

“How to easily explore hardware/software system configuration with Electronic System Level”



Ecole Polytechnique de Montréal
c/o Space Codesign – Dpt. GI #M4115
C.P. 6079 – Succ. C.-V.
Montréal, QC, H3C 3A7 CANADA
<http://www.spacecodesign.com>

This paper is the first of a series regarding the Space Codesign technologies. It presents the effectiveness of Electronic System Level design and how the Space Codesign technology let you efficiently benefit from these concepts.

Nowadays, technology innovations cause tremendous effects on circuit complexity; the hundred of millions of on-chip transistors needing to be shaped into increasing system components, crossed with growing customer needs, not mentioning the narrowing of the time-to-market window, leads to the necessity of rethinking and remodeling the way engineers design these circuits.

One exposed solution, Electronic System Level, relieves the pressure for designing these systems by raising the level of abstraction for system modeling, which handles both hardware and software components for describing communications between these components and capture system constraints. For instance, Transactional Level Modeling (TLM) lessens the design characteristics for hardware-software system design, improving designer productivity. Rather than dealing with abundant micro-architecture details, usually the case when dealing with RTL models, designers using a TLM can rapidly design a macro-architecture meeting expected performance requirements.

The **Space Codesign™** technology brings together state-of-the-art features including electronic system design and simulation, architecture selection, component partitioning and mapping, debugging, monitoring and system generation into a single electronic system level development environment.

Electronic System Level; Easier than you Think

Electronic System Level (ESL) [1] design methodologies progressively tend to be part of our vocabulary when talking about Electronic Design Automation, FPGA Design, Programmable Devices, Embedded System and even ASIC Design. This promising and fairly new design trail will change the design process of embedded systems within development teams. Several advantages assert this statement:

1. Reduced development costs and turnaround time;

By extensive reuse of components, performing simulation techniques, and high level specification capture, we can extensively reduce the total design time. ESL presents attractive and adjustable tradeoffs between the speed and the accuracy of system simulations.

2.1.Reduced needs for custom hardware in your products;

2.2.Performance criteria acknowledged even before hardware is ready;

2.3.Enhanced system flexibility;

You can explore the introduction of software components into your designs and measure the overall effect at a reliable simulation stage. This allows validation of the performance criteria very early in your design stage. Also, introducing software into your design will guarantee flexibility for late updates, bug fixing and new features while keeping the same physical platform.

3. Reduced uncertainties as the final system is integrated;

Managing an ESL solution within your design process increases the level of confidence of your final product. Hardware and software designs are overlapped in simulation, reducing the possibility of unwanted “surprises” generally happening in integration phases.

With ESL designs, exploration is facilitated and automated, letting you spend your limited design time on the deliverable application rather than on architecture considerations. Higher levels of abstraction enhance the speed of simulation, allowing you to validate your specifications (by launching them as executables) on your host machines. All architecture components are simulated to speedup behaviors while taking into account accuracy and timing concerns.

Advantages of using a single & standardized language

The Space Codesign technology handles the complexity of the SystemC v2.1 language (now IEEE 1666 standard) while letting you focus on the code you need to deliver. In short words, the Space Codesign environment greatly simplifies the difficulties related to using the SystemC language features, while keeping an intuitive and especially simplified user application interface. Readers who may be interested in learning about SystemC v2.1.1 may refer to [2] and [3].

The Space Codesign technology brings a unified programming language for hardware and software designs. This significantly reduces the effort required for system integration, hardware/software debugging and system description, as you do not have to deal with heterogeneous simulators, different design environment or language translators.

Virtual platforms: simulation brings confidence

The concept of platform-based design and more specifically the use of virtual platforms is the key to simplified and intelligent system designs, embedded systems and systems-on-chips. Platform-based design allows extensive reuse of components, which reduces the time-to-market for product first releases.

One can enhance the range of system configuration possibilities by making use of virtual platforms [4]. By recreating – in simulation – the full architectural environment of a system including the system’s algorithms themselves, we initiate the exploration of the design space; i.e. we try different design possibilities without the need of a physical prototypes, and with less design time invested. The final outcome might reveal usually unexplored solutions.

