# Design Guidelines for the High-Speed Dynamic Partial Reconfiguration Based Software Defined Radio Implementations on Xilinx Zynq FPGA

Ahmed Kamaleldin[1], Ahmed Mohamed[1], Ahmed Nagy[1], Youssef Gamal[1], Ahmed Shalash[1], Yehea Ismail[2] and Hassan Mostafa[1,2]

[1]Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt.
[2]Center for Nano-electronics & Devices, American University in Cairo & Zewail City for Science and Technology, Cairo, Egypt

***Abstract*—Reconfigurability of Field Programmable Gate Array (FPGA) makes it one of the most promising approaches in the implementation of the Software Defined Radio (SDR). FPGA Dynamic Partial Reconfiguration (DPR) feature emphasizes that approach by allowing the implemented SDR system to switch between multiple communications standards in runtime reusing the same FPGA hardware resources. Reconfiguration time is a significant parameter in DPR designs especially when a fast switching is required in real time system like SDR. In this paper, different designs of Partial Reconfiguration (PR) controllers are studied and evaluated according to their impact to improve the reconfiguration time of DPR-based SDR implementation. A multi-standard convolutional encoder design is implemented using DPR with different PR controllers as a case study. The design is implemented and tested on Xilinx Zynq evaluation board "ZC702". This comparative study provides important design insights and recommendations to the DPR-based SDR designers to help them select the best PR controller based on their system throughput requirement and power budget.**

***Keywords—Dynamic Partial Reconfiguration, Software Defined Radio, Field Programable Gate Array.***

## I. INTRODUCTION

Modern Wireless communication systems support multiple wireless standards using an adaptive reconfigurable wireless terminal that adopts dynamic communication chains based on the Software Defined Radio (SDR) concept [1]. SDR provides the flexibility to reconfigure the communication chain according to the required wireless standard without the need to change the hardware platform. Microprocessors are often used for SDR implementations to provide the required flexibility by a set of software routines. Obviously, microprocessors are not a suitable hardware platform for the high data rate and low power constraints required by the baseband signal processing [2]. Field Programmable Gate Array (FPGA) enables the implementation of advanced baseband systems within a reasonable area with a reduction in power consumption and a high data rate. Recently FPGA runtime Dynamic Partial Reconfiguration (DPR) has been widely used to provide the hardware flexibility for SDR implementation reusing the same hardware resources in the FPGA [3]. DPR provides the advantage to reconfigure a portion of the FPGA device at runtime while the rest of FPGA remains active. In addition, DPR reduces the static power consumption and the FPGA resource counts by sharing hardware resources during runtime.

Reconfiguration time is a key factor in SDR design which determines the required time to reconfigure the system from an operating mode to another. Minimizing the reconfiguration time is an essential target for fast switching between different wireless standards in runtime. This paper investigates experimentally the implementation of SDR using different DPR techniques taking into account the impact of reconfiguration throughput, and power consumption on the system performance. The implementation is done targeting the Xilinx Zynq FPGA device. This paper is organized as follows: Section II gives a background on DPR and previous works on SDR implementations. Section III presents different designs of self-reconfiguration controllers used in DPR. A multi-standard convolutional encoder design is presented in Section IV as a case study on the SDR implementation. Section V shows the system implementation and the experimental results. Finally, Section VI draws the paper conclusion.

## II. BACKGROUND AND PREVIOUS WORK

DPR offers the advantage of flexibility to reconfigure a wireless communication system with different wireless standards at runtime reusing the same hardware resources. Xilinx DPR design flow requires partitioning the system into a static part and a dynamic part [4]. The dynamic part contains the reconfigurable modules (RM) of the system, while the static part contains the static modules that are not be affected during the reconfiguration. The dynamic part contains multiple Reconfigurable Partitions (RPs), each RP has a set of RMs which can be swapped during runtime without interruption. A partial bitstream is generated for each RM to be mapped into a specific RP during reconfiguration. Partial bitstreams are loaded from a nonvolatile memory to the FPGA configuration memory through dedicated configuration interfaces. DPR are classified according to the configuration modes as internal or external reconfiguration techniques, based on the reconfiguration is handled internally within the FPGA or by an external device like a PC or another FPGA. Xilinx 7-series FPGAs have two internal configuration interfaces to the FPGA configuration memory [4]: (i) The Internal Configuration Access Port (ICAP) that is physically located on the FPGA fabric. (ii) Processor Configuration Access Port (PCAP) only available for the Xilinx

978-1-4673-6853-7/17/$31.00 ©2017 IEEE

7-series Zynq FPGA equipped with a hard macro ARM processor. Also, a three external configuration interfaces are used through the serial configuration ports: JTAG, Serial mode, and Select-Map. This paper focuses only on the internal configuration modes based on the fact of the reconfiguration time overhead of the external configuration interfaces is not suitable for real-time standalone applications like the SDR [5].

