Under the High Patronage of His Majesty The King Mohammed The Sixth

IEEE

EMI

ICM'2001 Proceedings

The 13th International Conference on Microelectronics

Ecole Mohammadia d'ingénieurs
Université Mohammed V-Agdal

October 29-31,2001, Rabat, Morocco

Under the High Patronage of His Majesty the King Mohammed the Sixth

IEEE

EMI

ICM'2001 Proceedings

The 13th International Conference on Microelectronics

Ecole Mohammadia d'Ingénieurs
Université Mohammed V-Agdal

October 29-31, 2001, Rabat, Morocco

Welcome message from

Organizing and Technical Program Committee Chairs

On behalf of the organizing and technical committees, it is our pleasure to welcome you to the thirteenth International Conference on Microelectronics (ICM'2001) in the Kingdom of Morocco. This conference is organized under the High Patronage of His Majesty the King Mohammed the Sixth. We wish you a pleasant stay in Morocco and hope that this conference will be of great benefit for you.

This year's conference is the thirteenth ICM in a series that include conferences held in Algeria, Syria, Egypt (twice), Tunisia, Turkey, Saudi Arabia, Malaysia, Indonesia, Kuwait and Iran. The next edition, ICM'2002, will be held in Lebanon in December.

The Technical Program of this year offers excellent refereed contributed papers as well as invited papers. Fifty nine papers have been chosen by the Technical Program from over ninety submitted papers. The international character of the conference is reflected in the fact that nineteen countries from all the world are included in the program. The authors of papers are from Canada, France, Egypt, Tunisia, Turkey, Iran, Germany, Italy, Lebanon, Russia, Saudi Arabia, UK, Finland, Poland, Spain, Switzerland, USA, Japan and Morocco.

The sessions cover a broad range of technical subjects related to new development in solid state technology, devices, circuits and systems. Two sessions will take place simultaneously. Six Invited talks, given by well-known scientists, are included in the program. A panel discussion on future of microelectronics industries is planed in the closing ceremony. We anticipate a strong interaction throughout the conference between all participants.

We appreciate the support of sponsors, listed in the next page, of this year's conference. Also, we would like, to take this opportunity, to thank all the authors who submitted papers, the reviewers, speakers, session chairs and panelists and all the members of ICM'2001 committees, whose contribution is the cause of the success of the Technical Program of ICM'2001.

Pr. R. Khalid
EMI, Mohammed V University

Pr. A. Bennouna
EMI, Mohammed V University

Pr. M. Eleuldj
EMI, Mohammed V University

Pr. M. I. Elmasry
University of Waterloo

ICM'2001 Sponsors

Co-sponsor

IEEE : Institute of Electrical and Electronics Engineers

Official Sponsor

ANRT : Agence Nationale de Réglementation des Télécommunications

Partner Sponsors

Ministère de l'Enseignement supérieur et de la Recherche scientifique

Secrétariat d'Etat auprès du ministre de l'Enseignement supérieur et de la Recherche scientifique, chargé de la Recherche scientifique

SEPTI : Secrétariat d'Etat auprès du premier ministre chargé de la Poste et des Technologies de l'Information et de Communications

CNRST : Centre National de Coordination et de Planification de la Recherche Scientifique et Technique

Rectorat Université Mohammed V-Agdal

Sicotel

Silvaco Data Systems

STMicroelectronics

Mentor Graphics

Commune de Agdal/Ryad

ONEP : Office National de l'Eau Potable

Barid El Maghreb

ICM'2001 Committees

General chair

Pr. R. Khalid, EMI, Rabat, Morocco

Conference chair

Pr. M. Eleuldj, EMI, Rabat, Morocco

Technical program Co-chair

Pr. M. I. Elmasry, University of Waterloo, Canada
Pr. A. Bennouna, EMI, Rabat, Morocco

Steering Committee

Pr. M. I. Elmasry, University of Waterloo, Canada
Pr. M. Kamel, University of Waterloo, Canada
Pr. M. Sawan, Ecole Polytechnique de Montreal, Canada

