

Envisioning the Future for Multiprocessor SoC

Multiprocessor SoCs are no longer an advanced research topic for academia. Ninety percent of SoCs designed in 130 nm include at least one CPU. Most popular multimedia platforms are already multiprocessor SoCs. This roundtable brings together key players from the semiconductor industry and from academia to discuss the challenges and opportunities brought by this new technology.

IEEE Design & Test thanks the roundtable participants: moderator Ahmed Amine Jerraya (CEA-Leti),

Olivier Franza (Intel), Markus Levy (The Multicore Association and EEMBC), Masao Nakaya (Renesas Technology), Pierre Paulin (STMicroelectronics), Ulrich Ramacher (Infineon Technologies), Deepu Talla (Texas Instruments), and Wayne Wolf (Princeton University). Jerraya gratefully acknowledges the help of Roundtables Editor Bill Joyner (Semiconductor Research), who organized the event. Special thanks go to the IEEE Computer Society for sponsoring the roundtable.

Jerraya: Our roundtable topic is, Why do we need multiprocessor systems on chip (MPSoCs)? How would you briefly define an MPSoC, and how is it different from classic system-on-chip or ASICs design?

Wolf: A multiprocessor SoC has two aspects. One is the multiprocessor aspect—we can debate how programmable those processors should be. The other aspect is distinguishing a SoC that's one chip—plus all the other chips we need to make the system work—but we need to distinguish that architecture from ASICs as well as from the multicore CPUs in personal computers these days. Multicore CPUs aren't designed to be systems; they're components. One difference with the SoC is that it's designed to be a complete system, often at least somewhat application specific.

Levy: We can view an MPSoC from a hardware and a software perspective. From the hardware side, vendors claim to build multiprocessing chips, but what does that really mean? Does it mean taking two cores and gluing them on a die, and somehow they communicate with each other? Or maybe they are two cores that perform independent of each other? Either way, it becomes a challenge when vendors add three, four, five, or even 100 cores. How do we write the software to get these cores to effectively communicate

with each other, and how do we design the hardware mechanisms to enable efficient, low-latency data transfers?

Nakaya: My definition is simple. An MPSoC has more than two processors on one chip.

Jerraya: Why do we need more than one processor on a single chip?

Ramacher: It depends on the applications. For multiple processors on the chip, I see three big areas: One is network processors, another is cell phones with baseband and multimedia processing, and the third area is in automotive driver assistance systems and in-car entertainment.

Franza: For general-purpose processor design, the key goal has been performance. Moore's law has charted a tremendous growth in microprocessor performance; however, we're in a phase where power consumption is limiting performance. Power efficiency and performance/watt are now critical metrics along with absolute performance. One approach to increase performance efficiency is by adding multiple cores and running them at a lower frequency and lower voltage.

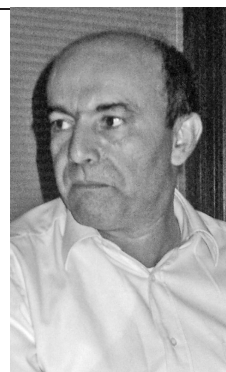
Levy: Because the automotive field is not an application in and of itself, manufacturers are starting to use multicore in some individual areas, such as power train and safety electronics—and of course, in-car entertainment systems. But where traditionally the power train electronics had a single CPU with high-performance peripherals around it, the processors just don't have enough horsepower to keep up with the hybrid technologies and the fuel efficiency requirements, so the power train is going multicore as well.

Ramacher: At Infineon, we're selling controllers or the TriCore, which comes with flash and some of the memory, but it's not really a multiprocessor. High-end BMWs, for instance, have approximately 95 controllers—the problem of embedded software on system design is for BMW, not for us as a semiconductor company.

Earlier when I mentioned the automotive application, I had in mind the driver assistance systems—for instance, imaging systems looking for track identification and so on. That means a lot of imaging power, and that comes not with one 5-GHz DSP but with several in parallel. The driving force for Infineon is to meet the applications' performance requirement. We need to supply reasonable power, and in the embedded arena—for handsets, for instance—it means 300, maybe 600, MHz, barely one gigahertz per processor. We need a baseband processor with 10 giga-instructions per second at least, and that means 15 to 33 processors on chip.

