

Fast Prototyping: a system design flow applied to a complex System-On-Chip multiprocessor design

Benoit Clement, +33-476-584-238
benoit.clement@st.com

Richard Hersemeule, +33-476-584-145
richard.hersemeule@st.com
STMicroelectronics
5bis, Chemin de la Dhuy
F-38240 Meylan France

Etienne Lantreibecq
etienne.lantreibecq@st.com
STMicroelectronics
850 rue Jean Monnet - BP 16
F-38926 Crolles Cedex, France

Bernard Ramanadin
bernard.ramanadin@st.com
STMicroelectronics
STAR US RnD c/o Hitachi HMSI
179, E. Tasman dr., San Jose, CA 95134, USA

Pierre Coulomb, +33-476-584-028
pierre.coulomb@st.com

Francois Pogodalla, +33-476-584-037
francois.pogodalla@st.com
STMicroelectronics
5bis, Chemin de la Dhuy
F-38240 Meylan France

1. ABSTRACT

This paper describes a new design flow that significantly reduces time-to-market for highly complex multiprocessor-based System-On-Chip designs. This flow, called Fast Prototyping, enables concurrent hardware and software development, early verification and productive re-use of intellectual property. We describe how using this innovative system design flow, that combines different technologies, such as C modeling, emulation, hard Virtual Component re-use and CoWare N2C™, we achieve better productivity on a multi-processor SOC design.

1.1 Keywords

System design, Hardware/Software (HW/SW) co-design, Virtual Component (VC) re-use, Fast Prototyping, system verification, system modeling.

2. INTRODUCTION

The trend towards Systems-On-Chip (SOC), now widely promoted in the industry, creates many issues that the semiconductor manufacturers have to face in order to, while improving the time-to-market, provide their customers with the best quality.

The productive re-use of Virtual Components (VC) is crucial since gate counts continue to escalate while market windows continue to shrink.

Modern SOCs require a significant amount of software (SW), that is not developed until a hardware (HW) prototype is available. This is a significant obstacle for First Time System Success, hence fast time-to-market. Moreover these SOCs are very often developed in collaboration with the customer, which generates the need to deliver early material

for target system integration before silicon.

Because these SOCs perform numerous functions and embed complex protocols, they are particularly difficult to verify [3,5], also the architecture generally requires tuning. Verifying the system at various abstraction levels is crucial for the design process to progress without costly iterations.

At STMicroelectronics a new methodology has been developed that addresses these issues:

- integration of various tools for true re-use of VCs
- generation of virtual prototypes at different abstraction levels, enabling the concurrent HW/SW development
- verification at different levels of abstraction and even where blocks are modeled at different abstraction levels
- any entry point, from full behavioral system modeling, HW/SW partitioning to hybrid RTL/C/gates prototyping

3. A SOC CHALLENGE

3.1 The CP4 SOC

CP4 is a 600k gates SOC targeted at consumer application. It is a multiprocessing system, containing mainly 3 parts: a high performance DSP cell, a microcontroller cell, and VCs designed and provided by the customer. Figure 1. shows a simplified block diagram of CP4.

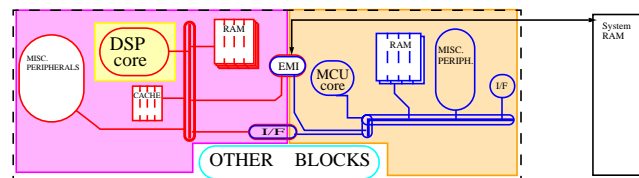


Figure 1. CP4 block diagram

The DSP and its peripherals are new designs, that are being conducted for the CP4 project. Hence we have there a new core development, alongside with the peripherals development (interrupt controller, DMAs, caches,...).

The microcontroller cell is based on an existing proprietary 32-bits core from STMicroelectronics. The majority of the microcontroller cell is direct reuse.

A software and an application team is working in parallel with the design team to produce tools, drivers and software application libraries.

