A Cost-Effective Dynamic Partial Reconfiguration Implementation Flow for Xilinx FPGA

Ahmed Kamaleldin[1], Islam Ahmed[2], Abulfattah M. Obeid[4], Ahmed Shalash[1], Yehea Ismail[3], Hassan Mostafa[1,3]
[1]Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt.
[2]IC Verification Solutions, Mentor Graphics, Cairo, Egypt.
[3]Center for Nano-electronics & Devices, American University in Cairo & Zewail City for Science and Technology, Cairo, Egypt.
[4]King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia.
{ah.kamal.ahmed@gmail.com, islam_ahmed@mentor.com, obeid@ieee.org, shalash@ieee.org, y.ismail@aucegypt.edu, hmostafa@uwaterloo.ca}

Abstract— **Reconfigurability of Field Programmable Gate Array (FPGA) makes it one of the most promising approaches in the implementation of reconfigurable systems. Partitioning the reconfigurable system to many Reconfigurable Modules (RMs) and allocating them into Reconfigurable Regions (RRs) on the FPGA is a challenging task for the system designer. Partitioning choices impact the area efficiency and the time of reconfiguration of the reconfigurable systems. In this paper, different partitioning techniques are studied and evaluated according to their impact on reconfiguration time and the area utilization. Also, a new proposed Dynamic Partial Reconfiguration (DPR) tool flow is presented that automates and optimizes the partitioning procedure based on a graph clustering algorithm, modifies the design's HDL files as per the partitioning results, and implements a routing switch to dynamically change routing between Reconfigurable Regions (RRs) during reconfiguration.**

Keywords— Design Automation; Field Programmable Gate Array; Dynamic Partial Reconfiguration.

I. Introduction

Reconfigurable systems are widely adopted by designers to implement complex applications to reduce hardware cost and improve the application performance. Implementation of Software Defined Radio (SDR) is an example of a reconfigurable system that can switch between different wireless standards at runtime reusing the same hardware platform [1]. Most recent family series of Field Programmable Gate Arrays (FPGA) are suitable hardware platforms for implementing reconfigurable systems due to their high performance and flexibility. Dynamic Partial Reconfiguration (DPR) technique offers the flexibility to reconfigure a portion of the FPGA while the rest of the FPGA remains active at runtime.

The DPR system consists of a number of Reconfigurable Modules (RMs). Each RM has a number of modes that are changed at runtime according to the DPR system operating modes. These RMs are physically implemented on specific locations on the FPGA defined by a partially Reconfigurable Region (RR) to isolate the reconfigurable parts of the system from the static parts. Partitioning is the process of determining the number and the size of RRs based on modes of the RMs and the system operating modes. Reconfiguration time is an important parameter while reconfiguring the system from one operational mode to another during runtime and it is proportional to the area of the RRs. Therefore, efficient partitioning schemes are needed to provide a reduction in the reconfiguration time and a high utilization of the area of the RRs.

In this paper, we propose a DPR tool flow based on Xilinx DPR tool flow. The proposed tool flow contributes to the Xilinx DPR tool flow by 1) automating and optimizing the partitioning process to reduce the reconfiguration time and the total area of the DPR system by modifying the partitioning algorithm in [3], 2) implementing a new HDL wrapper generation utility to modify the design's HDL files based on the partitioning results as it may require merging one or more modules in the same RR, 3) implementing a reconfigurable routing switch to re-connect the RRs during runtime to overcome the problem of static routing between RRs. Experimental results and analysis are carried out using Xilinx 7-series FPGA.

This paper is organized as follows: Section II shows the related work for partitioning process for Partial Reconfigurable (PR) systems, and the traditional DPR designs flow, and Section III presents the proposed DPR tool flow. Partitioning algorithms and routing techniques are discussed in Section IV. Section V demonstrates the experimental results and analysis. Finally, Section VI draws the paper conclusion.

II. Background and Related Work

Several works have proposed techniques to solve the partitioning problem to optimize the RR allocation for PR designs. Many types of optimization and task partitioning algorithms are taken from other research trends to find an efficient solution for the DPR partitioning case. In [2] an optimization method using Integer Linear Programming (ILP) is proposed to determine the optimal partitioning solutions with the minimum reconfiguration time and partitions area. An improvement in area and reconfiguration time is achieved.

