ISBA: An Independent Set-Based Algorithm for Automated Partial Reconfiguration Module Generation

Ruining He, Yuchun Ma, Kang Zhao, and Jinian Bian
Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing, P. R. China
myc@tsinghua.edu.cn

***Abstract*—Dynamic Partial Reconfiguration (DPR) on FPGAs has attracted significant research interests in recent years since it provides benefits such as reduced area and flexible functionality. However, due to the lack of supporting synthesis tools in current DPR design flow, leveraging these benefits requires specific designer expertise with laborious manual design effort. Considering the complicated concurrency relations among functions, it is challenging to properly select Partial Reconfiguration Modules (PR Modules) and partition them into groups so that the hardware modules can be swapped in and out during the run time. What's more, the design of PR Modules also impacts reconfiguration latency and resource utilization greatly. In this paper, we formulate the PR Module generation problem into a standard Maximum-Weight Independent Set Problem (MWISP) so that the original manual exploration can be solved optimally and automatically. Our proposed algorithm not only supports various design constraints, but also has the ability to consider multiple objectives such as area and reconfiguration delay. Experimental results show that our approach can optimize resource utilization and reduce reconfiguration delay with good scalability. Especially, the implementation of the real design case shows that our approach can be embedded in the Xilinx's DPR design flow successfully and it can save around 70% reconfiguration latency overhead compared with the heuristic PR Module generation approaches.**

Keywords-Dynamic Partial Reconfiguration; Partial Reconfiguration Module; Independent set-based model; FPGA

I. INTRODUCTION

A promising feature of an FPGA is the ability to reuse the same hardware for different tasks at different phases of an application execution. Moreover, the tasks can be swapped on the fly while other untouched part of the hardware continues to operate. This is known as dynamic partially reconfiguration (DPR) which presents interesting research challenges. Though DPR methodology is supported by modern FPGA architectures like Xilinx' Virtex series with DPR design flow EAPR [1], current tools still impose a number of limitations. The designer must manually define how to partition the designs into static part and Partial Reconfiguration Modules (PR Modules), requiring detailed knowledge of the FPGA architecture. The designer also needs to decide how to group the PR Modules into Partially Reconfigurable Regions (PR Regions) so that each group of PR Modules can be loaded or unloaded within their corresponding PR Region dynamically. To guarantee the feasibility of the DPR designs, the process of PR Module generation requires a laborious analysis which involves the time-based mutual exclusion relationship among hardware modules. What's more, the design of PR Modules heavily influences the trade-off between the cost of reconfiguration latency and the resource utilization. The systems with a large number of candidate PR Modules could be a big challenge. Without any supporting synthesis tool to aid the analysis, it is very hard for designers to make full use of DPR advantages and the poor design ability may lead to long development cycles or even design failure. Therefore, the lack of the automated PR Module design tools impedes the widespread usage of DPR technology. This inspires us to propose an automatic efficient algorithm which works for practical DPR design flow.

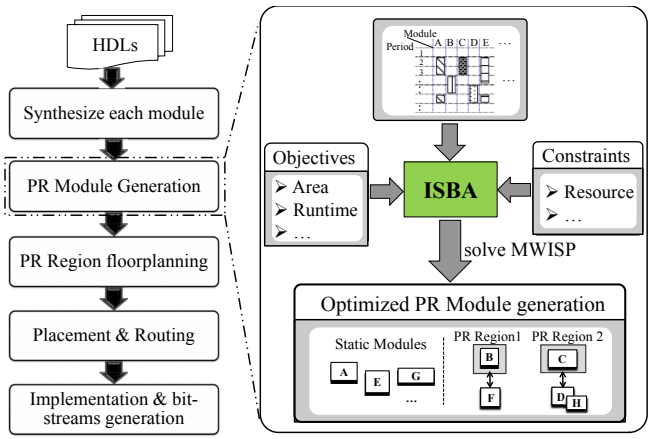


Figure 1. Diagram of Xilinx's EAPR design flow and our proposed ISBA.

In this paper, we propose an automated Independent Set-Based algorithm called ISBA which can be integrated into current design flow smoothly for PR Modules generation (as show in Fig.1). Instead of enumerating all the possible combinations, we transfer the original problem into a standard Maximum-Weight Independent Set Problem (MWISP) with design constraints. Therefore, ISBA can analyze the non-concurrency relations among candidate PR Modules and figure out optimal PR Module schemes. Based on ISBA, multiple optimization objectives such as resource minimization and reconfiguration delay minimization can be accomplished by solving a MWISP using well studied algorithms. The key contributions of this paper include:

- **A formal representation as a special Partial-Vertex Coloring Problem for the PR Module generation problem.** According to the analysis of the time-based non-concurrency relationship among candidate PR Modules, the PR Module generation problem is formally represented as a Partial-Vertex Coloring Problem (PVCP) with constraints.
- **Independent set-based optimization algorithm for PR Module generation.** To find the best PR Module combina-

This work was supported in part by NSFC 61076035, 60876030 and 61106030.

