ipPROCESS: A Development Process for Soft IP-core with Prototyping in FPGA

Marília Lima, Francielle Santos, João Bione, Tiago Lins, Edna Barros
Informatics Center. Federal University of Pernambuco
Recife, PE, Brazil 50740-400
{msml, fss, jfbs, tsl, ensb}@cin.ufpe.br

Abstract

Due to its increasing complexity the design of Intellectual Property cores (IP-core) have been a challenge for designers. The IP-core design involves many and distinct 'areas of concern' like specification, functional verification, simulation model, synthesis and prototyping. All these characteristics together make the design of an IP-core a complex task and demand a development team with different expertise. In this paper, we propose a development process for Soft IP-core design, called ipPROCESS, as a mechanism to facilitate and speed up designers learning and improve their productivity, as well as a support for designing complex IP-cores. Following the steps defined in the ipPROCESS, a designer's team may increase its chances to accomplish the design successfully due to the better understanding of all team members about the IP-core under development.

1. Introduction

The silicon capacity continues to increase along Moore's Law allowing us to build complex chips consisting of hundreds of millions of transistors. Furthermore, there are a new consumer market including more information appliances in small, mobile and ergonomic devices that provide information, entertainment and communications capabilities to consumer electronics, industrial automation and medical markets. These devices require complex electronic design and system integration, which must be delivered in the short time-to-market frames of consumer electronics. These new and even more features results in even more complex design architectures [C+99].

The demand for even more powerful products, and the increasing capacity of today's silicon technology have moved the design methodology to the system abstraction level. The integration technology supports nowadays the integration of a complete system in silicon (System-on-chip (SoC)) and design methodologies are more and more based on pre-defined and pre-designed Intellectual Property blocks (IP-core). The reusing of IP-cores has been an alternative to reduce the increasing gap between design productivity and chip complexity of emerging SoC designs [KB02]. As the complexity of the products under design increases, the need for development efforts increase exponentially. To keep these efforts in check, a design methodology that favors reuse and early error detection is essential [C+99]. Both reuse and early error detection imply that the design activity must be defined rigorously, so that all phases are clearly identified and appropriate checks are enforced. To be effective, a design methodology that addresses complex system has to start at high level of abstraction [C+99].

The design of an IP-core involves different 'area of concerns' like specification, implementation by using hardware description languages (HDL), simulation, functional verification, synthesis, prototyping and intellectual property protection. All these design aspects demand from designer teams various abilities in distinct expertise, as well as mechanisms to support teamwork. It is necessary to cope with the design of a complex system by dividing them in various modules, which can be designed by distinct team members. Efficient and correct communication mechanisms among them must be provided in order to accomplish the design successfully. Furthermore, the design of IP-cores has its own challenges like portability, reusability, standard interfaces, well-defined and useful documentation, project management and facility for integration [KB02].

Currently, one parameter for characterizing IP-cores is the degree to which the IP-core has been targeted toward a particular fabrication process [VSI97]: Soft IPs are delivered in the form of synthesizable RTL code and have the advantage of being more flexible and the disadvantage of not being as predictable in terms of timing, area and power; Hard IPs are optimized for power, size, or performance and mapped to a specific technology. They have the advantage of being much more predictable, but consequently are less flexible and portable due to process dependencies; Firm IPs are optimized in structure and in topology for performance and area through floorplanning/placement, possibly using a generic technology library. Firm IPs offer a compromise between Soft and Hard IPs. More flexible and portable than Hard IPs, yet more predictive of performance and area than Soft IPs, Firm IPs include a combination of synthesizable RTL, reference technology library, detailed floorplan, and a full or partial netlist.

In order to cope with some challenges of IP-core design, we propose, in this paper, a development process, called ipPROCESS, for Soft IPs with prototyping in FPGAs. The ipPROCESS approach defines the IP-core design task as a set of activities, where each activity determines 'what should be done, when and for whom', i.e., the process assigns activities and responsibilities to the right person at a right moment of the design lifecycle. According to the proposed methodology, the life cycle design of an IP-core starts by eliciting requirements and constraints. After that the IP-core structure, functionalities and behavior should be defined during the analysis design phase. In this phase, the structure, functionalities and behavior are modeled by using UML and Real Time UML [uml]. Only after this phase, the implementation in some HDL should take place. Due to the increasing complexity of IP-cores it is very important that the design teams acquire a clear and unique understanding of the IP-core functionality and behavior. The RTL Implementation phase aims to result a RTL specification of the IP by using some hardware description language. This phase can start with a behavioral description, which is refined until a RTL description is obtained, which can be synthesized. This refinement can be done manually or automatically, by using some synthesis tool. Concurrently with the RTL implementation, the functional verification must take place in order to guarantee that the RTL description have the same behavior as a reference model. The last phase is the prototyping phase, which aims to produce a prototype in FPGA.

