TITLE

System on Chip

BYLINE

Tanguy Risset,
Lyon University, INRIA
INSA-Lyon, CITI,
F-69621, Villeurbanne, France

SYNONYMS

System-on-a-chip, SoC

DEFINITION

A System on Chip (SoC) refers to a single integrated circuit (chip) composed of all the components of an electronic system. A SoC is heterogeneous, in addition to classical digital components: processor, memory, bus, etc. it may contain analog and radio components. The SoC market has been driven by embedded computing systems: mobile phones and handheld devices.

DISCUSSION

Historical view

Gordon Moore predicted in 1965 the exponential growth of silicon integration and its consequences on the application of integrated circuits. Following this growth known as *Moore's Law*, the number of transistor integrated on a single silicon chip has doubled every 18 months, leading to a constant growth in the semi-conductor industry for over 30 years. This technological evolution implied constant changes in the design of digital circuits, with, for instance, the advent of gate level simulation and logic synthesis. Amongst these changes the advent of System on Chip (SoC) represented a major technological shift.

The term SoC became increasingly widespread in the 1990's and is used to describe chips integrating on a single silicon die what was before spread on several circuits: processor, memory, bus, hardware device drivers, etc. SoC technology did not fundamentally changed the functionality of the systems built, but drastically enlarged the *design space*: choosing the best way to assemble a complete system from beginning (system specification) to end (chip manufacturing) became a difficult task. In 1999 a book entitled: "*Surviving the SoC Revolution: A guide to plateform-based design*", was published. It was written by a group of people from Cadence Design System, an important computer aided design tool company: SoC was driving a revolution of the *design methodologies* for digital systems.

The production of SoC has been driven by the emerging market of embedded systems, more precisely *embedded computing systems*: cellular phones, PDA, digital camera, etc. Embedded computing systems require strong integration, computing power and energy saving, features that were provided by SoC integration. Moreover, most of these systems required radio communication features; hence a SoC integrates digital components and analog components: the radio subsystem. The success of mobile devices has increased the economic pressure on SoCs, design cost and time-to-market have become major issues, leading to new design methodologies such as IP based design and hardware/software co-design.

Another consequence of integration is the exponential growth of embedded software, shifting the complexity of SoC design from hardware to software. Emulation, simulation at various level of precision and high level design techniques are used because nowadays SoC are incredibly complex: hundreds of millions of gates and millions of lines of code. Increasing digital computing power enables software radio, providing everything everywhere transparently. SoC are now used to provide convergence between computing, telecommunication and multimedia. Pervasive environments will also make extensive use of embedded "intelligence" with SoCs.

What is a SoC?

A system-on-chip or SoC is an integrated circuit composed of many components including: one or several processors, on-chip memories, hardware accelerators, devices drivers, digital/analog converters and analog components. It was initially named SoC because all the features of a complete "system" were integrated together on the same chip. At that time, a system was dedicated to a single application: video processing or wireless communication for instance. Thanks to the increasing role of software, SoCs are no longer specific, in fact many of them are re-used in several different telephony or multi-media devices.

Processors The processor is the core of the SoC. Unlike desktop machine processors, a wide variety of processors is integrated in the SoC: general purpose processors, digital signal processors, micro-controllers, application specific processors, FPGA based soft cores etc. The International Technology Road-map for Semiconductors (ITRS) indicated that, in 2005 more than 70% of application specific integrated circuit (ASICS) contained a processor. Until 2010, most SoCs were composed of a single processor, responsible for the control of the whole system, associated with hardware accelerators and direct memory access components (DMA). In 2008, the majority of SoCs were built around a processor of one of the following form of processor architectures: ARM, MIPS, PowerPC or x86. Multi-processor SoCs (MPSoC) are progressively appearing and making use of the available chip area to keep performance improvements. MPSoC appears to be a new major technological shift and MPSoC designers are facing problem traditionally associated with super-computing.

Busses The bus, or more generally the interconnection medium, is also a major component of the SoC. Many SoCs contain several busses, including a fast system bus connecting the major SoC components (processor, memory, hardware accelerator) and a slower bus for other peripherals. For instance, the Advanced Micro-controller Bus Architecture (Amba) proposes the AHB specification (Advanced High-performance Bus) and the APB (Advanced Peripheral Bus), STBus (ST-Microelectronics) and CoreConnect (IBM) are other examples of SoC busses. The use of commercial bus protocol has been a major obstacle to SoC standardization: a bus comes with a dedicated communication protocol which might be incompatible with other busses. Networks on Chip (NoC) are progressively used with MPSoC, the major problem of NoCs today is their considerable power consumption.