Paulin: The first driver in our applications is usually power; the second is flexibility. It's parallel systems—already parallel in hardware—in which we need more flexibility while keeping a reasonable power/performance ratio. Moving from hardware to a general-purpose processor at 1 GHz is too big a step. We're trying to find the right balance of power and flexibility. First and foremost, we're concerned with components previously implemented in hardware—extremely parallel and extremely low power—which are moving to more application-specific, domain-specific processors at a low frequency with lots of parallelism. That parallelism is in the application. We've been doing it for years in hardware; now we need to do it more flexibly through a combination of hardware accelerators and domain- or application-specific processors.

Jerraya:
When you say "multiple processor system on chip," do you mean the same processor duplicated several times or a different processor on the same chip?



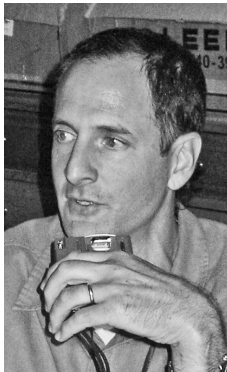
Wolf: Let's also remember the importance of real time. Unlike laptops, a lot of these SoCs must operate in real time and meet deadlines. Sometimes that's impossible—and, in general, less efficient—on a single processor. One of the basics of real-time system theory is that we cannot always use 100% of the CPU and also guarantee that we'll meet deadlines. If we have time-critical tasks, we often put them on separate processors to make sure they meet their deadlines; then we use another CPU to share the things that aren't so time critical. Real time is an important driver for multiprocessors, and as a side benefit, we get the power benefits of being able to run several processors at lower clock rates.

Jerraya: When you say "multiple processor system on chip," do you mean the same processor duplicated several times or a different processor on the same chip?

Nakaya: I mean that both cases are multiple processor systems on chip. Renesas has both examples. One is an application processor (SH-Mobile) for mobile phones. It has heterogeneous cores. The other is a reconfigurable processor (MTX) for audiovisual applications. It has many of the same cores.

Talla: We've seen scenarios that have heterogeneous processors on the same SoC. For example, at Texas Instruments we have our OMAP [open multimedia application platform] platform and the DaVinci platforms where we have a general-purpose processor, like an ARM, and we have a DSP, like a C64 or a C55. On the other hand, some applications, such as high-density voice, just have several DSP cores tied together.

Paulin: The applications we're dealing with range from the order of one GOPS [giga operations per



Levy:
We don't even know what types of hardware accelerators will be included in future MPSoCs, so the whole debugging issue just gets worse.

second] for high-end audio to hundreds of GOPS in the video, graphics, and imaging domain. Anything involving more than a few GOPS means that some part of the functions is implemented as ASIC hardware. We cannot implement that with a set of homogeneous RISC processors. If we're saying power is important, we're talking about a couple hundred megahertz processors, and they must be application- and/or domain-specific.

What I see happening is what I call regular heterogeneity, where heterogeneous systems, general-purpose hosts, then domain-specific processors, like DSPs, become more like VLIW processors. I agree with Ulrich Ramacher's presentation earlier today: that application-specific processors and hardware, and reconfigurable logic in the middle, will probably be implemented not on chip, but in a system-in-package. We're trying to achieve as much regularity and homogeneity as possible, but not more than we can afford in terms of our power budget.

Franza: Wayne's comment is interesting—that embedded processor and laptop processor requirements are completely different. In that sense, all legacy microprocessor design companies are fairly new to the world of MPSoC, because that wasn't what we were targeting. We were really targeting single-core performance, and when we hit the power wall, we became interested in MPSoC, so we're pretty new in that discipline. To answer your question, Ahmed, I would say that, for now, microprocessors are moving to multiple similar cores, homogeneous cores, on a chip. The future will depend on the applications. As we drive for increased performance and power efficiency, the use of dedicated processors for hardware acceleration may prove to be a viable solution introducing heterogeneous processors and greater integration of peripherals for MPSoCs.

