Specify-Explore-Refine (SER): From Specification To Implementation

A. Gerstlauer, J. Peng, D. Shin, D. Gajski
Center for Embedded Computer Systems
University of California, Irvine
{gerstl,pengj}@cecs.uci.edu,
{dongwans,gajski}@cecs.uci.edu

A. Nakamura, D. Araki
InterDesign Technologies, Inc.
43-16, Shiba 3-chome,
Minato-Ku
Tokyo 105-0014, Japan
nakamura.atsushi@interdesigntech.co.jp,
araki.dai@interdesigntech.co.jp

Y. Nishihara
Japanese Aerospace Exploration Agency
Tsukuba Space Center
Sengen 2-1-1, Tsukuba
Ibaraki 305-8505, Japan
nishihara.yuuji@jaxa.jp

ABSTRACT

Driven by increasing complexity and reliability demands, the Japanese Aerospace Exploration Agency (JAXA) in 2004 commissioned development of ELEGANT, a complete SpecC-based environment for electronic system-level (ESL) design of space and satellite electronics. As integral part of ELEGANT, the Center for Embedded Computer System (CECS) has developed and supplied the SER tool set. Following a Specify-Explore-Refine methodology, SER supports system-level design space exploration, interactive platform development and automatic model refinement and model generation. The SER engine has been successfully integrated into ELEGANT. With SER at its core, ELEGANT provides a seamless tool chain for modeling verification and synthesis from top-level specification down to embedded HW/SW implementation. ELEGANT and SER have been succesfully delivered to JAXA and its suppliers. Tools are currently being deployed in companies like NEC Toshiba Space Systems. Evaluation results prove the feasibility of the approach for design space exploratoin, rapid virtual prototyping and system synthesis resulting in tremendous productivity and reliability gains. In addition, ELEGANT has been commercialized for general market availability. The SER component has been licensed to InterDesign Technologies, Inc. (IDT) and it is available from, sold and supported by IDT.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; C.5 [**Computer System Implementation**]: Miscellaneous

General Terms

Design, Management, Reliability

Keywords

Electronic System-Level (ESL) Design

DAC 2008, June 8–13, 2008, Anaheim, California, USA

1. INTRODUCTION

The process of designing embedded computer systems is a tremendous challenge. System complexities are growing exponentially and the system design process is characterized by tight constraints and market pressures, not the least of which are cost and reliability. In many application areas such as space and satellite electronics, design problems can have serious and costly consequences, and traditional ad-hoc design methods are becoming infeasible.

Faced with growing cost and reliability issues, the Japanese Aerospace Exploration Agency (JAXA) in the early 2000s started to investigate alternative solutions. Having commonly invested large efforts into building several prototypes of satellites before their launch, JAXA was interested in concepts and technologies for early virtual prototyping. At the same time, design automation techniques for synthesis and verification should be applied in order to achieve required productivity, cost and reliability gains, following a correct-by-construction approach while ensuring that constraints are satisfied.

As outlined by many semiconductor technology roadmaps [12], a commonly accepted solution to increase productivity and handle the complexity is to move the design process to higher levels of abstraction, i.e. the so-called electronic system level (ESL). There are many approaches claiming to provide ESL solutions such as C-to-RTL tools implementing traditional high-level synthesis of a single hardware unit [5]. However, true system-level design has to encompass complete systems across hardware and software boundaries. Furthermore, while there are simulation-centric approaches for manual assembly of virtual system prototypes [2, 17] based on system-level design languages (SLDLs) such as SystemC [10], application of design automation techniques requires well-defined input, output and intermediate models to be captured on top of any underlying language.

As a result of their investigations, JAXA decided to assemble a complete ESL design solution supporting simulation, synthesis and verification in order to establish it as a common design environment for all of its electronic suppliers. In 2004, JAXA therefore commissioned development of the so-called ELEGANT[1] environment. As a basis for combining several tools into a common environment, ELEGANT was chosen to be built around SpecC technology [7, 8]. As a result, a common design environment and well-defined interfaces and formats of design models greatly reduces communication overhead and facilitates easy and error-free exchange between different suppliers.

