

Design Methodology for IP Providers

Jürgen Haase
SICAN GmbH, Hannover, Germany

Abstract

Based on their experience in the IP business including a large library of DesignObjects™ (Virtual Components) and lots of design-ins world-wide, SICAN has developed a design methodology for production of Virtual Components. This methodology includes appropriate design-flows for the IP-development as well as prerequisites and methods for successful application in customers' projects. This production flow will be completed by approaches for evaluation.

1 Introduction

Meanwhile it is accepted as a fact by most designers that for developing Systems-on-Chip comprising tens of millions of gates in the near future the application of reuse-methodology is mandatory. Already today application of reuse pays off in terms of development cost and time-to-market.

As first conclusion most system companies start programs to ensure reuse of the modules designed in-house [1]. On the other hand many companies including design houses, tool vendors and a lot of new start-ups have the vision to become a vendor of Intellectual Property (IP).

SICAN as one of the leading companies in the IP business started very early to provide its know-how to customers in form of reusable Soft Cores. They provided their DesignObjects™ (SICAN's trademark for IP) to a large number of users worldwide. Thus SICAN has accumulated substantial experience, technical as well as commercial, in this new market. Based on that experience this paper wants to present some prerequisites and solutions for being a successful IP vendor.

1.1 Types of IP Providers

IP providers can be grouped in two main categories: internal IP providers (reuse of modules designed in-house) and external IP providers. For external IP providers selling their IP is the main scope of their business. Up to now most internal IP providers are design teams in business units responsible for getting products out, in the future they have to be completed by corporate IP groups in order to make internal reuse a reality [1].

1.2 Types of Soft Cores

Soft Cores (synthesizable HDL-coded IP modules) can be divided into two basic classes as shown in Figure 1.

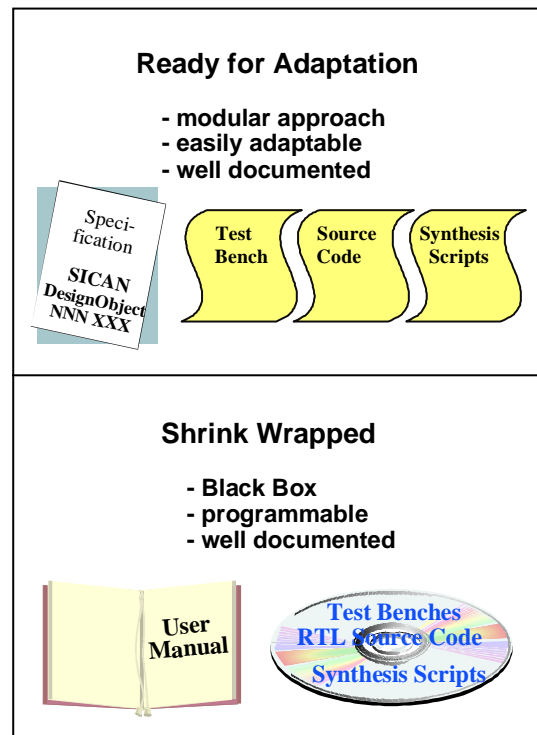


Figure 1: Classification of SICAN's DesignObjects™

”Shrink Wrapped” cores do not require any application specific modification, comply to appropriate standards and often are of small size. Their functionality is well suited for encapsulation. Examples are AC-3 audio decoder or interfaces like I2C-interface. They can be sold in high volumes.

”Ready for Adaptation” cores typically require adaptation to the targeted system. As their interfaces are not clear, they cannot be encapsulated well. They can be very complex, examples are video decoders and encoders compliant to H.263, MPEG-2 etc.

Complex cores have a limited market and are sold only in small volumes (for the Soft Cores) but are targeting high-volume products. Their development requires very substantial application and system know-how.