An automated partitioning tool is proposed by [3] based on a modified hierarchical clustering algorithm. The tool finds the optimal partitioning schemes with the minimum reconfiguration time and no optimization is done for the area of RRs. A simulated annealing based algorithm is proposed by [4] to determine modules allocation to RRs based on minimizing the partitions area but the reconfiguration time is not considered. An automated Hardware/Software (HW/SW) partitioning tool flow for HW/SW co-design systems is introduced in [5] by allocating RMs to a reconfigurable partition region on the FPGA or to be executed by a microprocessor. In [6], a run time temporal partitioning is proposed to reduce the reconfiguration time. An inter-module communication is implemented to re-connect the reconfigurable partition regions during reconfiguration to enable temporal partition assembly.

Currently, Xilinx and Intel (Altera) are the main FPGA devices vendors on the market. They provide a series of FPGA families that support the DPR design flow. In this paper, we consider only the Xilinx DPR design flow through Xilinx FPGA design software: Xilinx Vivado Design Suite or Xilinx PlanAhead tool.

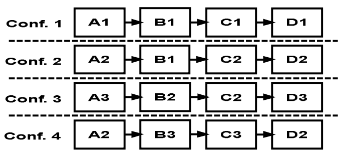

Fig. 1. An example of DPR design with four modes of configuration and four reconfigurable modules per configuration (A, B, C, and D).

Xilinx DPR design flow [8] requires the division of the DPR system into static logic blocks and reconfigurable logic blocks. A DPR system consists of a number of operating modes known as configuration modes (CMs) and each CM contains a number of RMs modes which are changed from a CM to another at runtime. Fig. 1 shows an example of a DPR system with four CMs (Conf.1, Conf.2, Conf.3, and Conf.4). Each configuration has four RMs (A, B, C, D) with three modes: A1, A2, A3; B1, B2, B3; C1, C2, C3; D1, D2, D3. The DPR design flow is implemented by Xilinx tools through the following steps: 1) synthesize the Reconfigurable Modules (RMs), 2) define the RRs (Partitioning), 3) floorplanning, 4) place and route, and 5) Partial Bitstream Generation.

For the partitioning, the RR includes different types of resources according to the requirements of its allocated RM modes for all possible configurations. There are two approaches for manual partitioning [2]. The first one, known as the single region partitioning approach, requires the allocation of all the RMs into a single large size RR that holds the most resources required by the largest configuration in the system, while the second one is a modular way for partitioning by assigning each RM into a single separate RR that could be reconfigured from one RM mode to another at runtime.

The single partition region approach has fixed time of reconfiguration, as switching from a CM to another requires the whole area of the RR to be reconfigured, and it has an advantage on area consumption since the area is equal to the minimum required reconfigurable partition area for a DPR design which is the area of the largest CM. For the modular approach, the number of RRs is equal to the number of RMs of the DPR system and the size of each RR is the size of the largest RM mode assigned to this RR. For example, in Fig. 2 there are four RRs as the system consists of four RMs, the sizes of RRs are shown in Fig.2 (RR1: A2; RR2: B1; RR3: C1; RR4: D1). This approach offers a less reconfiguration time compared to the first approach since the switching from a CM to another requires the reconfiguration of multiple small RRs without need to reconfigure the whole design as in single region partitioning approach, but it is least efficient in terms of area due to the total RRs size is equal to the sum of the largest mode size for each RM in the system as shown in Fig.2.

III. Proposed DPR Tool Flow

The proposed flow modifies the partitioning step in the Xilinx DPR design flow as shown in Fig. 3.

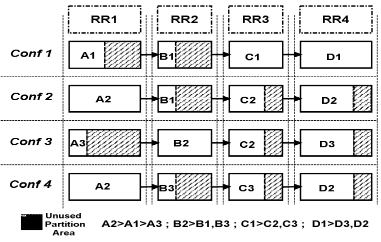

Fig. 2. An example of one module per region partitioning approach.

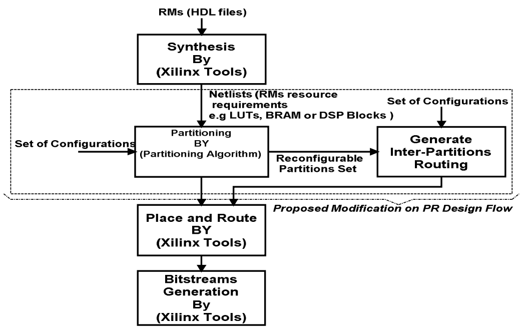
RMs (HDL files)
Synthesis By (Xilinx Tools)
Netlists (RMs resource requirements e.g LUTs, BRAM or DSP Blocks)
Set of Configurations
Set of Configurations
Partitioning BY (Partitioning Algorithm)
Reconfigurable Partitions Set
Generate Inter-Partitions Routing
Proposed Modification on PR Design Flow
Place and Route BY (Xilinx Tools)
Bitstreams Generation By (Xilinx Tools)

Fig. 3. Dynamic Partial Reconfiguration design flow with the proposed partitioning and routing tool.