Several works have discussed the benefits of using DPR for the implementation of SDR. McDonald [3] provides a full reprogrammed SDR physical layer implementation on Virtex-4 using internal and external configuration modes taking into consideration the reconfiguration time overhead of each configuration mode. Also in [6], a dynamic cognitive radios implementation is presented using the Xilinx Zynq FPGA. The proposed system uses the FPGA programmable logic (PL) to implement the baseband and the ARM processor for the MAC layer. Furthermore, a proposed custom partial reconfiguration controller is used to control the switching between different baseband modules, achieving a high reconfiguration speed [6].

Reconfiguration time (or reconfiguration throughput) is a critical parameter in DPR designs which affects the whole system performance implemented on the FPGA. Reconfiguration time depends on the dimension of RP and the generated partial bitstream data size. Also, the memory configuration setups used for data transfer increase the reconfiguration time overhead. Many contributions are proposed in the literature to improve the reconfiguration time and enhance the feasibility of DPR. In [7] authors minimizing the runtime DPR overheads by using streaming Direct Memory Acess (DMA) engines and a bitstream size compression technique for achieving a high reconfiguration throughput. Different versions of open source high-speed ICAP controllers are presented in [8-10] that significantly improve the reconfiguration throughput. Therefore, these controllers are utilized in this paper with the other conventional controllers to study the implementation of a high-speed DPR-based SDR system.

## III. PARTIAL RECONFIGURATION CONTROLLERS

Partial Reconfiguration (PR) controllers are proposed by researchers to control the DPR and to enhance the reconfiguration speed and maximize the reconfiguration throughput. PR controller provides the interface for transferring partial bitstream data to the FPGA internal configuration port (i.e., ICAP or PCAP) from an external or internal memory with a high data throughput. Moreover, some PR controller architectures have the capabilities of monitoring the system performance by measuring the reconfiguration time and determine the status of RMs. The time of reconfiguration is a key factor in the design of PR controller as it measures how fast the controller can handle the reconfiguration process. A lot of PR controllers are used in DPR whether they are conventional controllers provided by the FPGA vendors or novel controllers developed to carry out the DPR to be more efficient and reducing the reconfiguration time. PR controller is implemented by using a Finite State Machine (FSM) or a dedicated custom processor to control and manage the DPR.

### *A. Xilinx ICAP Controllers*

Xilinx provides several IP cores to interface the Xilinx's ICAP primitive with the user system design. ICAP Controllers enable an embedded microprocessor such as Microblaze or ARM processors to access the configuration memory. ICAP is a Xilinx FPGA hard macro that provides direct access to the configuration memory both in read and write modes [4]. The ratio of the ICAP interface data width to the configuration memory is 8, 16 or 32 bit wide in Xilinx 7-series. The ICAP provides a maximum theoretical reconfiguration throughput equal to 400 MB/S at a data width of 32 bits and a clock frequency of 100MHZ. practically, the reconfiguration throughput that has been measured in real time applications is less than the theoretical ICAP throughput due to the reconfiguration overhead added to the DPR at the system level design. XPS-HWICAP and AXI-HWICAP [11] are two ICAP Controllers designed for Processor Local Bus (PLB) and AXI buses interfaces which are connected to the buses as a slave peripherals. During DPR, the partial bitstream data are buffered from an external memory to a Write/Read FIFOs inside the core. An internal state machine observes the occupancy status of the FIFOs and continuously supplies partial bitstream data to the ICAP and then to the configuration memory. These controllers give a low reconfiguration throughput that's far below the theoretical ICAP throughput. The limitation comes from the low data rate of partial bitstream sent to the ICAP while using the microprocessor for fetching partial bitstreams data from an external memory to the controller FIFOs.