2 Fundamentals for IP Vendors

2.1 Goals and related problems

The mission of the IP provider should be to help the customer solve his problems and reach his goals. These goals include to benefit from reuse as follows:

- solving the key problems: handling of complexity, verification
- solution for design productivity problem: complexity increases 32% per year, design productivity only 20% per year
- reduced time to market
- cost reduction
- verified design with proven functionality
- extended functionality of the end product
- focusing on core competencies

To develop the IP the market demands with highest quality and as cost-effective as possible is the main goal of the IP provider. This requires addressing some prerequisites for successful IP products:

- market research
- design automation and productization of core development
- test environment
- support of differing design platforms
- scalability of Soft Cores
- parametrization of Soft Cores
- product documentation
- product support

In addition the IP Provider has to take into account the problems the IP user is facing:

- how can the customer verify that the core will play in his application and has the required functionality (behavioral models of the modules etc.)
- can the core be easily integrated in the customer’s design
- integration of cores from different suppliers
- automation (of synthesis etc.)
- verification of the modules’ functionality in the overall System-on-Chip
- test-reuse: test of the cores after production of the chip (boundary scan, BIST etc.)

The following chapters will give some approaches to tackle the listed problems.

2.2 Deliverables

For IP products fully verified functionality, compliance to standards etc. is mandatory. One approach to differentiate IP products from others is pricing, but much more important are the deliverables. SICAN recommends and provides the following deliverables for the three steps of purchasing Soft Cores:

- Evaluation support (loadable from web):
 - + datasheet
 - + interface specification
 - + behavioral model
 - + gate count estimation and memory sizes
- Deliverables for Soft Cores:
 - + user manual or technical specification
 - + implementation manual
 - + description of verification strategy
 - + test specification
 - + behavioral model (C, HDL, Cossap), optional
 - + RTL Source Code (VHDL and/or Verilog)
 - + verification set-up
 - + makefiles, scripts
 - + HDL testbench
 - + synthesis scripts and synthesis constraints
- Service and product support
 - + application and implementation support (e-mail, hot-line, on-site)
 - + training on-site

- + maintenance
- + updates and upgrades
- + guaranteed functionality and compliance to standards
- + technology mapping (optional)
- + test insertion (optional)
- + floor planning (optional)

2.3 Documentation

One of the most crucial issues of reuse is a good documentation. The documentation should contain all information the customer needs in order to handle the DesignObjects™ in his environment and to integrate them in his system. First of all a set of documents with a uniform format needs to be defined, which is provided with each DesignObject™. Two different types of documentation can be defined:

1. "Black Box"-Manual: Used for "Shrink Wrapped"-type DesignObjects™, which can be delivered "off-the-shelf".
2. "White Box"-Manual: Used for "Ready for Adaptation"-type DesignObjects™.

The "Black Box"-Manual consists of two parts. A functional part describes the functionality, architecture and interfaces of a DesignObject™. An implementation part describes the data structure, the way the DesignObject™ was designed and how it can be verified and implemented in the customers' system environment.

The functional part of the "Black Box"-Manual comprises:

1. Features and Functional Overview: *Describes the main features of the DesignObject™ and gives a short functional overview.*
2. Interface Description: *Lists all signals and accessible registers and describes their function and possible settings. Provides timing diagrams to describe the signal handshake.*
3. Programming Guide: *Describes the operation modes and functions of the DesignObject™ and how it can be programmed or configured.*

The implementation part of the "Black Box"-Manual contains:

1. Data Structure: *Provides information about the DesignObject™ database. This includes all information, which is necessary to handle the*

database (e.g. directory structure, design structure, setups, etc.).

2. Behavioral Model: *Gives all information, that is necessary to understand, compile and use the behavioral model of the DesignObject™.*
3. Implementation Information: *Provides information about the design and synthesis methodology. This includes information about clock tree, internal RAMs, asynchronous design parts etc. and how the DesignObject™ can be mapped to a target technology. Additional information is given to characterize the design in terms of frequency, size and power.*
4. Verification Information: *Describes the methodology which is used to verify the design. Gives information about the testcases and how new testcases can be created.*
5. Test Information: *Gives information about the implemented test structures.*

The "White Box"-Manual extends the "Black Box"-Manual by all information required to modify the internal structure of the DesignObject™.

