ISSN 1392 – 124X INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS, 2004, Nr. 3(32)

DESIGN OF DISTRIBUTED GENERIC EMBEDDED COMPONENTS

Robertas Damaševičius, Mindaugas Genutis, Vytautas Štuikys

Software Engineering Department, Kaunas University of Technology
Studentų st. 50, LT – 3031, Kaunas, Lithuania

Abstract. The increasingly complex nature of embedded system design is forcing designers to jointly use the available resources, both CAD tools and IP componentry, which are distributed between different design teams. We argue that the existing collaborative design practices demand that the abstraction level in design must be raised above the traditional soft IP. Markup languages can be used to structure and hierarchically organize the domain content. Such structuring makes the representation, processing, and distribution of a domain content much easier. This paper proposes to adopt XML for representation, parameterization, modification and distribution of embedded components. The XML-based representation of soft IPs can be considered as the next generation of IP - distributed generic embedded components (DGECs). We discuss their features, design problems and propose a design framework.

Keywords: collaborative design, distributed generic embedded component, XML.

1. Indroduction

Semiconductor design industry aims to design low-cost microelectronic components and subsystems for embedded systems (ES) as fast as possible, with maximum chance of success. These aims are pursued using pre-designed *Intellectual Property* components (IPs) assembled on a *System-on-Chip* (SoC). This solution frees a designer from the implementation details and allows focusing on integration of IPs at a higher abstraction level.

However, the list of required IPs may become so large and diverse that it is impossible for a single design company to develop, maintain, and distribute the entire catalog of IPs required by their customers. The increasingly complex nature of ES are forcing designers to join their resources: not only in terms of IP libraries, but in terms of distributed IPs, libraries and design tools connected into a global distributed *e-Design* framework, which brings new problems into focus: IP evaluation, IP delivery, IP protection.

Since the IPs developed by the third-party IP vendors usually use different interface standards and have different technological characteristics (clocking frequency, voltage, etc.), the problem of integrating IPs into a system is currently considered a top priority in ES design. We believe that the solution to this problem lies within the concept of a component as an ultimate design abstraction itself.

Abstraction is key to software (SW) design, and it must be so for hardware (HW) and ES design, too. Design complexity can be addressed by raising the level of abstraction to increasingly abstract and formal levels. However, at any level of abstraction, structural design information is as important as functionality. The representation of structural design data is an important problem, because it can be used to pre-specify other design issues, such as reuse and documentation of components and systems. Also customization and adaptation of domain components is much easier to achieve if applied to the structured representations of these components.

Currently existing HW description languages (HDLs), such VHDL, Verilog or SystemC, are not good for structuring domain components, because the structure and behavior of a system is usually mixed in a HDL specification. Such specifications are difficult to analyze and modify. Decomposing a HDL specification into an Abstract Syntax Tree and then analyzing its correctness on the syntactical level are common steps. Such low-level analysis also complicates further application of the derived design data for optimization or customization of domain components.

Therefore, we need to represent domain components more abstractly and structurally than now. One solution could be using a high-level specification language such as UML. For example, UML class diagrams allow representing the structure of domain systems using classes and relationships between them. However, the object-oriented techniques, which are good for conceptual high-level design of ES, may not be suitable when applied for representing individual domain components.

The other solution is to use the so-called *markup languages*. Markup languages are used to markup data, i.e., to encode data with information about itself.

One of the most popular and widely known markup languages is XML [1]. XML is a standard for storing and exchanging hierarchically structured data, as well as for defining other domain-oriented markup languages.

Our contribution is as follows: We (1) analyze and demonstrate the application of the XML technology in the domain of HW design, (2) propose a concept of distributed generic embedded component, which is a next generation of IP for ES design, and (3) discuss its features and design problems.

The structure of the paper is as follows. Section 2 reviews the related works. Section 3 discusses the IP generations used in HW design. Section 4 analyzes the role of XML in ES domain and presents a design framework. Section 5 presents a case study. Section 6 evaluates the features and problems of distributed IP design. Finally, Section 7 presents the conclusions.