Design teams usually experiment a full validation of their specifications by creating C/C++/Java models of their specifications instead of directly working with the final target platform. Working at this higher level of abstraction speeds up the validation of the specifications, as developers do not worry about micro-architectural details such as clock rates, exact pin outs, line signals, or bus protocol designs. Electronic System Level pushes forward the use of virtual platforms by intertwining the application model spoken above with a replica of the architecture itself, meaning a full simulation of this architecture executing the application. However, simulating such systems and architectures can be a very lengthy process. Several levels of abstraction are used to accelerate the simulations on one hand, and to bring accurate results (timings behaviors) on the other.

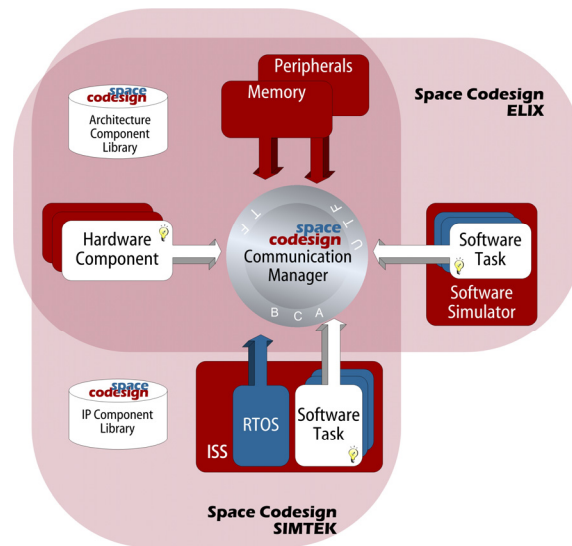
The Space Codesign environment offers to model such virtual platforms. One can instantiate predefined virtual platforms ready for use and simulation, or create a custom platform by selecting different components from our library.

Summary Technical Description of the Space Codesign Technology

The Space Codesign technology allows architecture exploration focusing on a refinement design flow for hardware-software embedded products development. The creation of C/C++/SystemC user blocks, user Intellectual Properties (IP), user modules or more simply modules, takes part of coding guideline that helps in creating transparent module, which can be swapped between hardware and software partitions of a system. This important feature creates a large choice of possible system design solutions (partitions). The Space Codesign technology lets the designer simulate each partition, to find the optimum one. This type of

modeling sometimes called Transactional Level Model (TLM) and can fasten simulation speed of up to 100 times compared to equivalent RTL model [5].

The Space Codesign technology is split into two levels each of them representing different abstraction levels that handle different features and functionalities to help you design and validate professionally your systems.



The technology integrates a wide variety of wonderful features for System Design. Each module is designed and developed based on an efficient application programming interface (API) that opens user code to the Space Codesign functionality.

The technology presents a thorough support for software-based solutions: software modules can be executed on different types of simulators to reproduce software behaviors and with true Real Time Operating System (RTOS) integration.

How to create a platform?

The Space Codesign technology, starting from the SpaceStudio (see below) tool suite, guides the user to the instantiation of custom modules or predefined IP blocks that constitute both the application and the underlying architecture. Timed and untimed crossbar and communication channels, memories and peripherals and architectural devices, such as timers, controllers and communication devices can all be selected from a library of IP. Different libraries exist and are adapted for different types of modeling. For instance, the *SpaceUTFChannel* component represents an untimed functional bus for early design stages, whereas the *OPBBus* component is used for bus cycle accurate model, before implementations. In software configurations, a software simulator, an ISS and the user selectable RTOS can be instantiated from our library as well.

How does communication work?

All user components are tagged with a unique identification number (ID) which simplifies the system definition. This ID, handled by the communication management system, is used for reaching modules on the system, as opposed to addresses. This creates a pleasant coding technique for module-to-module communications.

How software and RTOS is seamlessly supported in SystemC?