By defining all these phases as a set of well defined activities with actors and roles, we expected to contribute by improving the design productivity due to increasing probability of earlier error detection once the design starts at a higher level of abstraction, and due to the more reliable communication among team members supported by the use of the ipPROCESS concepts. The learning of the distinct aspects of a Soft IP design may be improved, once the ipPROCESS makes the design activities more predictable and clarify the abilities to execute them. Additionally, the ipPROCESS is described in SPEM, a UML profile, which can be used as CASE tool input, which can generate a management support for all process activities [iri05].

The next sections are organized in the following way: section 2 presents the most relevant related works, section 3 introduces the ipPROCESS concepts in more details, how the process has been modeled and defined, section 4 presents a case study, and finally, section 5 presents some conclusions and important features for next versions of the proposed approach.

2. Related Work

In line with the design for reuse trends, the Virtual Socket Interface Alliance (VSIA) has been formed by a group of major Electronic Design Automation (EDA) and Semiconductor companies with two goals. First, to establish a unifying vision for the chip industry, and second, to develop the technical standards required to enable the most critical component of that vision: the mix and match of IP-cores from multiple sources. VSIA later expanded that vision to meet the growing needs of the IC Industry by including software and hardware IP for SoC design [vsi]. The VSI Alliance intends to define, develop, ratify, test and promote open specification relating to data formats, test methodologies and interfaces, which will facilitate the mix and match and the reuse of intellectual property blocks from different sources in the design and development of system on chip [VSI97].

Another relevant work is the Reuse Methodology Manual (RMM), which outlines a set of best practices for creating reusable ASIC designs for use in a SoC design methodology. These practices are based on the authors' experience in developing reusable designs, as well as on the experience of design teams in many companies around the world [KB02]. The RMM is a chronicle of the best practices used by the best teams to develop reusable IP-cores and, as the VSIA, addresses the concerns of two distinct audiences: the creators/providers of IP-cores and chip designers who use or integrate these IPs [KB02,vsi].

The Rational Unified Process (RUP) is a software engineering process that provides a disciplined approach to assign tasks and responsibilities within a development organization. At the RUP context, a 'discipline' is a collection of related activities that are related to a major 'area of concern'; all activities related with the management of the project are grouped into a discipline called 'Project Management', for example. The RUP goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget [Kru98]. Furthermore, the RUP has a set of `Best Practices` that are important in the context of the reuse and early error detection such as: manage requirements and change, model visually (UML), use component architectures and continuous verify quality. Still from the Software Engineering, there is another important approach: the eXtreme Programming (XP). XP is actually a deliberate and disciplined approach for software development. This methodology emphasizes teamwork: manager, customers, and developers are all part of a team dedicated to delivering quality software [xp]. XP focus on the quality of the final code emphasizing on tests automation and in the use of a test suite for regression and validation test. The XP approach aims to adopt a set of simples Rules and Practices related to Planning, Designing, Coding and Testing [xp].

The first two works cited above, VSIA and RMM, do not intend to develop specifications related to the internal design of IP-cores, functional architectures of subsystem components nor fabrication processes [KB02,VSI97]. In other words, the VSI Alliance and the Reuse Methodology Manual are more concerned with the IP-core deliverables than how an IP-core is designed. So, the main idea of the ipPROCESS is to fill this gap by defining a Soft IP development process that describes, step-by-step, how it can be designed. Based on the RUP and XP processes, the ipPROCESS proposes a disciplined and documented way to design quality soft IP-cores, where the project lifecycle starts from a high-level abstraction analysis of the system to be designed. The ipPROCESS main concepts and details are described in the next section.

3. The ipPROCESS

As said in the previous section, the ipPROCESS has been defined based on two software processes: RUP and XP. The main idea is to use the expertise of software engineering in designing software systems by starting from a high abstraction level. In other words, the designer should conceive a system without implement it in some hardware description language. From the RUP processes, two disciplines related to high abstraction level have been adopted: Requirements and Analysis & Design. The Requirements discipline aims to eliciting and defining the requirements, whereas the Analysis & Design discipline, includes activities for architecture definition, and behavioral modeling from the elicited requirements. From the XP methodology Rules and Practices that enforce quality have been adopted. The reader can observe that the ipPROCESS includes various concepts of software engineering, which supports to start the design at a very high abstraction level. This allows the detection of errors at earlier design phases ensuring the quality of the final implementation.

The ipPROCESS concepts, activities and results have been modeled with the SPEM Metamodel. The SPEM, or Software Process Engineering Metamodel, is a UML Profile1 for defining processes and its components [spe]. In the following, the main constructors of SPEM 1.0 are described. These constructors have been used for modeling the ipPROCESS [spe,KB02].