Levy: It's obvious that for now, the majority of processor vendors are implementing multicore devices using the homogeneous and SMP approach. But what are the performance limits for a shared memory architecture? Intel, for example, is at two cores now, going to four, and soon to eight and perhaps higher. The standard SMP approach has performance limits beyond four cores, but vendors are implementing a variety of new techniques to minimize the impact with a greater number of cores. Alternatively, vendors can implement MPSoCs using a distributed memory architecture, which could include homogeneous or heterogeneous cores. While homogeneous MPSoCs make debugging much easier, with more complex chips, such as ST's Nomadik and TI's DaVinci, we need more proprietary solutions for debugging. Even those chips are relatively simple compared to where things are going in the future. We don't even know what types of hardware accelerators will be included in future MPSoCs, so the whole debugging issue just gets worse.

Ramacher: The landscape of multiple processors today is heterogeneous. For handhelds, we will see homogeneous architectures developed for the base-band and the multimedia part sharing one core and having different extensions by instructions. I wouldn't consider these different extensions to be heterogeneous, because the software development and compiler is to a great extent shared. The handheld scenario may be a special case, and a homogeneous platform clearly has a lot of advantages.

Paulin: We need to distinguish heterogeneous functionality and performance from the way we implement it. If we have a base core from which we can build 95% of our software tools in common and then derive different application- or domain-specific specializations in a clean, repeatable way with the same tools, then we have the best of both worlds. That is how to achieve heterogeneity with a homogeneous approach. That has big benefits in terms of having the same tools and the same verification environment. In the end, we have to build platforms for these and ISSs [instruction-set simulators], virtual platforms, and so forth.

Levy: One of the things the Multicore Association is trying to do is establish a common terminology in the industry, because everybody calls these cores different

names. Is a core processor a core or does it have to be a completely separate hardware accelerator with its own instruction set capability in order to be called a core?

Ramacher: We define a core processor to be a RISC or a VLIW processor, and we usually attach a small RISC core to an accelerator such that we have a homogeneous programming environment.

Talla: I agree, there's a lot of confusion as to what we call a core. If it's a simple accelerator tied to other accelerators, or if it's tied to an ARM or a DSP, it probably should just be considered an accelerator. But if we have a piece of IP that's essentially providing subsystem functionality—for example, if a camera phone is doing the complete capture functionality, or what we call the ISP [image signal processor], we could probably classify that as a core.

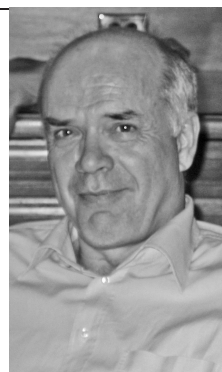
Levy: The question is not necessarily, "What's the market for MPSoC?" I think we have to look at what the applications are, and it doesn't really matter whether it's an MPSoC or a Pentium processor as long as it's doing the job within the price, performance, and power budget. The end user doesn't care.

Ramacher: Someone defined a multiple processor system on chip as simply more than two processors, and that could mean that individual programs are isolated, which we've known for 10 or more years. I think of the new applications—multistandard radios, for instance. The new challenge is that we can use a number of processors in parallel, working in parallel on a single program.

Paulin: The turning point in the industry is happening now. We currently approach design as a bottom-up assembly of independent subsystems to constitute a multiprocessor SoC, but we're not doing multiprocessing with a top-down approach. Moore's law applies to our bottom-up design of individual subsystems: Four chips will merge into one, so the traditional one RISC with one DSP is now becoming multiple RISCs with multiple DSPs for different subsystems combining audio, video, imaging, 3D graphics, and so on.

The key in this generational change is that we're now starting to design a system top-down from a set of applications onto a multiprocessor platform, and we're

Ramacher:
The new challenge is that we can use a number of processors in parallel, working in parallel on a single program.



starting to think about how those tasks interact. The first step is to synchronize the separate tasks. In those subsystems, we're reaching a point where one processor plus a coprocessor is not enough. Perhaps we're implementing four processors plus hardware coprocessors, so there's multiprocessing in the subsystem.