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA

tions represented by the partial vertex coloring, we transfer this Partial-Vertex Coloring Problem (PVCP) to a Maximum-Weight Independent Set Problem (MWISP) with constraints. Multiple objectives such as resource utilization and reconfiguration delay can be optimized under design constraints.

- **Support for modern DPR design flow**. Since our approach can replace the manual PR Module generation task in practical design flow, ISBA can be embedded into modern DPR design flow smoothly (as shown in Fig.1). By supporting traditional module-based high-level synthesis tools such as Mentor Graphics's Catapult® C Synthesis, ISBA successfully fills the gap between high-level synthesis and current DPR design flow, so that even the application specified with high-level programming language such as C/SystemC can be synthesized into partially reconfigurable hardware implementation.

The remainder of this paper is structured as follows. Section II presents some related works. Section III reports the problem formulation. In section IV, we discuss the formal representation of PR Module generation problem as a Partial-Vertex Coloring Problem (PVCP). In section V, we propose an independent set-based PR Module generation algorithm (ISBA) to solve this PVCP. In Section VI, we discuss how to optimize common design objectives based on ISBA. The experimental results on benchmarks are given in Section VII. We conclude our current work and discuss future work in Section VIII.

II. PREVIOUS WORKS

There are some studies which discuss DPR design flow such as [2] and [23]. In [2], Chris Conger et al. discussed DPR design framework in detail for special-purpose DPR system and multipurpose DPR system respectively. However, no solution to the selecting and partitioning of PR Modules are provided in their paper. In [23], a DPR design flow which tried to automate Xilinx's DPR design flow steps was proposed. However, they also didn't put forward how to solve this PR Module generation problem effectively. Due to the lack of the supporting tools, most of the current DPR designs are limited to only simple designs when non-concurrency constraints among alternative PR Modules can be derived directly from the application background [10, 25].

In [3] and [4], floorplanning methods for partial reconfiguration were described. But they still require the designer to provide the PR Module partitioning schemes manually. Authors of [25] proposed to efficiently allocate PR Regions for several given groups of PR Modules. But they mainly target a specific type of applications with several different execution modes and PR Modules have to be selected and partitioned into groups beforehand.

In [15], S. Ghiasi et al. proposed a reconfiguration sequence management problem which took scheduled operations as input. They supposed that all tasks are of equal area and focused on exploiting similarity between a given set of scheduled tasks. Their assumption is not practical because each task requires a particular group of active hardware modules in real designs. And reconfiguration overhead and resource requirement of operations are also various and they are key factors to design a highly efficient DPR system. Related studies such as online placement [5-7] attracted research interest in the past few years. But the increasing heterogeneous blocks and uneven routing in modern devices make these approaches very difficult to implement in reality and keep up with the times [16]. As a consequence, we feel it more meaningful and effective to address the challenges of DPR in the vendor-supported tool flows by developing algorithms to aid designers who are not architecture experts.

In reality, DPR imposes many physical constraints that need to be incorporated explicitly and existing related works often don't consider this special challenge. In this paper, we try to propose an effective algorithm to deal with PR Module generation in the vendor-supported DPR design flow. We not only support various real-life design constraints, but also try to consider multiple objectives such as resource utilization and reconfiguration delay.

III. PROBLME FORMULATION

According to the state-of-the-art DPR design flow EAPR, a PR Region is a designated rectangular region on the FPGA where a group of PR Modules can be loaded and unloaded dynamically. Each PR Region can accommodate at most one PR Module at any time. When implementing an application on a partially dynamically reconfigurable FPGA, the designer is required to select and partition PR Modules manually. Though some hardware modules such as the master module can be identified directly as static modules, analyzing the undetermined modules as candidate PR Modules still requires extensive labor from the designer, who is expected to know the details of the target FPGA architecture. In this paper, we build a model of the PR Module generation problem to provide efficient PR Module schemes for designers.

For a module-based application, the schedule of hardware modules during the application execution is represented with a Gantt chart derived from the outputs of high-level synthesis tools or provided by designers (as shown in Fig.2). Given a candidate PR Modules set $\mathcal{C} = \{M_1, M_2, \dots M_n\}$ with a Gantt chart $\mathcal{G}$ describing their schedule, the application execution life cycle can be divided into a series of periods and each module is active during certain periods. As shown in Fig.2, each column depicts the active periods of a specific hardware module. For example, module A is active during period i to i+1 and period i+4. For the purpose of demonstration, we will take the application from period i to period i+4 described in Fig.2 for an example throughout this paper.

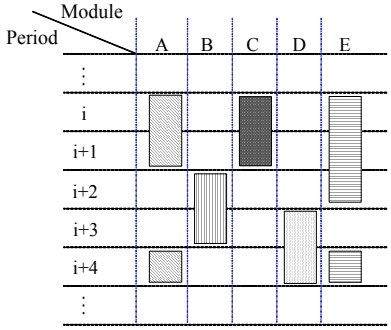


Figure 2. Schedule of hardware modules during application execution.