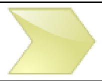
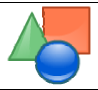
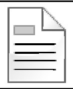
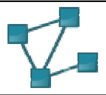
Stereotype /Concept	**Description**	**Notation**
Phase	Is the time between two major project milestones, during which a well-defined set of objectives is met, artifacts are completed, and decisions are made to move or not move in the next phase	
Discipline	Is a collection of activities, which are related to a major 'area of concern'.	
WorkFlow	Is a group of activities which are performed in close collaboration to accomplish some result. The activities are typically performed either in parallel or iteratively, with the output from one activity being the input to another activity	
Activity	Is the unit of work that a role may be asked to perform	
Role	Is the assignment of abilities and responsibilities to a person from the development team	
WorkProduct or Artifact	Is anything produced, consumed or modified by a process. Examples include plans, coding, scripts, documents etc.	
Guidance	Used to provide more detailed information to practitioners about the associated activity. Examples: Guidelines, Techniques, Tool mentors, Checklists, Templates etc	
Document	Is a kind of workproduct and represents a text document	
UML Model	Is a kind of workproduct and represents the UML diagrams	

Table 1: SPEM 1.0 Definitions and Notation

Figure 1 shows the overall architecture of the ipPROCESS, that has two dimensions: the horizontal axis represents time and shows the lifecycle aspects of the process, and the vertical axis represents disciplines, which group activities logically by its nature.

[1] A UML Profile is a kind of variant of UML that uses the extensions mechanisms of UML in a standardized way, for a particular purpose.

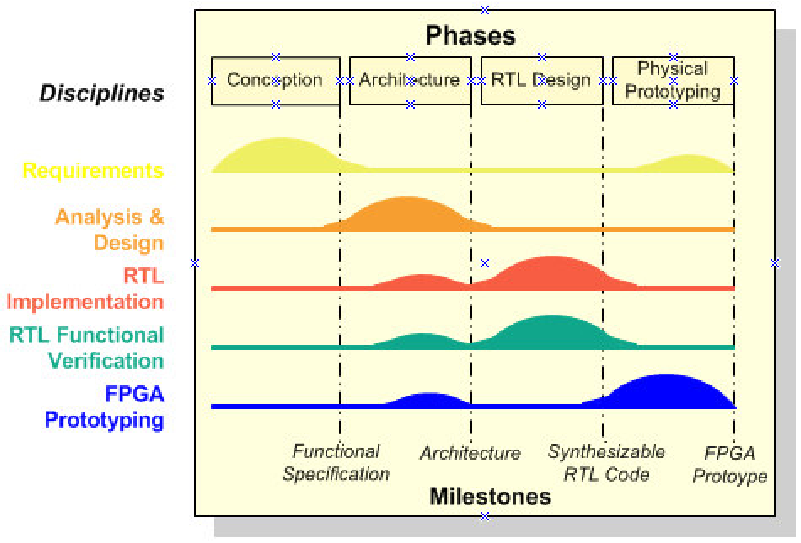


Figure 1: ipPROCESS Structure Overview

The first dimension represents the dynamic aspect of the process, i.e. how activities are distributed over the time. It is expressed in terms of Phases and Milestones. The second dimension represents the structural aspects of the process: how it is described in terms of disciplines, workflows, activities, artifacts and roles. The graph shows how the emphasis on one activity can vary over time. For example, in the first phase more time has been spent on requirements and in the last phase, more time has been spent on FPGA prototyping.

At the ipPROCESS website, [ipp], all details about the concepts listed above as well as the process architecture can be found.

3.1. The ipPROCESS Phases: An Overview

From a management perspective, the ipPROCESS is decomposed over time into four sequential phases, each concluded by a major milestone as show in Figure 2. The first two phases are related with the understanding of the problem ("what the IP should be") and with the modeling of its behavior ("how it should do") using UML and Real Time UML as a modeling language. In these phases, Conception and Architecture, the development team should focus on understanding the requirements and defining the IP behavior before implement it. The implementation should start at the RTL design phase. This delay to start with the implementation (coding on some HDL) aims to create a time slot in the design phase for discussing exhaustively all functionalities, constraints and possible architectures for the IP-core under development.

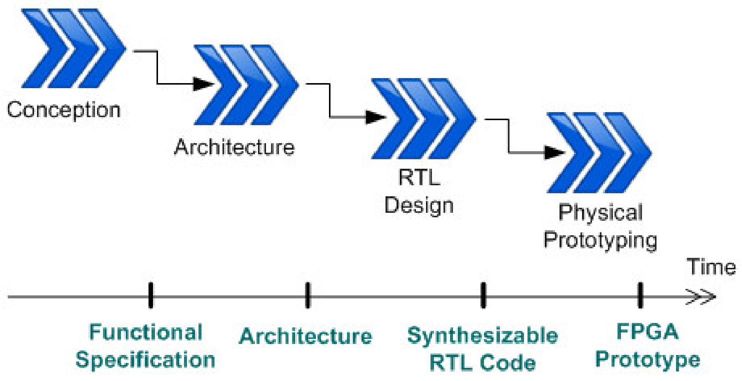


Figure 2: Lifecycle Phases of the ipPROCESS

In the following, the main objectives and milestone for each design phase of the ipPROCESS lifecycle are described.

