Implementation of Partial Reconfiguration Projects using OpenPR on the Xilinx development board ML507

Implementación de Proyectos de Reconfiguración Parcial usando OpenPR en el sistema de desarrollo ML507 de Xilinx

DORFELL LEONARDO PARRA-PRADA
Electronic Engineer
Connectivity and Signal Processing Research Group
Universidad Industrial de Santander
dorfell.parra@correo.uis.edu.co
Bucaramanga, Colombia

WILLIAM ALEXANDER SALAMANCA-BECERRA
Master in Engineering
Connectivity and Signal Processing Research Group
Universidad Industrial de Santander
williamsalamanca@gmail.com
Bucaramanga, Colombia

Fecha de recibido: 08/12/2013
Fecha de aceptado: 23/10/2014

Forma de citar: PARRA, Dorfell, SALAMANCA, William. Implementacón de Proyectos de Reconfiguración Parcial usando OpenPR en el sistema de desarrollo ML507 de Xilinx.Rev.UIS Ingenierías,2015,vol.14,n1,p.p 33 - 43.

RESUMEN

Este documento presenta la implementación de proyectos de reconfiguración parcial usando OpenPR, un *toolkit* alternativo de reconfiguración parcial de código abierto, en el sistema embebido de desarrollo ML507 de Xilinx. Se reproducen los resultados del proyecto Counter Project del repositorio de OpenPR y se crea un nuevo proyecto para una FPGA v5fx70t. Los archivos de configuración para los diseños estático y dinámico son generados e implementados en el sistema ML507 obteniendo un funcionamiento no esperado. Este funcionamiento conlleva a la revisión de los archivos de descripción nativa del circuito en FPGA EDITOR, herramienta de enrutamiento de Xilinx, para verificar la correcta ubicación de los busmacros en el diseño.

Luego se desarrolla la etapa de depuración en el chip usando Chipscope para capturar señales dentro de la FPGA en cada componente del diseño (e.g. registros, busmacros). Finalmente la evaluación integra las conclusiones y observaciones de las implementaciones y las depuraciones hechas en el desarrollo de los proyectos de reconfiguración parcial.

PALABRAS CLAVES: OpenPR, Open-source, Reconfiguración Parcial, FPGA.

ABSTRACT

This paper demonstrates the evaluation of partial reconfiguration projects implementation using OpenPR -an Open-source partial reconfiguration toolkit alternative-, on the Xilinx embedded system development board ML507. Results of the OpenPR repository counter example project are reproduced, and a new project targeting a different FPGA device, the v5fx70t, is created. Configuration files for static and partial design are generated and implemented in the ML507, getting a non-proper working in the board. This led to the native circuit description files check in FPGA EDITOR -a Xilinx FPGA routing tool- to verify the correct placement of the busmacros into the design-.

Then debugging on-chip stage is done using Chipscope to test the signals state inside the FPGA in each design component as registers and busmacros. Observations are made based on these tests. Finally, conclusions of all the observation process are integrated into this evaluation.

KEYWORDS: OpenPR, Open-Source, Partial Reconfiguration, FPGA.

1. INTRODUCTION

FPGAs are integrated circuits used to design and implement logical functions through a Hardware Description Language (HDL)(FPGA, 2012). FPGA applications include: digital signal processing, ASIC prototyping, medical imaging and almost every area that use a digital circuit. Moreover new techniques (as pipeline, partial reconfiguration) that improve the performance of the FPGA have been proposed.

One of these techniques is the Partial Reconfiguration. Partial Reconfiguration is the process of configuring a part of a programmable chip while the other is still operating (PR, 2012).

In figure 1 partial reconfiguration implementation in an FPGA is shown. The green partitions of the FPGA are the fixed portion of the hardware circuit that remains without changes and it is called static region. The black box is the dynamic region (Reconfigurable Partition). The yellow, blue and red boxes implemented here, are the hardware modules, (Reconfigurable Modules); one of them each time. The static region of the FPGA is configured first. Then, a partial bitstream (configuration file) with the information of the reconfigurable modules configures only the reconfigurable partition without erasing the configuration of the static region, neither stopping the execution of any process in that region. Constraints definitions are required by partial reconfiguration implementations, since according to the region chosen the reconfigurable module will have access to specific resources in the FPGA.

In the last few years several software tools which allow the management of partial reconfiguration projects have been developed.