[1]Electronic Design Guidance Tool for Space Use

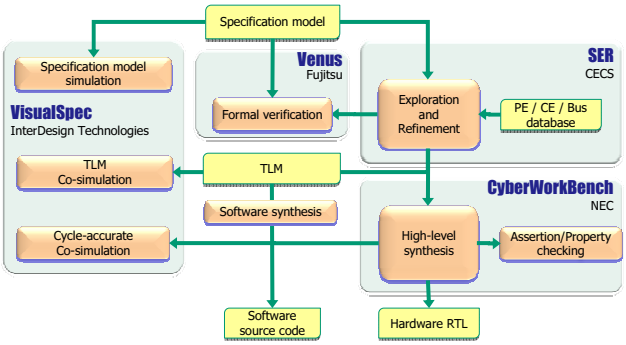

Figure 1: The ELEGANT environment.

1.1 ELEGANT Environment

ELEGANT (Figure 1) provides an environment for true top-down electronic system-level (ESL) design following a consistent and powerful methodology and tool chain from a top-level specification model all the way down to RTL design. Under JAXA's guidance, ELEGANT is a world-wide joint R&D project involving several partners, combining different point tools under a common umbrella. All tools in ELEGANT operate on and exchange design models in SpecC format as standardized by the SpecC Technology Open Consortium (STOC) [3]. Tools have been specifically developed or adapted to accept SpecC input and generate SpecC output, providing a seamless tool chain from top to bottom.

At the core of ELEGANT, the SER component provides system-level design space exploration and model refinement. Starting from an abstraction specification of the desired functionality written in SpecC, SER allows interactive platform definition and specification mapping. SER then automatically implements the specification on the platform and generates various transaction-level models (TLMs) of the design. The SER engine has been developed by the Center for Embedded Computer Systems (CECS) at UC Irvine as a derivate of CECS' original SpecC-based System-On-Chip Environment (SCE) [4], where SER has been adapted to JAXA requirements and databases have been filled with necessary components for space and satellite applications.

For synthesis of hardware components, ELEGANT includes the CyberWorkBench high-level, C-to-RTL synthesis tool originally developed at NEC [14]. As part of the ELEGANT project, Cyber has been adapted to accept SpecC input directly from the SER-generated models. Furthermore, it was extended to generate cycle-accurate (CA) SpecC models of the RTL output for co-simulation of the synthesized hardware with the rest of the system.

All models in the ELEGANT design flow are executable for validation through simulation. In ELEGANT, models are captured and simulated using VisualSpec, a SpecC modeling and simulation environment supplied by InterDesign Technologies [11]. VisualSpec provides visualization and debugging support and optional integration of third-party instruction-set simulators (ISS) into the SpecC simulation backplane. Furthermore, VisualSpec supports profiling and estimation capabilities for model analysis and design quality feedback, including software execution time estimation and timing back-annotation of TLM software models using so-called FastVeri technology.

Finally, ELEGANT contains a formal verification component, Venus, developed by Fujitsu Labs of America (FLA) and Tokyo University. Venus supports equivalence checking between different models in the ELEGANT design flow to guarantee correctness and model equivalence throughout the design process.

1.2 Specify-Explore-Refine (SER)

The SER component at the core of ELEGANT is based on SCE technology originally developed at CECS for the definition and synthesis of system platforms. As a derivative of SCE, SER supports and aids the manual, interactive design of system platforms by automatically generating platform design models at varying levels of abstraction. Using the tools inside SER, a system platform can gradually be developed in a step-by-step manner.

SER accepts architecture-independent, algorithmic specification models written in SpecC and it enables interactive and step-wise system-level design space exploration with HW/SW partitioning, network topology design, bus protocol selection and bus interface synthesis. Given the application specification and design decisions, SER automatically generates accurate transaction level platform models (TLMs) which enable performance analysis and software testing within the virtual platform simulator without the need for slow instruction-set simulation. In addition, SER generates C model descriptions of hardware components including all application functionality, bus interfaces and pin- and timing-accurate bus protocol state machines for backend behavioral synthesis of complete hardware processors down to RTL and implementation.

