Industrially Proving the SPIRIT Consortium Specifications for Design Chain Integration

Moderator: Christopher K. Lennard
ARM Ltd

Case Study 1: Victor Berman, Saverio Fazzari — *Cadence Design Systems*; Mark Indovina — *Improv Systems*; Cary Ussery — *Improv Systems*

Case Study 2: Marino Strik — *Philips Semiconductors*; John Wilson — *Mentor Graphics*

Case Study 3: Olivier Florent — *ST Microelectronics*; François Rémond — *ST Microelectronics*; Pierre Bricaud — *Synopsys*

Abstract

There has traditionally been significant engineering overhead required for the integration of multi-vendor tool and IP design methodologies. Making design-chain integration efficient is the key objective of the SPIRIT Consortium. This Special Session paper provides an insight into how the specifications of the SPIRIT Consortium are being adopted in the industry today. We present 3 production design-flow stories which show improved efficiency gained through use of the SPIRIT Consortium specifications. These include an IP generator for hierarchical VLIW processor design, a full hardware / software SoC integration design flow managed through generators, and methodology support for a flow from electronic system level (ESL) design through to the 65 nm CMOS process,

1. Introduction

It is a fact that most system-design companies integrate tools and IP from multiple vendors to handle a system-design as it passes through stages of refinement. Making this integration consistent and efficient requires a single, language-and-vendor neutral way to express IP and the configuration of a SoC design across all its views [1][2].

In this paper, we introduce the reader to the specifications of the SPIRIT Consortium [3]. This consortium is chartered to resolve the issue of multi-vendor design-chain integration. The SPIRIT Consortium specifications provide a way to describe and interpret metadata for IP integration and configuration requirements, and enable the encapsulation of configurable IP and point-tools as SPIRIT-compatible generators. Both of these concepts are gaining strong acceptance in the industry.

The development of the SPIRIT standard has been split into two phases, the first (SPIRIT 1.x) is to address RTL design and verification flow integration, and the second (SPIRIT 2.x) is to address electronic system-level (ESL) design. The first versions of the SPIRIT standard have already been proven on production RTL design exchange.

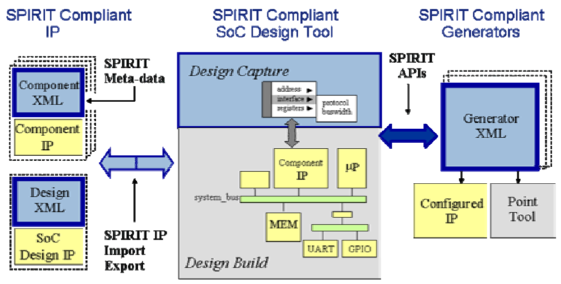

Figure 1: SPIRIT usage in a SoC design environment

The typical usage model for SPIRIT within a design-environment is shown in Figure 1. The SPIRIT specification provides an XML schema [4][5] and semantics to enable the unified authoring, exchange and processing of design meta-data (indicated in light-blue), and it also provides a complete API for meta-data exchange and data-base querying (indicated in dark blue). The user of such a design environment will be able to gather SPIRIT component descriptions (files) together into a library, along with SPIRIT-compatible definitions for any bus interfaces referenced within the components. The key contents of a SPIRIT 1.x component definition include: top-level I/O, bus interface declarations, definition of the memory map, the various views of the IP including the files required to 'see' a particular view, and implementation constraints which must be met in order for the IP to properly function within the subsystem. A design environment (integrated tool flow) can instantiate and connect SPIRIT components to form a design, which is itself represented as a SPIRIT design file expressing component instantiation and connectivity.

To enable a design-environment to be augmented with point-functionality, SPIRIT defines the concept of plug-in generators. Generators can be connected using a meta-data dumping mechanism, Loose Generator Interface (LGI), or the Tight Generator Interface (TGI) which is a full API based on SOAP [6] to guarantee language-independent generator integration.

The SPIRIT v1.2 Specification for RTL design is currently entering IEEE standardisation (P1685). A version 2.0 of the SPIRIT standard is also being developed as an extension to the SPIRIT 1.2 specification to address ESL design and transactional verification IP. Versions 1.2 and 2.0 will be kept compatible so that any design resolved to the RT level can be expressed in SPIRIT v1.2 [7] to a SPIRIT v2.0 environment.

The remainder of this paper describes three significant industrial adoption examples for the SPIRIT v1.1 specification.