The partitioning algorithm tries to determine the optimum partitioning scheme by finding the optimum sizes and numbers of RRs for a given DPR system reducing the total reconfiguration time and the total RRs area. It assigns different modes for different RMs to the same RR to save the total partitions area and reduces the reconfiguration time as shown in Fig. 4. As a consequence, the connections between the RRs should be changed from a configuration to another. For example, as shown in Fig. 4, in Conf.1, the output of RR1(B1) is connected to the input of RR3 (C1), after switching to the Conf.2 the output of RR1 (B1) should be connected to the input of RR2 (C2) to maintain the functionality of the DPR system, a proposed routing switch is used to handle that problem by re-routing the input/output signals between the RRs during the reconfiguration, the routing switch is a custom reconfigurable IP Core in the static region.

The algorithm as well may merge two or more RMs modes of the same design CM to the same RR as example {A1, B1} or {A1, C1}, so a proposed HDL wrapper is used to generate an HDL wrapper for this case. The proposed tool introduces the term of "*Virtual Flexible Reconfigurable Partition*" by combining the partitioning algorithm and the routing switch together to virtually change the partition size during reconfiguration to fit with the size of the RMs modes as a solution to save the total area as shown by the red dashed circles in Fig. 4 for RM (A).

IV. Partitioning Algorithms and Routing Technique

A. Partitioning Algorithms

An open source hierarchical clustering algorithm proposed in [3, 7] is used for the automatic partitioning scheme in the proposed flow. In this section, we describe 1) the original algorithm proposed in [3], and referred to it by (Vipin partitioning algorithm). 2) Our proposed modification on Vipin partitioning algorithm, and referred to it by (proposed partitioning algorithm).

1) Vipin Partitioning Algorithm: A clustering partitioning algorithm [3] represents the DPR design by a graph network, the node weight is the number of times the mode occurs in the configurations and the edge weight between two modes is the number of times these two modes occurs concurrently. First step; the algorithm iterates on the graph network to generate a base partitions set which is the set of all possible partitions for the design, this set contains partitions with single and mergeable RM modes partitions (e.g. Fig.1 {(A1);(C2);(A1,B1);(B2,C2)}).

Second step; a candidate partitions set is selected from the base partition set, the selection of candidate partitions is done such that 1) the selected partitions contain all the RM modes to cover all the possible configurations of the PR design, and 2) all the base partitions that contain mergeable non-consecutive RM modes are excluded. For example in Fig.1, a partition {A2-C3} contains a non-consecutive RM modes that are not physically mergeable inside one partition due to routing constraints. A valid mergeable partition could be for example {A2-B1} or {B1-C2}.

Third step; it searches for the compatible set of partitions from the candidate partitions sets. Compatible partitions are two

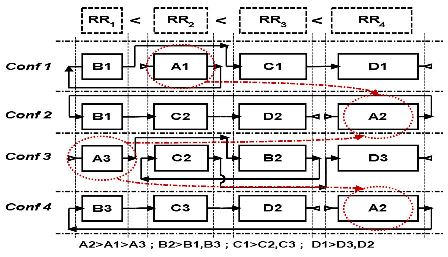

Fig. 4. A Partitioning solution where different modes belong to different RMs share the same RR, so re-routing is required between RRs. The red dashed circles shows the concept of virtual flexible reconfigurable partition.

or more candidate partitions in which their RM modes do not co-occur in any of the configuration in the same time. Final step; the algorithm iterates on the compatible set of partitions to find sub-optimal solutions of the final set of partition regions. The optimization is based on minimizing the total reconfiguration time of the DPR design. This algorithm reduces the total reconfiguration time at the expense of total partitions area by assigning large size RM modes to static partitions as some of RM modes are not always active in all possible configurations which lead to a reservation of unutilized area during runtime.

2) Proposed Partitioning Algorithm: In some cases of PR designs, Vipin partitioning algorithm minimizes the total reconfiguration time at the expense of the total area. The proposed partitioning algorithm solves this problem by minimizing the total area and the total reconfiguration time simultaneously. In The proposed algorithm, the selection of candidate partitions set includes the base partitions that contain mergeable non-consecutive RM modes into the candidate partitions set for example {A2-C2}, {B3-D2} shown in Fig. 1. The proposed partitioning algorithm takes a long runtime compared to (Vipin partitioning algorithm) since the number of iterations is increased to find the optimal compatible partitions set from the candidate partitions set which is augmented by adding the new mergeable partitions in the set.