On top of platform models generated through SER, application functionality can be developed and/or refined. Using SER platform models, applications can be developed and co-simulated with the whole system even before the actual system hardware is available. Moreover, since SER supports stepwise platform development, intermediate platform models at high levels of abstraction are available even before details of the complete system architecture are available or decided. In addition, intermediate high-level models provide fast simulation and feedback. Therefore, SER platform models at various levels serve as system prototypes for rapid, early application and platform co-development.

2. SER DESIGN FLOW

SER follows a *Specify-Explore-Refine* methodology [6]. It starts the design from a model representing the design functionality (*Specify*). At each following design step, SER users first make certain design decisions by exploring the design space (*Explore*) through scripting or a graphical user interface. SER then automatically generates a new model at lower abstraction level by integrating the decisions into the previous model (*Refine*).

Figure 2 shows the SER design flow. Platform design in SER starts with the specification model. With the specification model, the user defines the desired application platform, framework and requirements. SER then consists of of three platform design tasks: *Architecture Exploration*, *Network Exploration* and *Communication Synthesis*. During architecture exploration, the processing platform is defined and the computational aspects of the specification are implemented on top of a set of processing elements (PEs), i.e. software processors, custom hardware blocks and memories. During network exploration and communication synthesis, the communication platform is defined and abstract system communication requirements are implemented over the given network of buses and communication elements (CEs), i.e. bridges and transducers.

Tasks generally operate on, read and generate models for the platform design being developed. All models within SER are described and captured using the syntax and semantics of the standard SpecC system-level design language. The five models for these tasks are the *Specification Model*, the *Architecture Model*, the *Network Model*, and the two communication model variants: the *Transaction Level Model (TLM)* and the *Pin-Accurate Model (PAM)*. All models in the SER design flow are executable for vali-

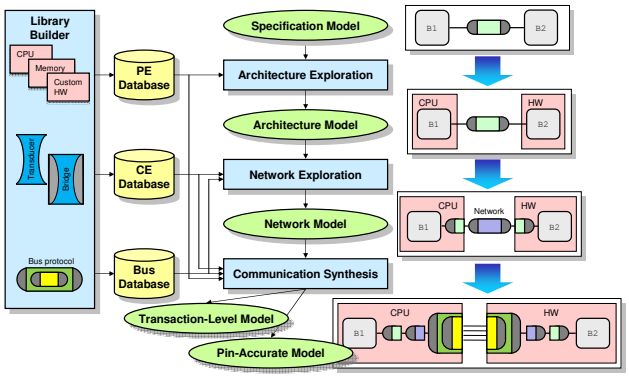

Figure 2: Specify-Explore-Refine (SER) design flow.

dation through simulation. In general, design models generated and read by SER tools can exported and imported in SpecC format for co-simulation, visualization, capture or verification by other tools.

Definition and development of platforms within SER is based on system components stored in the SER databases. For this purpose, SER maintains three component databases for PEs, CEs and buses. With the help of the separate *Library Builder* tool that is part of SER, databases can be maintained by adding, removing or modifying components through a graphical user interface.

2.1 Specification

The specification defines the desired application framework in terms of its computation and communication functional requirements. Computational requirements are specified through the hierarchy of SpecC behaviors that encapsulate the various computation tasks available in the application. Leaf behaviors at the bottom of the hierarchy contain application task algorithms in the form of ANSI C code. Behaviors can then be composed hierarchically in arbitrary serial-parallel fashion. At each level, a sequential, parallel or state machine composition of sub-behaviors is supported. SER will then later generate an implementation that ensures that the tasks run in the specified order in the final system platform.

Communication requirements between tasks in the application framework are specified in the form of variables or channels connecting the behaviors. Behaviors can use shared variables for exchange of data. In addition, especially in the case of synchronization between concurrent behaviors, typed or untyped channels out of the standard SpecC channel library (semaphores, queues, or single and double handshake channels for synchronous or asynchronous message-passing) can be used for communication between tasks. SER will then later generate an implementation of all inter-task communication over the given communication architecture and bus network.