### *B. Xilinx Partial Reconfiguration Controller*

The Xilinx Partial Reconfiguration Controller (Xil-PRC) core provides management functions for self-controlling partially reconfigurable designs, the core is equipped with AXI4-Lite bus interface [12]. Xil-PRC is released for enclosed systems where all the design RMs are known to the controller. Xil-PRC consists of a 32 Virtual Socket Managers (VSM) and each VSM can contain up to 128 RMs. The Virtual Socket is a term that used to refer to a RP that is managed by the Xil-PRC. VSMs are connected to a fetch path that fetches the partial bitstream data from an external memory to the ICAP without passing by the processor. Each VSM operates independently of the others and has its own AXI4-Stream interface for partial bitstream data transfer. An external trigger is sent to a VSM to select the required RM to be loaded into the RP, each RM has its own trigger number. Xil-PRC accesses the external memory using an AXI External Memory Controller (axi_emc) or a Memory Interface Generator (MIG). Xil-PRC does not depend on the processor to fetch partial bitstreams from an external memory, which leads to a high reconfiguration throughput close to the theoretical ICAP throughput.

### *C. High-speed Open Source ICAP Controllers*

Various Open Source ICAP Controllers are proposed by researchers to improve the reconfiguration time while using the ICAP through an embedded microprocessor. Moving partial bitstream data to the ICAP using a microprocessor is inefficient and reduces the feasibility of DPR. In [9], an open source ICAP controller is proposed, reaching a reconfiguration throughput of 399.80MB/S. The controller fetches the partial bitstream data from a DDR using the DMA controller to an asynchronous

FIFO connected to the ICAP and then to the configuration memory. A custom reconfiguration controller for Xilinx Zynq FPGA (ZYCAP) is presented in [10]. ZYCAP achieves a reconfiguration throughput of 382 MB/S. ZYCAP is a DMA based AXI-HWICAP controller equipped with two AXI slave bus interfaces connected to the ARM processor. The AXI4 bus interface is used by the DMA to receive partial bitstream data from DDR memory and the AXI-Lite bus interface for control signals.

### D. Software-controlled Partial Reconfiguration

Xilinx Zynq FPGA provides the potential to implement a software-controlled (S-C) DPR through the processing system (PS) device configuration (DevC)/PCAP interface [13]. This scheme does not require any programmable logic (PL) resources during DPR. The DevC unit is controlled by the ARM processor to select the internal configuration interface to be PCAP or ICAP according to the user system design. PCAP interface contains AXI-PCAP bridge to access the configuration memory in the PL side from the PS side. In a S-C PR, the ARM controls the DPR using an internal DMA engine to transfer partial bitstream data from an external memory to PCAP interface and then to the configuration memory on the PL side. The PCAP has a 32-bit wide data interface that operates at a clock frequency of 100 MHZ, achieving a theoretical reconfiguration throughput of 400 MB/S similar to the theoretical reconfiguration throughput achieved by the ICAP. The drawback of that scheme is that it blocks the ARM during DPR and prevents it from doing other software tasks.

## IV. A MULTI-STANDARD CONVOLUTIONAL ENCODER

In order to analyze the impact of DPR techniques on the implementation of SDR, a multi-standard convolutional encoder is used as a case study. Convolutional encoders are one of the Forward Error Correction (FEC) coding schemes widely used in wireless communication systems to reduce the effect of noisy channels. Three main parameters are used for describing convolutional codes $(n, k, l)$, where $n$ represents the number of output bits, $k$ represents the number of input bits and $l$ represents the constraint length or the number of shift registers. The Convolutional encoder rate is $\frac{k}{n}$. In our experiment, a seven encoder schemes shown in Table I are used as a benchmark. DPR is used to switch between the seven encoders RM at runtime reusing the same RP on the FPGA [14].

TABLE I. CONVOLUTIONAL ENCODERS SCHEMES

| Encoders | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
|---|---|---|---|---|---|---|---|
| System | 2G | 2G | 2G | 3G | 3G | LTE | WIFI |
| Rate $\frac{k}{n}$ | 1/2 | 1/3 | 1/2 | 1/2 | 1/3 | 1/3 | 1/2 |
| Constr.length | 5 | 7 | 5 | 9 | 9 | 9 | 7 |

## V. IMPLEMENTATION & RESULTS

The experiment aims to study the impact of applying DPR to the SDR implementation using different designs of PR controller which are discussed in Section III and compare between them with respect to the reconfiguration time, power consumption, and utilized area. The experimentation is carried out using four different PR controllers for SDR designs : 1) AXI-HWICAP, 2) Xil-PRC, 3) ZYCAP [10] , 4) S-C/PCAP.