B. *Inter-partition routing technique*

Routing or communication between RRs is modified in the proposed partitioning algorithms since the optimal partitioning solution is a set of partitions that different modes from different RMs share the same RR in different configurations as shown in Fig. 4. A rerouting technique between RRs is needed to maintain the validation of the design data flow. A proposed routing switch is shown in Fig. 5. The routing switch is implemented with a number of multiplexers and decoders in the static region of the PR design. The switch input ports are connected to all RR output ports and the switch output ports are connected to all RR input ports. A finite state machine (FSM) is designed to control the routing switch by sending control signals to the switch to change the routes between input ports and output ports in the time of reconfiguration. The routing switch has a routing map of all RM modes locations for every configuration in the DPR system.

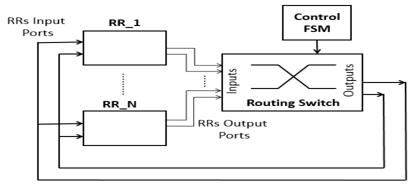

Fig. 5. Inter-partition routing technique

V. EXPERIMENTAL RESULTS AND ANALYSIS

In this paper, we propose an automatic cost-effective DPR flow considering the 7-series Xilinx FPGA. Four ways of partitioning schemes are considered in this analysis: 1) the single region; 2) the one module per RR 3) Vipin automatic partitioning algorithm [3]; 4) the proposed automatic partitioning algorithm. The two automatic partitioning algorithms are implemented using Python programming language. The performance analysis is based on 1). The total reconfiguration time is calculated as:

$$t_{total} = \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} tcon_{i,j} \qquad j > i \qquad (1)$$

Where c is the total number of configurations per design and $tcon_{i,j}$ is the reconfiguration time required to change the system from mode i to mode j which is measured by the number of configuration frames. 2) The total RRs area used by the PR system which is calculated as:

$$A_{total} = \sum_{i=1}^{N} RRarea_i \qquad (2)$$

Where $RRarea_i$ is the area of the RR_i calculated by the number of tiles and N is the total number of RRs used by DPR system. A number of PR designs are required as benchmarks to evaluate the performance of the algorithms. Unfortunately, a few number of PR designs are available in literature. Hence 16 synthetic PR designs are generated to test the algorithms as shown in Table I.

Fig. 6, shows a comparison of the total reconfiguration time, the time of the single module per region scheme is the highest since the switching between modes requires the reconfiguration of all RRs, and the RR size for each RM is equal to the size of the largest mode belong to that RM. The time for proposed partitioning algorithm is better than Vipin's one in all DPR designs cases. Since the optimum solution of merging some of the non-consecutive RM modes of the same configuration into a single RR saves the total reconfigured area and correspondingly the reconfiguration time is reduced. Moreover, from Fig. 6 (c, d) when the number of RMs per configuration decreases, the time for the two partitioning algorithms is approximately the same since the merging of non-consecutive RM modes in that design cases does not lead to an impressive reduction in the total RRs area. The two algorithms improve the total reconfiguration time by 30% to 40% compared to the traditional partitioning schemes.

Fig. 7, shows a comparison of the total RRs area, the single region partitioning scheme has the least RRs area among the other schemes. Vipin's algorithm consumes a total RRs area more than the single region scheme in all the DPR design cases. The proposed portioning algorithm reduced the total RRs area by allowing the merging of some non-consecutive RM modes in the same configuration into the same RR. In addition, from Fig. 7 (c, d) at design cases with a small number of RMs the total RRs area obtained by the proposed algorithm is approximately equal the minimum partitions area that a PR designs could reach.

The proposed algorithm improves the total reconfiguration time in average by ~5% and the total RRs area by ~10% compared to Vipin's one. As the number of RMs increase in PR designs, the proposed algorithm reaches a significant improvement percentage of ~20% over Vipin's one. For example, by applying the proposed algorithm on a PR design with 8 RMs and 10 configurations the total reconfiguration time is improved by a 21% and the total partitions area is improved by 16% compared to Vipin's one. It's clear that the proposed algorithm offers a significant improvement by reducing the cost of partitioning of DPR designs with a large number of RMs and configuration.