Permission to make digital/hardcopy of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 99, New Orleans, Louisiana
(c) 1999 ACM 1-58113-109-7/99/06..\$5.00

3.2 Product design requirements

The requirements identified in this project can be split in 4 categories:

- *architectural & microarchitectural refinements*: in this development, the gross architecture and microarchitecture are already frozen, but still need some tools to tune specific parts (cache behaviors, bus priorities,...).
- *reference simulation*: a reference model is needed to support the RTL verification activities
- *VCs integration*: a significant part is direct reuse
- *system prototyping*: the parallelisation of tools, internal application software and customer integration developments put a specific requirement of usable system prototypes early in the design process

4. EXISTING TECHNOLOGIES

There are many technologies available for modeling and design, each with their own merits and drawbacks:

- tools like Cadence Bones™ are difficult to integrate in a design flow. The models produced require intensive works for correlation with the RTL implementation
- data-flow based tools like Synopsys COSSAP™ and Alta SPW™ are very application-specific, and mostly dedicated to DSP-based dataflow architectures.
- C prototyping is a solution that has the problem that a path down to implementation is missing
- RTL simulation or emulation: models are very accurate with the implementation, but the time to get a system model is long, thus delaying the integration work.
- high-level languages such as SDL, although ramping-up rapidly, are still not yet supported by industry-proven flows-to-silicon. Besides, most generally they require a radical change in the design practices, rather than a smooth transition from RTL-based methodologies

Looking inside most of the semiconductor companies today, we find a strong will to broaden VC re-use, specifically at the RT-level. But the vast majority of the VCs in the industry is not available even as an RTL. The profit is not in developing different models (C, BoNES,...) for VCs, but in re-using them. Hence a VC-friendly design flow would have to give the capability to build a system model whatever the available view of a VC (gates, transistors, RTL,...).

Based on these considerations, it is clear that the more traditional technologies for system modeling do not satisfy our requirements, and that the most advanced ones are not directly applicable in a business-driven company. Therefore it was necessary to develop a new methodology.

5. FAST PROTOTYPING

This methodology defines a flow for fast SOC's design. Its main characteristics are:

- focus on producing system prototypes (i.e. models) early in the design flow
- maintain the consistency of the models by keeping a uni-

fied development framework throughout the design process

- provide the designers with a fast turn-around to refine the architecture of the new parts of the system

The Fast Prototyping methodology relies on CoWare N2C™ system design tool as a front-end [2], and on Mentor Graphics SimExpress™ hardware emulation as a back-end. The basic reasons for choosing these technologies include:

- CoWare N2C™ provides, through Register-Transfer-C (RTC) a seamless flow that allows, by successive iterations, to refine a behavioral C description down to a clocked-C which is close to a VHDL RTL (path down to implementation)
- CoWare N2C™ provides a powerful co-simulation engine that allows hybrid prototyping (i.e. prototypes made of C, RTC, RTL), unlike Synopsys Eagle-I™ or Mentor Graphics Seamless™, that only allow ISS/VHDL co-verification
- emulation provides simulation power for VCs that are not available under the form of a high-level view
- emulation is a system functional sign-off platform for a design, hence is a natural ultimate prototype for a SOC integration

The capability of handling hybrid prototypes is key for building early system models. As shown in Figure 2., the design of the functional blocks of a SOC can start at various levels, and although all of them will end up under the form of an RTL or a software library element, they will go through various description levels in the design process.

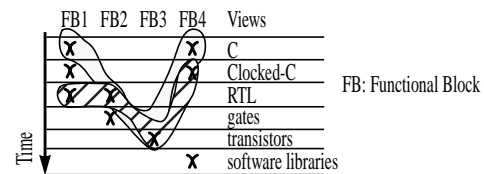


Figure 2. Hybrid prototyping

Hence cosimulating these views becomes essential. Only this capability allows to keep a consistent framework throughout the design process, as indicated in Figure 3.

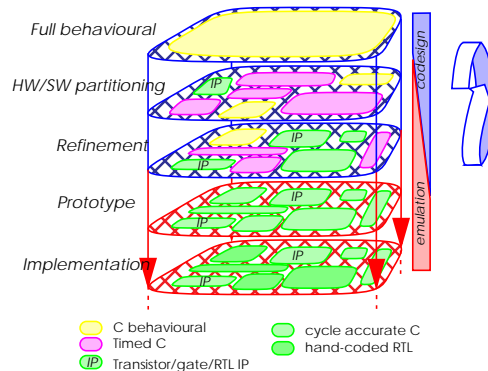


Figure 3. Fast Prototyping framework

If the different levels of description are not interoperable,

then each model produced at a stage raises correctness issues, and more important, all the blocks of the SOC have to be at the same level of description at the same time to build a prototype.

In this design flow, early availability of the Instruction Set Simulator (ISS) is extremely important, as the first model available for a core. We made use of an ST proprietary technology called FlexWare [6] that offers a design framework for fast and semi-automatic generation of tools for a core: ISS, assembler, C compiler, debugger. Using FlexWare, we can get access to an ISS far before any RTL is available for the core being developed.