The key to reusability is not only the transfer of reusable code, but the transfer of the knowledge to handle the functionality therein. The importance of documentation quality cannot be overestimated and is one of the key characteristics for an IP provider.

2.4 Verification

The verification strategy is the key to success for the customer, design reuse must include test reuse. SICAN prefers a verification method based on file I/O with stimuli and reference data generated from C-models which results in an HDL independent verification environment (see Figure 2). The testbench itself is self-checking which enables an automatic verification flow. All required tests are stored in a test data base. Certain critical tests are tagged as regression tests.

After code modification it is possible to run regression or full tests without any user interaction. Reports are automatically generated indicating success or failure of the executed simulations. All files to be delivered to customers are under control of a revision control system. In case of a delivery all delivered files are tagged with a symbolic name representing the version being delivered. Branches are generated for each

customer in which the environment of the customer is mirrored.

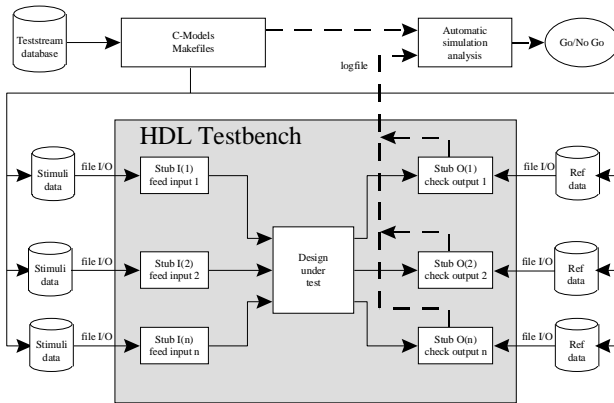


Figure 2: Testbench environment

2.5 Business Aspects

Every guideline for application of Virtual Components includes evaluation of the Virtual Components available on the market. The usual solution is to provide behavioral models of your Virtual Component that enable the customer to evaluate its functionality with respect to his demands. However, in practical projects due to tight schedules it is often not possible to perform lengthy evaluations. Thus the customers need other sources for qualifying the Virtual Component, preferred are working silicon and references from previous customers. For "Shrink Wrapped"-type Virtual Components guaranteed datasheets are accepted.

Besides identifying Virtual Components with the required functionality it is essential to find adequate business models. Possible models range from one-time licenses to licenses with unlimited use, with and without royalties. Having full information about available IP the customer can take his make-or-buy decision, taking into account:

- Doing the design himself requires special know-how and available resources (designers).
- Application of available IP allows for reduction of time to market, which is far more important than minimizing development cost.
- IP with proven functionality reduces risk.
- Use available IP for standard functionality, concentrate your resources on differentiating your designs from competition's products.

It is agreed upon, that the effort for developing high quality IP is approximately 2.5 times the effort of classical designs. The IP vendor has to make sure to get the return on investment. Profitability of developing Virtual Components depends on pricing and on the number of sold licenses.

Internal IP providers face the problem, that the business unit's teams are under pressure to get products out, productization of cores and providing the core support will have minor priority for them. As minimal solution they require internal business models that reward groups for developing reusable cores, in addition corporate core groups are required (to some extent) as internal IP providers to make internal reuse a reality.

IP Providers, internal as well as external, need to develop their chip designers to IP-library builders, which requires a special mind-set and training.

The key to success is customer support, see the section on deliverables for details on service.

3 IP Production

One possible approach for providing IP is to start development from existing modules, which requires productization of these modules to reusable cores. However, this should only be a short-term approach. IP providers have to provide not merely designs but real IP products, requiring replacement of the classical design flow by a product development process. This process has to include:

- methodology driven design
- documentation standards
- naming styles, coding conventions, coding standards [2]
- standardized design flows
- core quality standards
- compliance to standards (application specific like MPEG, reuse-specific like VSIA)
- reviews
- quality assurance
- release management

3.1 Workflow Management

The IP product development process should be supported by a workflow management methodology, which guides the designer through the development process as well as through the whole product life

cycle, manages all data related to the product and allows for automation of all steps in the product development process.