The next step is where we have potentially 32 processors for four individual audio, video, imaging, and general-purpose control subsystems working together. Then we'll start thinking, "I never have all 32 running at the same time, so I can start migrating tasks from an unused subsystem," especially if we have a homogeneous implementation of subsystems in which we use a general-purpose RISC subset. Then we could borrow resources that aren't using other subsystems—loosely coupled tasks that can be performed on an unused subsystem. At that point we arrive at the third step, which is migrating tasks and sharing resources across the entire subsystem.

Those are the three steps: (1) achieve a bottom-up design of loosely coupled subsystems that are not designed to work together but are glued together; (2) design subsystems that communicate efficiently and are planned that way; and finally, (3) implement resource sharing across subsystems for load balancing at the top level.

Wolf: Pierre is right about the turning point. Remember that during much of the 1980s and 1990s the reason for VLSI was cost reduction. We took things that were on a board and put them on a chip. We've certainly seen systems on chip that represent cost reduction but where we also achieve benefits in power—they're really not MPSoC. But now we're starting to see true multiprocessors with multiple parallel threads of execution, and that also means a different design methodology from the applications. It's much more software driven,



Wolf:
We have to undo all the problems with the software and then figure out how to design the platform.

so the hardware architects have to work more closely with the software people to understand what the software will do. That's a big change.

Ramacher: We have two changes: one is related to hardware people needing to program this programmable processor, and another change concerns the software people. They used to have a single-processor architecture; now they have a many-processor architecture.

Wolf: More readers of *IEEE Design & Test* are hardware designers than software designers, but clearly both hardware and software people need a lot of adaptation to work in an MPSoC world.

Paulin: With respect to challenges, parallel MPSoC platforms are the least challenging. A harder challenge is to write parallel applications—that's the whole education of an entire software community in describing, validating, and specifying parallel systems. The hardest task is mapping those parallel applications on that platform. In the industry, we've got it completely backward. We're focusing first on the platforms, second on the expression of parallelism, and third on the mapping. The mapping technology will take 10 years to develop, the platform innovation cycles typically a couple of years, so we need to turn this around.

We can prototype an MPSoC platform in one year so that it's ready in three years. We basically already know what we can do in platform mapping technology. Anything more sophisticated will take 10 years—software mapping technologies are more complicated than innovative platforms, so we need to educate platform designers about what software mapping technologies we think we can build in the next three

years. Then we can limit the scope of what architectures we should build in that same three-year period. That is key: to achieve synergy in the three areas of applications, tools, and platforms.

Wolf: Another problem related to mapping is that we keep talking about software people, but in fact, much of the software is from things like multimedia and communications, which isn't written by software people. It's written by signal processing and communications people who have little or no formal background in computer science. In fact, if you look at the reference implementations that come out of standards committees, they are typically highly sequentialized in ways that are difficult to unravel. It would be great if the code that was handed to MPSoC designers by application designers had nice clean expressions of parallelism. Unfortunately, many of these programs aren't written with any notion of parallelism in mind—in fact, some have been sequentialized in ways that are very hard to unravel. We have to undo all the problems with the software and then figure out how to design the platform.

Nakaya: My background is hardware design, so I am not familiar with software design. My question is: in the hardware design, there are synthesis tools for each design step. So, design productivity of hardware increases dramatically, but I am not sure there are good synthesis tools for software design. Are there any good synthesis tools for software development?

Wolf: If you're talking about something like C, we have good compilers, but if you're talking about a concurrent application like a digital camera or a cell phone where many things run at once, we don't have good enough tools where someone can write a high-level parallel description, push a button, and get a good software implementation.

Jerraya: My impression is that we are leaving the structured world where we had a company design the processor, a company design the system, another company design the operating system, and then a customer would write programs. What is the case for MPSoC?

