DESIGNING A SET OF SOFT IP-CORES FOR GAMES WITH PROTOTYPING IN FPGA BASED ON IPPROCESS

João Carlos Nunes Bittencourt, Anfranserai Morais Dias

Technology Department, State University of Feira de Santana
Av. Transnordestina S/N, Novo Horizonte – 44.036-000 – Bahia, Brazil.
{joaocarlos,anfranserai}@ecomp.uefs.br

ABSTRACT

Searching for new process models which satisfy the embedded systems demands, has been reflected in efforts to develop reusable IP-cores based on the Rational Unified Process (RUP). Among the alternative process, highlight the ipPROCESS, a Brazilian initiative in order to create a standard and enhance the development of integrated circuit design in the country. In this context, this paper presents an application of ipPROCESS designing a set of IP-cores for FPGA game projects. The design is based on a car racing game to support the construction of IP-cores considered essential for game project, such as control, handling and visualization.

1. INTRODUCTION

Technological advances in the microprocessors development has grown exponentially in the last decades, making them smaller and shipping a larger number of transistors. This growth rate was referred from Moore's Law [1] and verified by periodic insertion of new integrated circuits (ICs) on the market with large processing capacity. This law is due to Gordon E. Moore, who in 1965 observed that the density of components on integrated circuits was doubling in regular time periods, implying that this behavior would persist for a long time [2].

This elevated growth rate results in a short space of time for a new design to reach the market. Allied to this, there have been the increasing complexity of projects due to the high density of components in the ICs, where a complete system can be embedded - System-on-Chip (SoC). An alternative way to the development of these ICs is the creation of Intellectual Property Cores (IP-cores) based on the RUP (Rational Unified Process) [3] with prototyping in FPGA devices (Field Programmable Gate Array).

To suit the demands of new products with increasingly varied requirements, have been developed methodologies for IC design using IP-cores. Among them stand out the Virtual Socket Interface Alliance (VSIA), Reuse Methodology Manual (RMM) and recently ipPROCESS [4]. This process model aims to decompose the design into well defined stages, meeting the requirements established in the initial stage. Its main purpose is to reduce design time, by dividing the efforts required to design an IP-core.

This paper introduces an ipPROCESS implementation to development of IP-cores for games in FPGA. The design is based on a car racing game, similar to the old 8-bit consoles that were popular in the 80s. Documentation and RTL modules were developed with in order to its reuse. The RTL model was programmed in Verilog and embedded in an FPGA Development Kit.

The approach employed in this study surrounds all development stages, from initial design up to the current stage of the project. The sections 2 and 3 presents a brief description of the concepts necessary for developing the project. Sections 4, 5, 6 and 7 show the ipPROCESS stages of the project cycle, to one of the IP-cores developed. Finally the results will be reported until the present stage, as well as future prospects.

2. IP-CORE

In last decade there has been an increasing demand in the market for equipment that add a growing set of features (such as communication, entertainment and information access). In this context, the IP-cores is introduced as a complementary element to the development paradigm called System-on-Chip (SoC). From the use of cores, entire the embedded system can now be integrated and implemented on a single chip, by reusing pre-built and pre-verified components, aiming to reduce the time and effort to project [5].

The IP-cores of this work were classified as soft cores, since the circuits are open to new implementations, and its synthesis can be repeated for different programmable logic technologies without changing its operation. In addition, all documentation will be available the extent to which all requirements are met. The framework described is able to receive changes. Once known for its architecture, the modules can be reused in applications which demands for its resources.

3. IPPROCESS

The need of process models for the SoC development resulted in the advent of sophisticated new methodologies for create IP-cores. In most cases, these methods uses paradigms frequently used in software engineering, such as modeling by diagram, extensive documentation, and functionality testing.

The ipPROCESS is a process model for development of IP-cores with FPGA prototyping, based on RUP [3]. This process is intended, through a set of activities assigned to specialized roles in the organization, to transform the requirements into an IP-core, fulfilling time constraints, cost and quality.