Organizing Committee

Publication :	Pr. Z. Guenoun, EMI, Rabat, Morocco
Publicity :	Dr. N. Bennis, ENSET, Rabat, Morocco
International affairs :	Pr. , T. Bennani, SEPTI, Rabat, Morocco
Industrial liaison :	Mr. P. Foulon, ST Microelectronics, Rabat, Morocco
	Mr. S. Avoski Omtech, Rabat, Morocco
	Dr. M. Mohssine, , EMI, Rabat, Morocco
Academic liaison :	Pr. L. Zenkouar, EMI, Rabat, Morocco
	Pr. A. Kherras, EMI, Rabat, Morocco
Logistic :	Pr. M. Maaroufi, EMI, Rabat, Morocco
	Pr. H. Ammor, EMI, Rabat, Morocco
Financial affairs :	Pr. L. Bahi, EMI, Rabat, Morocco

Technical Committee

A. Kherras, EMI, Rabat, Morocco
E.M. Aboulhamid, University of Montreal, Canada
L. Zenkouar, EMI, Rabat, Morocco
H. Ammor, EMI, Rabat, Morocco
M. Drissi, INSA, Rennes, France
M. Eleuldj, EMI, Rabat, Morocco
M. Kamel, University of Waterloo, Canada
M. Maaroufi, EMI, Rabat, Morocco
M. Mohssine, EMI, Rabat, Morocco
M. Sawan, Ecole Polytechnique de Montreal, Canada
N. Bennis, ENSET, Rabat, Morocco
T. Bennani, SEPTI, Rabat, Morocco
Z. Guenoun, EMI, Rabat, Morocco

ICM'2001 Secretary

A. Lahmili
J. Bouchiah
I. Bayine
H. Arji
H. Messoudi
L. Benmouya

The Organizing committee would like to express cordial gratitude to the staff and student of Ecole Mohammadia d'Ingénieurs of Mohammed V University.

Table of contents

INVITED PAPERS

MICROELECTRONICS TECHNOLOGY

DESIGN AND APPLICATION OF INTEGRATED CIRCUITS AND SYSTEMS

COMPUTER-AIDED DESIGN FOR MICROELECTRONICS

Invited speakers

New Hardware/Software Design Methodologies

E.Mostapha Aboulhamid
Université de Montréal
aboulham@iro.umontreal.ca

Abstract

This paper describes the use of the multi-paradigms aspects of SystemC to develop hardware libraries, to reuse deigns, to accelerate simulation, and to allow efficient design space exploration both at the RTL and architectural levels. Concepts of commonality and variation are used to compare SystemC capabilities to those of VHDL.

1. Introduction

SystemC is a modeling platform consisting of C++ class libraries and a simulation kernel for design at the system-behavioral and register-transfer-levels. Designers create models using SystemC and standard ANSI C++[1]. Different models of computation and design methodologies may be used in conjunction with SystemC. The design libraries and models needed to support these specific design methodologies are considered to be separate from the SystemC core language standard. This work may be considered as a design methodology for RTL and architectural models.

By its nature, SystemC inherits the capabilities of C++, like the support of multiple paradigms: classes, overloaded functions, templates, modules, ordinary procedural programming, and others. This richness makes some hardware designers very skeptical due to the freedom that a designer has, letting him develop models that are very difficult to synthesize or comprehend. Our objective in this paper is to show that this multiparadigm environment is rather a blessing than a curse. We will show that we will be able to develop more resilient specifications and models, express different behaviors and view of the same "entity", facilitating design exploration and ease specification and development of libraries.

We will illustrate this by comparing SystemC capabilities to what VHDL can express. VHDL lack of some useful paradigms has been already describe as early as 1991 as summarized in [2]. Early attempts tried to augment VHDL with other constructs, but since systems have more and more programmable components, designers are leaning toward a common language for hardware and software modeling. We will describe the system and RTL design capabilities of SystemC (like Channels and interfaces). System level capabilities using C++ and/or SystemC are well described elsewhere [3, 4]. This may serve also as guideline for VHDL designers to comprehend and probably adopt the SystemC methodology. In our comparison we will choose the concept of Commonality and variation developed by Coplien [5, 6] to compare the two environments.