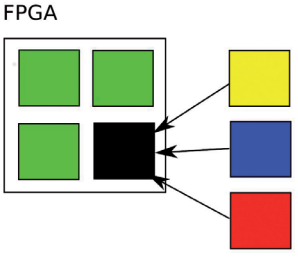


Figure 1. *Partial Reconfiguration Process. Adapted from (PR PlanAhead, 2010)*

Some of these tools are: JBits; it creates Xilinx Virtex bitstreams form Java code, Torc; an open-source infrastructure and tool set for reconfigurable computing, Xilinx Partial Reconfiguration Toolkit; which proposed a partial reconfiguration flow using PlanAhead (PR PlanAhead, 2010) and OpenPR; an open-source Partial Reconfiguration Toolkit for Xilinx FPGAs.

Private software tools such as the Xilinx Partial Reconfiguration Toolkit offers a friendly interface for the user, but its cost could be up to two thousand dollars and its source code is private. Open-source tools are in a continuing process of development and nowadays more users are working to expand their capabilities and improve their features.

Reviews of these tools, as well as performance benchmarks are needed by users who are interested in partial reconfiguration and don't want to waste time nor money.

This paper presents an evaluation of the implementation of partial reconfiguration projects using OpenPR in an Embedded System Development Board ML507 (in figure 2) starting with a brief explanation of the OpenPR partial reconfiguration flow in section 2. Creation of a partial reconfiguration project is made in section 3. Debugging results for the implementation of OpenPR partial reconfiguration project using Chipscope are exposed in section 4.

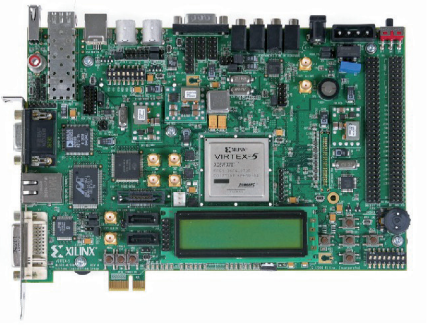


Figure 2. *Virtex-5 FX70T FPGA ML507 Evaluation Platform. Taken from (Virtex 5, 2012)*

Finally conclusions and observations of this work are in section 5.

2. THE OPENPR PROCESS (OPENPR, 2012)

OpenPR is an Open-Source Partial-Reconfiguration Toolkit for Xilinx FPGAs, presented by its designers in 2011.Although

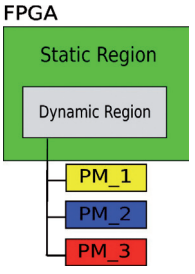


Figure 3.*OpenPR Partial Reconfiguration Design*

OpenPR only works for Xilinx FPGAs, its functionality can be expanded to Altera FPGAs with the proper documentation. This section briefly describes the software partial reconfiguration flow demonstrated by the authors in (OpenPR, 2012). Basic partial reconfiguration concepts are kept in OpenPR. Its flow for creation of partial reconfiguration projects is shown in figure 4.

The dynamic region in figure 3, represents the reconfigurable partition where the reconfigurable modules, known as the partial modules (PM) will be implemented. OpenPR aims to be an easy and almost free-manual intervention required tool. In figure 4 the partial reconfiguration flow is shown. The process is divided into the four main stages described below.

A. OpenPR XML Project File

The first step is to create the OpenPR XML project file which has the project information needed by OpenPR. Some important parameters contained in this file, such as the *busMacroPrefix* are listed in table 1. Bus macros are templates that allow OpenPR to connect partial and static design as shown in figure 5.

An example of the OpenPR XML project file can be found in the OpenPR repositories (OpenPR, 2012).

B. Floorplanning

In this stage, the designer uses PlanAhead or another tool to draw the *dynamic* region, create an area group and export the constraints to the OpenPR user constraints file (ucf), which contains all the design restrictions. This process is called floorplanning.

The area group must have the same name as the *dynamicAGName* parameter in the XML project file.

C. Static Design Flow

Every project in OpenPR has a Makefile for the static directory and a Makefile for the partial directory. These makefiles run scripts with the routines needed to execute the partial reconfiguration. Some of these routines are to synthesize, translate, map, place & route and bitstream generations. The map and place & route processes have a Native Circuit Description (NCD) file output, where the physical description of the design is made (NCD file, 2012). The bitstream generation process has two outputs, one file to configure the static design in the FPGA, and the second one for the partial design. Once the constraints defined in the floorplanning stage have been added to the ucf file, it's time to use the Makefile. The designer can create the bit file for the static design with the command:*make static*.