Another important tool for the introduction of hard VCs is called LayBooL [3]. This tool is a layout abstraction tool. It reads in a transistor netlist, extracts its boolean equations and translates it in an RTL.

Finally, adding emulation to this design framework brings us the VC re-use efficiency, and the powerful simulation capability. The Figure 4. details the overall Fast Prototyping flow.

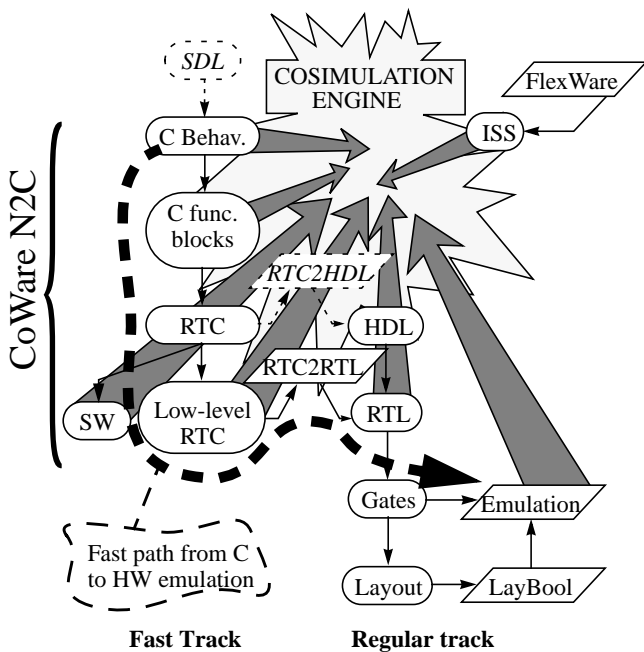


Figure 4. Fast Prototyping flow

6. FAST PROTOTYPING ON CP4

6.1 High performance megacell

The challenge in the DSP megacell is important in that it involves not only peripherals design, but also a DSP core design. Clearly, designing a DSP core is a long effort that generally gates the system modeling effort in traditional approaches, due to late availability of the RTL.

In CP4, we decided to implement the peripherals in RTC for quick development, at a level that is architecturally correct. The models are bit-accurate, and their cycle-behavior is correct at their IO level. This allows to get good simulation

performance (better than RTL), and an architecturally correct model, which can be used to tune the megacell performance thanks to its top-level accurate cycle behavior.

This step of peripherals development is very useful for the architecture refinement, as developing fast models helps sorting out holes in the architecture manual (undefined values, unclear behaviors,...).

The ISS is integrated in the design, and is generated using FlexWare (see 5.FAST PROTOTYPING). The ISS being purely behavioral, and because we chose to build a top-level cycle-accurate model, we assigned it a cycle-behavior through a Bus Functional Model (BFM). This model, written in RTC, expands the instruction execution (fetches, load/store) into bus cycles to connect to the rest of the peripherals.

This prototype of the DSP megacell is described in Figure 5.

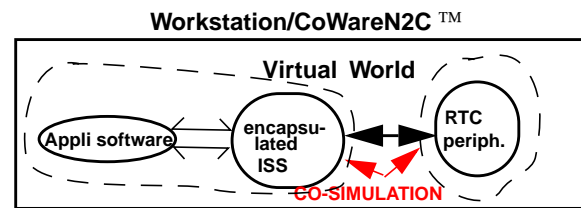


Figure 5. Reference model

This model constitutes the reference simulation platform for this megacell. It is usable as a plug-and-play platform for RTL development. The software application team uses it to start developing code, and also to develop verification tests. This model is called **golden model**.

As the RTL design progresses, whenever a peripheral block is available, its RTC description is removed and replaced with the VHDL. The prototype can then be run to validate the new RTL part, inside the full megacell prototype. This way we can keep block-level, testbench-based verification to a low level, while validating all the blocks in the system. At the end of this plug-and-play process, the model is composed of the ISS, the BFM, possibly the firmware, and all the peripherals in VHDL, and we call it the **hybrid model**. This model enables the peripherals verification before the core RTL is available.

Another important model is the **core verification model**. This model is delivered to the team in charge of designing the core. It is made of all the peripherals in RTC, and the ISS+BFM replaced by the core RTL that is under development. This provides the core design team with a core validation platform that brings simulation performance (peripherals in RTC), without the overhead of peripherals debug. This clearly shows that the design teams in charge of different parts of the project can share the different models, allowing full parallelisation of core, application, SOC design efforts.