2. Case Study 1. SPIRIT enabled VLIW processor design platform

2.1 Introducing a design generator

Many innovative IP designs have failed to gain traction in the market place because of the difficulties of integrating them into full system design and verification flows. This case study describes how the Improv Systems Jazz VLIW Processor Platform for the creation of processor architectures has used the emerging SPIRIT standard to solve the flow integration problem.

The Jazz Composer Tool Suite allows designers to hierarchically create custom VLIW processors and platforms. Each platform consists of multiple processor blocks, embedded memories, and integration blocks (e.g., AMBA interface). The tool suite includes a tool for creating these platforms, and a complete set of generators for creating a full RTL database together with EDA tool scripts, test benches, and emulation mappings.

The Improv development system, which has been in production use for more than five years, already uses design meta-data and interfaces that are conceptually very close to the SPIRIT standard. What was lacking for scaleable integration of the Improv tool suite into multiple design methodologies was an industry agreed standard syntax and semantics for describing components and their interconnections. The SPIRIT schema and semantics rules provide that standard, enabling the interoperability of IP and tools outside of the proprietary Improv database.

.

2.2 Making an existing IP generator design flow SPIRIT-enabled

The Improv system is based on an internal database for components and a series of generators that build customized platforms from that data based on user constraints. The current version of this system still relies on internal formats for its software definitions and overall system architecture but brings this information into a SPIRIT database for configuring the hardware platform for implementation.

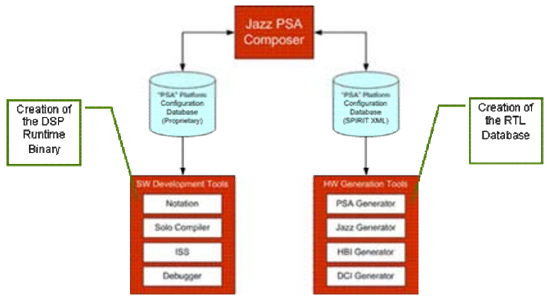
Jazz PSA Composer
"PSA" Platform Configuration Database (Proprietary)
"PSA" Platform Configuration Database (SPIRIT XML)
Creation of the DSP Runtime Binary
Creation of the RTL Database
SW Development Tools
Notation
Solo Compiler
ISS
Debugger
HW Generation Tools
PSA Generator
Jazz Generator
HBI Generator
DCI Generator

Figure 2. Deploying SPIRIT in hardware IP generation for the Improv design flow

To create the SPIRIT enabling of the Improv tool suite, the internal database is transformed into a parallel SPIRIT database which contains the various levels of schema description for components, designs, platforms and buses (interconnections). This parallel database approach is depicted in Figure 2. External SPIRIT components, including those from 3rd-parties, can be imported into the design database for final configuration and the full design can be exported as a SPIRIT compatible description for use by other SPIRIT compatible design environments (DEs) and EDA tools.

The design generators enabled from the SPIRIT database are PSA for architecture level, Jazz for the processors, HBI for the bus interfaces, and DCI for the digital channel interfaces are used to automate the production of an RTL database that constitutes the fully configured design. Platform configurations are represented as full SPIRIT XML specifications including 'hints' files. The generators read in SPIRIT specifications from the platform components as input to the generation process.

The current integration of the SPIRIT specification into the Improv IP generation tools allows for direct integration with SPIRIT based design and verification tools and flows. This standards based approach has the potential to open up the SoC design process to a richer set of innovative IP and significantly reduce the bottle necks in producing efficient, verified chips. Even greater efficiency benefits are expected from migration to a native SPIRIT database, system-level model support with the SPIRIT Consortium specification v2.0, and the migration of the software runtime binary to a SPIRIT compatible methodology in future releases.

3. Case Study 2. Creating an efficient derivative design flow using the SPIRIT specifications

3.1 Introducing an SoC design methodology

Establishing an efficient reuse methodology has had priority within the Philips Semiconductor organization for many years [8]. IP creation with reuse in mind is common practice. A central organization operates within the company providing reusable IP with configurable infrastructure IP, standard communication interfaces and memory controller IP. These highly reusable components must be described in such a way that methods for creating derivatives can be automated, including IP configuration and integration. The emergence of well defined standards, including the SPIRIT Consortium specification, is the reason why reuse is becoming highly effective.

In most design projects, IP is obtained from sources both inside and outside the company. It is a necessity that these IP are able to be rapidly imported into complex multi-vendor design-tool flows. To achieve this, the design tools must recognize the structure and configurability of the IP automatically, and this cannot depend on the IP source coding style, design-language and file packaging.