With the specification model, the application developer is provided with an abstract, parallel model for programming the platform. The specification defines the services that have to be made available by the platform being developed. It is guaranteed that the same exact services are available in any platform model at any abstraction level created with SER. As such, SER in the end generates and provides a semantically equivalent implementation of all specified computation and communication functionality on top of the platform developed within SER.

2.2 Architecture Exploration

Architecture exploration (AE) starts with the specification model captured by the designer. It then allows the designers to specify the processing architecture by selecting system components (software and hardware processors, memories) from a database of processing elements (PEs). Following this, the designer can perform hardware-software co-design and process and storage partitioning by mapping the behaviors and variables in the specification to selected (hardware or software) PEs and memories. Architecture exploration then generates and exports the new architecture model that implements the specification computation on the PE architecture according to the selected mapping.

As part of behavior partitioning, architecture exploration will automatically generate all necessary synchronization to preserve the original execution semantics. For example, if two sequential blocks are mapped onto concurrent processors, additional event channels and synchronization calls will be inserted into the architecture model to ensure that the second task will only start once the first task has finished.

During architecture exploration, all variables shared between PEs have to be mapped into physical storage for implementation. In general, variables can be mapped to PE-local or global, shared memories. For all shared variables mapped into a synthesizable hardware PEs, external memory interfaces are generated in the hardware PE and variable accesses by other components are refined into memory-mapped I/O between PEs. If variables are not mapped to specific memories, distributed, local copies of the variable are generated in each PE, and additional message-passing channels and calls are inserted to automatically exchange updated data values at synchronization points.

The specification model at the input of architecture exploration may contain complex communication channels such as semaphores, queues or mutexes. During architecture exploration, complex channels are converted into an implementation following a client-server model. For each complex channel, a server task will be generated in a PE as determined by the user-defined channel mapping. Complex channels can be mapped by the designer to a newly allocated, dedicated PE or to any existing hardware or software processor next to other behaviors and variables. Architecture exploration will then refine all complex channel communication into message passing between clients and server using a remote procedure call (RPC) type mechanism. As a result, all communication in the architecture model at the output of architecture exploration occurs exclusively through memory interfaces, events or synchronous and asynchronous message-passing channels.

2.3 Network Exploration

Network exploration (NE) starts from the architecture model and allows designers to define the network topology by allocating buses and CEs and by defining the connectivity of PE and CE ports to system buses. Given the topology and connectivity, the designer then defines the routing of architecture channels via stations and links between neighboring PEs and CEs. As a result of network exploration, SER inserts transducers, merges and refines channels from the architecture model and generates and exports the refined network model implementing the upper communication layers for realization of end-to-end channels over point-to-point logical communication links between PEs and CEs.

In general, networks are segmented and CEs are introduced during network exploration to implement communication between two or more bus protocols. CEs in the database are distinguished between bridges, which are IPs that transparently connect two timing-compatible buses directly at the protocol level[2], and transducers, which are synthesized to connect any two incompatible buses based

[2]Examples of bridges are memory controllers or typical bus bridges that connect high-speed processor and low-speed peripheral buses.

on a store-and-forward principle[3]. In order to reduce buffer requirements in intermediate transducers, network exploration supports the packeting of large messages into a stream of smaller packets if a transducers is in the path between two PEs.

In the architecture model, PEs and memories communicate via abstract, typed end-to-end message-passing channels and memory interfaces. The architecture model may contain multiple channels between two components. As part of the network model, such channels will be merged if possible, i.e. if they are accessed mutually exclusive in time and if their transactions are guaranteed to never overlap. Inside each PE and CE of the network model, the network explorer inserts the necessary protocol stacks for implementation of channel routing, packeting and channel merging. In addition, network protocol stacks handle the conversion of abstract application data types into untyped byte streams supported by the network. This includes conversion between incompatible byte layouts and endianess in different PEs and memories. As part of data type conversion, memories and memory interfaces are refined down to the individual word level according to the memory's data layout.