In this talk, we will describe the desirable characteristics of a design environment; introduce the commonality and variation concepts; deal with design reuse and hardware libraries; and discuss configuration and variation issues.

2. Characteristics of a Design Environment

In our opinion, the desirable characteristics a hardware designer is looking for are:

•Reuse: hierarchy is one of the mechanisms that allow reuse. Structural hierarchy is available in VHDL but not behavior hierarchy, as it will be illustrated later.

•Refinement and automatic path to synthesis

•Families of design: often a design house will be interested in developing a family of products such as processors, controllers, and busses. All express some commonality but inside the same familty it is desirable to be able to express variation by changing parameters or other mechanisms.

•Express design constraints related to timing, testability number of processors etc.

•Development of methodologies that go beyond the basic concepts either of SystemC or VHDL, and this is possible using Design patterns methodlogy for example.

•Development of powerful test-benches, it seems that the time spent on test bench development may take as much as 70% of the time allotted to a project, we think that by its C++ nature, SystemC opens the door to the reutilization of smart code like STL libraries[7], windowing libraries[8] to develop a very hign level test benches.

•During the design exploration phase it would be very helpful to break the cycle, modeling, analysis and elaboration, simulation (execution) by allowing online configuaration of a model leading to more freedom in comparing results, a shorter time for the analyzer to evaluate its design, The cycle would be: model, compile, and examine the execution phase by dynamically reconfiguring the model. If the module are fully interfaced, implemented and debugged, their construction could be controlled via a GUI at execution time. Once satisfied, a smart synthesis environment can take the snapshot of our current design and continue with synthesis.

•Link to the outside world and extension of the capabilities of the environment: this is almost New Hardware/Software Design Methodologies inexistent with VHDL. Since SystemC is an open source environment we have seen some efforts in developing other tools around the SystemC core as well as methodologies to have interoperable environments [9].

•Fast simulation during design space exploration: preliminary results show that SystemC is 10 times faster than VHDL for the same level of abstraction.

•Minimize the number of environments and languages to deal with: in the case of systems containing both hardware and software, SystemC seems to have an advantage.

•Avoid pitfalls: Both VHDL and SystemC have their disorienting characteristics for a new designer. In VHDL, there are always two ways of expressing the same concept: concurrent and sequential. Some constructs seem at the first glance very similar, like the use of attributes S'stable and S'event; however the use of S'Stable is more time consuming during simulation.

•Have delay models that fit the need of the different levels of abstraction. Delay models are very restrictive in VHDL (inertial, delta and transport delays). These seem sufficient for describing hardware components. However they are very inefficient when developing test-benches. Test-benches and abstract architectures necessitate FIFOs, queues and stacks for example. This is very possible in SystemC especially with channels.
•Documenting tools for the models: the models would be more usable and more resilient, if well documented. In this case, SystemC seems to have a big advantage compared to VHDL. It is open source
and the models are developed using the same paradigms as those used to develop SystemC itself. GNU tools like Doxygen[10] may be used to both document SystemC as well as the models developed using SystemC [11]. Graphical documentation can also be easily obtained using UML [12].

3. Commonality and Variation

Similarly to Software engineers, Hardware designers are under pressure to develop new system versions in less time. In the case of software systems, The authors [5] show the benefits of explicitly identifying the common and variable aspects of the different versions of a system. We will illustrate the ability of SystemC, in comparison to VHDL, to express commonality as well as variation in modeling of hardware systems. VHDL [13] allows the expression of commonality by what is called design units. A design unit is a VHDL construct that may be independently analyzed and inserted in a design library. These design units are:
•Entity declaration: describes the interface view of a component (like a Data Book description). It is implementation independent.
•Architecture body: describes an implementation of an entity (like a single schematic diagram). A single interface may have alternative architectures.
•Package (declaration and body): contains information common to many design units. This information consists of functions, types, signals, and constants. It hides details, simplifies design, and may invoke other packages.
•Configuration: relates local entity and architecture references to actual units in libraries (like a parts reference list).
Commonality can also expressed by generate statements and generic constants for regular structures like a ripple carry adder or an interconnection network.
Variation is obtained by configuration, which allow the designer to choose an architecture among many others during design space exploration; by giving a specific value to a generic parameter; by overloading functions and subprograms in coordination with packages allowing the reuse of models even if we change the basic data types, like going from a bit type to a 9-valued standard logic. Commonality between processes is very limited except if we choose very complex ways like concurrent procedure calls rendering models quite cryptic.
In SystemC, at the system level, commonality is expressed by abstract classes such as interfaces, then variation is encapsulated from interfaces with a possibility of multiple inheritance. Commonality can also expressed as use of design patterns [14] and other means as described later.