The life cycle proposed by the ipPROCESS is decomposed into four sequential phases (Figure 1). At the end of each one, a verification of the criteria

requirements is performed. Only after this validation it's possible to proceed to the next phase [4][6].

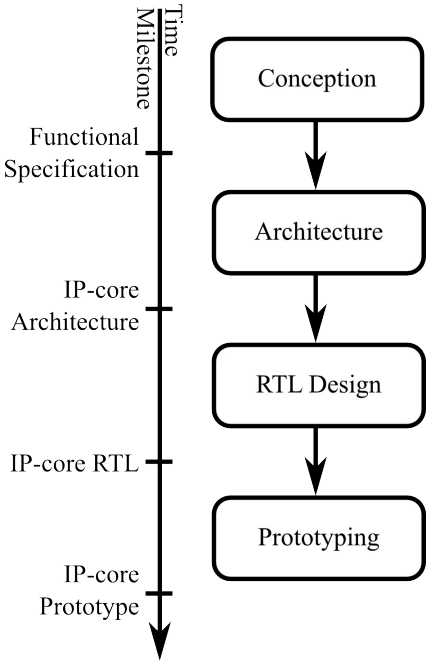

Figure 1 – Life cycle of ipPROCESS.

The process steps are described below [6]:

1. **Conception:** planning stage, where it is defined the scope and project requirements. The expected result is a Functional Specification document containing the project requirements.
2. **Architecture:** stage where a stable architecture is designed for the project.
3. **RTL Design:** The goal of this phase is to build a prototype and verify the IP-core based on the architecture defined during the previous stage. This step concentrates the highest level of effort within the process.
4. **Prototyping:** The focus of this phase is to create a prototype of the design and finalization of the user documentation.

4. METHODOLOGY

The ipPROCESS was used to develop a set of soft IP-cores that will be used for the creation of simple games. The development stages were established in order to attend the full process cycle for each IP-core. The project has been developed over six months (including learning period). The development team consists of two members (graduate students with no design experience).

The design of the (soft) IP-cores consists, at the present time, in five modules: video controller, multiplexer 8-bit color, positioning controller of objects on the screen, analyzer of crashes and the main controller. Were also implemented the following graphic elements: car model, rectangular objects and characters.

The user documents, also called artifacts, have been designed as the project life cycle was restarted, initiating the development of a new IP-core. Throughout this process were carried out reviews/audits. Revisions are performed during the verification process, when design flaws are identified [6]. A change of requirement in an IP-core, which has not yet completed its project cycle, can also imply in considerable changes in stable modules . In this case it is necessary to identify the affected IP-core and restart the process to meet new requirements or restrictions.

Due to the large amount of content for the developed modules in this work, the methodology adopted by this paper presents the results (diagrams and implementations) to only one module (position control).

5. CONCEPTION

The project proposal was drawn from the work started in [7], in order to expand its scope. The goal was to develop a set of IP-cores to support the design of FPGA games.

This is a car racing game that operates in graphic mode. The player can control the car by using buttons or an analog stick to change the speed and position. The game output is displayed on a VGA monitor. The scenario consists of a road with obstacles, as well as other cars, simulating a racing environment. The screen also shows information about the distance and the “lives” left to the player. Figure 3 illustrates a graphical view designed along this step.

Figure 2 – Illustrated perspective of the project.

The estimated effort for this stage of the process life cycle was the lowest since it's the initial phase of the project and the designer has no control over the final architecture model [6]. The following sections provide an overview of the artifacts presented as a result for this step.

5.1. Vision Document

This is the project business card, since it presents an IP-core overview for the client. This document clarifies possible questions with a more detailed project proposal, and possible solutions. This document presents details from the initial project design, including:

1. **Product description:** it's a car racing game that operates in graphic mode. The player can

control the car through buttons or an analog stick (joystick);

2. **Stakeholders:** the team is formed by two members, which the first is responsible for preparing the documentation and describe the RTL modules, and the second to plan and perform the verification;
3. **Product Perspective:** the game should allow the use of two control interfaces, and operate based on the used patterns, providing a good gameplay experience;
4. **Product Features:** the final product is a prototype of the input/output circuit (control and video interface) embedded in an FPGA that contain the game programming;
5. **Standards:** the project is based on the standard VGA video with a resolution of 640x480 pixels;

5.2. Requirement Specification

The purpose of this document is to specify all the project requirements, functional and nonfunctional. Functional requirements describe the actions that should be able to run. This information is taken from the development of use cases, responsible for documenting the inputs, processes and generated outputs. On the other hand, non-functional requirements represent the characteristics that the IP-core should have, or restrictions under which it will operate. These characteristics are related to techniques, algorithms, technologies, standards and system features.