Talla: Semiconductor companies cannot afford to just sell MPSoCs and expect other partners or vendors to pick up the rest of the components like software and


reference designs. Today, most semiconductor companies need to become systems companies. That's what many companies are becoming, in fact, and the customer basically expects us to deliver, not only the MPSoC and the documentation, but also the basic software, tools, hardware, and sometimes a complete turnkey reference design. That lets customers run to production quickly and not spend resources on developing mundane tasks. For example, if you're developing a video camera, there's little reason for the end customer to be developing the MPEG-4 or the H.264 codec. The customer now expects the semiconductor vendor to develop the codec.

Paulin: System houses are definitely expecting more, and expecting more of a turnkey solution, from the semiconductor companies. At the same time, they're asking for more openness into multimedia subsystems. They need a basic package, which is 90% of what they want to sell to the marketplace, but increasingly they also want to access those optimized multimedia subsystems to program their own image quality improvement, noise filtering, and whatever else they think is the specific value they bring to the market. So we have a conflict of delivering more while keeping the system open so system houses can add their 10% additional functionality in terms of complexity, which for them might be a significant market differentiator.

Talla: That's exactly what we are trying to address with the DaVinci platform. It's an open platform, fully programmable. Customers can choose what value they want to add at a higher level of abstraction or go deep into the software stack to change any of the core routines.

Levy: The more complex the SoC becomes, the fewer components we want the customer to access. If we have a SoC with a hundred cores, and customers want to add value for a multimedia application, then that's the part we'll open to them through some sort of API. For example, take the case of the PowerQUICC architecture from Freescale. This is a multicore device containing a PowerPC core and a special RISC core for accelerating various networking functions. Freescale kept the latter core basically hidden for many years; it was only accessible by using their library calls. In other words, the PowerPC core was completely open, and they provided the APIs for customers to access the code inside the RISC accelerator.

Franza: Looking forward, the evolution of multiprocessing capabilities on a single chip will continue with multiple cores and also virtualization, which is another way to increase that capability.



Ramacher: The more complex APIs get for SoCs, the more has to be developed by the semiconductor house producing that SoC. How can you design an API and the architecture of an MPSoC without having all the application code?

Nakaya: At Renesas, we are not expanding our business alone. As a semiconductor vendor, we have relationships with our customers—mobile-phone makers like Fujitsu, Mitsubishi, and infrastructure vendors like NTT DoCoMo. We can establish a joint development with infrastructure vendors and with mobile-phone/cellular-phone makers. Consequently, we need to consider the value chain to increase total values. If we focus only on direct customers, we will be in a tough situation in the future. Our development cost has continually increased, but our price has not equally increased.

Franza: Multiprocessing systems with multiple CPU chips have been available for many years. Operating systems and software programs have evolved to leverage this parallelism. When multithreading was introduced, it let multiple programs simultaneously run on a single die. Windows and Linux operating systems can handle multitasking. Looking forward, the evolution of multiprocessing capabilities on a single chip will continue with multiple cores and also virtualization, which is another way to increase that capability.

A general-purpose processor is different from an embedded processor: it runs hundreds of thousands of different applications—it therefore cannot be optimized for one specific application to the detriment of others. However, programs running under operating system control can provide information to the cores to manage power performance and can exploit the multiprocessing capabilities such processors offer.



Paulin:
We have to distinguish between designing useful MPSoCs and delivering useful applications on an MPSoC.

Levy: You're coming at it from the desktop and the server side, but Intel's also begun a big push into the embedded world with its dual-core architecture. In addition to providing lots of ready-made applications or libraries to support that, you also have to maintain an open architecture for people to program.

Franza: That's true; one of the advantages is that the X86 architecture is a very mature and robust architecture and has a huge existing code base. It's been around for a long time, so developers know it well and can achieve high levels of performance.

Jerraya: How difficult is it to design this MPSoC?

Levy: We'd have to look at it from the EDA perspective as well as understand the different functions that need to be integrated. To synchronize all of the activities, access memory, and deal with all the high-level issues is a significant challenge.