4. Conclusion

The objective of this talk was to describe the use of the multi-paradigms aspects of SystemC to develop hardware libraries, to reuse deigns, to accelerate simulation, and to allow efficient design space exploration both at the RTL and architectural levels. Concepts of commonality and variation are used to compare SystemC capabilities to those of VHDL. The use of the presented methodology will allow the development of resilient and well documents models, it may also help in developing efficient test-benches.

5. References

[1] Open SystemC Initiative (OSCI), Functional Specification for SystemC 2.0, http://www.systemc.org, 2001.
[2] S. Swamy, A. Molin, and B. Covnot, "OO-VHDL: Object-Oriented Extensions to VHDL," *Computer*, vol. 28, pp. 18-26, October 1995.
[3] D. Verkest, J. Kunkel, and F. Schirrmeister, "System Level Design using C++," Proceedings of Design, Automation and Test in Europe, Paris, France, 27 - 30 March, 2000. [4] Open SystemC Initiative (OSCI), SystemC, http://www.systemc.org, 1999-2001.
[5] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," vol. 15, pp. 37-45, November/December 1998.
[6] J. Coplien, *Multi-Paradigm Design for C++*. Reading, MA: Addison-Wesley, 1999.
[7] Silicon Graphics Computer Systems Inc., Standard Template Library Programmer's Guide, http://www.sgi.com/tech/stl/, 1993-2001.
[8] Trolltech AS, Qt On-Line Reference Documentation, http://doc.trolltech.com, 1996-2001.
[9] L. Charest, M. Reid, E. M. Aboulhamid, and G. Bois, "A Methodology for Interfacing Open Source SystemC with a Third Party Software, "Proceedings of Design Automation and Test in Europe Conference & Exhibition, Munich, Germany, 13-16 March, 2001, pp. 16-20.
[10] D. van Heesch, Doxygen, http://www.stack.nl/~dimitri/doxygen/index.html, 1997-2001.
[11] L. Charest, SystemC documentation, http://www.iro.umontreal.ca/~chareslu/, 2000-2001.
[12] G. Booch, I. Jacobson, and J. Rumbaugh, *The Unified Modeling Language User Guide 1*/e:Addison Wesley, 1999.
[13] *IEEE Standard VHDL Language Reference Manual*, 1076,2000 ed: IEEE, 2000.
[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*: Addison Wesley, 1994

Biography

El Mostapha Aboulhamid was born in Midelt Morocco, studied in Lycée Tarik, Azrou, then in Lycée Lyautey, Casablanca. He received his Engineer degree from Institut National Polytechnique (Grenoble France) in 1974. He returned to Morocco from 1974 to 1978 where he participated to the formation of the first ingénieurs analystes in Morocco (INSEA). He received his Master and Ph.D. degree from Université de Montréal, in 1979 and 1985 respectively. He was Professeur at Université du Québec à Montréal from 1981 to 1985, then from 1985 till now Professeur at Université de Montréal. He participated actively to the introduction of new hardware design methodologies in Canada, first the design for testability through Canadian Microelectronics Corporation, then by the introduction of VHDL. He gave courses related to hardware modeling and synthesis in different countries, including France, Czechoslovakia and Morocco and to different industries in Canada: Nortel, Miranda, Spar Aerospace etc.

His scientific interests are modeling, synthesis and verification of hardware/software systems. He is member of ACM and IEEE. He has collaborated with industries in Canada, and is member of Micronet Center of Excellence. He has been on the board or the executive committee of GRIAO (renamed ReSMiQ) since its creation; he was also the director of that Center for three years. He has been on the program committee or organizing committee of different International Conferences. He is currently the General Chair of the International Symposium on System Synthesis, which will be held in Japan in October 2002.