Also the external customer, who is the final user of the SOC, is keen on having early models for his software development. We deliver to our customer the golden model

(ISS+BFM+RTC), which brings him simulation performance and architectural accuracy, before any VHDL is available and validated.

Hence this DSP megacell has been developed with a methodology that allowed to build several different prototypes, all consistent, for different purposes.

6.2 MCU megacell

As stated earlier, the MCU megacell is not a new development. Some of the peripherals were developed in VHDL, but for the core itself it was decided to use a hard VC (i.e. a layout view).

Hence the flow we used was based on LayBool (see 5.FAST PROTOTYPING). The RTL generated by LayBool from the transistors netlist, was re-synthesized and mapped onto SimExpress™ emulator [3]. This flow guarantees that the netlist emulated matches exactly the physical implementation, thus tackling the validation problems.

Having the core in the emulator, the peripherals themselves were mapped from their RTL description.

The MCU megacell would not be complete without reusing also the software tools associated with the core, i.e. essentially the software source-level debugger. This debugger interacts with the silicon through the JTAG lines that are used to carry a dedicated debug protocol. We reproduced this setup with the emulator, in a in-circuit experiment. This emulation platform, and the intensive reuse in the MCU megacell, allowed us to build very quickly a prototype that was usable both for some peripherals validation, and for early software development.

6.3 System prototyping

The full system prototype is achieved by integrating the emulator into CoWare N2C™ cosimulation engine, as shown on Figure 6.

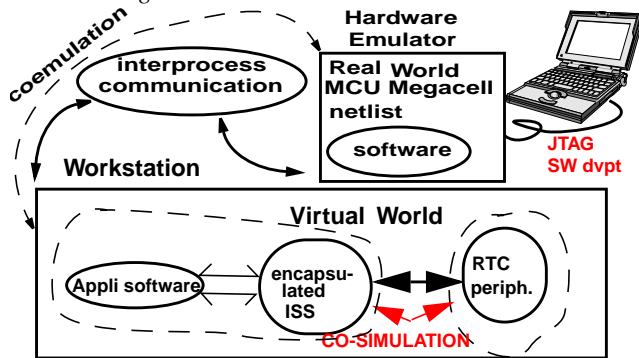


Figure 6. Full system prototype

In a common development with CoWare and Mentor Graphics, we extended CoWare N2C™'s co-simulation capability to the emulator, thanks to the emulation technology which provides a co-emulation library that contains all the functions to control the emulator. It is also remarkable that, although used in a co-emulation mode, the emulator can be at the same time in-circuit (JTAG part).

The power of this full system prototype is that it is available very early. The use of emulation makes the simulation performance higher than that of an RTL simulation-based prototype.

Another strength of this prototype is that it can seamlessly be transformed into a fully emulated prototype, the system functional validation sign-off platform, by integrating the RTL blocks as they become available, and mapping them into the emulator.

Another way of achieving a full emulation prototype, before all the RTL is available, is to use RTC to VHDL translation. We have assisted CoWare in their development of a tool that transforms an RTC description into a synthesizable RTL. This tool is not intended to be a behavioral synthesis tool, but rather a translator. Hence the RTC itself has to be refined precisely before being translated, synthesized and mapped. But still, it provides a faster path to full system emulation than waiting for the full RTL to be ready.

7. RESULTS AND FUTURE WORKS

7.1 Results on CP4 project

Overall, the results of this Fast Prototyping technology were very positive on the project. First the time-to-first-prototype was a breakthrough for that kind of complex SOC:

- the emulation bring-up of the MCU megacell was achieved in 4 weeks, including the software debugger bring-up, thanks to both the re-use strategy and the flexibility of our emulation technology
- the RTC/ISS prototype was built in 4 months, with 2 engineers, a good performance knowing that the SOC architecture was evolving at the same time, and that the VHDL peripherals development required 6 months with 3 persons (hence a productivity gain of 55%)
- hence the full system prototype was achieved between 4 and 5 months. This is remarkable specially in regard to the DSP core VHDL availability: it was available, as in-system integrable RTL, 6 months after this first full system prototype

This shows that a consistent Fast Prototyping methodology, together with a strong VC reuse policy can overcome some of the SOC challenges.

The customers for the different prototypes that we built are numerous, as described earlier. Being able to build these prototypes enables the parallelisation of the work that is required to meet today's time-to-market challenges (core, hardware, software, SOC). Both the application team and the customer started their developments early.