The PR Module generation problem is to select a subset $\mathcal{S} = \{M_{i1}, M_{i2}, \dots M_{im}\}$ of $\mathcal{C}$ as PR Modules and partition $\mathcal{S}$ into appropriate groups S_1, S_2, …, S_p such that $\bigcup_{k=1}^{p} S_k = \mathcal{S}$ and $\forall k1 \neq k2 \in \{1,2, \dots, p\}, S_{k1} \cap S_{k2} = \emptyset$. im and p are unknown variables to be determined. This process needs to optimize the target design objective under given constraints such as

- **Non-concurrency constraint**: According to modern DPR design flow, a PR Region can accommodate at most one PR Module at the same time. Therefore for each group of PR Modules sharing a PR Region, there must be at most one active module in this group in each period of the application execution. For example, as we can see from Fig.2, since module B and module C is not concurrently scheduled, it will satisfy this non-concurrency constraint if module B and module C are selected as PR Module*s* and share a PR Region on the FPGA fabric.
- **Resource requirement**: Each hardware module (static

module or activated PR Module) requires a specific group of resources whose number can be depicted by a *Resource Requirement Vector* [9]. *Resource Requirement Vector* is a 3-tuple vector $R_m=(m_{clb}, m_{ram}, m_{dsp})$ which represents the number of CLBs, BRAMs and DSPs required by a module *m*.

- **Design constraints**: Optional constraints such as the limit of available resources on target FPGA and the upper bound of the number of PR Regions can be restricted by the designer. The designer can also constrain the area discrepancy of PR Modules sharing a PR Region to reduce fragments caused by various resource requirements of the PR Modules. For applications where a strict limit of reconfiguration latency must be met, an upper limit of total reconfiguration delay can also be set as a constraint for design space exploration.

With these constraints considered, there may be several common situations the designer needs to address in real life. One case is that the designer needs to generate PR Module combinations with the minimized total reconfiguration latency and enable the application to fit into a given FPGA chip. Another case is that for applications which have no strict time constraint, the designer tries to minimize the resource utilization of the implementation with little concern for the reconfiguration latency. Therefore, the optimization approach of PR module generation needs the ability to explore the trade-off between resource utilization and reconfiguration delay.

In this paper, we propose a novel flexible formal model for this PR Module generation problem and an algorithm called ISBA to solve the problem efficiently. For different situations when different optimization objectives are required, ISBA succeeds in transferring the PR Region problem into a Maximum-Weight Independent Set Problem (MWISP) with constraints well considered.

IV. Exclusion Graph-based Represnetation of PR Module Generation Problem

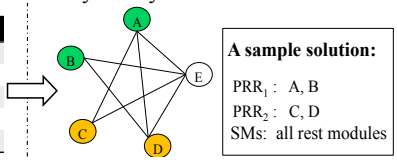
Considering the non-concurrency relations among hardware modules, we build a graph (*i.e. Exclusion Graph*) to formally describe the PR Module generation problem. *Exclusion Graph* is used to describe whether any two hardware modules meet the non-concurrency constraint discussed in Section III. First of all, we define several concepts and discuss what we can derive based on these concepts. At the end of the discussion, we will find that the PR Module generation problem can be formally described as a special Partial-Vertex Coloring Problem on the *Exclusion Graph*.

Definition 1 *Exclusive: Module A and module B are* ***exclusive*** *iff they are both active in one or more periods.*

Based on Definition 1, we can say that module A and module B meet the non-currency constraint if and only if they are not *exclusive* according to Definition 1. Now we construct a graph (called *Exclusion Graph*) to describe the *exclusive* relation between each pair of hardware modules. To accomplish this, for each period, we identify all active modules and we get the execution table as shown in Fig.3 A). Then we build a graph with 5 isolated vertices representing the 5 modules respectively. Next, for each period we complete a pair-wise connection between vertices which represent active modules in this period and finally we get Fig.3 B).

Similarly, we can say that a set of modules satisfy non-concurrency constraint if and only if any two modules in this set

All periods	Active modules
i	A, C, E
i+1	A, C, E
i+2	B, E
i+3	B, D
i+4	A, D, E



A) Execution Table B) Exclusion Graph

Figure 3. Build Exclusion Graph according to the module schedule.

are not *exclusive* which means there is no connection between any two modules in the set. Therefore, we can conclude that a solution to the PR Module generation problem satisfies non-concurrency constraint if and only if it consists of several intersection-free sets on *Exclusion graph* such that for every two vertices in each set, there is no edge connecting them.

In fact, to represent multiple independent sets on the *Exclusion Graph*, PR Module generation problem can be formally described as a Partial-Vertex Coloring Problem (PVCP):

✧ ***PVCP****: Consider an undirected graph G = (V; E), (i.e. Exclusion Graph in this paper) where V is the set of vertices and E is the set of edges. The Partial-Vertex Coloring Problem (PVCP) requires selecting certain sets (S_1, S_2, S_3, …, S_k) of vertices and assigning each set a unique color in such a way that*

1) No intersection: $\forall i, j \in \{1,2,\ldots,k\}, i \neq j, S_i, S_j \subseteq V, S_i \cap S_j = \varnothing$;

2) No connection: $\forall i \in \{1,2,\ldots,k\}, \forall\ v_a, v_b \in S_i, v_a \neq v_b$, *$v_a$ and v_b are not adjacent.*

Note that *k* corresponds to the number of PR Regions and it is unconstrained by default.