Conception Phase. The Conception is the first phase of the ipPROCESS, when designing an IP-core. Its goal is to elicit the requirements and to define the scope of the project. The primary objectives include: (1) to understand the functional and non-functional requirements, (2) to define the functional specification and (3) to obtain an agreement about the IP-core scope. The milestone of this phase is a well-defined set of functional specifications. At this point, the evaluation criteria are: (1) an agreement that the right set of requirements have been elicited and that there is a share understanding of them and (2) important terms and expressions have been captured and documented.

Architecture Phase. The goal of this phase is to define the architecture of the IP-core in order to provide a stable basis for the next phase. The architecture of the IP-core must take into account the most significant requirements. The primary objectives of the Architecture Phase include: (1) to define the components of the architecture and its interfaces, (2) to demonstrate that the defined architecture will support the requirements of the IP-core, (3) to plan the components integration and (4) to plan the RTL Functional Verification. The project milestone at the end of this phase is the IP-core architecture (components and interfaces). The evaluation criteria of this phase include: (1) to ensure that the defined architecture supports all requirements and (2) that requirements and architecture are stable enough.

RTL Design Phase. The RTL Design Phase aims the development of a RTL simulation model based on the defined architecture, which can be synthesized. The primary objectives of this phase include: (1) to create the RTL simulation model for the IP-core, (2) to construct the RTL Functional Verification components and (3) to ensure that all detected bugs have been corrected. At the end of the RTL Design Phase the milestone are the RTL simulation model (which can be synthesized) for the IP-core and its testbenches. The evaluation criteria for this phase consist in answering the following questions: (1) are the IP-core functionalities stable enough? and (2)are all the components of the architecture synthesized?

Physical Prototyping Phase. The focus of this phase is to create a physical prototype to ensure that the IP-core can be distributed to its end users (system integrators). The project milestone at the end of this phase is a FPGA prototype of the IP-core (Soft IP). The primary evaluation criteria for the Physical Prototyping Phase consist in answering the following questions: (1) is all the IP-core synthesized and tested?, (2) are the synthesis scripts created? and (3) are the synthesis constraints documented?

3.2. The ipPROCESS Disciplines: An Overview

As mentioned previously, a Discipline includes all activities you may go through to produce a particular set of artifacts and it has been described at a detailed level, called 'workflow details'. A workflow detail shows how roles collaborate, and how they use and produce artifacts [Kru98, spe]. In order to help the description of each discipline, they have been organized into a logical, sequential order of workflow details.

The table below shows the relationship between the ipPROCESS disciplines and the main areas of concern when designing an IP-core. In the following, each discipline is detailed described: the activities, roles and produced artifacts.

Discipline	**“Areas of Concern” of an IP-core design**
Requirements	Specification
Analysis & Design	HW Architecture
RTL Implementation	Implementation and Simulation
Functional Verification	Tests (development and automation)
FPGA Prototyping	Synthesis and Physical Prototyping

Table 2: ipPROCESS Disciplines

Requirements Discipline: the purposes of this discipline are: (1) to establish and to maintain agreement with the costumer on what the IP-core should do, (2) to provide IP-core developers with a better understanding of the requirements, and (3) to define an external interface for the IP-core, focusing on the needs and goals of the end users (system integrators). Figure 3, part (a), shows the sequence of the Requirements discipline workflows.