Dedicated components A SoC will include standard control circuits such as UART for the serial port, USB controller and control of specific devices: GPS, accelerometer, touch pad, etc. It might also include more compute intensive dedicated *intellectual properties* (IPs) usually targeted to signal or image processing: FFT, turbo decoder, H.263 coder, etc. Choosing between a dedicated component or a software implementation of the same functionality is done in early stages of the design, in the so-called *hardware/software partitioning*. A hardware IP will save power and execution time compare to a software implementation of the same functionality. Dedicated IPs are often mandatory to meet the performance requirements of the application (radio or multi-media), but they increase SoC price, complexity and also reduce its flexibility.

Other components Analog and mixed signal components may be included to provide radio connexion: audio front-end, radio to baseband interfaces, analog to digital converters. Other application domains may require very specific components such as microelectromechanical systems (MEMS), optical computing, biochips and nanomaterials.

Quality criteria The only objective indicator of the quality of a SoC is its economical success, which is obviously difficult to predict. Among the quality criteria the most important are: power consumption, time to market, cost and re-usability.

Power consumption has been a major issue for decades. Integrating all components on the same chip reduces power consumption because inter chip wires have disappeared. On the other hand, increasing clock rate, program size and number of computations has a negative impact on power consumption. Moreover, it is very difficult to statically predict the power consumption of a SoC as it heavily depends on low level technological details. A precise electrical simulation of a SoC is extremely slow and cannot be used in practice for a complete system. Minimizing memory footprint and memory traffic, gating clocks for idle components and dynamically adapting clock rate are techniques used to reduce power consumption in SoCs. In recent submicronic technologies, static power consumption is significant, leading to power consumption for idle devices.

It is usually said that, for a given product, the first commercially available SoC will capture half of the market. Hence, reducing the design time is a critical issue. Performing hardware and software design in parallel is a designer's dream: it becomes possible with virtual prototyping: simulation of software on the virtual hardware. High level design and refinement are also widely used: a good decision at a high level may save weeks of design.

The cost of a SoC is divided into two components: the non-recurring engineering (NRE) cost which corresponds to the design cost and the unit manufacturing cost. Depending on the number of units manufactured and the implementation technology, a complex trade-off between the two can be made, bearing in mind that some technological choices can shorten the time to market such as field programmable gate arrays (FPGA) for instance. NRE cost integrates, in addition to the design cost of the hardware and software part of the SoC, the cost of computer aided design tools, computing resource for simulation/emulation and development of tools for the SoC itself: compilers, linkers, debuggers, simulation models of the IPs etc. To give a very rough idea, in 2008, the development cost of a complex SoC could reach several millions dollars and could need more than one hundred man years. In this context, is seems obvious that the re-use of IPs, tools and SoCs is a major issue because it shrinks both NRE costs and time to market.

An example The figure 1 shows the architecture of the Samsung S3CA400A01 SoC, designed to be associated with another chip containing processor and memory through the Cotulla interface. The Cotulla interface is associated with Xscale processor (strong ARM processor architecture manufactured by Intel). This chip was used for instance in association with the PXA250 chip of Intel in the PDA of Hewlett Packard iPaQ H5550. This SoC illustrates the different device driver components that can be found, one can also observe the hierarchy of busses mentioned before. It also shows that there is no *standard SoC*, this one have no processor inside. Thanks to the development of the open source software community, many SoC architectures have been detailed and Linux ports are available for many of them.

Why is it a SoC "revolution"?

Although its advent was predicted, SoC really caused a revolution in the semiconductor industry because it changed design methods and brought many software design problems in the micro-electronics community.

Hardware design has been impacted by the arrival of SoCs. The complexity of chip mechanically increases with the number of gates on the chip. But the design and verification effort increases more than linearly with the number of