So far, we have represented the PR Module generation problem as a Partial-Vertex Coloring Problem (PVCP). Fig.3 B) also presents a sample of possible solutions. As we can see from this figure, {A, B} and {C, D} are two intersection-free sets assigned with two colors respectively. Green vertices {A, B} can be designated as PR Modules and share a PR Region PRR_1, while orange vertices {C, D} can be designated as PR Modules and sharing PRR_2. All the rest vertices are designated as static modules and mapped to static region on the FPGA fabric. Apparently, Theorem 1 can be established.

Theorem 1*: There is a one-to-one correspondence between the solutions to PR Module generation problem and the solutions to this Partial-Vertex Coloring Problem.*

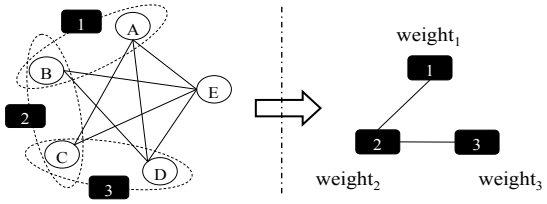
V. Independent Set-Based Algorithm (ISBA) For PR Module Generation Problem

In the previous section, we have formulated PR Module generation problem as a special Partial-Vertex Coloring Problem (PVCP) on the *Exclusion Graph*. In this section, we will discuss how to build a model to solve this special problem.

A. Identify alternative independent sets on Exclusion Graph

In graph theory, an *independent set* (or *stable set*) is a set of vertices in a graph, no two of which are adjacent [24]. According to the discussion in Section IV, due to the no connection requirement of PVCP, each valid solution must consist of a group of *independent sets*. Therefore, we can firstly identify all alternative *independent sets* on the *Exclusion graph* and then we can analyze relationships among them and choose an optimal group of *independent sets* under design constraints.

In this step, given the *Exclusion graph* (see Fig.4 A)), we identify all the alternative *independent sets* which constitute a set $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_t\}$. Fig.4 A) shows the situation that all *independent sets* are identified. Certain ineligible *independent sets*



A) Identify alternative independent sets. B) Intersection Graph.

Figure 4. Build the Intersection Graph.

can also be discarded according to given design constraints. Constraints like limiting the upper bound of the area discrepancy of PR Modules sharing a PR Region can be handled in this step. If the resource requirements of the PR Modules sharing a PR Region vary considerably, there would be many unused fragments when the smaller PR Module is active. Restricted by such constraints, ineligible *independent sets* that don't meet these constraints are discarded and only alternative *independent sets* are preserved, which not only favors the later-on optimization by limiting the solution space, but also controls the possible PR Module design to be uniform in terms of resource utilization.

B. Build Intersection Graph of alternative independent sets

In this step, given the set of alternative *independent sets* $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_t\}$ from the previous step, we cannot directly get the feasible coloring solution for PVCP since there may be some intersection between different *independent sets*. For example, as shown in Fig.4 A), the intersection of *independent set* 1 (*i.e.* {A, B}) and *independent set* 2 (*i.e.* {B, C}) is {B}. Therefore the two *independent sets* cannot be assigned with two different colors, which means it is unfeasible to include both of them in the solution to PVCP. According to the no-intersection requirement of PVCP in Section IV, we need to find an intersection-free subset of the alternative *independent sets* as $\mathcal{A}' = \{\mathcal{A}_{j1}, \mathcal{A}_{j2}, \dots, \mathcal{A}_{jh}\} \subseteq \mathcal{A}$ with each element in $\mathcal{A}'$ to be assigned a unique color and $\forall u \neq v \in \{j1, j2, \dots, jh\}, \mathcal{A}_u \cap \mathcal{A}_v = \emptyset$ because we cannot assign two different colors to a single vertex.

To describe the *intersection* relations between each pair of *independent sets* in $\mathcal{A}$, we construct an *Intersection Graph* (*G' = (V'; E')*) whose vertices represent the alternative *independent sets* and edges represent *Intersection* relations (see Fig.4 B)). If there is intersected part between two *independent sets*, their corresponding two vertices will be connected by an edge.

Apparently, every solution to the Partial-Vertex Coloring Problem corresponds to a set of vertices *SL* on the *Intersection Graph* such that $SL \subseteq V'$ and no two vertices of *SL* are adjacent. This means *SL* is an *independent set* of the *Intersection Graph* and we can solve the Partial-Vertex Coloring Problem (PVCP) by finding an *independent set* of the *Intersection Graph* with objectives optimized under design constraints. By assigning each vertex with the optimization objective evaluation as a weight, we can solve the Partial-Vertex Coloring Problem (PVCP) by solving a Maximum-Weight Independent Set Problem (MWISP) on the *Intersection Graph*. Design constraints such as limited resources or the upper limit of reconfiguration delay can be used to prune during the searching process to obtain some speed-up. The whole optimization algorithm based on the *independent set* model to solve the Partial-Vertex Coloring Problem (PVCP) is called ISBA.