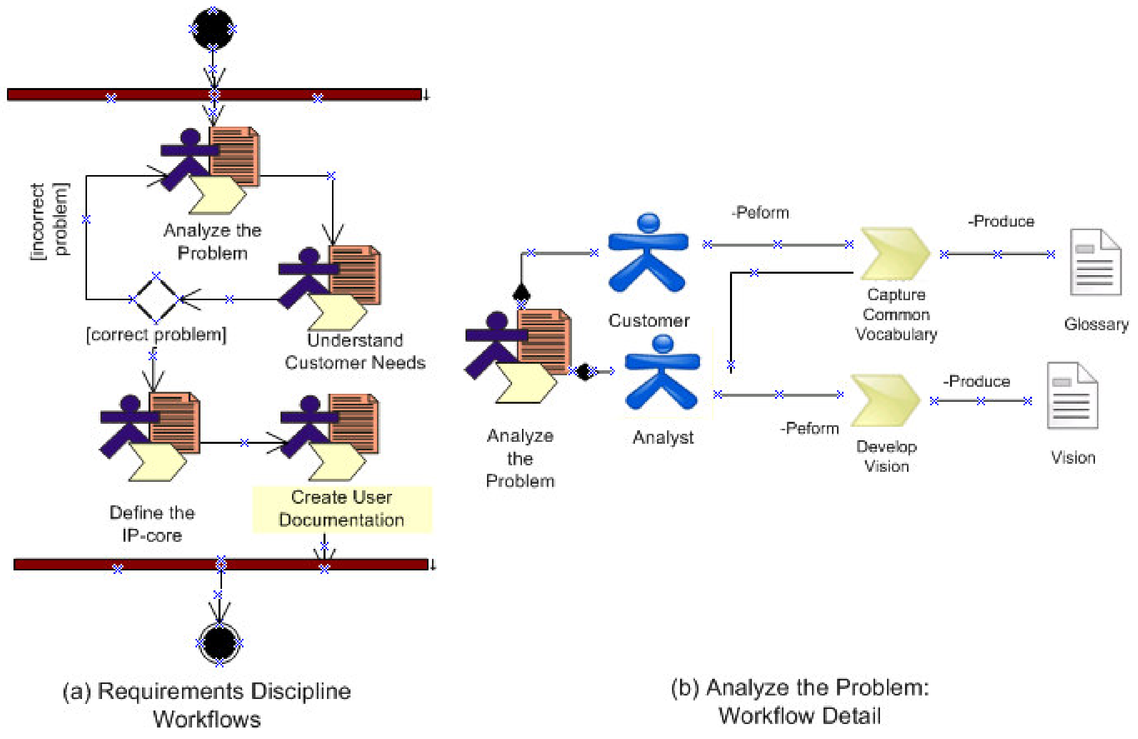


Figure 3: *Requirements Discipline* Workflows

Each workflow represents a key skill that needs to be applied to perform effective eliciting of the requirements. The workflows Analyze the Problem, Understand Customer Needs and Define the IP-core are executed during the Conception phase, whereas the emphasis on Create User Documentation is given during the Physical Design phase. Table 3 describes the main objective of each workflow from the Requirements Discipline.

Workflow	**Description**
Analyze the Problem	It is the first contact with the customer. The main objective is to identify the users that will define the requirements as well as the commonly used terms and expressions.
Understand Customer Needs	The main objective is to identify and detail the requirements.
Define the IP-core	The main objective is to define the scope of the IP-core with the customer.
Create End User Documentation	Create the documents that should be delivered to the end users (integrators) with the RTL code.

Table 3: Description of the Requirements workflows

Figure 3, part (b), shows the details of the workflow Analyze the Problem. The purpose of this workflow is to identify the users that will define the requirements as well as the commonly used terms and expressions. In order to reach this purpose, the Analyze the Problem workflow has been organized into two activities that are executed by the collaboration of two roles(Customer and Analyst), and resulting in two artifacts (Glossary and Vision). The activity Capture Common Vocabulary aims to obtain a common terminology in order to avoid misunderstandings. The common terms and expressions are registered in a document called Glossary. The other activity is the Develop Vision, which aims to identify the end users and its views of the IP-core to be built. This information are organized in a document called Vision.

In the following, a short overview of the others workflows is given. Due to paper length limitations, it is not possible to describe all workflows for each discipline of the ipPROCESS in detail. More information about them can be found in the ipPROCESS website [ipp].

Analysis & Design Discipline: the goals of the Analysis & Design workflows are: (1) to design the IP-core architecture and (2) to assign the functionalities to the components of the architecture. Figure 4 represents the sequence of the Analysis & Design workflows, and all of them are executed during the Architecture phase.

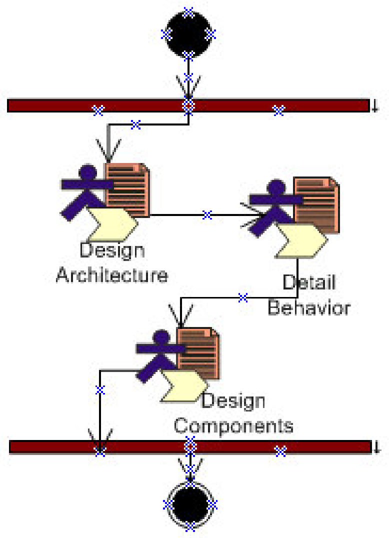


Figure 4: *Analysis & Design* Discipline Workflows

Table 4 describes the main objective of each workflow from the Analysis & Design Discipline.

Workflow	**Description**
Design Architecture	Identify the components of the architecture by taking the IP-core requirements into account. The architecture should be modeled through a Class Diagram using the elements of the Real Time UML profile.
Detail Behavior	The purpose is to identify and understand the functionality of each component. These functionalities should be modeled through a UML Use Case Diagram. [uml]
Design Components	The main objective is to detail the behavior of each component through UML State Diagrams (State Machines). [uml]

Table 4: Description of the Analysis & Design workflows

The Real Time UML (UML-RT) Profile has been chosen for describing the architecture since it defines new stereotypes useful to represent some constructors of hardware description languages [NS+04].

RTL Functional Verification Discipline: the goals of this discipline are: (1) to validate that the requirements have been implemented appropriately, and (2) to find and to document defects in the

RTL simulation model. Figure 5 (part (a)), shows the sequence of the RTL Functional Verification workflows.