2. Related works

Research on e-Design practices in the ES domain focuses on (1) distributed CAD tools, environments and frameworks for collaborative web-based design [2-12]. (2) Virtual libraries (databases, exchange systems, reuse management systems) of IPs [13-15]. These systems usually contribute to the reuse of IP, automatic web-based exchange of design data, IP searching, remote evaluation, simulation and validation of IPs, IP protection and delivery issues. IP exchange systems are usually implemented using the traditional client/server model [16] or different extensions to it (see, e.g., [17]). The proposed systems usually aim at integrating design tools, data and services. However, the aspect of integration of IPs into target systems is usually left to a local designer.

Application of XML technology in ES design domain is steadily increasing. Most often XML is used to represent structured design information such as VHDL components or intermediate design data, and generate design documentation [18-20]. Other researchers use XML to structure the domain at a higher level of abstraction than components and represent the entire ES architectures and platforms [21-23]. In IP exchange frameworks, XML usually serves as an interface between IP provider and IP user as well as represents infrastructure and content of IP catalogues and provides support for IP management and reuse [13, 14]. The most recent research stream focuses on XML as a metalanguage capable of specifying parameterized domain components [24].

Our novelty is that we suggest distributing the effort of design and integration of IPs, by introducing a concept of *distributed generic embedded component* (DGEC), which is a continuation of our previous research on embedded component design [25]. We understand the DGEC within the context and framework of the existing distributed internet-based CAD environments and virtual IP libraries. Therefore, we do not focus on such general aspects as IP protection that should be the same for all types of IP componentry. Instead, we focus on structural representation and parameterization aspects using XML, which are unique to our embedded component model.

3. Generations of IP Components

IP components used in HW design have evolved in generations, which correspond to quantum leaps in IC design methodologies, tools and design practices. According to [26], currently there are three generations of IP. Further, we consider these generations in brief.

The first generation of IP is *hard IP* and *firm IP* cores. Hard IPs are technology-dependant cores that have been fully implemented into the mask-level data required for implementing the block in silicon. They are described in GDSII data format, which is used for transferring/archiving 2D graphical design data. Firm IPs have been floor-planned and synthesized into one or more silicon technologies to get performance, area, and power estimated values. Firm IP is delivered in technology-specific netlist formats.

The second generation of IP is *soft IP* - the most common type of commercial IP today. Soft IPs are specified in RTL code described using a high-level standard HDL, such as VHDL or Verilog, that can be read and interpreted by a synthesis tool. Soft IP has virtually no physical information.

The third generation of IP is user-*configurable customizable (generic) soft IP*. These components represent the application of SW development methodologies such as SW generators, multi-language design, metaprogramming, scripting, preprocessing, etc., to the HW domain. Generic IPs are usually described using multiple languages (e.g., a HDL for expressing domain content, and a metalanguage (scripting language) for parameterization), and may have graphical user interfaces for selection of parameters. The user-guided configurability minimizes the need to perform modifications by hand in order to adapt IPs to the specific SoC requirements.

In [25], we introduced a concept of *generic embedded component* (GEC), which belongs to the third generation of IP. GEC is a parameterized description of a family of reusable domain (SW or HW) components intended for integrating into ES designs. A family includes several (from dozens to hundreds or even thousands) reusable component instances that differ in functionality and characteristics. The environment for generic embedded components is a generic library of IP providers or large ES design organizations.

Additionally, we envision the next generation of IP: *distributed generic embedded component (DGEC)*, which is an embedded component (subsystem, system) assembled from multiple geographically distributed sources and transferred via the Internet to a local designer. The fourth generation of IP represents the application of WWW-based technologies to HW and ES design domain. This vision is in compliance with recent trends towards the distributed collaborative design and E-commerce in future silicon IP market.

In the next section, we describe representation and implementation of DGEC using the XML technology.