Franza: Another view is that, by putting multiple, simpler cores on chip, those simple cores supposedly would be easier to design or at least not as hard as a single multiple-way multithreading core. Designing each element of the MPSoC should be easier and more manageable—because they're smaller and require less complexity. Assembling everything on the chip, however, requires extra effort.

Once everything starts to get multiplied—cores, power domains, clock domains, I/O ports, and so on—a multitude of new problems arise and actually increase the design's overall complexity. For example, to name only a few, timing verification, validation, test, and debug of multiple (possibly heterogeneous) domains is a challenging and somewhat new activity required for high-quality MPSoC design.

Talla: Yes, MPSoC design is getting fairly complicated and continues to do so, given that the performance of applications is growing and the number of applications where we need to run simultaneous threads is accruing, coupled with the fact that we're integrating more components—more analog—onto the latest digital processor. On the other hand, we do have some tools we can take advantage of. For one thing, platforming helps a lot so that we don't have to redo the SoC from scratch for each generation or for each application. Seventy to eighty percent of the SoC does not have to be regenerated. Most components can be reused. Also, having experienced designers on board is another way to attack the complexity.

Paulin: We have to distinguish between designing useful MPSoCs and delivering useful applications on an MPSoC. The former is clearly a tough engineering task. Examples like the cell are impressive, but although there are hundreds of different design teams able to build these platforms, only a couple offer tools to program them efficiently.

Levy: Don't forget that companies like Freescale, IBM, and Intel are still building multicore processors running at 2 to 3 GHz, which is a lot different than an embedded multicore processor running at 300 MHz.

Ramacher: We have seen new qualities in design. In the past 10 years, we mostly dealt with multiple processors on chip that were working on their individual program and not communicating with other, isolated processors. These systems contained often coprocessors or accelerators. During that time, we looked at the algorithms and developed an idea of what the hardware macros for the accelerators or coprocessors should look like. Now, with multiple processors for applications that must cooperate in parallel, there's no way to continue to do it in the same fashion. Hardware-wise, building a scalable architecture with multiple processors is not the problem; it's the mapping: partitioning the code, scheduling the threads, synchronizing data.

Nakaya: Accordingly, as the integration level of MPSoCs becomes higher, it takes a long time to do verification, validation, and testing, in addition to debugging. Therefore, it is more difficult to meet the time to market requirement. These are big issues that present an economical problem with future MPSoC design.

Jerraya: If we have 100 companies making MPSoCs and only a few providing the environments to program them, will this technology be a differentiator in the future for semiconductor companies?

Ramacher: I don't see 100 companies—at least, not surviving. Because the application areas are all in the consumer arena, that means mass-produced products, not niche markets. We have about one billion handsets, and six or seven semiconductor houses coming up with platforms—in the future I think only a few semiconductor houses will develop these consumer applications—including fabless semiconductor companies—and I don't see many tool developers.

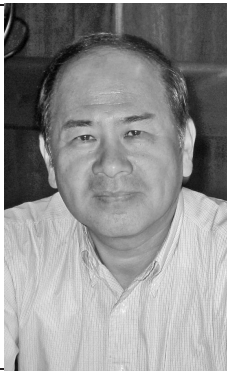
Wolf: The question of programmability gets back to this question of what constitutes a salable chip. If customers expect the semiconductor house to deliver the video codec or the communications software with the chip, then customers aren't going to be writing that code, and the semiconductor house can use whatever methods and however many people it chooses to develop the software.

Tools are certainly a good idea, but as Ulrich Ramacher points out, that may not provide a market for third parties who sell those tools. If we're talking about applications, clearly customers want to add software to differentiate their products, but what sort of software will that be? Highly paralleled software, or will it be Pac-Man running on the host processor? I suspect that, in many products, much of the programming done by the systems house and not by the semiconductor house will be the less parallel variety and where more standard programming environments will be enough.

Jerraya: We know about some successes for the MPSoC platform. The TI OMAP has already earned Texas Instruments about one billion dollars, which already covers the cost of the platform's development. Any other success stories?