MULTIMEDIA MULTI-CHALLENGES

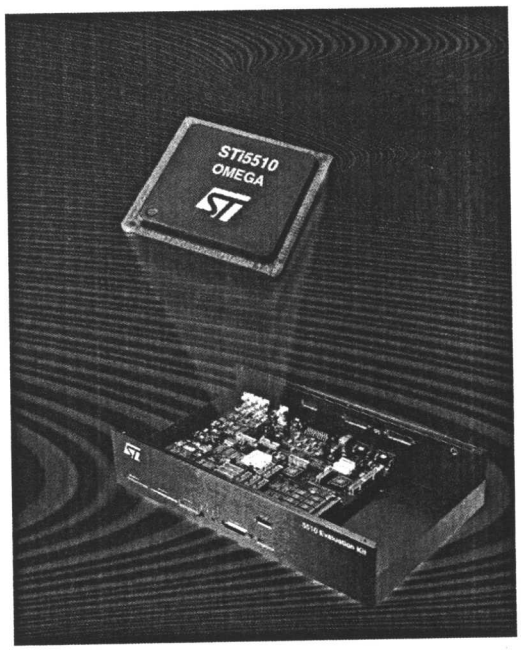
Jean-Marc CHATEAU
STMicroelectronics
Director of Design, Consumer and Microcontroller Groups
jean-marc.chateau@st.com

Multimedia is now one of the most popular expressions when talking about high tech. However the e-technology is now somewhat taking over the fashion of the multimedia era. Is there a shared definition of multimedia? It's something about images and sounds reaching you by some electronic technology means. Multimedia is the easy answer to the question: what is it? Especially when you are facing something that is a mix of the 3 "old " segments: computer, consumer and telecommunication. Many electronic corporations have built their organization around this natural split of the market; this is true for electronics manufacturing and components. Moreover, this multimedia is climbing into the car mixing up with the 4th segment, which was automotive. Talking about microelectronics, which is the domain we are all interested in today, this means the biggest challenge ever. Not only do we have to rethink the way we are organized and who is our customer, but also we must be able to mix all types of functions related to these mixed applications. Computer was advanced CMOS, high clock frequency, large memories, very short lifetime, not necessarily low cost. Consumer was high volume, very price-sensitive, mixed signal CMOS.

Telecom was driven standards, DSPs and RF. Automotive meant a very long qualification time, price- sensitive, smart power, heavy constraints on voltage and temperature. Now we have to mix all the constraints at a crazy speed when standards are still moving, features are increasing every six months and prices are dragged to the bottom.

Following the model of the PC which stands for *personal* computer, the phone, the TV and the car, which were shared by a family, are now being customized to be differentiated and finally personal. Apart from the effect on volume, the emphasis has now moved more onto the personal service than the equipment itself. Phones, PCs and set top boxes are given away for a long-term subscription to a service: so much for the electronics industry that must follow up on the escalation of the services, reducing the cost at the same time.

Multimedia has forced microelectronics to change many paradigms. Semiconductor companies have been obliged to integrate complete systems on chips and to develop software, leaving most of the traditional silicon design problems to the EDA community at least for the digital portions.



You cannot sell a multimedia chip without providing a reference demo board which looks close to the end product, the software drivers, and a friendly development and debug environment for your customer to add their own final touch. This final layer (on top) is generally more or less driven by the service provider. So OEMs are more and more squeezed between the System on Silicon and the Service, with little room for customization.