C. ISBA-based PR Module generation flow

In this subsection, we conclude the PR Module generation flow based on our *independent set*-based algorithm (ISBA). As shown in Fig.5, firstly we build an undirected graph (*i.e. Exclusion Graph*) to describe whether any two given modules satisfy the non-concurrency constraint. And then the PR Module generation problem is formally represented as a PVCP with certain design constraints. To solve the PVCP, we identify all alternative *independent sets* on the *Exclusion Graph* and use an *Intersection Graph* to analyze intersection between each pair of these *independent sets.* With constraints and optimization objectives considered, the PR Module generation problem is finally formulated into a Maximum-Weight Independent Set Problem (MWISP) with certain PR Module design constraints.

Note that there is usually a few vertices (several dozen) on the *Intersection Graph* due to the few number of candidate PR Modules in real life. Exact algorithms which target optimal solutions can be employed. These algorithms have been well studied by many previous works such as [17-19]. When the problem size is large in extreme cases (for example, thousands of vertices on the *Intersection Graph*), there are two strategies which can be adopted by the designer. One strategy is that more static modules could be extracted manually so that less number of candidate modules are needed to be formulated by ISBA. The other is to employ some heuristic algorithms, which have also been well studied [20-22], to find approximate optimal solutions within reasonable time.

VI. Design Optimization Based On ISBA

In this section, we discuss common design optimization based on ISBA. Resource optimization and reconfiguration delay optimization are the two most important design objectives which often need to be addressed in real life.

A. Resource optimization based on ISBA

In practical work, the designer often needs to economize the available hardware resources on the FPGA chip. ISBA can help to minimize the resource consumption of the DPR implementation of the given application by providing optimal PR Module schemes which save the maximum resources.

In this paper, resource consumption is estimated according to the utilized area on the target chip. Area can be calculated in terms of the number of CLB's size. For example, as shown in Fig.6, the area of a BRAM is 3 CLBs, and the area of PRR_1 is 36 CLBs. A hardware module's area can be calculated by adding up the areas of all its required resources. Since a PR Region is shared by a set of PR Modules, its resources must be a superset of the all kinds of resources required by the PR Modules. Therefore its area can be estimated according to the maximum demand for each type of resource from the accommodated PR Modules.

$$area\_PRR_i = \sum_{k=1}^{num\_types} (area_k \times max\_demand_k) \quad (1)$$

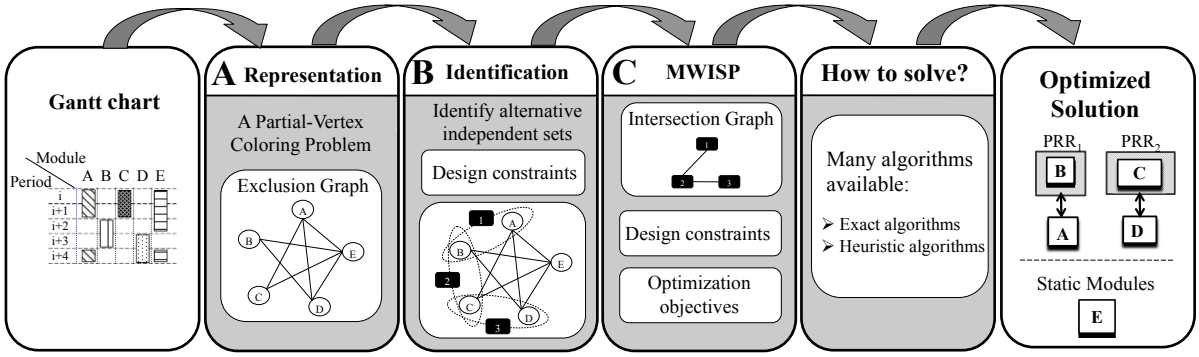


Figure 5. Work flow of our proposed ISBA for PR Module Generation.

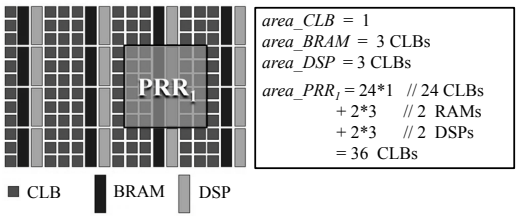


Figure 6. A schematic diagram of area estimation.

$area_k$ is the area of resource of type *k,* and max_demand_k is the maximum resource demand for type k from all the PR Modules sharing PRR_i. Since a set of PR Modules share a PR Region on the FPGA fabric, the area saved by a PR Region PRR_i can be calculated by

$$saved\_area\_PRR_i = (\sum_{PRM_k \in PRR_i} area\_PRM_k) - area\_PRR_i \quad (2)$$

Apparently, for a given feasible PR Module scheme, the saved area can be calculated by

$$saved\_area = \sum_{PRR_i \in \text{Partition}} saved\_area\_PRR_i \quad (3)$$

where $saved\_area\_PRR_i$ is calculated according to (2).