Paulin: STMicroelectronics' set-top box platform. ST is the market leader in that segment and has been for many years. The set-top box started as an analog platform and moved to digital with a single processor, an ST20 old-fashioned RISC processor, and moved up to the latest generation of dual, high-definition, set-top box HDTV decoders, which have 10 to 20 processors.

Nakaya: Accordingly, as the integration level of MPSoCs becomes higher, it takes a long time to do verification, validation, and testing, in addition to debugging.



Franza: Strictly speaking, it's hard to claim a traditional microprocessor is an MPSoC, but as more and more functionality is integrated onto the CPU to gain lower power and form factor benefits, the microprocessor will become a major MPSoC player.

Nakaya: Renesas has developed a platform—the name is EXREAL. We developed it in collaboration with our customers, cellular phone vendors, and infrastructure vendors. We expanded this platform beyond mobile for automotive and audiovisual applications. The EXREAL is configured using a new, interconnected scheme to handle a variety of hardware, software, and evaluation/validation design tasks, such as scalable on-chip bus, multilayer API, and performance and power evaluation.

Jerraya: Everyone needs MPSoC, then; so what's the next step?

Wolf: The question for the MPSoC evolution is, Will architectures become less regular or more regular as the chip size and number of processors grow? Arguments can go both ways. Heterogeneous architectures often give lower energy consumption, which is important, but more regular processors are easier to program; also, because we can load balance, we can play system-level power management tricks that we can't do with a highly heterogeneous architecture.

Paulin: Factors favoring platform homogeneity include design for manufacturability. With mask set costs continuing to rise, if we can build a regular system that's overdesigned for classic applications but has a high-end performance for a niche market and we build it that way, then we get 50% yield, for example.



Talla:
For cell phones, it will probably be another two or three years before we record and play back single-channel HD.

But on that 50% yield, we might have one third of the system that's completely functional. If it's regular, we can test the functional parts, disconnect the non-functional ones, and offer a much cheaper product with a simpler package. Besides yield management and cost management, long-term reliability and fault tolerance also favor homogeneity. If we overdesign—building 12 processors when we need 10, say—we'll still have a fully functional, 10-processor system at all times; and in the telecom space, for example, that's important. Power and cost are strong factors in our design decisions now, but as we go to 45 nanometers and beyond, other forces will come into play.

Ramacher: For me, the most important question is how many different platforms we will see in the future, and how many will survive. Will set-top boxes have a different platform than the handsets? There are good reasons for expecting spin-offs from the handhelds, because of their advantage in economy of scale. Also, we see the same set of applications on the various terminals. Hence, we can expect convergence of the architectures for the set-top boxes, home router systems, and the Customer Premises Equipment.

Wolf: So you're suggesting that the set-top box of the future for cable TV could actually have a cell phone processor inside?

Ramacher: Definitely. Not only is it the set-top boxes, but think of the automotive applications, too. Currently, we have entertainment or imaging applications implemented in cars, which could be supported by a good multimedia processor developed for handhelds.

Talla: It's one thing to use some of the IP and the infrastructure and another to reuse the whole SoC. I don't think we will see anytime soon where the whole

SoC in a cell phone can be used for a set-top box, given that the set-top box SoC performance requirements will be significantly more than what the cell phone is capable of doing. Today we are talking about HD, dual HD, quad HD set-top boxes. For cell phones, it will probably be another two or three years before we record and play back single-channel HD.

Paulin: There are strong forces in the application space that are driving convergence between STMicroelectronics' application divisions. ST is diversified in both the mobile and set-top box spaces, and we're seeing more discussion and effort to homogenize the platform components as a first step. Ultimately, the platform programming models, sharing different IP, essentially become one virtual platform with a lot of available plug-and-play software and hardware IP. Both mobile and set-top box spaces share the same competences in image processing, video processing, audio, and networking. Homogenizing the platform components reduces the number of total platform variants, which becomes a key competence.

