that HSA platforms conform to the HSA specifications so that you can be sure you get the right answers wherever your program runs. However there's no "HSAF Seal of Approval" that certifies your software application uses HSA correctly. If you develop application software, it's easiest to think about "HSA conformance" the same way you think about conformance to "x86," "Windows," "Android," "Linux" or other platforms. HSA, like those other platforms, includes the tools you need to create software, but it's up to you to make sure that software works properly. The HSAF does plan to develop a procedure that lets application developers contribute code to HSA test suites used to validate HSA development tools and implementations.

22. **How do you pronounce "Heterogeneous?"**
The word "heterogeneous" can be pronounced het-er-*uh*-jee-nee-*uhs*, or het-er-*uh* -jeen-*yuhs*. It is sometimes confused with the word "heterogenous," pronounced het-*uh*-roj-*uh*-*nuhs*. The two words have slightly different meanings; heterogeneous means "different in kind or unlike or incongruous," while heterogenous (used primarily in biology and pathology) means "having its source or origin outside the organism or having a foreign origin." Isn't English a wonderful language?

23. **What is the AMD Developer Summit 2013?**
The AMD Developer Summit 2013, also known as APU13, is the follow-on to the AMD Fusion Developers' Summits held in 2011 and 2012. It should be a good place for people interested in HSA to get the latest information on developments in heterogeneous computing, and to interact with others in the growing HSA ecosystem. APU13 will be held on November 11-14, 2013 at the San Jose Convention Center in northern California. On November 12th, a day the organizers have designated "HSA Day," AMD's HSAF partners will have a chance to discuss their HSA plans. The average temperature in San Jose in November is 64°F, and it can be rainy, so pack accordingly.

24. **Where can I find out more about HSA?**
The HSA Foundation's website (http://hsafoundation.com/) contains the latest versions of HSA specifications, along with information about how organizations can join the HSAF and participate in its programs.

AMD's Developer Central website (http://developer.amd.com/) contains a variety of presentations and tutorials relating to HSA and parallel programing.

Hot Chips 25, a technical conference sponsored by the IEEE with a focus on high performance chips, offered a four-hour tutorial on HSA in August 2013. The session included speakers from AMD, ARM and Qualcomm. A video recording of this session is available from the Hot Chips organizers. (http://www.hotchips.org/)