4. Implementation of dgec using XML

4.1. Introduction to XML technology

XML [1] is a markup language that provides a common syntax for describing the structure of data according to its content or meaning. XML also is a metalanguage that allows defining domain-specific markup languages for different types of data. Additionally, XML allows to simplify data export, transfer and storage, creation and usage of string libraries and description of the metalevel functionality (e.g., generation of target code).

XML has several advantages as a representation language. The hierarchical organization of data reflects the logical structure of data. The XML tags data embed the information structure within the data, which makes processing of data easier. The advantage of XML over other markup languages is that a designer is not restricted to a limited set of tags defined by proprietary vendors. By defining his own set of tags, a designer can create a markup language oriented at a specific domain of application.

XSLT is a language for describing transformation of XML documents. An XSLT program (*style sheet*) is a set of template rules for transforming a source tree into a result tree. Each template rule has two parts. A *pattern* is used to match nodes in a source tree of the input XML document. A *template* is used to form a result tree of the output XML document. The transformation is achieved by associating patterns with templates. The structure of the result tree can be completely different from the source tree, thus allowing a wide range of transformations.

In the domain of ES design, XML technology can be applied for the following tasks. (1) *Representation* – to structure, hierarchically organize and represent the domain content. (2) *Parameterization* – to describe generic domain components and architectures (patterns). (3) *Modification* – to describe automatic modification of components for adaptation to user- and application-specific requirements and optimization. (4) *Translation* – to translate XML-based domain component representation into domain language(s) of the designer's choice using XSLT. (5) *Generation* – to generate API documentation in various formats using XSLT. (6) *Distribution* – to distribute domain component s between geographically distributed design teams.

4.2. Design framework

The distributed ES design framework is presented in Figure 1. Different DGECs may be located in different remote IP libraries on different servers. The client retrieves the IPs represented in XML (IP_{XML}) and assembles a target system (S_{XML}). The IPs that may require adaptation can be modified using XSLT style sheets that describe design transformation processes (DP_{XSL}). A design process [27] is a representation of a common designer actions aimed at implementing a well-proven domain model or design pattern. Finally, the XML-based representation of a system is translated into HDL (S_{HDL}) and further used for modeling and synthesis.

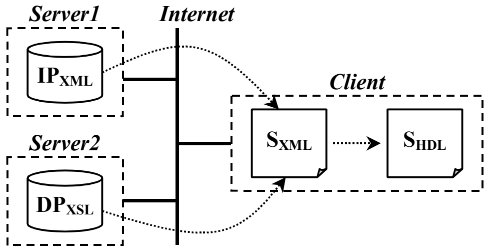

Figure 1. Distributed design framework

An abstract design flow is presented in Figure 2. The translation tools are used to derive the DGEC, which is the XML-based representation of soft IP(s). The transformation code in XSLT is used to modify the DGEC. The result is the XML-based specification of the target system. A lower-level implementation of the target system is obtained when the XML-based specification is translated into the domain language specification using XSLT.

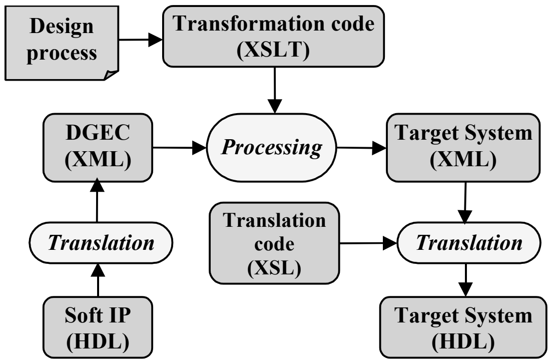

Figure 2. Design flow

5. Case study

In this case study, we demonstrate the application of XML technology for specifying HW wrappers. An example of the wrapper specification for a particular DGEC is presented in Figure 3, a. It uses <WRAPPER> tag to concisely specify a handshake wrapper (HsWr) of the *core_6502* component with common *clock* signal. The wrapped IP is specified using <IP>