Another important result is due to the consistency of these models. Having this common framework from which to derive prototypes provides lots of cross-checking capabilities, thus maximizing the functional validation of the models.

7.2 Future works

In this project, we have not used the capability to describe

functional blocks at a pure behavioral C level. This has two consequences: first, we did not get the maximum simulation performance; and second, we did not leverage the capabilities of N2C™ for hardware/software partitioning. However, we plan to use future customer projects to validate this capability.

Additionally, we plan to extend the coverage of the flow to the very high-level system capture, possibly by adding other tools as earlier front-ends (SDL-based, for example) [8].

Also, as this project was a pilot project, we minimized the risks by developing concurrently in RTC and VHDL. Our next step is to cancel the VHDL development until the first system prototypes are validated, hence until the architecture itself is satisfactory. To achieve this, we shall rely on both the RTC prototyping, and also on the RTC-to-VHDL translation, which should allow a quick path from RTC to VHDL, hence to emulation. Being able to go so fast to emulation, even before having started any real VHDL development, we believe will improve the system architecture assessment by providing:

- huge cycle emulation power
- significant application profiling
- connectability to external devices

Another important work direction is to integrate these models generated through Fast Prototyping into our test generation flow. ST has developed advanced verification technologies, one of which being a test (assembler programs) generator targeted at functional verification of cores [3,9]. The natural next step is to raise the capabilities of this test generation environment to the system level, by integrating system simulators to generate accurate system reference data.

The area of verification also encompasses formal verification, which we are investigating as a mean to formally prove the equivalence of the descriptions we build throughout the flow (C vs. RTC, RTC vs. VHDL). This will secure the consistency of the different descriptions, hence raise again the quality level of the models produced.

8. CONCLUSION

In this paper we presented the Fast Prototyping system design framework that we put in place within STMicroelectronics. We demonstrated how this environment, based on CoWare N2C™ (system design tool) and Mentor Graphics emulation, was applied to a real industrial project to increase the productivity and the quality of the SOC design effort.

We believe that the capability of producing system prototypes very early in a SOC design flow is a major key of success in today's industry:

- to secure the system architecture work
- to support hardware/software codesign
- to parallelize the different design tasks instead of serializing them

- to distribute usable prototypes to internal and external customers
- to start the functional verification as early as possible

The results achieved so far in terms of prototyping and productivity increase are encouraging enough so that we can consider further developments of this Fast Prototyping technology both horizontally (i.e. by applying it to new SOC projects) and vertically (i.e. by developing the technology itself).

9. ACKNOWLEDGEMENTS

We wish to thank here all groups and people involved in this development, with special attention to CoWare, Gerard Mas, Andre Picco, Marco Carilli, Roberto Fantechi.

10. REFERENCES

- [1] M. Genoe, Alcatel, "Requirements capturing and specification of Systems-on-Chip", MEDEA/ESPRIT conference on HW/SW codesign, 1998
- [2] S. Tsasakou, C. Dre, H. Kharatanasis, A. Birbas, Univ. of Patras/Intracom SA, "Combined assessment of an industrial current practice and CoWare's methodology for the codesign/cosimulation problem", MEDEA/ESPRIT conference on HW/SW codesign, 1998
- [3] C. Berthet, G. Mas, F. Pogodalla & al., STMicroelectronics, "Functional verification methodology of Chameleon processor", 33rd DAC, 1996
- [4] K. Hashmi, A. C. Bruce, "Design and use of a system-level specification and verification methodology", EURO-DAC 95
- [5] J. Monaco, D. Holloway, R. Raina, "Functional verification methodology for the PowerPC 604 microprocessor", 33rd DAC, 1996
- [6] P. Paulin, "A flexible hardware/software development environment and its application to consumer multimedia products designs", CODES/CASHE'98
- [7] A. Sangiovanni-Vincentelli, J. Liu, M. Lajolo, "Software timing analysis using HW/SW cosimulation and instruction set simulator", CODES/CASHE'98
- [8] A.A. Jerraya, J.M. Daveau, G. Marchioro, "hardware/software codesign of an ATM network interface card: a case study", CODES/CASHE'99
- [9] M. Benjamin, D. Geist, A. Hartman, G. Mas, R. Smeets, Y. Wolfsthal, STMicroelectronics and IBM Science and Technology, Haifa Research Lab. "A Study in Coverage-Driven Test Generation", DAC'99