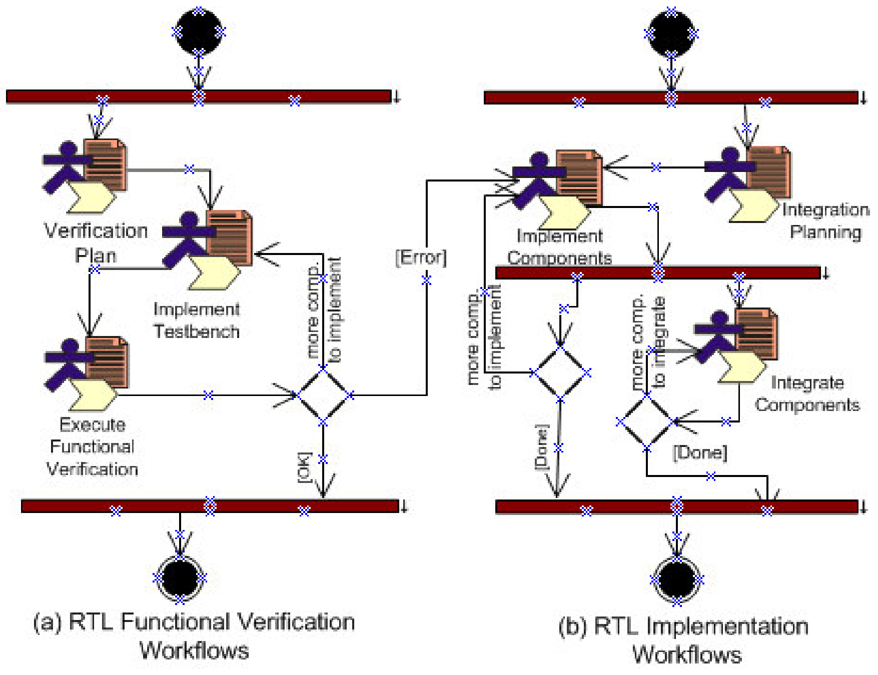


Figure 5: *Implementation* and *Functional Verification* Disciplines Workflows

Verification Planning is executed during the Architecture phase, whereas the emphasis on *Implement Testbench* and *Execute Functional Verification* is given during the RTL Design phase. Table 6 describes the main objective of each workflow from the Analysis & Design Discipline.

Figure 6 shows the functional verification approach that has been adopted in the ipPROCESS. The *Source* is responsible for generating stimulus to the *Design Under Verification* (DUV) and the *Reference Model*, and the *Checker* compares both output from DUV and *Reference Model* [SM+04].

Workflow	**Description**
Verification Plan	The purpose of this workflow is to define how the components of the architecture should be verified and to specify the test cases (stimulus) [SM+04]
Implement Testbench	The objective of this workflow is to implement the testbench components (see fig. 6) and to define its automation (For example, create regression scripts). Using tools to create the UML Diagrams, like Rational Rose [Kru98], the Reference Model (see fig. 6) can be generated automatically from the State Diagrams. [Kru98]
Execute Functional Verification	The main objective is to execute the verification and report the bugs. The *Bug Tracking* Guideline explains the bug lifecycle.

Table 5: Description of the RTL functional Verification workflows

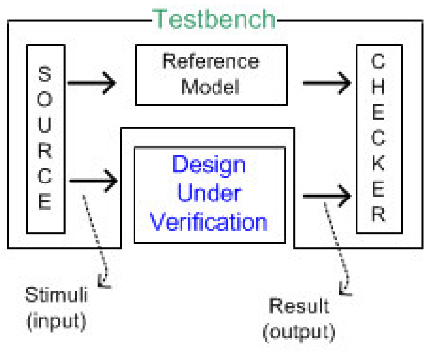


Figure 6: RTL Functional Verification Approach

The DUV is the main result of the *RTL Implementation* Discipline. The goals of the RTL Implementation Discipline are (1) to implement each component defined in the architecture and (2) to define how they should be integrated. Figure 5 (part (b)) shows the sequence of the RTL Implementation workflows. *Integration Planning* is executed during the Architecture phase, whereas the emphasis *on Implement Components* and *Integrate Components* is given during the RTL Design phase. Table 7 describes the main objective of each workflow from the RTL Implementation Discipline.

Workflow	**Description**
Integration Plan	The purpose of this workflow is to define how the components of the architecture should be integrated and tested
Implement Component	The objective of this workflow is to implement each component. The implementation should follow the *Coding Standard* Guideline.
Integrate Components	The main objective is to integrate the components as defined by the Integration Plan.

Table 6: Description of the RTL Implementation workflows

FPGA Prototyping Discipline: this discipline aims: (1) to transform the synthesizable RTL simulation model into a physical prototype in FPGAs, and (2) to validate that the requirements have been implemented appropriately in the physical prototype. Figure 7 represents the sequence of the workflows *FPGA Implementation*. *Integration Planning* is focused during the Architecture phase, whereas the emphasis on *Implement Components in FPGA* and *Integrate Components* in FPGA is given during the Physical Design phase. Table 8 describes the main objective of each workflow from the RTL Implementation Discipline.