Levy: Is it conceivable that a future generation general-purpose processor could consist of many different heterogeneous cores, where you supply every chip with every type of peripheral and accelerator on it, whether you use it or not, because soon transistors will be basically free? Is that a relatively conceivable model?

Paulin: Leakage problems exist and will get worse with scaling. I don't see that we can just say "transistors are free." They're free perhaps from an area perspective only, but not from the power perspective, which is now dominating MPSoC design and development.

Talla: I don't see such a model happening in the next five to ten years. Transistors are getting cheaper, but they're not free. Today, we talk about selling cell phones in Brazil, Russia, India, and China for \$20—that's a basic GSM [Global System for Mobile Communications] phone with no fancy features. What's to say in the next three to four years that we might not sell the high-end phones for \$20? Suddenly we cannot expect in the next 5 to 10 years to throw in the whole kitchen sink and meet the price points that we need to reach.

Nakaya: I believe the problem will be design cost. The value of LSI products per wafer (8-inch equivalent)

stays unchanged, and the production cost per wafer also remains unchanged for these past 15 years, although advanced process technologies have been introduced and the numbers of integrated transistors have been increased. Therefore, profits per wafer will decrease as the design cost increases. In order to keep profitable, we have to change the situation.

Jerraya: Any final comment about MPSoCs, and what will be the next step after MPSoCs?

Talla: MPSoC integration is going to continue for at least two to three processor generations. It's getting more challenging, given the amount of analog integration needed.

Paulin: Regularity is becoming increasingly strong, which I call "regular heterogeneity," where we have a regular approach to building heterogeneous subsystems. A good way to achieve that is Ulrich's proposal: a set of VLIW general-purpose processors with application-specific accelerators attached to them. In the next five years, there will be an impetus to build more easy-to-program MPSoCs. Getting the product to market will always dominate, and that will force us to be more efficient on the mapping side. What's next? Multiprocessor systems in packages, when we figure out how to combine a DRAM, a flash, an FPGA, and an MPSoC with some to-be-designed feed-through mechanism, it will connect the pins and some grid, whether it's drill-through vias or some other mechanism.

Franza: Intel has introduced a full line of dual-core products, expanding beyond dual cores and moving toward the world of many-core MPSoC. Intel's research teams have shown research projects with tens of cores integrated on a chip. This is in line with the industry performance trend.

Integration is another important benefit of MPSoC because it lets us give more performance in as small a form factor as possible, so integration of more capacities onto the chip is where the industry is moving. What's next after that? Carbon nanotubes, quantum computing, and all these exotic techniques are far away, but they will eventually become mature and probably come into the mainstream.

Levy: The current MPSoC has probably another 10 years before people maximize its efficiency. From my perspective in running the Multicore Association, new companies are continually asking questions about this technology, and most are software companies. A lot of the software companies have become fabless semiconductor companies to support their software. This trend will continue for quite some time before we've reached its capacity.

Wolf: Rest assured, there are lots of applications where we need more processing power than current MPSoCs can provide. Robust speech recognition takes huge amounts of computing power. Even relatively simple vision tasks take huge amounts of computing power, and, of course, people want to do these on mobile platforms with zero power consumption. So, we have lots of challenges ahead.

About the participants

Ahmed Amine Jerraya, our moderator, is research director at CEA-Leti in France.

Olivier Franza is a senior staff engineer in the Digital Enterprise Group at Intel in Hudson, Massachusetts.

Markus Levy is president of the Multicore Association and EEMBC, with headquarters in El Dorado Hills, California.

Masao Nakaya is executive general manager of the LSI Product Technology Unit at Renesas Technology in Japan.

Pierre Paulin is the director of SoC platform automation, Advanced System Technology Group at STMicroelectronics in Ontario, Canada.

Ulrich Ramacher is senior director of the Innovation Department in the Communication Business Group at Infineon Technologies in Munich, Germany.

Deepu Talla is a system architect of the Imaging & Audio Group at Texas Instruments, in Dallas, Texas.

Wayne Wolf is a professor in the Department of Electrical Engineering at Princeton University in Princeton, New Jersey.