In order to improve the project management scope and facilitate the priorities establishment throughout the project, the requirements presented in this document are classified according to their level of importance:

1. **Important:** without this requirement the system works, but not as well.
2. **Essential:** this requirement must be attended for the system to work.
3. **Desirable:** requirement that does not compromise the system operation.

The requirements are identified and classified for each IP-core. The position controller requirements, according to the above classification, are shown in Table 1.

Table 1 – Position controller requirements.

Important	Set initial value of offset for the object;
Essential	Setup the register of speed of the objects on the screen;
Desirable	Not apply;

6. ARCHITECTURE

This step is the conversion of ideas into an architectural model [6]. During this phase are elaborated documents that support the IP-core design.

The major milestone of this stage is to identify the architecture components, once considered the requirements of IP-core. The elements are modeled in class diagrams and use cases. In some cases it is also possible to describe the processes through finite state machines (state diagrams) [8].

Eight elements were identified and make up the architecture of the racing cars game, which are presented as a class diagram in Figure 4. The architecture model uses the Model View Controller (MVC) nomenclature to sort the Soft IPs.

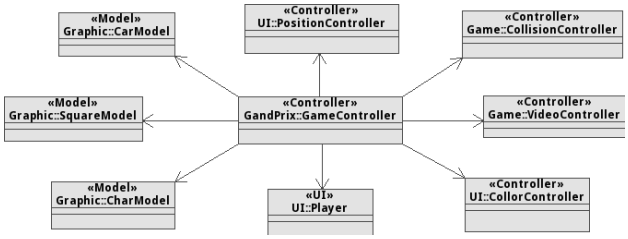

Figure 4 – Project design representation.

The *GameController* is responsible for performing the mapping of all rules and requirements. of graphics and operating the game To it were connected the *CollisionController*, which is dedicated to determine the collision of objects. The *PositionController* is assigned to control the movement of objects (in both axes) on the screen. The *VideoController* and *CollorController* module are part of the IP designed to displaying images on the screen, according to default VGA resolution. The structures named with Model suffix represents the graphics. These models contains information on size and position of each graphic. The classes in this scope also requires the a exclusively designed controller to manipulate the information of its respective model.

6.1 User Cases

The aim of this document is the specification of project use cases. That's includes the following information: actors, flow of primary and secondary events, special requirements, preconditions and post-conditions, non-functional requirements and extension points.

In [8] is presented a study case of ipPROCESS application using Real Time UML (UML-RT). However, the modeling can be described using UML 2.0 without any lost in the architecture interpretation. The following sections present the artifacts of this stage.

The diagram shown in Figure 5 presents an example of use case used in the position controller documentation. This IP-core is responsible for moving an object on the screen, according to the controls flags from the game controller (*Game Ctrl*).

In this structure, each module/object is represented by a actor, and actions are labeled in the circles. This diagram indicates the flow of actions over a control operation. After the player presses a button, the *Game Ctrl* object identify which one was pressed, as it enables the *Position Ctrl* module. The new position is validated and transmitted to the moved object from its own controller by *Game Ctrl*.

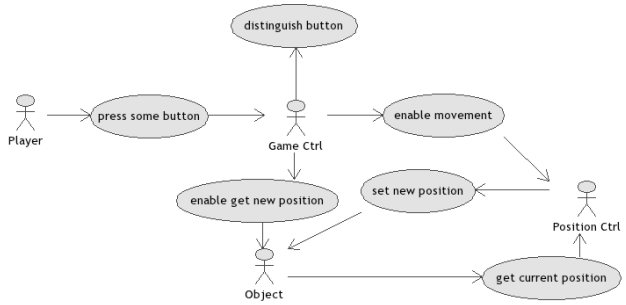

Figure 5 – Use case diagram of position controller.