Software execution incorporate user-defined modules (threads) interlocked with an RTOS emulation mechanism handled by the SystemC to RTOS Translator named *Tor*. The *Tor* is first responsible of leading the system's initialization sequence, which consists of creating software task, launching the RTOS scheduler and synchronizing with the hardware platform. The *Tor* interface's second role is to provide mapping between system functions proposed in SystemC 2.1 and RTOS functions through a platform-standardized API. For instance, a SystemC *SC_THREAD* is mapped onto the *taskCreate* function of this API, which can forwardly be mapped to, for instance the *OSTaskCreate* of μ C/OS-II [Z], an RTOS we support. Finally, the *Tor* provides a communication manager, which establishes connections between software modules and platform hardware. Using specific read/write methods, the *Tor* answers task requests to communicate with other software or hardware modules. RTOS are compiled with the appropriate Hardware Abstraction Layer (HAL), depending on the platform and the abstraction level the simulation is in.

In short and simple words, for hardware engineers, the *Tor* allows to quickly and seamlessly execute your process in conjunction of any RTOS, while keeping the code you write intact (consider any type of code), in a multithread or multi-process environment.

What does partitioning bring me?

An automated backbone for module partitioning also is a key feature of the Space Codesign technology. Without changing a single line of code, modules can be swapped from hardware to software, software to hardware, software to software (from one processor to another) or from one bus to another. This feature extraordinarily opens the design space and lets users entirely benefit from their architecture.

Using the designer's experience, coded modules can be instantiated either in the software or the hardware part of the system. Once a new hardware and software configuration is created, a simple recompiling of the hardware (SystemC platform code) and software (cross-compiled) partitions is needed to reconstruct the new system, ready for new co-simulation. The Space Codesign ID mechanism allow to easily move blocks from hardware to software or vice versa. In a couple mouse clicks, one can explore a variety of different configurations and evaluate different solutions. A monitoring engine (see below) supports the users in this task. In addition, automated algorithms based on cost functions filled with engineering criteria can help in deciding acceptable partitioning decisions.

The SpaceStudio Environment

As several vendors of the EDA domain already did, SpaceStudio is a development environment constructed on the Java Eclipse v3.1 platform [7]. Eclipse aims at the creation, simulation and monitoring of hardware or software electronic systems in general.

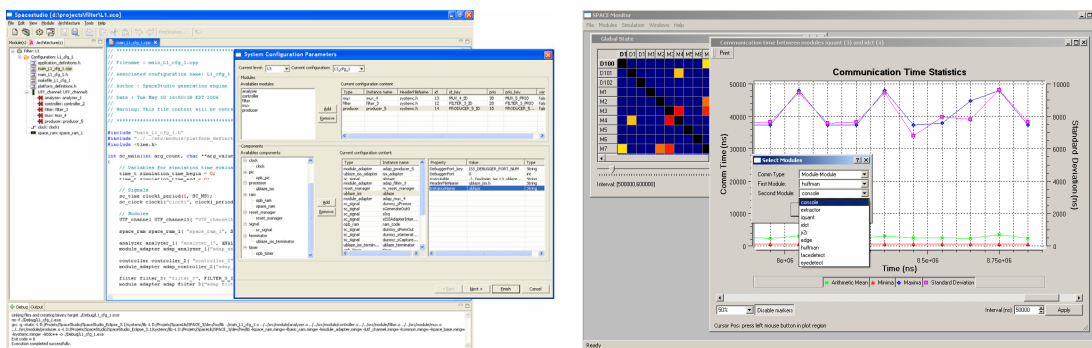
SpaceStudio lets designers browse through architecture and component libraries to create a parameterized system or insert their own custom C/C++/SystemC blocks or C/C++ software tasks. Architectural exploration lets you discover different system configurations (e.g. number of processors and buses, code mapping on hardware or software, hardware coprocessors, etc.) typically not considered when working at the Register Transfer Levels (RTL), because the work implied to experiment such possibilities is enormous. With the help of SpaceStudio, one can drag and drop SystemC blocks into a hardware-software architecture of its own. System parameter setting is kept simple and is centralized in a configuration window accessible through wizards that help you dig out the best from your development environment.

What does SpaceStudio support or integrate?

SpaceStudio is a layer of system creation atop a console based environment. We use SystemC v2.1.1 for hardware and system simulations, Cygwin-based [8] or Linux console commands for makefile creation and execution, compilation or cross-compilation (when selecting target processors or ISS) and the GNU GCC compiler environment [9] and GNU GDB [10] debuggers. Our library of components is XML-oriented, which open to the possibility of importing new components.