Workflow	**Description**
Integration Plan	The purpose of this workflow is to define how the components of the architecture should be integrated and tested in the FPGA.
Implement Component in FPGA	The objective of this workflow is to synthesize each component and test it in the FPGA.
Integrate Components in FPGA	The main objective is to integrate ant test the components in the FPGA as defined by the Integration Plan.

Table 7: Description of the FPGA Implementation workflows

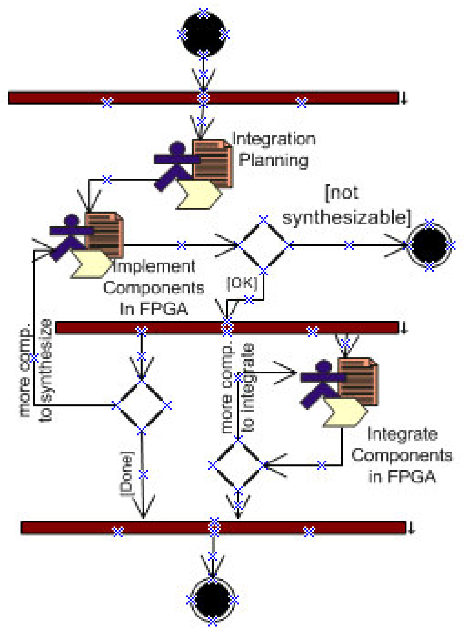


Figure 7: FPGA Implementation Discipline Workflows

4. Using the ipPROCESS for Designing an 8051 Micro-controller Soft IP: A Case Study

The complete ipPROCESS includes 16 class diagrams, 9 tool tutorials and 16 artifact templates all of them available at process site. The ipPROCESS has been used for designing a Soft IP-core for the 8051 Micro-controller. The designed 8051 IP-core (Soft IP) has the following features: 8-bit CPU with a set of 255 instructions, Serial Interface with 4 operation modes, 2 Timer/Counters, 4 Parallel Input/Output ports, 256 bytes Internal RAM, an Interrupt Manager with 4-level of priorities, 12MHz clock and OCP-IP compliance External Interface [ocp].

The project has been developed in ten months (including learn and design time), and the development team has included seven people (undergraduate students all of them without design experience). The 8051 functionalities and features have been implemented in six modules: CPU, serial interface, interrupt, OCP-IP interface, timers and ports. Each team member has performed distinct roles during the project. For example, to the same person was assigned the role of analyst during the Conception phase, whereas in the Architecture phase this person has performed the Designer role and so on. As a Analyst this person has specified one module, as Implementer has implemented another module and when performing the Test Designer role, he/she has verified another one. Through this strategy, it has been provided to each team member: (1) the opportunity to take awareness about the various aspects of a Soft IP design, (2) a broader understanding over the core being developed and (3) the possibility to identify specification errors sooner, thus minimizing the communication and integration errors among the core modules. In the Table 8 are listed all artifacts that have been produced in each phase of the process. All of them are available at the ipPROCESS website [ipp].

Process phase	Artifacts
Conception	Requirements Specification, Functional Specification (use cases in UML-RT – see Figure 8)
Architecture	IP-core architecture (class diagrams – see Figure 9 for OCP architecture)

RTL Design	Synthesizable RTL simulation model: 26,539 lines of SystemC RTL code
	Components for RTL functional verification: 10,566 lines of SystemC TL, test bench with 817,623 test cases (including random), and Regression Scripts
Physical Prototyping	FPGA Prototype in a Xilinx Virtex II XC2V1000-4FG456C, occuping 40% of 4-input LUTS and 9% of slice flip-flops.