Based on ISBA, we know that each vertex on the *Intersection Graph* represents an alternative group of PR Modules which will share a PR Region if this vertex is included in the final PR Module scheme. Therefore, there is a potential gain (*i.e.* the saved area) for each vertex on the *Intersection Grap*h. The gain can be calculated according to (1)(2) and attached to the vertex as a weight.

The ISBA-based resource optimization turns out to be a standard Maximum-Weighted Independent Set Problem (MWISP). In other words, we need to find an *independent set* of the *Intersection Graph* and maximize the sum of the weights of the vertices in the selected *independent set*.

B. Performance optimization based on ISBA

Due to the swapping of PR Modules during system execution, reconfiguration overhead imposes a delay (*RD*) on the total application runtime. Therefore, the overall reconfiguration latency should be optimized with the trade-off between performance and resource utilization. Since all tasks have been scheduled and execution time of each task is already fixed, the application runtime is minimized *iff RD* is minimized. In other word, *RD* optimization equals runtime optimization in this problem, and therefore we put them together to discuss.

The total reconfiguration delay of a DPR design during the whole application life cycle (*TRD_DPR*) can be calculated by adding up the total reconfiguration delay (*TRD_PRR*) of each PR Region.

$$TRD\_DPR = \sum_j TRD\_PRR_j \quad (4)$$

Xilinx design flow estimates reconfiguration latency according to the maximum bandwidth of configuration port and the number of configuration frames to be reconfigured [1]. Therefore, we can estimate TRD_PRR_j according to the reconfiguration speed and the total amount of configuration frames to reconfigure PRR_j during the whole application life cycle.

$$TRD\_PRR_j = fn_j \times W_f / Rs \quad (5)$$

fn_j represents the total amount of configuration frames to reconfigure PRR_j. W_f is the frame lengths measured in bits. Rs is the maximum bandwidth of the configuration port. Actually, fn_j can be realistically estimated by (6) based on the number of configuration frames and switches of PRR_j.

$$fn_j = cf_j \times num_j \quad (6)$$

cf_j is the number of configuration frames occupied by PR Region PRR_j. num_j means the number of switches of PRR_j during the whole application periods. This calculation is reasonable because modern DPR technology requires a complete refresh of a PR Region whenever a PR Module is being swapped in it. In (6), cf_j is calculated by adding up the number of configuration frames of each resource type in PR Region PRR_j.

$$cf_j = \sum_{k=1}^{num\_types} (area_k \times max\_demand_k / area\_RF \times u_k) \quad (7)$$

area_RF means the area of a reconfigurable frame. Reconfigurable frame is the smallest size physical region that can be reconfigured and *area_RF* rests with the device series. u_k represents the number of configuration frames corresponding to a reconfigurable frame of resource type *k*, and it is also constant depending on specific FPGA chip.

To calculate num_i in (6), we need to analyze the execution table in Fig.3. Take the vertex 2 in the *Intersection Graph* (see Fig.4 B)) for example. There are two modules {B, C} in this vertex. Assume that they are chosen as PR Modules and sharing PRR_2 on the FPGA fabric. The calculation of num_2 for PRR_2 is shown in Fig.7.

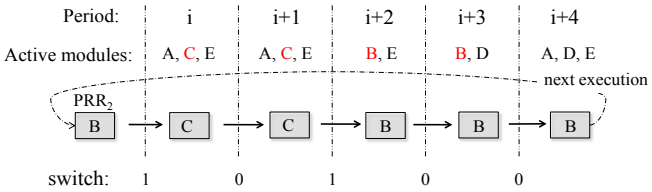


Figure 7. The calculation of num_2 for PRR_2.

Before the application begins to execute, module B occupies PRR_2. It is not module C because the initial state comes from the last execution which left module B in PRR_2. Since the application begins to execute, PRR_2 may switch once each time when a new period arrives. And the total number of switches of PRR_2 is 2 in one execution cycle. Therefore we can calculate num_i by

$$num_j = \sum_{k \in Periods} switch_k \quad (8)$$

Algorithm 1 presents the pseudo code to calculate num_i. Function *module*(*x*) means active modules in period *x* or all PR Modules sharing PR Region *x*. For example, *module*(*period i*) = {A, C, E} (see Fig.7).

Algorithm 1 Calculate num_j of PRR_j

```
FindRelatedPeriod (relatedPeriodSet);   // find all period related to PRRj
currentModule := module(PRRj) ∩ module(last period in relatedPeriodSet) ;
switch := 0;
For each period p in relatedPeriodSet   // in ascending order
    If module(PRRj) ∩ module(p) ≠ currentModule Then
        switch ++;
        currentModule := module(PRRj) ∩ module(p) ;
    End If
End For
Return switch ;
```