2.4 Communication Synthesis

Communication synthesis (CS) takes in the network model, and it allows designers to define implementation of point-to-point logical links for each bus segment in the system. Designers define link and bus parameters like addressing, interrupt and arbitration assignments for each system bus. Communication synthesis then inserts protocols and bus-functional component descriptions from the databases, and it generates the refined communication model which implements the communication links in each segment over the actual, shared bus protocols and bus wires. In addition to the pin-accurate communication model (PAM), the communication synthesizer can generate a timing-accurate transaction-level model (TLM) which abstracts away the pin-level details of individual bus transactions in order to achieve faster simulation speeds.

The generated communication models are composed out of bus-functional models for each component that include the necessary lower layers of communication stacks for implementation of synchronization, addressing, data slicing, arbitration and bus protocol timing. If available, such bus-functional component models are taken from the PE and CE databases. In all other cases, bus interface and bus driver implementations are synthesized on top of bus protocol models taken from the bus database. For this purpose, the bus database contains multi-level bus models that provide a canonical abstraction of all bus accesses from individual bus transactions up to generic transfers of raw blocks of bytes. As such, the bus concept in SER is generic in the sense that any communication media that can provide reliable, error-free byte block transfers can be captured and stored in the bus database[4].

Unless synchronization is built into the bus protocols themselves, buses generally require communication partners to be synchronized before the start of each actual bus transfer. During communication synthesis, SER insert any such necessary synchronization between masters and slaves (or senders and receivers) into all participating system components. Communication synthesis supports synchronization via dedicated or shared interrupts and through polling. In case of interrupt-based synchronization, components are connected to interrupt wires and interrupt generation and detection logic is inserted. In case of polling or shared interrupts, driver code to query bus-mapped flags on the slave side is inserted into the components.

[3]Examples of transducers are UARTs or other serial bus controllers.

[4]Examples can range from typical master/slave buses up to node-based buses such as RS232 or other network protocols.

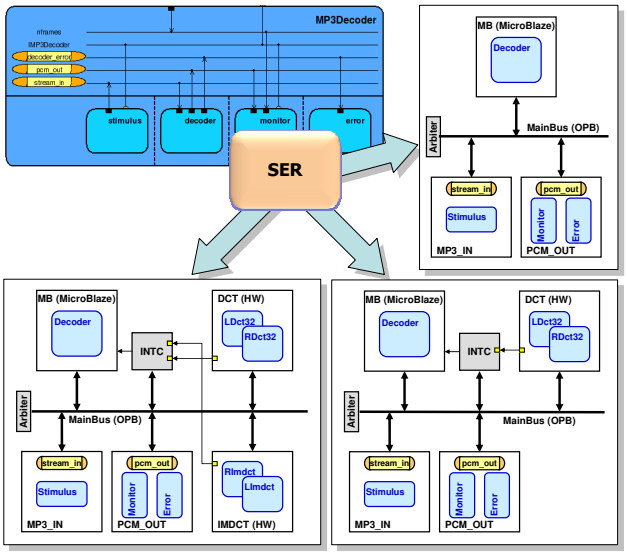

Figure 3: MP3 decoder explorations.

In general, all links within a segment require unique address/interrupt assignments, and each communication link in the network model is assigned a separate interrupt and a separate bus address (or an address range in case of polling or interrupt sharing). Communication synthesis then generates any necessary bus protocol stacks for implementation of synchronization and bus addressing on top of the raw data streams provided by the bus interfaces in the database.

Note that in case of software processors, bus-functional models taken out of the PE database consist of a hardware abstraction layer (HAL), a model of the processor hardware including the processor's interrupt handling behavior and models for any interrupt controllers. Communication synthesis then generates all bus drivers and interrupt handlers inside the processor as necessary. As a result, TLM and PAM contain high-level models of programmable processors that, together with back-annotated estimated execution times, provide fast and accurate software prototyping compared to traditional, slow instruction-set simulators (ISS).

3. EVALUATIONS AND RESULTS

The first phase of development of SER tools and the ELEGANT environment was successfully completed in 2006. Next to continuation of the development process, JAXA initiated several evaluation projects in order to investigate and demonstrate the feasibility and benefits of the approach. Tools were deployed and applied to various design examples in several of JAXA's suppliers and other independent investigators with positive results. In addition, feedback from the evaluation process is used to drive further continued development and extension of the SER and ELEGANT tool sets.