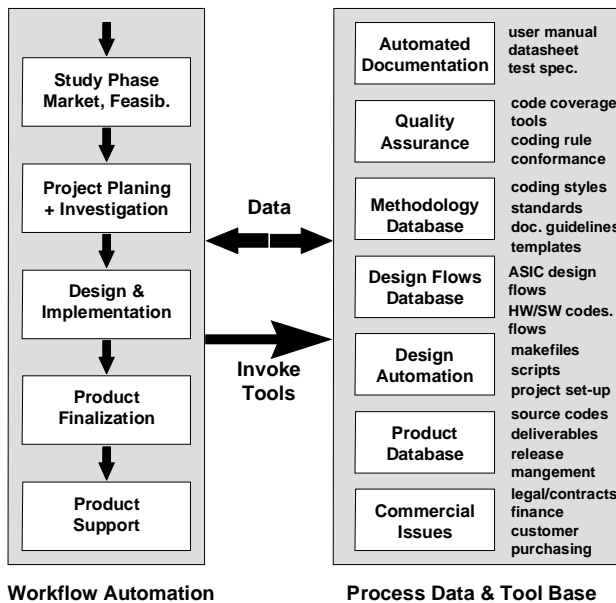


Fig. 3: IP Product Development Process

As shown in Figure 3, this methodology should enable and support

- design data management
- documentation management
- design automation
- invoking tools, makefiles, scripts for synthesis etc. automatically
- release management
- maintenance

3.2 Scalable DesignObjects™

The IP provider faces the problem to support more than one HDL. The best solution for the IP provider regarding effort and costs for implementing, verifying and maintaining the DesignObjects™ would be to have one source code base and derive different versions of the DesignObject™ from that source code base automatically [3],[5].

Such parametric or scalable DesignObjects™ provide the means to tailor their functionality to what the customer really requires. Unfortunately neither VHDL nor Verilog include this as part of the language definition. Although VHDL and Verilog offer methods to pass parameters or generics to a module to scale

certain values like data path width, this kind of scalability does not allow to manipulate the overall architecture or functionality of a module.

A solution to extend the VHDL and Verilog functionality is to use C-like preprocessing to generate the desired derivative from the source code base. The source files have to be extended by certain keywords to mark areas of the code belonging to a certain part of the functionality of the DesignObject™. A preprocessor removes these lines from the code base and extracts only the VHDL or Verilog code indicated by the setting of the preprocessor directives.

4 Conclusions

This contribution presented SICAN's approach for development of IP products based on their technical and commercial experience in this field. Besides developing its own reuse methodology an IP vendor has to comply to standards. The VSIA (Virtual Socket Interface Alliance) does a lot of important work on this. However, by providing standards for the technical issues only one part of the problems is solved. Making reuse a commercial success requires the development of adequate business models for selling and applying IP. It will take some time until any standard business models become accepted on a worldwide basis. In addition to that the user of IP products has to change his design approach, too. He has to overcome "not invented here"-problems in his teams and has to establish a practice to work with the cores and functionality available on the IP market.

5 References

- [1] U. Schlichtmann, B. Wurth: "Implementing a Corporate Design Reuse Strategy", Proceedings IP 98 Europe, Sept. 1998
- [2] M. Keating, P. Briccaud: "Reuse Methodology Manual", Kluwer Academic Publishers, 1998
- [3] J. Haase, T. Oberthuer, M. Oberwestber: "Design Methodology for IP Providers", in: "Reuse Techniques for VLSI Design", Kluwer Academic Publishers, 1999
- [4] R. Seepold: "Survey of Reuse", Proceedings HW/SW Codesign Conference, Sept. 1998
- [5] J. Karrfalt, T. Oberthuer: "IP Design Flow Centers on Automatic HDL Translation", Integrated System Design (ISD) Magazine, April 1998