SpaceStudio integrates co-debugging and real time co-monitoring tools to maximize programmers' efficiency. With the Space Central Monitor engine, one can analyze communication statistics, bus usage, or software behaviors in true real time or post simulation.

This highly configurable engine pushes one step further the development of hardware/software systems.

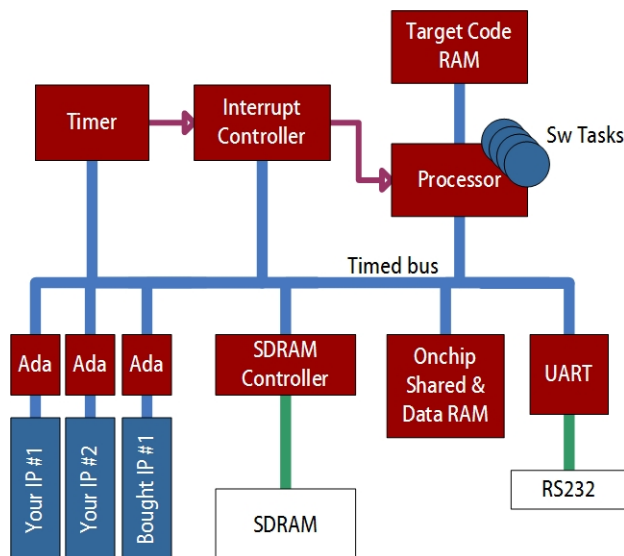


The Space Codesign Technologies

The **Space Codesign Elix** Technology regroups tools for early stages of the design flow and principally serves for model or application validation and fast architecture exploration. With Elix, one can, in a straightforward but specialized SystemC environment, create an untimed or timed functional model of your own or your customer's specifications.

You can benefit from all the tools that help preparing your design for physical implementation. All simulations will bring preliminary results about system performance and expected hardware/software behavior, as you integrate your RTOS and create a set of RT tasks. The early integration of a RTOS helps you fix threads priorities, strengthen software target code while in simulation, and this even before the hardware prototypes are ready to be used.

The **Space Codesign Simtek** technology reproduces, in simulation, a cycle accurate replica of a real architecture and its behaviors, including transactions over a specific bus protocol, hardware wrappers to encapsulate Users Custom IPs, a collection of user accessible platform-specific IP devices, such as memories, timers, controllers and I/Os, as well as performing Instruction Set Simulators (ISS) for true cycle software execution. This abstraction is called Bus Cycle Accurate (BCA) and is primarily used to validate the timing of a future on-chip or on-board implementation. As this abstraction reproduces realistic behaviors in simulation, it produces results that can easily be analyzed, debugged and monitored.



The Space Codesign Elix and Simtek technologies are accessible through the SpaceStudio design environment.

Who is Space Codesign intended to?

Three categories of engineers or designers can expect to greatly benefit for ESL design environments, more specifically from the Space Codesign technology:

Software Engineers have access to a complete and typical software development environment: software threads coding, debugging & profiling. However, rather than executing and debugging this software in a standalone manner on a host machine, or on the physical target board, designers execute their code on a virtual platform representing their final system. This accelerates the design V-cycle process, as the software coding can start a lot earlier, much before a physical hardware prototype is finished.

Systems Engineers are also targeted to benefit from such an environment. Application specifications can be quickly validated using the Space Codesign Elix technology, as a system engineer can very quickly build a hardware/software simulation platform and focus on the deliverable.

Also, systems engineers can explore different system architectures based on the Simtek technology. They can evaluate different system configurations or hardware/software mappings. Moreover, system's performances can be evaluated using the Space Codesign monitoring tools.

Hardware Engineers have the possibility to reproduce a bus cycle accurate model of the hardware to design, in order to validate their processes' algorithmic and timings. They can easily create test benches applicable to all levels of technologies (Elix, Simtek) to enable an early verification process of the specifications (e.g. apply verification process at high level). They can focus on application code rather than on communication complexity or bus synchronism and they can migrate down to a physical implementation using Gen-X Pro.