3.1 MP3 Decoder Design Example

As a test case and design example for demonstration of SER capabilities, SER tools were applied to the design of an MP3 decoding application previously developed at CECS [9][5]. The SpecC specification model of the MP3 decoder was originally developed based on an open-source C reference implementation we obtained

[5]The SpecC code of the MP3 decoder specification model is freely available under the GPL on the CECS web pages [1].

from the internet [16]. Due to the fact that SpecC is an extension of ANSI C, the C reference code could be directly used as a basis for conversion into a SpecC specification. The resulting SpecC specification model of the MP3 decoder has 14,045 lines of code distributed over 44 behaviors (out of which 29 are leaf behaviors).

Using SER's exploration and platform development capabilities, the MP3 decoding application was mapped and implemented on several different candidate architectures, ranging from a pure software implementation up to various levels of hardware acceleration (Figure 3). For demonstrational purposes, a Xilinx-based implementation target (MicroBlaze CPU with OPB system bus) was assumed for all MP3 explorations.

At the top level, the MP3 specification consists of the actual decoder running concurrently with I/O behaviors for MP3 frame input (*Stimulus*), audio output (*Monitor*) and *Error* detection. I/O behaviors communicate with the decoder through a set of FIFO queue channels. In all target architectures, I/O behaviors and associated queue implementations have been mapped to dedicate hardware I/O units for MP3 input and PCM output where the CPU accesses I/O units through memory-mapped hardware registers and polling.

In a pure software implementation (upper right), the complete *Decoder* behavior hierarchy is mapped onto a single software CPU. Considering that a software implementation of the MP3 algorithm will not meet necessary frame deadlines, increasing levels of hardware acceleration were additionally explored. In a first candidate (lower right), discrete cosine transform (*DCT*) blocks from the PCM synthesis stage of the decoding algorithm were moved into a dedicated, custom synthesized hardware co-processor. In a second stage (lower left), inverse modified DCT (*IMDCT*) blocks in the MP3 frame decoding stage were moved into a second custom hardware unit for acceleration. In both cases, hardware units contain two concurrent instances of DCT and IMDCT blocks in order to exploit available parallelism between left and right channel decoding. Hardware co-processors communicate with the CPU over the main system bus and synchronization is implemented via interrupts. As such, accelerated target architectures contain an additional interrupt controller (*INTC*).

Using SER, transaction-level and pin-accurate models at varying levels of abstraction were automatically generated for each of the design alternatives. All models were simulated to validate functionality and performance. Using SCE tools, models for each target implementations were generated in less than a minute. Furthermore, including time needed for validation and simulation of models, the complete design space exploration process was completed and an optimal architecture was generated in less than an hour.

In all cases, final pin-accurate models are ready for further hardware and software synthesis. DCT and IMDCT hardware co-processors are currently being synthesized down to RTL implementations using the CyberWorkBench high-level synthesis process, targeting an FPGA-based prototyping of the complete MP3 platform.

3.2 SpaceWire Evaluation

SpaceWire (SpW) is a standard for high-speed and high-reliable interconnection networks in space and satellite applications, supporting data communication between high-performance on-board data handling systems and ground stations. SpaceWire is a packet-based network. It supports asynchronous communication with speeds between 2 Mbps and 400 Mbps, fault tolerance to rapidly recover from link failures and arbitrary topologies such as bus, star, ring or tree. Equipment implemented according to the SpaceWire standard is compatible at both component and sub-system levels, reducing cost of development and development timescales.

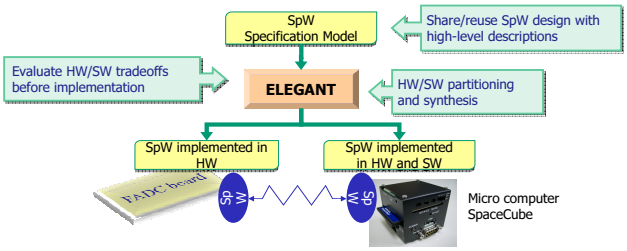

Figure 4: SpaceWire (SpW) evaluation.