Table 1. *OpenPR Project File Parameters. Adapted from (OpenPR, 2012)*

Parameter	Description
designName	Top-level design entity.
staticPath	Full path static design directory.
partialPath	Full path partial design directory.
isPartial	Boolean value indicating whether project is a sandbox or a partial module.
ucfPath	Full path to UCF file with pin/timing constraints for static design.
dynamicAGName	Name of dynamic region AREA_GROUP as defined in UCF file.
busMacroPrefix	Naming prefix for bus macro defining target architecture and bus macro type.
passThroughNet Name	For sandbox, identify nets passing through dynamic region; for partial design, identify nets connecting virtual IO pins to dynamic region inputs/outputs.
clkNetNames	Collection of clock nets in the design.
busWidth	Width of dynamic region datapath.
deviceName	Full device name.
regionDefined	Boolean value indicating whether OpenPR region is defined yet.
x/y Min/Max	Vertices of dynamic region.

D. Partial Module Design Flow

The partial directory has a similar static design Makefile, but in the XML project file the *isPartial* parameter is set to 1, to indicate that the directory contains a partial design. This makefile must be used after successfully running the static design makefile to compile the files for the partial design.

Partial bit file is created with the command: *make partial*.

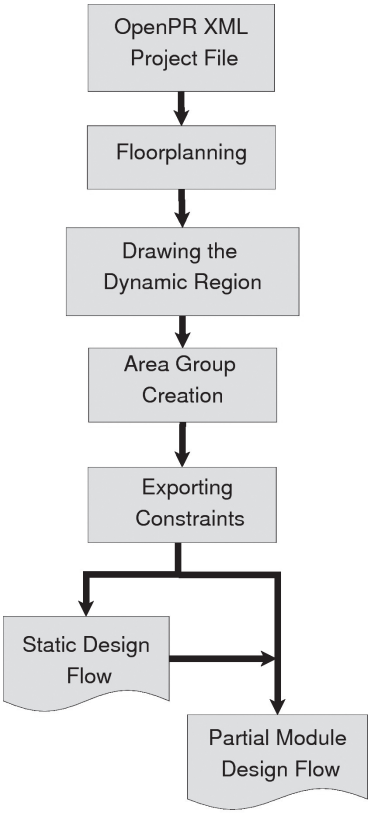


Figure 4. *Overview of the OpenPR design process*

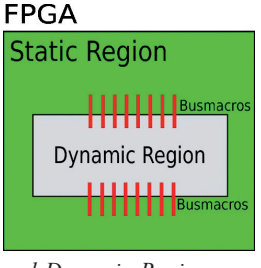


Figure 5. *Static and Dynamic Region connection using bus macros. Adapted from (Sohanghpurwala et al., 2011)*

3. CREATING RECONFIGURATION PROJECTS WITH OPENPR

This section demonstrates the tracking of the OpenPR Partial Reconfiguration Project flow using the Counter demo project, which is an OpenPR project example for Virtex 5 LX FPGAs. Then, a new project is created to evaluate the OpenPR capabilities.

A. Counter project

The Counter demo is available from OpenPR repository, which has Documentation, source-code and OpenPR example projects. OpenPR projects have a directory structure shown in figure 6.

There are OpenPR Project files in both the Sandbox and Partial module (PM) folders with the parameter *isPartial* defined to indicate the software whether the folder has a Static or a partial design. Static design files are in a Sandbox folder, while the partial designs are in PM folders. Routines needed to create bit files for the static and partial design are in the Makefile.

Hardware description language files (*.vhd) have design descriptions and are used to synthesize the digital circuits; Physical constraints are in an ucf file and a precompiled folder has bus macros templates.

- **Sandbox**
  - ↦ **Makefile.....Static Design Make-file.**
  - ↦ **top.proj.....OpenPR Sandbox Project File.**
  - ↦ **src**
    - → **sandbox.vhd.....Static Design Top Level.**
    - → **system.ucf**
    - → **precompiled**
      - ↪ **busmacro_*.....Pre-built Bus Macros.**
- **PM_0**
  - ↦ **Makefile.....Partial Module Make-file.**
  - ↦ **top.proj.....OpenPR PM Project File.**
  - ↦ **src**
    - → **sandbox.vhd.....PM Top Level with Bus Macros.**
    - → **prm.vhd.....PM Implementation.**
    - → **system.ucf**
    - → **precompiled**
      - ↪ **busmacro_*.....Pre-built Bus Macros.**
- **PM_1**
- **PM_2**

Figure 6. *OpenPR Partial Reconfiguration Projects Directory Structure. Taken from(Sohanghpurwala et al., 2011)*