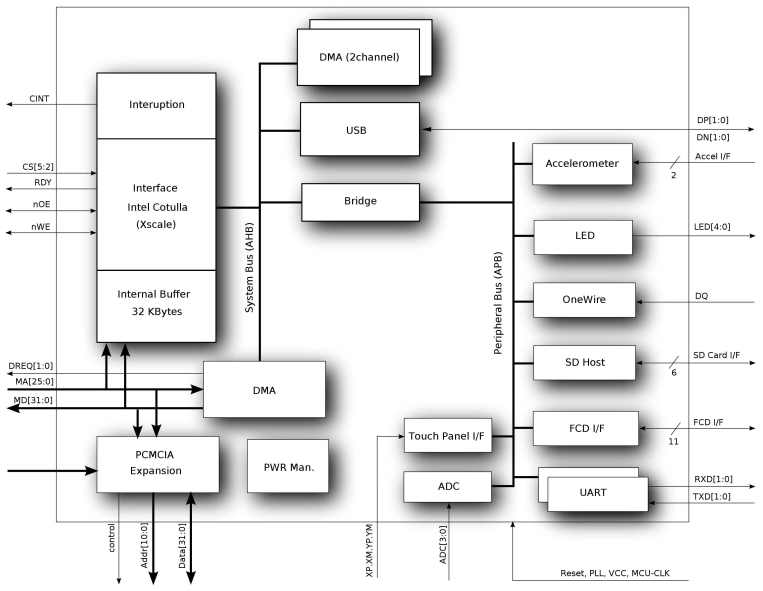

Figure 1: Architecture of the Samsung S3CA400A01 SoC

gates, this is known as the *design gap*: how can the designer efficiently use the additional gates available without breaking design time and design reliability? A SoC requires different technologies to be integrated on the same silicon die (standard cells, memory, FPGA). The clock distribution tree is a major source of power dissipation leading to the advent of globally asynchronous, locally synchronous systems (GALS). Routing wires became a real problem partially solved by using additional metal layers.

Hardware design methods have drastically changed with the constant arrival of new Electronic Design Automation (EDA) tools. RTL design methods (RTL stands for *register transfer level*) have become the standard for hardware developers replacing transistor level circuit design. The hardware description languages VHDL and Verilog are widely used for hardware specification before synthesis. However, higher level hardware description language are needed, partly because SoC simulation time became prohibitive, requiring huge FPGA based emulators. In addition, the networked nature of embedded applications, and the non-determinism introduced by parallelism in MPSoC greatly complicates SoC simulation. As mentioned before, cost and time to market imposes pressure on engineers productivity leading managers to new management methods. The hardware/software nature of a SoC require a constant dialog between hardware and software engineers.

Embedded software has been introduced in circuit design, this is, in itself, a revolution. In 2006, the Siemens company employed more software developers than Microsoft did, and since the mid-2000's, more than half of the SoC design efforts were spent in software development. Originally composed of a very basic infinite loop waiting for interrupts, embedded software has evolved by adding many device drivers, standard or real time operating system services: tasks or threads, resource management, shared memory, communication protocols, etc. Even if an important part of embedded code is written in C, there is a trend to use component based software design methodologies that reduces the conceptual difference between hardware and software objects. Verifying software reliability is very difficult, even if an important effort is dedicated to that during the design process, complete formal verification is impossible because of the complexity of the software. This explains the bugs that every-body has experienced on their mobile devices.

Both hardware and software design methods have been drastically modified, but the most important novelty concerns the whole SoC design methodology that tries to associate in a single framework hardware and software design.

SoC Design Methodology

There are many names associated with SoC design methodologies, all of them refer more or less to the same goal: designing hardware and software in the same framework such that the designer can quickly produce the most efficient implementation for a given system specification. The term frequently used today is *system level design*, the generic scheme of system level design is the following: *i*) derive hardware components and software components from the specification, *ii*) map the software part on the hardware components, *iii*) prototype the resulting implementation and *iv*) possibly provide changes at some stage of the design if the performance or cost is not satisfactory.

There is a global agreement on the fact that high level specifications are very useful to reduce the design time. Many design frameworks have been proposed for system level SoC design. The idea of refinement of the original specification down to hardware is present in many frameworks, high level synthesis (HLS) was once seen as the solution but appears to be a very difficult problem in general. In HLS, the hardware is completely derived from the initial specifications. During the 90's, the arrival of Intellectual Properties (IP) introduced a clear distinction between circuit fabrication and circuit design, leading some companies to concentrate on IP design (ARM for instance). On the other hand, IP based design and later *platform based design* propose to start from fixed (or parameterizable) hardware library to reduce the design space exploration phase. Improvement in simulation techniques permits, with the use of systemC language, to have an approach between the two by using virtual prototypes: cycle true simulation of complete SoC before hardware implementation.

MPSoC and future trends

An open question remains at the present time: what will be the dominant model for MPSoCs that are predicted to include more than one hundred processors? There are two main trends: heterogeneous MPSoCs following actual heterogeneous SoC architecture or homogeneous SPMD-like multi-processors on a single chip.

A heterogeneous MPSoC includes different types of processors with different instruction set architectures (ISA): general purpose processors, DSPs, application specific instruction set processors (ASIP). It also contains dedicated IPs that might not be considered as processors, DMAs and memories. All these components are connected together through a network on chip.

Heterogeneous MPSoCs are a natural evolution of SoC, their main advantage is that they are more energy efficient than homogeneous MPSoCs. They have been driven by specific application domains such as software radio or embedded video processing. But, for the future, they suffer from many drawbacks. As they are usually tuned for a particular application, they are difficult to re-use in another context. Because of incompatibility between ISAs, tasks cannot migrate between processors inducing a much less flexible task mapping. Moreover, in the foreseen transistor technology, integrated circuits will have to be fault tolerant because of electronics defaults. Heterogeneous MPSoC are not fault tolerant by nature. Finally, their scalability is not obvious and code re-use between different heterogeneous platforms is impossible. However, because heterogeneous MPSoCs provide better performance with lower power consumption and lower cost there are lots of commercial activities on this model.