6.2. Class Diagram

This document presents the module analysis, based on their use case. Each class diagram describes the modules operations, inputs and outputs, as well as its relationship with other IP-cores.

The descriptions are provided by a sequence diagram. The relationships shown in this diagram and the actions taken are transcribed as one or more classes. The Figure 6 depicts the controller position class diagram.

<<Control>> **PositionController**
-_start_boud int = 0 -_end_bout int = 479 + enable : bool + direction: bit + speed: int - move_counter_reg int = 10 * speed + current_position: int + offset: int + new_position: int + ready: bool
+ set_move_register(): int + set_new_position(): int + new_position_validate (new_position: int): bool

Figure 6 – Example of a class representation.

The parameters are identified with the "_" character before its name, representing the internal scope. The input and output ports are highlighted as public attributes (preceded by the symbol "+"). The internal use registers and operations are defined as private attributes (preceded by the symbol "-").

7. RTL DESIGN

This stage consists in the IP-core development and verification, according to the specifications and architecture [4]. The projects were described in Verilog and synthesized using a design, synthesis and simulation tool from Altera Quartus II v9.0 [9]. The RTLs have been developed for the following modules: position controller, video controller, color multiplexer, and graphical models of rectangular elements and the car representation.

In parallel programming the module, are checked for inconsistencies in the documentation and implementation failures. Generally these problems are identified during the functional verification. For example, the positions controller verification plan is responsible for: ensure of new position validation, check the system response to input signals and the behavior of output signals.

8. CONCLUSION AND FUTURE WORK

Before the proposal designed in [7], and by the adoption of ipPROCESS, it was possible to provide elements that allow the continuity to the project. This characteristic is due to the extensive documentation designed during the development of the first stage.

The major difficulty throughout the project was due the change of paradigm. Because of experience lack, the work to execute the life cycle of ipPROCESS demanded greater effort during the first stages of documentation. Currently the development cycle has been following the effort scales established by Lima [8]. Using this process model allowed the division of game design in well defined stages, with specific objectives and predetermined standards.

The project partial result is the documentation of design and planning architecture and the RTL graphic's core (graphic objects and video controller), and game control elements (position controller and collision).

The next step is to review the artifacts, improving the core game control and design the complementary elements to the game: obstacles generator; interface for monitoring the game progress. Finally, the design should be prototyped and made available.

REFERENCES

[1] R. R. Schaller, Moore´s law: Past, present and future, IEEE Spectrum 34, 6 (June), 52–59, 1997.

[2] D. C. Brock, *Understanding Moore's law: four decades of innovation*, Chemical Heritage Foundation, 2006.

[3] P. Kruchten, *The Rational Unified Process: An Introduction*, 3 ed, Boston: Pearson Education, 2003.

[4] M. S. M. De Lima, "ipPROCESS: Um processo para desenvolviemnto de IP-cores com Prototipação em FPGA". Recife : *Master's Thesis*/UFPE, 2005.

[5] J. C. Palma, F. Moraes and N. L. V. Calazans, "Métodos para Desenvolvimento e Distribuição de IP Cores", *Master's thesis*, Faculdade de Informática – PUCRS, 2001.

[6] F. S. dos Santos, "Reestruturação do ipPROCESS e Inclusão dos Processos Fundamentais do Ciclo de Vida", Recife: *Master's Thesis*/UFPE, 2009.

[7] J. C. N. Bittencourt, I. S. Santos, A. M. Dias, "Desenvolvimento de um Jogo de Corrida em FPGA", *Proceedings* VIII Brazilian Symposium on Games and Digital Entertainment, Rio de Janeiro, 2009.

[8] M. Lima, F. Santos, J. Bione, T. Lins, E. Barros, "ipPROCESS: A Development Process for Soft IP-core with Prototyping in FPGA", MSE, 2005.

[9] ALTERA, "Quartus II Web Edition Software", *online* [http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html]