Based on ISBA, each feasible PR Module scheme is an *independent set* of the *Intersection Graph*. Similar to the discussion about resource optimization, each vertex on the *Intersection Graph* can be attached with the total reconfiguration delay of its represented

candidate PR Region as a weight. This weight is calculated according to (5)(6)(7)(8). Since available resources are limited, the final implementation of the design must be small enough to fit into the target FPGA. According to (1), for a given PR Module scheme, the area occupied by all the PR Regions can be estimated by

$$area_{PRR} = \mu \times \sum_{\mathrm{j}} area\_PRR_{\mathrm{j}} \quad (9)$$

where μ is a coefficient which enlarge area estimation by a certain ratio to make up for the underestimation due to the uneven distribution of available resources on the FPGA fabric. As all the rest hardware modules are designated as static modules, each of them will consume an independent area on the FPGA fabric. According to (9), we can estimate the area required by the PR Module scheme by

$$area\_total = \mu \times \sum_{\mathrm{j}} area\_SM_{\mathrm{j}} + area_{PRR} \quad (10)$$

In (9) and (10), μ is set according to the resource distribution on the specific FPGA chip. Finally, we conclude that a PR Module scheme satisfies resource constraints if

$$area\_total \leq area\_FPGA \quad (11)$$

where *area_FPGA* means available area on the given FPGA fabric.

In conclusion, ISBA-based performance optimization turns out to be a Minimum-Weight Independent Set Problem on the *Intersection Graph* under resource constraint. In other words, we need to find an *independent set* of the *Intersection Graph*, minimizing the sum of the weights of the vertices in the selected *independent set* while guaranteeing the DPR implementation can fit into the given FPGA chip. To solve it, we can transform it into an equivalent Maximum-Weighted Independent Set Problem (MWISP) by making the weight attached to each vertex to be its additive inverse. Then we can solve the MWISP by adding a pruning strategy based on the area constraint to the traditional MWISP algorithm framework. With the ISBA approach, it is also possible to optimize the resource utilization and reconfiguration cost at the same time by combining weights from different optimization objectives for each vertex on *Intersection Graph*.

VII. Experimental Results

In this section, we present experiments to demonstrate the efficiency of ISBA based on an exact MWISP algorithm [17] to find the optimal PR Module schemes. The goals of our experimental study are three-fold: 1) to demonstrate that ISBA can find optimal solutions using exact MWISP algorithms while spending little time, 2) to demonstrate ISBA's ability to effectively explore the PR Module generation space, and 3) to give a real-life example in which ISBA are helpful.

A. Efficiency of ISBA algorithms

To show the exploration ability of ISBA, we have implemented two ISBA-based approaches to find optimal PR Module combinations with different optimization objectives. The first algorithm minimizes the area of the design (**Resource Optimization**); the second algorithm minimizes the *RD* of the design under given resource constraint (***RD* Optimization**). The two approaches are implemented according to what we discussed in Section VI. In this experiment, we also constrain that the total number of PR Regions is no more than seven. This constraint is probable because too many PR Regions often generates high reconfiguration overhead and may exceed the available storage capacity of external memory according to DPR design flow [1].

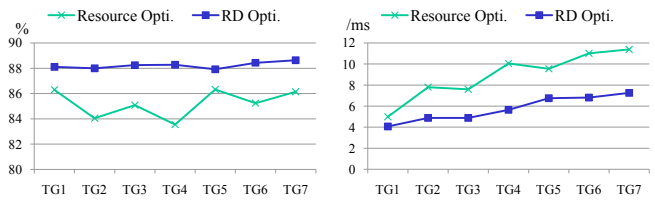
We take the XC5VLX330T fabric from Xilinx Virtex-5 series as our target FPGA chip. Using SelectMAP mode or the ICAP, the maximum bandwidth of configuration port is 3.2 Gbps. Due to the fact that no standard DPR benchmark is available, we have manually designed 7 groups of benchmarks with big problem scales according to the execution mode of common applications to evaluate our proposed ISBA approach. Each group consists of 10 different benchmarks which vary in parallelism degree and continuity of active periods to represent various types of applications. The *Resource Requirement Vector* [9] of each module is generated randomly while guaranteeing that the number of required CLB is between 200 and 700 and the number of required BRAM and DSP is 5%～10% the number of required CLB. For the ***RD* Optimization** algorithm, we also generate the available resource vector. The number of available resources on XC5VLX-330T is set to 85%～90% of that required by the original design.

Table I demonstrates the 7 groups of benchmarks (TG1, TG2, …, TG7) and their experimental results. In Table I, ***#V*** means the average number of vertices on the *Intersection Graph*. ***Avg. Origin Area*** means the average original area of the benchmarks before optimization. ***Avg. Area%*** shows the average percentage of the optimized area; ***Avg. RD*** represents the average reconfiguration delay after optimization; ***Avg. RT*** means average algorithm runtime. Note that there may be no solution to some benchmarks for ***RD* Optimization** algorithm due to the shortage of available resources for the given application. We don't take them into account in Table I when collecting statistics. As we can see from Table I, **Resource Optimization** approach provides PR Module scheme with the least resource utilization, but the *RD* is relatively high. ***RD* Optimization** approach needs to fit the application into a FPGA fabric with 10%～15% shortage of resource while to minimize *RD* at the same time. Limited available resource provides an additional pruning condition when solving the MWISP, which makes ***RD* Optimization** take less time than the **Resource Optimization**.