Many requirements relevant for individual reusable IP also need to be apply to subsystems as they need to be integrated into a larger chip context. This is the next major step in IP re-use, as depicted in Figure 3. Like an IP, a subsystem must be transferable from one group to the other. A subsystem must be self contained and provide a verification view and an implementation view. It is most productive to verify IP component properties at the component level, and not to repeat component verification at the subsystem level. Complementary to the verification flow, the automation of implementation tasks such as synthesis and DFT on subsystems makes these processes more repeatable and the validation process more straight forward [9].

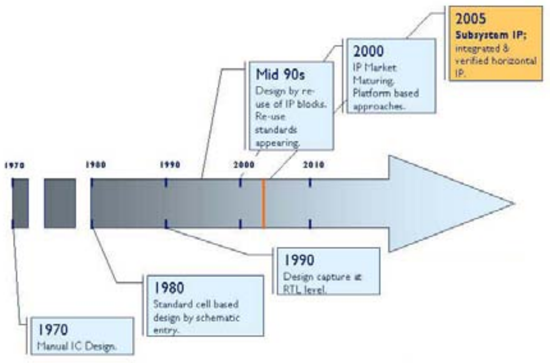
2005
Subsystem IP;
integrated & verified horizontal IP.
2000
IP Market Maturing. Platform based approaches.
Mid 90s
Design by re-use of IP blocks. Re-use standards appearing.
1970
1980
1990
2000
2010
1990
Design capture at RTL level.
1980
Standard cell based design by schematic entry.
1970
Manual IC Design.

Figure 3. The increasing adoption of IP re-use

Re-use of subsystems with verification IP and automated implementation tasks is a fundamental aspect of iterative and derivative design is known as the single-build use model. It drives the need for verification IP encapsulation using common structural descriptors for integration and configuration requirements, as well as capture of automatable implementation and verification tasks as design-flow generators [10].

The Philips flow performs the integration of IP blocks by accessing a repository to configure IP and then creating a top level RTL description. Complementing this, all scripting for the verification and implementation is generated with Makefiles as the user interface. These enable the user to invoke every flow step individually. The script generation takes a matter of seconds, but executing of these scripts can take hours.

3.2 Improving flow integration using SPIRIT

SPIRIT is an important standard to enable the single-build use model. It enables IP integration and configurability information to be described by the IP provider such that it is able to be interpreted in any SPIRIT compatible environment.

The determination of configuration parameters is required before the configurable IP can be downloaded from a repository. Two mechanisms are applied based on the XML information to determine IP parameter values. All parameters for which no trivial value can be concluded form the subsystem context are offered to the user. For about 75% of the parameters the value can be derived automatically using a generator.

The SPIRIT XML points to a generator which can access the XML information of the complete system and all its IP components. This context analysis is invoked in the build process. The user starts the build process thereby informing the system that all interactive design composition tasks are completed.

Underlying the SPIRIT XML processing is a SPIRIT-Enabled Design Environment as depicted in Figure 4. The DE can read the SPIRIT XML data into a database, enables the designer to configure IP, creates the configured designs at the block-level, and records the IP and design structures and configurations.

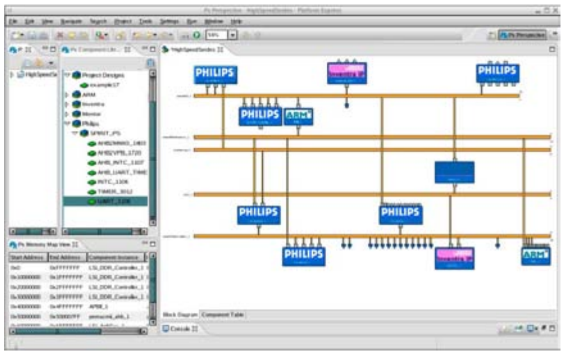

Figure 4. An example of a SPIRIT-enabled design environment

It is from this database that the generators access the design data to create new design information. The new design information derived from the SPIRIT XML data typically includes RTL design descriptions, software to execute on the design, automatic setup of various design and verification tools used in the design process as well as many other applications.

One of the key productivity features has been the ability to extend the core functionality of our SPIRIT DE, the Mentor Graphics Platform Express tool. This enabled links directly to our IP repository. Using generators, design context and configuration information was extracted from the DE's design database and sent to our IP repository. The IP repository creates a correctly configured IP package which is returned to the DE, enabling the DE to automatically update the design with the correctly configured IP.