### A. System Implementation and Setup

The system has been implemented using Xilinx Zynq XC7Z020LG484-1 FPGA and tested with a ZC702 board [15]. The DPR flow has been carried out using Xilinx Vivado tool. The complete system is developed as shown in Fig. 1. The system design is controlled by the ARM Cortex-A9 microprocessor on the PS side, the ARM processor communicates with the PL using the AXI bus interface. The static part of the system consists of the PR controller, the ICAP for internal configuration, the AXI bus connections, and the static blocks in the wireless communication chain. The dynamic part contains a single RP for the reconfigurable encoder. The encoder has 7 RMs (C1-C7), each module is synthesized in 37 LUTs. A single encoder RM is active at a time. Encoders RMs (C1-C7) partial bitstreams data are stored on SD flash memory to switching between them during DPR by replacing the existing encoder RM with another RM on the RP. The system is connected to an external PC using UART interface to interact with the ARM processor through software codes.

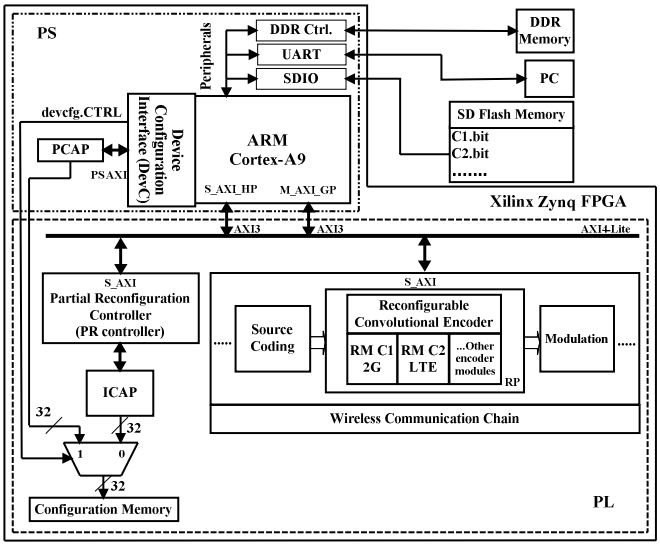


Fig. 1. Complete System Overview

### B. Reconfiguration Time and Throughput

As mentioned in Section II, reconfiguration time is proportional to the size of the RPs located on the FPGA fabric. Hence, the size of convolutional encoder RP is varied along the experiment in order to check the variations in reconfiguration time values as shown in Fig. 2 with different PR controllers.

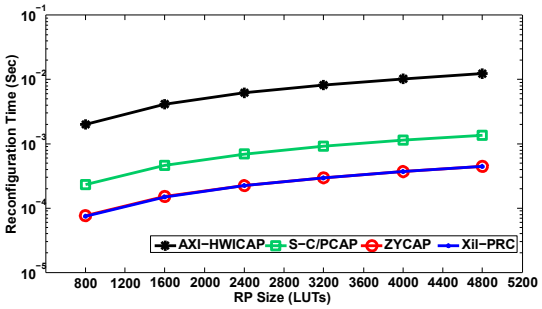


Fig. 2. Reconfiguration Time for various RP Sizes

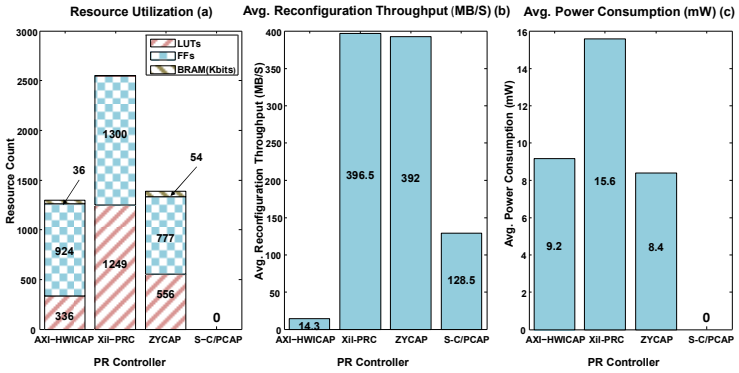


Fig. 3. (a) Resource Utilization , (b) Avg. Reconfiguration Throughput and (c) Power Consumption Comparisons between Different PR Controllers.

In this experiment, the reconfiguration time is computed by measuring the number of clock cycles required to complete the DPR task after a switching decision is issued by the ARM processor. The reconfiguration time is measured by AXI-Timer implemented on the PL side. (Fig. 3 (b)) shows the reconfiguration throughput of the four PR controllers used for SDR implementation. SDR designs using Xil-PRC and ZYCAP achieve an average reconfiguration throughput of {396.5, 392 MB/S} approximately equal 98 % of the ICAP throughput.