Together with its electronic suppliers at NEC Toshiba Space Systems and others, JAXA initiated a project to evaluate a scenario for automated SpaceWire design using the ELEGANT tool set. As shown in Figure 4, starting from a top-level specification model, the SpaceWire protocol is automatically implemented on various targets. Using SER tools, hardware/software partitioning and exploration of the target architectures was performed. In the process, exploration was supported and driven by quantitative estimation (using VisualSpec estimation and profiling capabilities) to help select an appropriate implementation.

Using ELEGANT, a single SpW specification was realized as both a pure hardware implementation on the sensor side as well as a mixed hardware/software design for the on-board electronics. Both design variants were successfully implemented and synthesized using ELEGANT, SER and Cyber tools. For prototyping of the mixed HW/SW solution, it was implemented on JAXA's *SpaceCube* platform consisting of a MIPS-based proprietary CPU, several FPGAs, peripherals and external interfaces. As a result, a demonstrator running an application on the SpaceCube that reads out sensor values over the synthesized SpaceWire implementations on both sides was successfully built and showcased at various trade shows such as DAC'07.

3.3 MPEG4 Evaluation

In addition to the SpaceWire evaluation by its suppliers, JAXA initiated a project for evaluation of ELEGANT by independent consulting and design service company Applistar, Inc. using an MPEG4 application supplied by Japan's Semiconductor Technology Academic Research Center (STARC) [15]. Using ELEGANT, a single, top-level SpecC specification of the MPEG4 decoder algorithm was implemented on JAXA's proprietary HR5000 platform. HR5000 is built by High-Reliability Components Corporation (HIREC) of Japan around a 64-bit MIPS 5kf core [13] specifically for space applications. Several architectural alternatives were explored by varying the amount of FPGA-based hardware accelerators attached to the CPU via its banked and configurable memory bus interface.

Table 1 shows the results of the various MPEG4 decoder explorations. In all cases, the main body of the MPEG4 algorithm is executed in software on the MIPS core. In addition to a pure software solution, architectures were realized in which the inverse discrete cosine transform (IDCT) and/or parts of the motion compensation (MC) blocks are implemented in hardware. In all cases, the CPU, the buses and the FPGA are assumed to run at 25 MHz clock speed.

For all alternatives, transaction-level and pin-accurate models of the design were generated using SER. Models of the software behaviors running on the MIPS core were back-annotated with estimated timing data obtained using VisualSpec's FastVeri technology. Using CyberWorkBench, FPGA hardware units were synthesized into RTL implementations and cycle-accurate SpecC models

Design	Dec. delay	Sim. time
CPU	4.176 ms	2.000 s
CPU+FPGA(MC)	3.823 ms	1.827 s
CPU+FPGA(IDCT)	3.119 ms	3.020 s
CPU+2xFPGA(MC,IDCT)	2.761 ms	2.917 s

Table 1: MPEG4 evaluation results.

directly from the SER-generated pin-accurate SpecC code. Resulting cycle-approximate CPU and cycle-accurate hardware models were then co-simulated in VisualSpec's SpecC simulation environment using the same testbench for all designs.

Simulation results (Table 1) confirm the effect of hardware acceleration for performance gains: moving functionality into hardware generally increases performance and reduces MPEG4 decoding delays. A design with hardware implementation of both MC and IDCT blocks has by far the best performance and can be considered optimal.

4. SUMMARY AND CONCLUSIONS

ELEGANT is an ESL design solution originally commissioned by the Japanese Aerospace Exploration Agency (JAXA). It combines several design tools under a common, SpecC-based framework. ELEGANT provides a seamless tool chain from top-level specification model down to embedded HW/SW system implementation, and it forms a complete ESL tool set for modeling, synthesis and verification. Model generation and high-level synthesis achieve high productivity and flexibility for exploration of system architecture and hardware/software implementation. Simulation and verification support rapid prototyping, performance evaluation and functional validation of system design specification and architecture.

At the core of ELEGANT, the SER component provided true system-level design, exploration and synthesis. Given an abstract, high-level specification of the desired system functionality, SER allows interactive architecture partitioning, network topology definition and bus communication synthesis. SER then automatically implements the given specification on the user-defined platform and generates transaction-level and pin-accurate models of the design at varying levels of abstraction. As such, SER automates tedious and error-prone tasks such as model rewriting while leveraging the human expertise and insight for high-level decision making, system architecture definition and design space exploration.