3.3 Automotive example and conclusions

The SPIRIT-enabled single-build design-flow is being used on production designs. One example is an ARM9 processor based design. The on-chip AHB based communication infrastructure, serial communication interfaces, embedded memory and the external memory interfaces were integrated using integration automation enabled through the SPIRIT Consortium specifications.

The benefits of this first project were very clearly recognized when the second platform design was executed. Many generic solutions were able to be reused even though many IP components and much of the infrastructure changed which a change in the processor. This complete new AXI based system was created in 6 months, a record time, and its use will reduce the derivative design time by 50% [11].

By enabling IP integration and configuration information to be described with a single XML schema, and enabling configurability and tool-scripting to be captured in generators, SPIRIT has played key role enabling the separation between IP and tooling. Philips can create subsystems faster and larger than what was possible manually before.

The next application of the single-build use-model will be a heterogeneous multiprocessor design with automated flow from component composition to ready-for-placement netlist, including DFT and to FPGA image for physical prototyping.

4. Case Study 3. A 65 nm SoC design based on the emerging SPIRIT standard

4.1 SPIRIT in SoC design methodologies

With increasing time to market pressure, it is key to automate as much as possible the front-end design process, implementing 'correct by construction' system integration and speeding up the time from RTL to netlist. The approach should be flexible enough to encompass multi-abstraction system model integration, Transaction Level Modelling (TLM) to RTL, to enable IP selection, system verification and early software development. It should also maintain consistent system configurability data through to physical implementation. Most crucially, this consistency of flow must be easily supported for a SoC built with IP blocks from various sources.

Past focus on IP reuse has concentrated on design quality metrics such as those driven by VSIA [12] with the QIP industry quality metrics. While this is a critical element in judging the 'goodness' of design component, it does not address how to deliver IP for automated integration of IP blocks within tool-flows. The SPIRIT Consortium was created to build specifications resolving this issue. To ensure success from the beginning, this initiative began with a Steering Committee consisting of the top 3 EDA vendors, the major processor IP provider and two European SoC integrators and manufacturers. The SoC integrators are part of the Crolles 2 Alliance and this is considered the most advanced nanometer silicon Research and Development effort in Europe. Since its inception, the Consortium has built a significant membership from leading design-chain and systems integrator companies.

On the technical side the SPIRIT standardization effort can be separated into two parts: the XML IP design meta-data, and the API for IP generator and point-tool integration. The IP design meta-data enables machine-interpretable design IP, integration requirement specifications, and consistent information across all design views. The generators enable point-tool launch, IP configuration launch, and tool-flow integration. Both of these aspects of the SPIRIT specification have been used in the creation of the 7200 design.

4.2 Deploying SPIRIT in the 7200 chip design flow

Designed to cover high-end set top box requirements, this 65 nm chip features dual HD MPEG4 decode, audio decoding, HDD and DDR2 interfaces and extensive set of communication channels (Sata, Ethernet…). The master bus is running at 200MHz, with several internal DSPs. Estimated synthesized gate count is 24 million, and the circuit embeds 6 Mbits of static RAM. The circuit is built using 40 design IP components (sourced internally and externally) and 30 custom blocks. The architecture, depicted in Figure 5, is composed of 8 major functional subsystems.

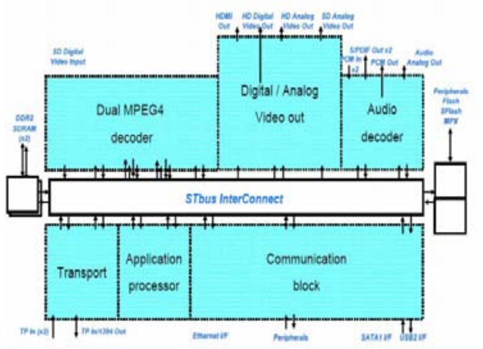

Figure 5. Overview of the 7200 Set Top Box SoC architecture

Originally every SoC library element was available as a standard ST IP package. A custom ST toolset has been developed to automatically derive the SPIRIT component XML from existing IP data, and add this as part of the standard ST IP package. It extracts in batch mode boundary description, bus interface, memory map, but also records references to existing views such as SDC, RTL and TLM.