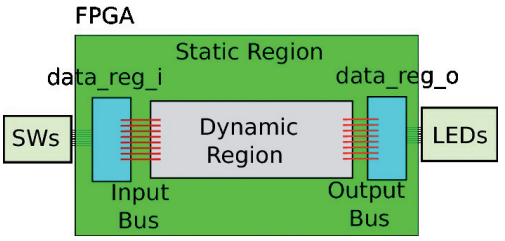


Figure 7. *Counter Example Circuit*

Table 2. *Partial Modules Description*

Partial Module	Description	Implementation
Passthrough	Input-Output bus macros direct Connection	Leds show SWs state
Increment	Ascending counter	Leds blinking
Decrement	Descending counter	Leds blinking

Data_reg_i and data_reg_o are the registers dedicated to the LEDs and SWs peripherals, as it is shown in figure 7. Bus macros, in red, allow the connection of the dynamic region with the static design.

The partial modules names in the Counter example are: "passthrough"', "increment"' and "decrement". In the passthrough module the SWs state is shown in the LEDs. Increment and decrement modules are counters with their first value given by the SWs state. The dynamic region definition for any of the three modules can be made using PlanAhead. Table 2 summarizes their behavior.

In figure 8 a) the architecture of a Virtex 5 LX series FPGA is shown.The rows in the FPGA are known as clock regions. There are 6 of them.

Figure 8 b) is a detailed view of how FPGA logic resources like: Slices -in blue-, Ram block (BRAM) -in Red-, and Digital Signal Processing (DSP) -in green-, are organized in columns of the FPGA. A slice is a set of 4 CLB.

Indications of specific places in the FPGA are made using slices coordinates.(i.e. SLICEX0Y0, where X and Y are the positions of the slice) (UG190, 2010).

The area group selected for the counter project is shown in figure 8 c). The constraints for this region were:

```
AREA_GROUP "counter" RANGE=SLICE_X8Y60:SLICE_X23Y79;
AREA_GROUP "counter" RANGE=DSP48\_X0Y24:DSP48_X0Y31;
AREA_GROUP "counter" RANGE=RAMB36_X0Y12:RAMB36_X0Y15;
```

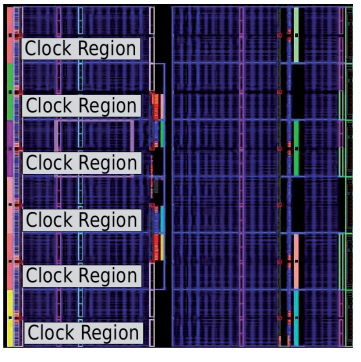
FPGA_editor is a Xilinx Tool that allows the edition of the internal FPGA circuit routing (FPGA Editor, 2008). Figure 9 shows the *.ncd file generated with the bus macros correctly placed. The blue rectangle is the FPGA and the red point is the Busmacro position.

Implementation of the project was made executing the Makefiles for the static and partial design. That process resulted in the successful generation of the static and partial bit file.

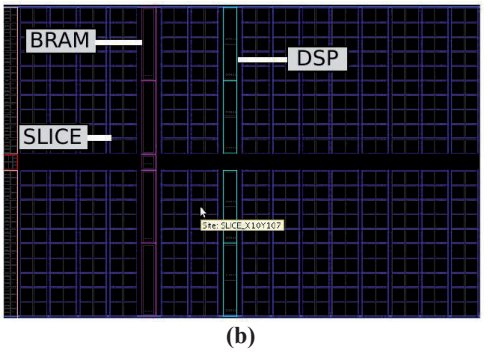
B. Creating a New Project

A new Counter project based on the demo was created to evaluate the proper working of OpenPR in the ML507 development board.

Makefiles were changed to use the ML507 FPGA as well as the place for the area_group. This new position of the area group is shown in figure 10 where the black box represents the PowerPC hardcore processor for this Virtex 5 FX FPGA.

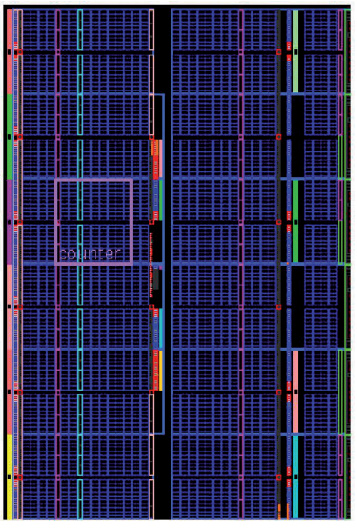


(a)



(b)