tag and is located on a remote server specified using XInclude tag. The DGEC is also represented using XML (see Figure 3, b, the interface is given only). We have adopted the VHDL-based description style, where <ENTITY> tag specifies component interface and <PORT> tag specifies the I/O signals.

```
<WRAPPER type="handshake" clock="clk">
    <IP name="core_6502" xmlns:xi="http://www.w3.org/2001/XInclude">
        <XI:INCLUDE href="http://soften.ktu.lt/~damarobe/xml/free6502.xml"/>
    </IP>
</WRAPPER>                                                              (a)
```

```
<DGEC>
  <ENTITY name="core_6502" source="http://www.free-ip.com/6502/">
    <PORT name="clk"     dir="in"    type="std_logic"         width="1"/>
    <PORT name="reset"   dir="in"    type="std_logic"         width="1"/>
    <PORT name="irq_in"  dir="in"    type="std_logic"         width="1"/>
    <PORT name="nmi_in"  dir="in"    type="std_logic"         width="1"/>
    <PORT name="addr"    dir="out"   type="std_logic_vector"  width="16"/>
    <PORT name="din"     dir="in"    type="std_logic_vector"  width="8"/>
    <PORT name="dout"    dir="out"   type="std_logic_vector"  width="8"/>
    <PORT name="dout_oe" dir="out"   type="std_logic"         width="1"/>
    <PORT name="we_pin"  dir="out"   type="std_logic"         width="1"/>
    <PORT name="rd_pin"  dir="out"   type="std_logic"         width="1"/>
    <PORT name="sync"    dir="out"   type="std_logic"         width="1"/>
  </ENTITY>
</DGEC>                                                                 (b)
```

Figure 3. XML-based specifications of handshake wrapper (a) and DGEC (b)

Table 1. DGEC source code increase and Synthesis results of VHDL code

Soft IP	VDHL	XML/XSL increase	Area, cells (soft IP)	Area, cells (HsWr)	Overhead
Free-6502 [29]	**100%**	**10%/30%**	**4670**	**471**	**10%**
i8051 [30]	**100%**	**15%/45%**	**24258**	**1016**	**4%**

We generate a VHDL code from the HsWr specification using XSLT style sheet. The architecture of the HsWr is presented in Figure 4. The Handshake wrapper uses the handshake protocol to control communication of IP with its environment. Note that the IP data signals are represented abstractly as *Data_in* and *Data_out* signals. More details about wrapper architecture can be found in [28].

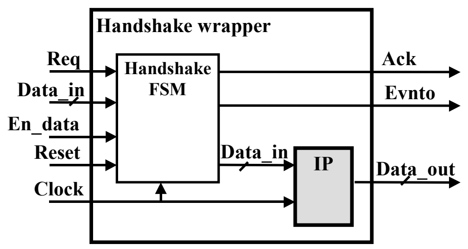

Figure 4. Architecture of handshake wrapper

In our experiments, we have used 2 freely available third-party soft IPs. The synthesis results of the original soft IPs and the generated handshake wrappers (Synopsys; CMOS 0.35 um) using XML are presented in Table 1.

6. Evaluation of Distributed IP design

Collaborative design is gaining more and more support in EDA community, because integration of multiple third-party soft IPs from different IP vendors onto a single chip requires a large variety of design skills that may not be available to a local design team. There are several aspects in distributed CAD: (1) distributed tools, (2) distributed libraries, and (3) distributed components. Further, we will discuss those aspects in detail.

The distributed CAD environments are based on the following principles: (1) there are many actors in the design process (IP designers, IP providers, IP users), (2) the design information is geographically distributed (distributed IP), (3) the designed system is a joint effort from many design institutions. The latter implies the need for IP exchange networks and IP protection.

Storing generic IPs in distributed libraries has several benefits as follows: (1) the genericity of IPs allows them to be offered to a large number of designers, (2) common library allows all designers to share design data, (3) generic component specifications are separated from component instances.