Once the XML descriptions for the components are available, a SPIRIT based tool is used to build the SoC. This is done within Synopsys CoreAssembler, a SPIRIT-compatible tool that uses automatic protocol based connection to enable correct-by-construction bottom-up IP assembly. Once any remaining ad-hoc wire connectivity has been completed for the system, the SPIRIT On-Chip Bus Generator is called to configure the bus micro architecture. Finally, the core level signals and interfaces are exported to match the top-down definition of the pad-ring of the chip. This process of design assembly can be either interactive using the CoreAssembler GUI, or batch-mode using the tool's TCL interface. This enables batch replay of the assembly to update the design database quickly and reliably after design specification changes.

Using the design XML database, many different tasks are addressed: the assembled netlist and the promoted SDC constraints are extracted to start the flow into hardware implementation [13] and physical design exploration. Several simulation models can be built that reflect different views of the same system thanks to an internally developed netlister with the ability to mix different levels of the SoC components (RTL, TLM, Instruction Set Simulation). This enables standard RTL verification as well as early development of application software and drivers. The memory maps are also automatically consolidated at SoC level, taking into account the various address shifts introduced by the On-Chip Bus. A C-language program is then generated and executed on the embedded processor to verify system level memory map and integrity of register accesses.

4.3 Benefits of SPIRIT in the ST design flow

There are several immediate advantages seen in the adoption of SPRIT into the design flow: 1. the adoption of the SPIRIT specifications allowed the ST flow to be built on an open format database, thus enabling the development and integration of a custom toolset; 2. the protocol based design capture has shown a significant productivity increase compared to traditional wire level design entry as the fully scripted implementation responds easily to design respins due to design specification changes; 3. the cost of design flow adoption is reduced as automatic translators allow the creation of an XML 'system-design skeleton' from RTL/SDC data; and 4. the unifying the SoC description both for implementation, verification and early software development prevents costly iterations arising from misalignment of separate design databases.

These design flow advantages demonstrated on a real life system design are helping to guarantee the broad-based adoption of a SPIRIT-compatible methodology within ST Microelectronics.

5. General conclusions

The benefits of IP re-use methodologies have been understood for many years, culminating in the vision of full subsystem and platform re-use [14]. While the vision has been correct, the implementation and design-chain support for that high degree of complex configurable IP re-use has been lacking.

The specifications of the SPIRIT Consortium are designed to address the needs of complex IP re-use in multi-vendor flows. By using the W3C standards for meta-data exchange [4][5], the specifications are able to guarantee description of IP integration and configuration requirements that are independent of IP architecture, IP authoring style, and design language. The use of SPIRIT Consortium APIs, also using W3C standards [6], enables the encapsulation of configurable IP generators and implementation processes in a design-environment neutral way.

The SPIRIT Consortium specifications are getting significant adoption in the industry, from the IP and EDA suppliers through to system integrators. The case studies presented here represent just a few of the growing number of multi-vendor design-flow integration successes being achieved through use of the SPIRIT Consortium specifications.

6. References:

[1] Lennard, Granata, "The Meta-Methods: Managing design risk during IP selection and integration". European IP 99 Conference, November 1999

[2] John Wilson, "IP Reuse Simplifies SoC Design and Verification", EE Times, 10 November 2004

[3] SPIRIT Consortium, "SPIRIT 1.1 Specification", www.spiritconsortium.org, June 2005

[4] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0" Third Edition, 2004

[5] World Wide Web Consortium, "XML Schema Part 1: Structures", Second Edition; "XML Schema Part 2: Datatypes" Second Edition, 2004

[6] SOAP Specifications: www.w3.org/TR/soap

[7] Grun, Shin, Baxter, Noll, Madl, Lennard, "Integrating a Multi-Vendor ESL-to-Silicon Design Flow using SPIRIT". IP SoC Conference, December 2006

[8] Geoff Mole, "Philips Semiconductors Next Generation Architectural IP ReUse Developments for SoC Integration" IP SoC Conference, December 2004

[9] Ron Wilson, "Revolutionary Pay-Off". EE Times, 28 February 2005

[10] Denis Bussaglia, "Automated Implementation Flows based on IP-level constraints and synthesis intent in XML". IP SoC Conference, December 2005

[11] Philips Semiconductors *SJA2510* SoC press release www.semiconductors.philips.com/news/content/file_1191.html

[13] Bricaud, Pierre-Keating, Michael, "Reuse Methodology Manual for SoC Designs". Springer

[14] Alberto Sangiovanni-Vincentelli, Grant Martin, "A Vision for Embedded Systems: Platform-Based Design and Software Methodology". IEEE Design and Test of Computers, Volume 18, Number 6, November-December, 2001, pp. 23-33.