UIS Ingenierías
REVISTA DE LA FACULTAD DE INGENIERÍAS FISICOMECÁNICAS



(c)

Figure 8. *FPGA Virtex 5 LX. a) Virtex 5 LX FPGA architecture. b) FPGA resources. c) Area Group Counter selected*

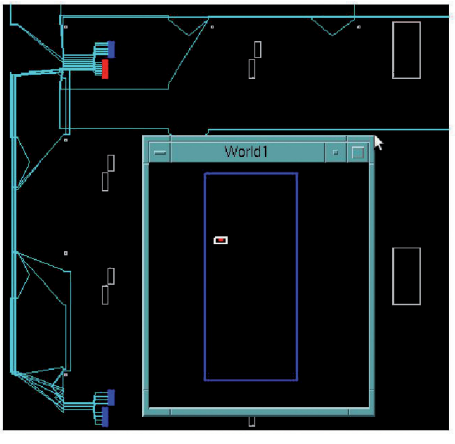


Figure 9. *Counter example bus macro position in the LX FPGA*

Constraints for this new position were:

AREA_GROUP "counter" RANGE=SLICE_X4Y20:SLICE_X29Y39;
AREA_GROUP "counter" RANGE=RAMB36_X0Y4:RAMB36_X1Y7;

After floorplanning stage implementation process produced some errors. In the next section these errors are shown as well as the explanation of the tests done for the project.

4. DEBUGGING IMPLEMENTATION OF PR PROJECT

The counter project for the ML507 development board created in the previous section was used to evaluate the proper operation of the OpenPR. Implementation process and an analysis of the results are presented in this section.

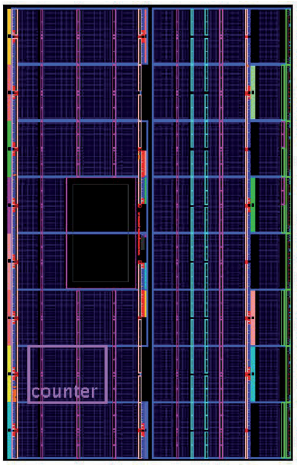


Figure 10. *Area Group Counter Selected for the New Project*

All the tests made can be summarized in two cases with different dynamic regions selected.

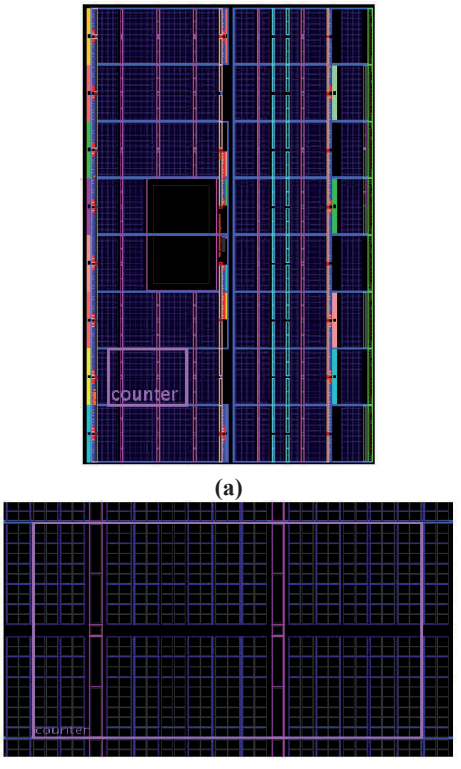
A. Region 1

The Area group counter in region 1 for the reconfigurable modules was placed in:

AREA_GROUP "counter" RANGE=SLICE_X4Y20:SLICE_X29Y39;
AREA_GROUP "counter" RANGE=RAMB36_X0Y4:RAMB36_X1Y7;

as is shown in figure 11 a). The resources in this area group are magnified in figure 11 b). Once the Makefile was executed the mapping process produced one error shown in figure 12.

In the mapping report this error was repeated by each bus macro component of the Counter project. There are two possible causes:



(a)

(b)

Figure 11. *Area group counter location. a) Area group counter place in region 1. b) Area group counter resources in region 1*

- The first is that OpenPR assumes that the design requires DSP resources, but the region doesn't have these resources, so it produces an error message saying that the other resources needed by the components are blocked; *"Some of the logic associated with this structure is locked"* in figure 12.

- The second possible reason is that OpenPR assumes the whole left side of the FPGA as a region, and it has a PowerPC hardcore processor, so the PowerPC resources are considered to be locked and OpenPR cannot place the bus macros nor the other components of the project.