The features of DGECs are as follows (some of them are inherited from the previous generations of IP): (1) *high level of abstraction* – not less than a (sub-)system level; (2) *technology and language-independence* – DGEC is represented independently from the implementation HDL, i.e., the solution of a particular HDL should be the designer's last choice; (3) *genericity* – DGEC represents families of domain components; (4) *customizability* – DGEC can be adapted to the context of application automatically; (5) *mobility* – DGEC can be easily moved across the Internet, while preserving the requirements for IP protection.

The distributed CAD environment should integrate the existing design tools and IP libraries using common media (Internet), XML-based domain representation formats, integration standards, and legal agreements. It must integrate multiple, independently managed design frameworks and have a flexible structure adapted to new E-business models and electronic services. The entire EDA community currently moves towards a single, integrated, distributed E-design environment, which encompasses distributed CAD tools, IP libraries, IP design, qualification, acquisition and distribution services, and is constantly evolving to meet increesing user requirements.

7. Conclusions

We argue that the next generation of IP should be implemented using XML – a flexible technology for creating domain-oriented markup languages. It allows for convenient representation, access, management, and distribution of various types of the structured domain information, including HW/SW components and ES architectures. Furthermore, the designer has the ability to implement automatic modification of IPs and generation of documentation files in various formats using XSLT style sheets. Systematic application of XML/XSLT for ES design could contribute to increase in design productivity and IP reuse and remote share as well as to provide the better documentation capabilities.

References

[1] **T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler.** Extensible Markup Language (XML). W3C *Recommendation*, 2000.

[2] **M.J. Silva and R.H. Katz.** The Case for Design Using the World Wide Web. *Proc. of Design Automation Conference (DAC 1995), June* 12-16, *San Francisco, CA, USA*, 1995, 579-585.

[3] **L. Benini, A. Bogliolo, G. De Micheli.** Distributed EDA tool integration: the PPP paradigm. *Proc. of IEEE Int. Conf. on Computer Design (ICCD),* October 7-9, 1996, *Austin, TX, USA*, 448-453.

[4] **M.D. Spiller, A.R. Newton.** EDA and the Network. *Proc. of Int. Conf. on Computer Aided Design (ICCAD), November* 9-13, 1997, *San Jose, CA, USA*, 470-476.

[5] **H. Lavana, A. Khetawat, F. Brglez, K. Kerminski.** Executable Workflows, a Paradigm for Collaborative Design on the Internet. *Proc. of Design Automation Conference (DAC'*97*), June* 9-13, 1997, *Anaheim, CA, USA*, 553-558.

[6] **K. Hines, G. Borriello.** A Geographically Distributed Framework for Embedded System Design and Validation. *Proc. of Design Automation Conference (DAC* 98*), June* 15-18, 1998, *San Francisco, CA, USA*, 140-145.

[7] **G. Konduri, A. Chandrakasan.** A Framework for Collaborative and distributed Web-based Design. *Proc. of Design Automation Conference (DAC'*99*), June* 21-25, 1999, *New Orleans, LA, USA*, 898-903.

[8] **A. Fin, F. Fummi.** A Web-CAD methodology for IP core analysis and simulation. *Proc. of Design Automation Conference (DAC* 2000*), June* 5-9, 2000, *Los Angeles, CA, USA*, 597-600.

[9] **M. Dalpasso, A. Bogliolo, L. Benini, M. Favalli.** Virtual Fault Simulation of Distributed IP-based Designs. *Proc. of IEEE Design Automation and Test is Europe Conference (DATE* 2000*), March* 27-30, 2000, *Paris, France*, 99-103.

[10] **K. Yang, A. Windisch, T. Schneider, J. Mades, W. Ecker.** IVE: An Environment for Internet Based Distributed VHDL Design. *Proc. of the* 16th *World Computer Congress, August* 21-25, 2000, *Beijing, China*, 516-525.

[11] **D. Kirovski, M. Drinic, M. Potkonjak**. Hypermedia-Aided Design. *Proc. of Design Automation Conference (DAC* 2001*), June* 18-22, 2001, *Las Vegas, NV, USA,* 407-412.