VI. Conclusion

The techniques of partitioning and resources allocation of the system RMs to RRs have a direct impact on the performance of PR systems. A performance evaluation of four partitioning techniques is presented in this paper considering the reconfiguration time and total reconfigurable area. It is shown that manual techniques of partitioning are not suitable for time critical DPR systems. A modified DPR tool flow is proposed in this paper to improve the time of reconfiguration by 40% compared to the traditional flow, and it reduces the design area by virtually changing the RRs size during reconfiguration to be fit with the size of active RMs using a routing switch to reconnect the RRs according to the active CM.

VII. 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.

TABLE I. The 16 generated DPR designs used as benchmark

16 DPR Designs	# Of conf./design	# Of RM/conf.	4 modes per RM
1-4	3,4,5,6	6(A,B,C,D,E,F)	A1-A4; B1-B4; C1-C4;D1-D4; E1-E4;F1-F4
4-8	3,4,5,6	5(A,B,C,D,E)	
8-12	3,4,5,6	4(A,B,C,D)	
12-16	3,4,5,6	3(A,B,C)	

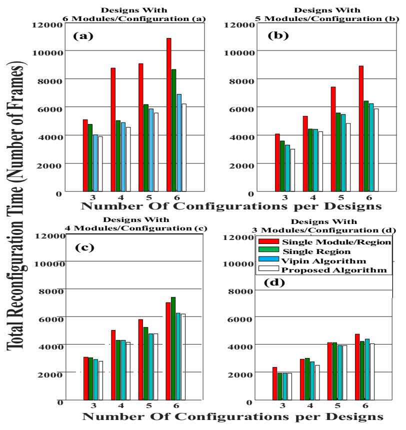

Fig.6. Total Reconfiguration time of the four partitioning approaches for DPR designs with (a) 6 RMs,(b) 5 RMs,(c) 4 RMs,(d)3 RMs per configuration.

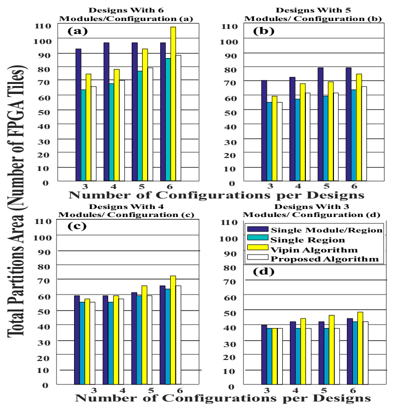

Fig. 7. Total RRs area of the four partitioning approaches according to DPR designs with (a) 6 RMs ,(b) 5 RMs , (c) 4 RMs , (d) 3 RMs per configuration

REFERENCES

[1] A. Kamaleldin, A. Mohamed, A. Nagy, Y. Gamal, A. Shalash, Y. Ismail, H. Mostafa, "Design Guidelines for the High-Speed Dynamic PartialReconfiguration Based Software Defined Radio Implementations on Xilinx Zynq FPGA", 2017 IEEE International Symposium on Circuits and Systems (ISCAS 2017), Baltimore, USA, May 2017, pp 1-4.

[2] K. Vipin and S. A. Fahmy, "Efficient region allocation for adaptive partial reconfiguration," 2011 International Conference on Field-Programmable Technology, New Delhi, 2011, pp. 1-6.

[3] K. Vipin and S. A. Fahmy, "Automated Partitioning for Partial Reconfiguration Design of Adaptive Systems," 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops, and Phd Forum, Cambridge, MA, 2013, pp. 172-181.

[4] A. Montone, M. Santambrogio, D. Sciuto, S. Memik, "Placement and Floorplanning in Dynamically Reconfigurable FPGAs",ACM Transactions on Reconfigurable Technology and Systems, vol. 3, pp. 1-34, 2010.

[5] S. Yousuf and A. Gordon-Ross, "An Automated Hardware/Software Co-Design Flow for Partially Reconfigurable FPGAs," 2016 IEEE Computer Society Annual Symposium on VLSI, Pittsburgh, PA, 2016, pp. 30-35.

[6] A. Jara-Berrocal and A. Gordon-Ross, "Runtime Temporal Partitioning Assembly to Reduce FPGA Reconfiguration Time," 2009 International Conference on Reconfigurable Computing and FPGAs, Quintana Roo, 2009, pp. 374-379.

[7] M. Srinivas and C. K. Mohan, "Efficient clustering approach using incremental and hierarchical clustering methods," The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010, pp.1-7.

[8] Xilinx Inc. "Partial Reconfiguration User Guide UG909" v2016.1. 2016