B. Region 2

The area group counter in region 2 for the reconfigurable modules was placed in:
RANGE=SLICE_X40Y120:SLICE_X67Y139; Logic Resources
RANGE=DSP48_X0Y48:DSP48_X1Y55; DSP Resources
RANGE=RAMB36_X3Y24:RAMB36_X4Y27RAM BLOCK

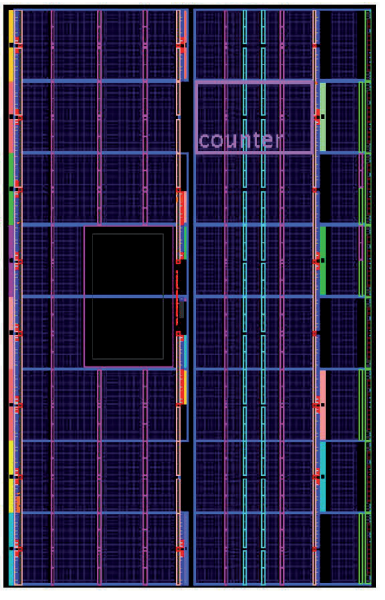
Phase 1.1 Initial Placement Analysis ERROR: Place:346 - The components related to Macro Instance <busmacro_xc5v_async_vert_input_0_0> of the Macro Definition <busmacro_xc5v_async_vert_input_0> cannot be placed in the required relative placement form.

The following components are part of this structure: LUTbusmacro_xc5v_async_vert_input_0_0/$COMP_3, locked to site SLICE_X5Y39 LUTbusmacro_xc5v_async_vert_input_0_0/$COMP_4 LUTbusmacro_xc5v_async_vert_input_0_0/$COMP_5 LUTbusmacro_xc5v_async_vert_input_0_0/$COMP_0

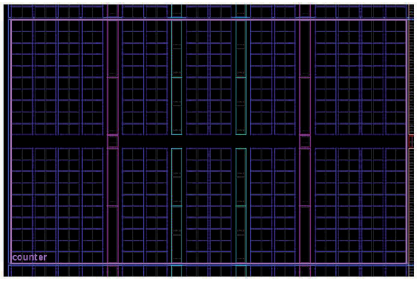
The reason for this issue is the following: Some of the logic associated with this structure is locked. This should cause the rest of the logic to be locked. A problem was found at site SLICE_X4Y39 where we must place LUT busmacro _xc5v_async_vert_input_0_0/$COMP_0 in order to satisfy the relative placement requirements of this logic. This site appears to be prohibited.

Figure 12. *Mapping Process Error Message*

as it is shown in figure 13a). A detailed view of the resources in that region is shown in figure 13b), where it can be seen that the area group now has DSP resources. Placing the area group in region 2 allowed OpenPR to successfully finish the execution of the *Makefile*.



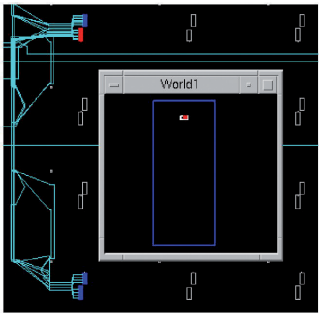
(a)



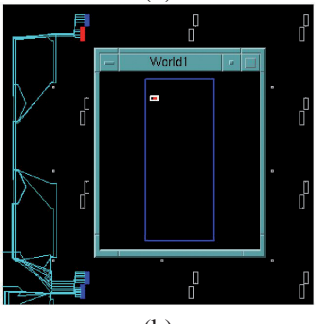
(b)

Figure 13. *Area group counter new locations. A) Area group place in region 2. b) Area group resources in region 2*

The *.ncd files for the sandbox and the passthrough partial modules were checked using FPGA_editor. In figure 14a) the input and output bus macros have been automatically placed by OpenPR in the boundaries of the reconfigurable region for the static design in sandbox, but in figure 14b) the .ncd for the partial module Passthrough, analyzed in FPGA_editor, shows that the bus macros appear in another position different to the position seen before in the static design.



(a)



(b)

Figure 14. *Native Circuit Description for Static and Partial design. a) Bus macros structure and position for Sandbox. b) Bus macro structure and position for the PM Passthrough*

Therefore, there is no match between the static design bus macro location and the dynamic design bus macro location.

After the .ncd files check, testing the generated .bit files is the next step. Configuration of the FPGA was made using the impact tool provided by Xilinx, downloading first the static bit file and then the partial bit file. The bit files generated are listed in table 3. Partial bit file has a smaller size than the static bit file because the partial module circuit is smaller than static circuit.