Looking at the roofs of Rabat, blooming with satellite dishes, I found my first example with which to illustrate my talk. The market of digital set top boxes has been flourishing for the last two years for both cable and satellite transmission, reaching 30M units per year. The boxes are more and more interactive proposing web TV access, pause and play, individual program guides and recording, phone over cable and games. The next generation will be a kind

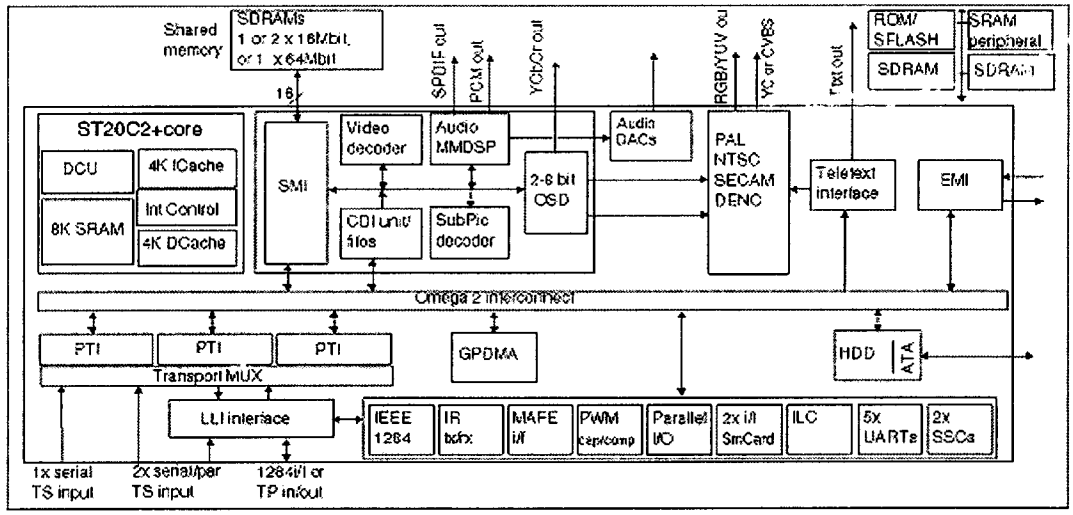
of home gateway between the telecom media external to the home and the different terminals inside the home. But it could well be the display that has the brain to decode all the services without the need for a box to be put on top, leaving only the physical interconnection problems to the gateway box. Who knows?

Basic Pay TV systems require now no more than two systems on chip: the tuner and demodulator in one chip; the transport demultiplexer and the decoders in another chip. All you need to add are the flash for software, the SDRAM for image and sound data buffering, an audio DAC and a MAFE modem. The connections to the satellite dish, to the TV, the VCR, the optional hard disk for pause and replay, the remote control and the smart card readers are included. These chips are today in 0.18 micron CMOS. The next generation will include the MAFE and the audio DACs. Front end and back end chips could be integrated into one; what refrains the merge is the difficulty to choose how many channels of which type we want to integrate.

Today, a typical decoder for satellite TV contains:

- a 32bits controller running at 120MHz, with its data and instruction caches, an additional SRAM and a diagnostic controller
- a shared memory interface 16bits wide to a 64Mbits SDRAM, running at 120MHz
- a programmable external memory interface to connect RAMs, ROMs, Flash or an optional external micro
- From 1 to 3 programmable transport interfaces to demultiplex multiple DVB or Direct TV streams, several descramblers to support multiple standards
- an MPEG-2 video decoder
- a graphic engine doing blittering, alpha blending, antialiasing, antiflutter and antiflicker filtering using CLUT entries and link list control
- a compositor for 5 display planes
- a hard disk interface (ATA-5 compliant)
- an audio decoder supporting MPEG1, MPEG2, Dolby Digital, MP3 and Dolby Prologic
- a PAL/NTSC/Secam encoder
- communication peripheral such as UARTs, smart card interfaces, I2C, PWMs DMAs, IR blaster, IEEE1284, low power controller

The chip integrates 7 million transistors and it is packaged in a 35x35 PBGA388.



The design of this kind of chip requires several hundred of skilled engineers. It starts with the architecture, follows with the RTL, the gates and the layout, not forgetting the evaluation board, the software drivers and the test programs. As the market requires several variants of decoders every year, it is not possible to run independent projects. Looking at the functions (or IPs) needed to do the decoding of a DVD, you will find a very similar list. Digital TV chassis are also embedding similar functions. Resources and skills are spread over the planet in multi-site projects; there is a natural need to group the talents into centers of competence dedicated to a given IP. This is IP Reuse. But IP Reuse doesn't exist; each product is calling for some level of customization. IP Reuse means IP easy to modify and integrate. It requires discipline all over a corporation to establish IP Reuse standards in the spirit of VSIA (Virtual Socket Interchange Alliance). Hardware description is now a software file to be architectured and documented for reuse. The way Verilog or VHDL descriptions are written must ensure readability

by the CAD flows in use all over the design community, taking into account not only the possibility of synthesizing, but also the constraints from hierarchical timing and signal integrity tools, DFT strategy, clock strategy, formal proof etc....