Down to the chip using Gen-X Pro

To physically implement Hw/Sw partitioned solutions, we extend the Space Codesign technology to a lower level model called *implementation*. At this stage, hardware (muxes, ALUs, decoders, memories, etc.) is described in terms of register transfers executed in each clock cycle (i.e. the RTL), and software is described in terms of a sequence of instructions and RTOS function calls on the selected processor. The transformation from a transactional level (Elix, Simtek) to an implementation level is referred to as *hardware/software co-synthesis*. It is considered one of the most challenging tasks in embedded design involving system-on-chips. For the case of an implementation based on the Xilinx Virtex II-Pro FPGA, an implementation layer is already provided by the *EDK (Embedded Development Kit)* from Xilinx [10]. EDK is an application for designing embedded programmable systems based on the Virtex II-Pro. This pre-configured kit includes all the tools and IPs that you require for designing the implementation level of a hardware/software system. Therefore, the Space Codesign technology performs an

automated co-synthesis process, an automatic translation between Space Codesign Simtek and EDK in 3 steps:

- Software generation is completely automatic. The user software modules/tasks and the RTOS are compiled for the embedded processor. As a first step, the μ C/OS-II RTOS from Micrium has been selected. μ C/OS-II offers all the advantages of a micro-kernel: task preemption, a priority based task scheduler and an interrupt system. As explained in the above sections, the SystemC function calls are mapped to μ C/OS-II equivalents using the "Tor" technology.
- Hardware synthesis is accomplished with behavioral synthesis tools that transform timed functional models into fully timed RTL models. This synthesis is not performed by the Space Codesign technology, but by external FPGA/IC development tools. The Space Codesign approach is based on IP-reusing. We package IPs for reuse at different levels of abstraction in the design cycle, such as the functional (SystemC transactional) level and the RTL (VHDL). IPs can come from within a company or from third-party vendors.
- Communications synthesis is also processed. IBM CoreConnect wrappers are required to connect IPs to a channel (for the transactional level) and to a bus (for RTL). As RTL IP wrappers called IPIF (Xilinx Intellectual Property InterFace) can be automatically generated using the Xilinx Import Peripheral Wizard, a corresponding timed functional SystemC library of IPIF is reproduced.

Ultimately, Gen-X Pro generates the optimum platform partitioning in two files input into EDK: one contains C-code and RTOS-specific function calls cross-compiled for processor (MicroBlaze and/or PowerPC), while the other file contains a list of IPs and processors along with their connections to the different buses instantiated.

The Space Codesign Gen-X Pro technology is currently under development and announcements are expected by mid-2007.

Design Example

Please refer to [11] for a complete description of a design example, the guiding system of a Space Rover.

It's up to you now!

The Space Codesign technology is applicable in various helpful ways to simplify system specification capture, system validation, application monitoring and debugging in C/C++/SystemC, for really quick prototyping, evaluation and co-debugging of software configurations on a hardware virtual platform and for simulating cycle true platforms before to go on-chip. SpaceStudio, an intuitive graphical user interface is supplied to extract the maximum out of your design environment.

References

- [1] John Donovan. "Electronic system-level tools for portable design". Portable Design. http://pd.pennnet.com/articles/Article_Display.cfm?ARTICLE_ID=259962&p=21. September 2006.
- [2] SystemC <http://www.systemc.org/>. September 2006.
- [3] Space Codesign FAQ Sheets on SystemC. <http://www.spacecodesign.com/> September 2006.
- [4] Alberto Sangiovanni-Vincentelli. "Defining platform-based design." May 2002. <http://www.eetimes.com/news/design/showArticle.jhtml?articleID=16504380>
- [5] Space Codesign FAQ Sheets on TLM – <http://www.spacecodesign.com/>. September 2006.
- [6] J.J. Labrosse, MicroC/OS-II, The Real-Time Kernel, 2nd ed., CMP Books, 2002.
- [7] Eclipse.org. <http://www.eclipse.org/>. September 2006.
- [8] Cygwin. <http://www.cygwin.com/>. September 2006.
- [9] GNU GCC. <http://gcc.gnu.org/>. September 2006.
- [10] GNU GDB. <http://sourceware.org/gdb/>. September 2006.
- [11] Xilinx. <http://www.xilinx.com>. September 2006.
- [12] Space Codesign. "System Design Example using the Space Codesign technology". White Paper #2. September 2006. <http://www.spacecodesign.com>.