Module	Bit file	Size
Sandbox	genesys_top.bit	3.2 [MB]
Passthrough	top_full.bit top_partial.bit	3.2 [MB] 86.5[kB]

Table 3. *Bit files generated*

The circuit was tested using the board, but according to the behavior listed in table 2, there wasn't a proper working of the example. This led to the debugging process.

C. Debugging Process using ChipScope

ChipScope was used to get a better image of the behavior of the circuit. ChipScope offers the ChipScope Pro Core Inserter to create IPcores as ILA, ICON, IBERT, etc, which capture signals inside the chip, and the ChipScope Analyzer to show the signals that are being checked.

The generation of the IPcores ICON and ILA involved the parameters in table 4 which are related to the names in figure 7.

Data_reg_i and data_reg_o were associated to data channels [0:15], and the module_in and module_out, which are the signals that connect the input/output registers to the busmacros, were associated with data channels [16:31].

Once the ILA and ICON core were inserted into the design, ChipScope Analyzer was used to inspect the signals.

Table 4. *ILA and ICON parameters*

Parameters	Value
Input Netlist	top.ngc
Device Family	Virtex 5
Trigger Width	1
Match Units	1
Data Same As Trigger	Unchecked
Data Width	[0:31]
Data Depth	1024 samples
Clock Signals	Clk_BUFGP
Trigger Signals	Clk_BUFGP
Data CH [0:7]	data_reg_i[0:7]
Data CH [8:15]	data_reg_o[0:7]
Data CH [16:23]	module_in[0:7]
Data CH [24:31]	module_out[0:7]

Table 5 shows the leds and switch status for the sandbox implementation. The connections between SWs and the data_reg_i are working as seen in table 6. It is important to note that there is no connection between the data_reg_i and data_reg_o because any partial bit file has been downloaded to the FPGA.

Table 5. *SW and LEDs state for Sandbox implementation*

SWs	LEDs
0x00	0xff

Table 6. *Data Channel Outputs for Sandbox*

Data Channel	Output
DataPort [0:7]	0x00
DataPort [8:15]	0xFF
DataPort [16:23]	0x00
DataPort [24:31]	0xFF

Table 7 shows the behavior of the implementation of the top_full bit file. It seems similar to the sandbox bit file described in table 5, but as it is shown intable 8, all the signals have the boolean value 1 as output. Therefore, there are no connections between the data_reg_i and the dip switches.

Table 7. *SWs and LEDs state for top_full.bit implementation*

SWs	LEDs
0x00	0xff

Table 8. *Data Channel Outputs for Partial Module Passthrough*

Data Channel	Output
DataPort [0:7]	0xFF
DataPort [8:15]	0xFF
DataPort [16:23]	0xFF
DataPort [24:31]	0xFF

In order to continue with the process the partial bitstream was downloaded to the FPGA, but instead of configuring just the area group counter, it changed the whole configuration of the FPGA, erasing the chipscope IPcores. In table 9 the SWs and LEDs status are shown.

Table 9. *SWs and LEDs state for top_partial.bit implementation*

SWs	LEDs
0x00	0xC4

5. CONCLUSIONS

This paper has demonstrated an evaluation of OpenPR working on the development board ML507. The evaluation process follows the design flow proposed by the authors (Sohanghpurwala et al., 2011), including a debugging on-chip stage using Chipscope. The main conclusions of this work will be shown below.

The OpenPR partial reconfiguration design flow has been documented giving a brief description of each stage of the process, furthermore, the Partial Reconfiguration OpenPR project creation using the OpenPR examples and a review of the walk-through is demonstrated.

Although the implementation on the ML507 board was not successful, the information obtained in the debug stage led to the following observations.

The automatic location of the bus macros depends on the resources outside the partial region that are available. OpenPR does not take into account the resources when it places the bus macros for the partial region, because as seen before, bus macros encountered logic resources blocked for the XC5V70T-ff1136-1 where the left chip architecture is different than the right, due to the PowerPC hardcore processor. The causes of this leak of information might be in the structure definition made in the Virtex5DeviceInfo file.

Once that OpenPR has located the bus macros for the static region in sandbox, the location of the bus macros in the partial modules differs from the previous place chosen in the static design, leading to mismatches in the implementation of the FPGA partial configuration. A consequence of these mismatches is the loss of design parts.

Partial bitstream generation is not being correctly made. It is shown by tests, that when a user downloads the partial bitstream, part of the static design is changed. This issue is expected to be corrected in future OpenPR updates.