SER supports applications with arbitrary task graphs hierarchically composed in a sequential, parallel or state machine fashion. Tasks can communication via a rich set of communication primitives such as shared variables, events, semaphores, queues or synchronous and asynchronous message passing. SER will automatically implement an application on a user-defined target architecture. Computation consisting of behaviors and variables is mapped down to hardware and software processors and memories. Communication channels, on the other hand, are implemented over the buses, bridges and transducers forming the system communication architecture. In summary, SER supports automatic implementation of an abstract, high-level platform programming model on a wide variety of target architectures with multiple CPUs, custom hardware blocks and memories connected by a network of buses, bus bridges and transducers.

Using ELEGANT and SER, accurate platform models for virtual prototyping of all design can be generated within minutes. Compared to traditional ISS-based prototypes, generated models simulate extremely fast, enabling rapid and early design space exploration. As such, generated models provide a feasible alternative to traditional ISS-based virtual prototyping and validation. Furthermore, the complete RTL code for all hardware units covering the whole range from application functionality down to pin-accurate bus interfaces and bus protocol state machines is automatically synthesized from high-level models, ready for download into FPGAs or futher hardware logic and physical design.

Several phases of ELEGANT development have been successfully completed and tools are currently being introduced and deployed for first production use in JAXA suppliers such as NEC Toshiba Space Systems. Initial evaluation results have proven the feasibility of the approach for design space exploration, rapid virtual prototyping and system synthesis resulting in tremendous (1000x) productivity and reliability gains. The ELEGANT system has also been been commercialized and is being introduced into the general market. Specifically, the SER component has been licensed to InterDesign Technologies, Inc. (IDT) and it is available from, sold and supported by IDT.

5. REFERENCES

[1] Center for Embedded Computer Systems. SpecC home page. `http://www.cecs.uci.edu/~specc`.

[2] CoWare. Virtual Platform Designer. `http://www.coware.com`.

[3] R. Dömer, A. Gerstlauer, and D. Gajski. *SpecC Language Reference Manual, Version 2.0*. SpecC Technology Open Consortium, `http://www.specc.org`, December 2002.

[4] R. Dömer, A. Gerstlauer, J. Peng, D. Shin, L. Cai, H. Yu, S. Abdi, and D. Gajski. System-on-Chip Environment: A SpecC-based Framework for Heterogeneous MPSoC Design. *EURASIP Journal on Embedded Systems*, 2008.

[5] Forte Design Systems. Cynthesizer. `http://www.forteds.com`.

[6] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong. *Specification and Design of Embedded Systems*. Prentice Hall, 1994.

[7] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Design Methodology*. Kluwer Academic Publishers, 2000.

[8] A. Gerstlauer, R. Dömer, J. Peng, and D. D. Gajski. *System Design: A Practical Guide with SpecC*. Kluwer Academic Publishers, 2001.

[9] A. Gerstlauer, D. Shin, S. Abdi, P. Chandraiah, and D. D. Gajski. Design of a MP3 decoder using the System-On-Chip Environment (SCE). Technical Report CECS-TR-07-05, Center for Embedded Computer Systems, November 2007.

[10] T. Grötker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Kluwer Academic Publishers, 2002.

[11] InterDesign Technologies, Inc. VisualSpec. `http://www.interdesigntech.co.jp`.

[12] International technology roadmap for semiconductors. `http://public.itrs.net`, 2007.

[13] MIPS Technologies, Inc. *MIPS64 5Kf Processor Core Datasheet*, 2001.

[14] NEC System Technologis, Ltd. CyberWorkBench. `http://www.necst.co.jp/product/cwb`.

[15] Semiconductor Technology Academic Research Center. `http://www.starc.jp`.

[16] Underbit Technologies Inc. MAD: MPEG audio decoder. `http://www.underbit.com/products/mad`.

[17] VaST Systems. VaST tools and models for embedded system design. `http://www.vastsystems.com`.