Table 8: Soft IP design of an 8051 Micro-controller: phases overview

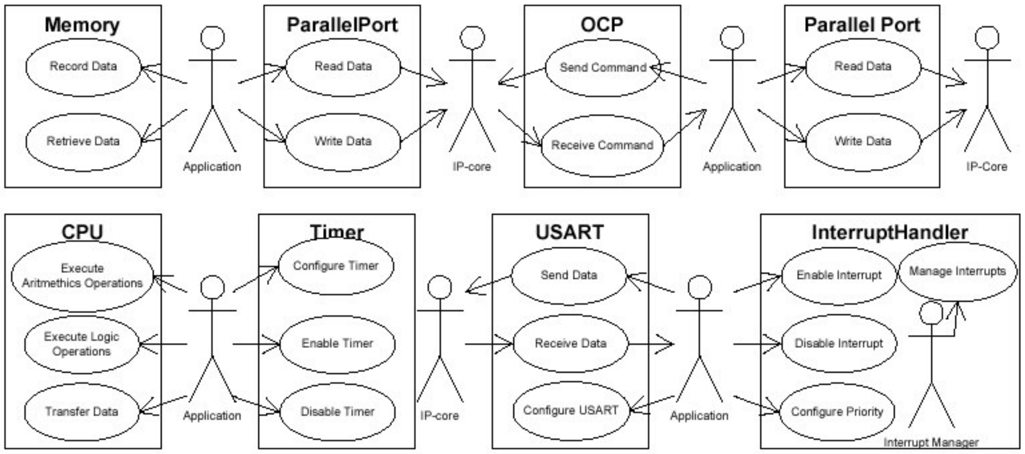


Figure 8: Use Cases for the 8051 micro-controller

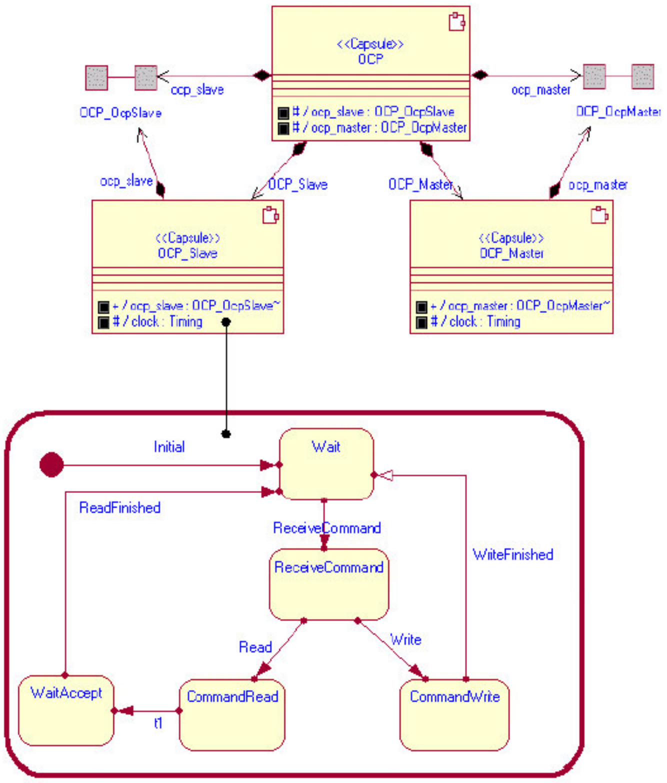


Figure 9: Class Diagram for OCP control

5. Conclusions and Future Work

A process for Soft IP development, called ipPROCESS, has been proposed. The ipPROCESS has been structured in terms of phases and disciplines, where each discipline is related with an 'area of concern' and has been defined in terms of workflows (activities, roles and artifacts) that extend from the understanding of what an IP-core design should provide until its physical prototyping. Its utilization for designing an 8051 Soft IP, as mentioned in the previous section, has shown that pre-defined workflows, document templates, pre-validated coding standards and practical tool tutorials are very useful during the design process. Furthermore, the organization of related activities into disciplines has helped the designers to become aware about the distinct aspects of the IP-core design like specification, RTL implementation, functional verification, synthesis and so on. This broad view of the design process and the support for providing reliable communication among team members, can improve the team productivity when designing a complex IP-core composed of various modules.

As future work we are planning to define/update the specification and templates for all deliverables (artifacts that should be delivered to the system integrators) in order to make them compliant with the VSI Alliance standards [5]. Thus we expect to help the designers to deliver IP-cores that fit the industry requirements. To include additional disciplines and the related workflows for supporting the design of Firm IPs and Hard IPs is also planned in the context of this work.

References

[Ber00] Janick Bergeron. Writing Testbenches: Functional Verification of HDL Models. Kluwer Academic Publishers, 2000. Second Edition.

[C+99] Henry Chang et al. Surviving the SoC Revolution. Kluwer Academic Publishers, 1999.

[ipp] ipPROCESS Website. Available at: http://www.brazilip.org.br/ipprocess.

[iri05] Iris Case Tool: Iris Suite, 2005. http://www.osellus.com.

[KB02] Michael Keating and Pierre Bricaud. Reuse Methodology Manual for System-on-a-Chip Designs. Kluwer Academic Publishers, 2002.

[Kru98] Philippe Krutchen. The Rational Unified Process. Addison-Wesley, 1998.

[NS+04] K.D. Nguyen, Z. Sun, et al. Model-Driven SoC Design via Executable UML to SystemC. In 25th IEEE Real-Time Systems Symposium, 2004.

[ocp] OCP - IP: Open Core Protocol International Partnership. Available at: http://www.ocpip.org.

[uml] UML - Unified Modeling Languague. Available at: http://www.uml.org.

[SM+04] Karina Silva, Elmar Melcher, et al. An Automatic Testbench Generation Tool for a SystemC Functional Verification Methodology. In Symposium on Integrated Circuits and Systems Design, 2004.

[spe] SPEM - Software Process Engineering Metamodel. version 1.0 Available at: http://www.uml.org.

[vsi] VSIA - Virtual Socket Interface Alliance. Available at: http://www.vsi.org.

[VSI97] VSI Alliance Architecture Document, 1997. version 1.0. Available at: http://www.vsi.org

[xp] eXtreme Programming. Available at: http://www.extremeprogramming.org.