In spite of these bugs, OpenPR is free, allowing more users to get involved in the process without paying for expensive tools. Its Open Source code admits to include others FPGAs that are not supported by private tools these days. OpenPR sets a strong alternative in FPGA reconfiguration tools, and its development process will be the key for the next improvements, like user graphics interfaces, and support for the Virtex 5 FX series.

6. REFERENCES

FPGA. *Field-programmable gate array*, 2012 [Online] Available:*h*ttp:/www.altera.com/products/fpga.html?GSA_pos=3&WT.oss_r=1&WT.oss=FPGA

STARBOARD. *StarBoardDesign: Specializing in rapid, effective, analog and mixed-signal electronic design*, 2009 [online] Available: http://www.starboarddesign.com/examples.html

KUON, I; ROSE, J. *Measuring the gap between FPGAs and ASICs,* Proceedings of the international symposium on Field programmable gate arrays-FPGA'06, 2006

VIRTEX 5. Virtex-*5 FXT FPGA ML507 Evaluation Platform,* 2012 [Online] Avaliable: http://www.xilinx.com/products/boards-and-kits/HW-V5-ML507-UNI-G-image.htm

MONEY, D; *HARRIS,* S. *Digital Design and Computer Architecture,* Morgan Kaufmann Publications, 2007

PR, Partial re-configuration, 2012 [Online].Avaliable:http://en.wikipedia.org/wiki/Partial_re-configuration

GUCCIONE, S; LEVI, D; J Bits*: A Java-Based Interface to FPGA Hardware,* Xilinx, 1998.

GUCCIONE, S; LEVI*, D; JBits: A Java-Based Interface for Reconfigurable Computing*, Xilinx, 1998.

SOHANGHPURWALA, A; ATHANAS, P;FRANGIEH, T; WOOD, *A.OpenPR:An Open-SourcePartial-Reconfiguration Toolkit for Xilinx FPGAs*, IEEE, 2011.

STEINER, N. et al. *M.Torc: Towards an Open-Source Tool Flow,* Proceedings of the 19th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2011 (Monterey, California).

STEINER, N; A *Standalone Wire Database for Routing and Tracing in Xilinx Virtex, Virtex-E, and Virtex-II FPGAs.* Virginia Polytechnic Institute and State University, 2002, (Blacksburg, Virginia).

PR PlanAhead. *Using PlanAhead to manage the Partial Reconfiguration flow,* 2010 [Online]. Available: http://www.xilinx.com/tools/partial-reconfiguration.htm

JBits SDK, 2012 [Online]. Available: http://www.xilinx.com/products/jbits/index.htm

OpenPR, *OpenPR FPGA ToolKit, 2012.* [Online]. Available: http://openpr-vt.sourceforge.net/OpenPR/OpenPR.html

STEINER, N et al. *French, M.Torc: Tools For Open Reconfigurable Computing,* 2011 [Online]. Available: http://torc-isi.sourceforge.net/documentation.php

NCD file, 2012 [Online].Available: http://www.xilinx.com/itp/xilinx10/help/iseguide/mergedProjects/floorplanner/html/fp_d_ncd_file.html

UG190, Virtex-5 FPGA User Guide UG190, 2010 [online]. Available: http://www.xilinx.com/support/documentation/virtex-5_user_guides.htm

FPGA Editor, 2008 [Online].Available: http://www.xilinx.com/itp/xilinx10/isehelp/ise_n_fed_navpage.htm

GCC, the GNU Compiler Collection, 2012. [Online]. Available: http://gcc.gnu.org/

Boost C++ Libraries, 2012 [Online]. Available: http://www.boost.org/

Bison\-GNU Parser Generator, 2012 [Online]. Available: http://www.gnu.org/software/bison/

Flex*: The Flast Lexical Analyzer*, 2012 [Online]. Available:http://flex.sourceforge.net/

7. ACKNOWLEDGEMENTS

The author would like to thank God, to his mother Elisa,his father Felix, his two little sisters Dori and Nubia and his girlfriend Laura.The author would also like to say thanks to CPS, Sergio, Carlos A, Carlos F, for their support and especially to William for his patience. And last but not least, he thanks his friends for their support, understanding and love.

8. CURRICULUM

Dorfell L. Parra Prada is a Master student graduated from the Universidad Industrial de Santander. He is a hard working reserved man, who is now involved in the digital design, robust control, RTOS, and GPGPU applied to the Seismic research field.

William A. Salamanca B. is a PhD student who has a master degree in electronic engineering from the Universidad Industrial de Santander. He is interested in high performance computing using alternative devices such as GPGPU and FPGA applied to the oil industry.