Homogeneous MPSoC are more flexible, they can implement well known operating system services such as code portability between different architectures, dynamic task migration, virtual shared memory. Homogeneity can help in being fault tolerant, as any processor can be replaced by another. However, with more than a hundred processor on a chip in 2020, the MPSoC architecture cannot be completely homogeneous, it has to be hierarchical: clusters of processors tightly linked, these clusters being interconnected with a network on chip. Memory must be physically distributed with non-uniform access.

Ensuring cache coherency and providing efficient compilation and parallelization tools for these architectures are real challenges. In 2009, the ITRS road-map predicted for 2013 the advent of a concurrent software compiler that will "Enable compilation and software development in highly parallel processing SoC", this will be a critical step towards the advent of homogeneous MPSoC. However many unknowns remain concerning the programming model to use, the ability of compilation tools to efficiently use the available resources, the re-usability of the chip and the re-usability of the embedded code between platforms, and the real power consumption of these homogeneous multi-processors on chip.

A trade-off would be a homogeneous SoC with some level of heterogeneity: dedicated IPs for domain specific treatments. New technological techniques, such as 3D VLSI packaging technology or more generally the arrival of so-called *system in package*, mixing various nano-technologies on a single chip might also open a new road to embedded computing systems. But, whatever will be the successful MPSoC model, it will bring highly parallel computing on a chip and should lead to a revival of parallel computing engineering.

RELATED ENTRIES

BIBLIOGRAPHIC NOTES AND FURTHER READING

A number of books have been published on SoC, we have already talked about "surviving the SOC revolution" [CCH+99]. Many interesting ideas were also present in the Polis project [ea97]. Daniel Gajski [GAGS09] worked on SoC from the beginning. Wayne Wolf published a general presentation [Wol02]. More recently Chris Rowen [Row04] proposed an interesting view of SoC design.

A good overview of the status of high level synthesis can be found in Coussy and Moraviec book [CM08]. More practical details can be found on the Steve Furber book on ARM [Fur00].

A number of useful information are present on the web, from the International Technology Road-map for Semiconductors (ITRS [fS09]), from analyst or engineer [Des09] or open source software developer for embedded devices [fd09, ffeL09]. A simple introduction to semiconductor industry can be found in Jim Turley's book [Tur02].

A good introduction on MPSoC is presented in [JW04], but most of the interesting work on this subject are presented in the proceedings of the international conferences on the subject: Design Automation Conference (DAC), Design automation and Test in Europe (DATE), International Forum on Embedded MPSoC and Multicore (MPSOC), etc.

BIBLIOGRAPHY

References

[CCH+99] Henry Chang, Larry Cooke, Merrill Hunt, Grant Martin, Andrew J. McNelly, and Lee Todd. *Surviving the SOC revolution: a guide to platform-based design.* Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[CM08] P. Coussy and A. Morawiec, editors. *High-Level Synthesis : From Algorithm to Digital Circuit.* Spinger, 2008.

[Des09] Embedded System Design, 2009. `http://www.embedded.com/`.

[ea97] F. Balarin et al. *Hardware-Software Co-Design of Embedded Systems: The POLIS Approach.* Kluwer Academic Press, 1997.

[fd09] Linux for device, 2009. `http://www.linuxfordevices.com/`.

[ffeL09] Open Embedded: framework for embedded Linux, 2009. `http://wiki.openembedded.net`.

[fS09] International Technology Roadmap for Semiconductors, 2009. `http://www.itrs.net/`.

[Fur00] Steve Furber. *ARM System-on-chip Architecture*. Addison Wesley, 2000.

[GAGS09] D.D. Gajski, S. Abdi, A. Gerstlauer, and G. Schirner. *Embedded System Design: Modeling, Synthesis and Verification*. Kluwer Academic Publishers, 2009.

[JW04] Ahmed Jerraya and Wayne Wolf. *Multiprocessor Systems-on-Chips (The Morgan Kaufmann Series in Systems on Silicon)*. Morgan Kaufmann, 2004.

[Row04] Chris Rowen. *Engineering the complex SOC : fast, flexible design with configurable processors*. Prentice-Hall Press, 2004.

[Tur02] Jim Turley. *The essential guide to semiconductors*. Prentice Hall Press, 2002.

[Wol02] Wayne Wolf. *Modern VLSI design: system-on-chip design*. Prentice Hall Press, 2002.