In the past, we had librarians, taking care of standard cells and memories. The new librarian is managing 1M gates IPs generating variants along their own roadmap like dashboards, and engines have their own life independent of the car models they will integrate.

Concurrent engineering at all levels is the rule to be ready to assemble quickly the right combination for the right customer at the right time. The product team is now more an assembly team than a design team. But they face the Jigsaw syndrome. We still have many places where IP Reuse rules are not applied. The result is you receive a lot of pieces. Not all the pieces are needed. You don't know where the right pieces are. Lots of pieces look similar, but only one will fit correctly. You can only be sure we have the right pieces when we get near the end.

ST Microlectronics has been a pioneer defining IP Reuse as a corporate policy since April 1997. The Blue Book gathering all the rules has been sent to all designers as early as 1998.

Rules to develop the data are not enough. We are talking about several hundreds of Gygabytes to be shared across several continents. You need:

- a mechanism to provide controlled release of high quality design data to other groups
- a configuration management system and detailed audit trail
- a mechanism for referencing the released work of others
- a mechanism for transferring databases, maintaining consistency between many sites.

On top of data management you also have to install a bug tracking system able to work all over the dimensions of System On Chip. A bug discovered by a cable TV subscriber at his home must be able to propagate its way down to the faulty variant of a hardware or software IP, and then up to all users of this version of IP. It means, for semiconductor companies, deploying a bug entry system to all field engineers propagating to all the possible causes of failure: a customer software, an API, a middleware port, microcode or a silicon block ; each developer is reached step by step until the bug is confirmed .

The design process of a multimedia system on a chip can benefit from the IP Reuse approach, careful data management, and bug tracking. It is still not enough to achieve the fast turn around market for which each product characteristic is frozen at the last minute in order to better match the end-user expectations. Design is a relatively fast operation for such systems if you consider the effort required by verification. In order to validate all the possible combinations of operations offered by a rich set of functions, you need several people working several months in parallel on the first packaged samples plugged onto the reference board. You cannot afford to have many reworks. This means you have to do all the system validation at an early stage of the design, and across the different abstraction levels. Design productivity comes from platform-based design. Let's define it as a stable microcontroller-based architecture that can be rapidly extended, customized for a range of applications, and delivered to product groups for quick deployment. With commercial system level design tools such as N2C from Coware used by ST, you are able to describe pieces of your system, to mix them even at different abstraction levels, to synthesize interfaces to co-simulate the hardware IPs and the code on the processor. It helps to separate the communication interfaces from the functionality of the IP blocks. When designing a platform, we generate the appropriate hardware interfaces using the interface synthesis capability of N2C, as well as the software interfaces to the hardware (for instance, the memory maps for registers and the software drivers). This makes swapping out one piece of IP an easier process and also allows designers to evaluate the potential of one piece of IP in a design versus another.

IP Reuse for Multimedia Systems doesn't make sense without a common and standardized backbone to plug the IPs. VSIA has made recommendations for On Chip Bus; the architecture of STBus takes them into account. The toolkit is composed of:

- Parameterized Interconnect crossbars nodes (arbiters and routers; 3 types depending on complexity)
- Buffer used as a FIFO between two nodes
- Converter (size and type)
- IP interfaces (either initiator or target or both)

The capture can start at the architecture level Parametrizing the STBus backbone for the requirements of your multimedia application. Simulating the behavior of both hardware and software will bring you, step by step, to the level of optimization you wish. Then VHDL code will be synthesized automatically for the STBus as well as for the peripherals you have described to a bit and cycle accurate level.

To illustrate this talk, let's take an example: Pocket Multimedia. We know the PDAs of today. Nobody knows what will be the successful device of tomorrow. There are so