### C. *Resource Utilization and Power Consumption*

The resource utilization for the four PR controllers used in SDR designs is shown in (Fig. 3 (a)). Xil-PRC utilizes 4X LUTs as compared to the AXI-HWICAP, while ZYCAP utilizes 556 LUTs approximately less than 2X of AXI-HWICAP LUTs. (Fig. 3 (C)) shows the average estimated power consumption of the four PR controllers used in the SDR designs. ZYCAP consumes the minimum power in comparison with the other PR controllers implemented on the PL side.

### D. *Design Insights and Recommendations*

Xil-PRC and ZYCAP will be always recommended for SDR designs that require high reconfiguration speed as shown in Fig. 2. Low power DPR-based SDR designs that use S-C / PCAP do not require any resources on the PL side for DPR. Therefore, the power consumption is the power consumed by the ARM during the reconfiguration as shown in (Fig. 3(a, c)), but the drawback of this scheme is that the ARM processor is blocked from doing other software tasks during reconfiguration time. Xil-PRC achieves the highest reconfiguration throughput at the expense of a large resource utilization. Meanwhile, ZYCAP provides an acceptable reconfiguration throughput with a reduction in power consumption and resource utilization making it a good choice for DPR-based SDR designs.

## VI. CONCLUSION

In this paper, a four partial reconfiguration controllers are used to implement a high-speed reconfigurable SDR system targeting a Xilinx Zynq FPGA. A reconfigurable convolutional encoder is implemented as a benchmark to evaluate the perfo-rmance of the four partial reconfiguration controllers. This work provides essential design guidelines for the DPR-SDR designers to choose the suitable PR controller based on their system requirements.

## ACKNOWLEDGEMENT

This research was partially funded by Cairo University, ITIDA, NTRA, NSERC, Zewail City of Science and Technology, AUC, the STDF, Intel, Mentor Graphics, SRC, ASRT and MCIT.

## REFERENCES

[1] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," in IEEE Personal Communications, vol. 6, no. 4, pp. 13-18, Aug 1999.

[2] J. Delahaye, G. Gogniat, C. Roland and P. Bomel, "Software radio and dynamic reconfiguration on a DSP/FPGA platform", Frequenz, vol. 58, no. 5-6, pp. 152-159, 2003.

[3] E. McDonald, "Runtime FPGA partial reconfiguration", in 2008 IEEE Aerospace Conference, 2008, pp. 1-7.

[4] Xilinx Inc. “Partial Reconfiguration User Guide UG909" v2016.1, April 2016.

[5] A. Hassan, R. Ahmed, H. Mostafa, H. A. H. Fahmy and A. Hussien, "Performance evaluation of dynamic partial reconfiguration techniques for software defined radio implementation on FPGA," 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS), Cairo, 2015, pp. 183-186.

[6] S. Shreejith, B. Banarjee, K. Vipin and S. Fahmy, "Dynamic cognitive radios on the Xilinx Zynq hybrid FPGA", in International Conference on Cognitive Radio Oriented Wireless Networks, 2015, pp. 427-437.

[7] S. Liu, R. Pittman, A. Forin and J. Gaudiot, "Minimizing the runtime partial reconfiguration overheads in reconfigurable systems", The Journal of Supercomputing, vol. 61, no. 3, pp. 894-911, 2012.

[8] M. Liu, W. Kuehn, Z. Lu and A. Jantsch, "Run-time Partial Reconfiguration speed investigation and architectural design space exploration," 2009 International Conference on Field Programmable Logic and Applications, Prague, 2009, pp. 498-502.

[9] K. Vipin and S. A. Fahmy, "A high speed open source controller for FPGA Partial Reconfiguration," Field-Programmable Technology (FPT), 2012 International Conference on, Seoul, 2012, pp. 61-66.

[10] K. Vipin and S. A. Fahmy, "ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq," in IEEE Embedded Systems Letters, vol. 6, no. 3, pp. 41-44, Sept. 2014.

[11] Xilinx Inc. “AXI HWICAP PG134” v3.0, October 2016.

[12] Xilinx Inc.“Partial Reconfiguration Controller PG193” v1.1, October 2016.

[13] C. Kohn “Partial Reconfiguration of a Hardware Accelerator on Zynq-7000 All Programmable SoC Devices XAPP1159”, Xilinx Inc. Application Notes, 2013.

[14] A. Sadek, H. Mostafa and A. Nassar, "Dynamic channel coding reconfiguration in Software Defined Radio," 2015 27th International Conference on Microelectronics (ICM), Casablanca, 2015, pp. 13-16.

[15] Xilinx Inc. “ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC UG850”, September 2015.