Fig.8 A) and B) compares the optimization effects of the two ISBA-based approaches. **Resource Optimization** provides the PR Module schemes with the least possible area consumptions, as shown by Fig.8 A). As the benchmark increase in complexity, the reconfiguration delay achieved by the two approaches tends to

TABLE I. EFFICIENCY OF THE TWO ISBA-BASED EXACT APPROACHES

Benchmark Groups			Avg. Origin Area	#V	Resource Optimization			RD Optimization		
ID	*#Candidate PR Modules*	*#Periods*			*Avg. Area %*	*Avg. RD (ms)*	*Avg. RT (s)*	*Avg. Area %*	*Avg. RD (ms)*	*Avg. RT (s)*
TG1	30	20	24278.0	44	86.30	4.983	< 0.01	88.10	4.058	< 0.01
TG2	35	30	27981.8	65	84.05	7.800	0.01	87.99	4.879	0.01
TG3	40	40	31962.5	114	85.08	7.596	0.09	88.24	4.876	0.05
TG4	45	50	35829.7	144	83.55	10.043	1.95	88.27	5.641	0.85
TG5	50	60	39698.2	154	86.32	9.548	4.83	87.91	6.748	1.97
TG6	55	70	43036.2	237	85.24	11.018	68.56	88.43	6.806	54.03
TG7	60	100	47034.4	295	86.14	11.385	78.91	88.63	7.258	70.07
ratio			1		0.85	1.55	1	0.88	1	0.82



A) Area optimization effect. B) RD optimization effect.

Figure 8. Optimization Effects of different approaches.

grow at different rates. Therefore the ability to effectively explore the trade-off between resource utilization and reconfiguration delay for PR Module generation is more critical.

In Table I, column *#V* shows the scale of the problem in terms of the number of vertices in *Intersection Graph*. Building the *Intersection Graph* is pretty efficient that we can build the corresponding MWISP in less than 10 ms for all the cases. Most of the runtime are spent on solving the MWISP so that the runtime of the approach relies on the scale of the *Intersection Graph* greatly. From Table I we can see that ISBA's problem scale tends to increase linearly as the benchmarks increase in complexity. As shown in Table II, even for the largest cases with 60 candidate PR Modules and about 300 vertices in the *Intersection Graph*, our ISBA-based approaches can find the optimal PR Module schemes within just a few minutes. Therefore, ISBA with exact algorithms to solve MWISP can deal with current modern applications efficiently.

B. A case study

Nowadays, advanced algorithms applied in medical, wireless, and consumer applications are more complicated than ever before. High-level synthesis (HLS) tools such as Mentor Graphics's Catapult accelerate design implementation by enabling C, C++ or System C specifications to be directly synthesized into production quality RTL, which provides designers with a faster and more robust way of delivering high quality designs. Our proposed ISBA can analyze the output of HLS tools and provide efficient dynamic partially reconfiguration schemes for the synthesized design. Fig.9 presents the complete tool chain which can be used by advanced algorithms to be implemented with DPR feature.

In this subsection, we demonstrate the design process by implementing an advanced algorithm GSM. GSM performs the linear predictive coding analysis of global system for mobile communications and we get it written in C++ code from

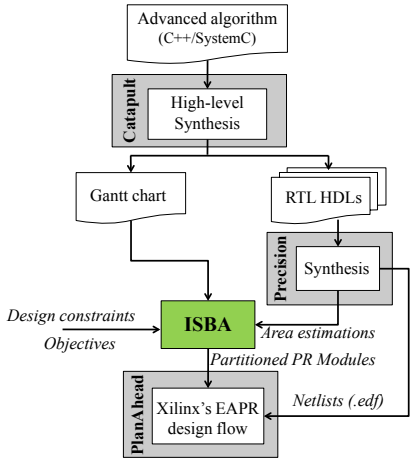


Figure 9. ISBA bridges the gap between HLS and current DPR design flow.

CHStone benchmark suite [11]. First of all, we synthesize GSM with Catapult targeting XC5VLX50T-FF1136 FPGA chip from Xilinx Virtex-5 series. We identify seven functions in the algorithm to be hardware modules by setting their hierarchy attributes to be "Block". They are *Autocorrelation*, *Quantization_and_coding*, *Reflection_coefficients*, *Gsm_mul*, *Gsm_div*, *Transformation_to_Log_Area_Ratios*, and *Main* respectively and they are all chosen as candidate PR Modules. The rest of the algorithm will be synthesized into some control modules, which are directly designated as static modules.

After the hierarchical synthesis, Catapult generates synthesizable RTL HDL files as well as a Gantt chart from which the schedule of hardware modules can be derived. Then we synthesize the HDL files using Precision Synthesis RTL 2011a.61 and get the *Resource Requirement Vector* of each module. Available resources on XC5VLX50T and the resources required by the hardware modules of GSM are described by Table II. As we can see from Table II, the available number of Slices on XC5VLX50T is not sufficient to implement GSM if DPR technology is not adopted.

Using our proposed ISBA approach, efficient DPR implementation schemes can be generated automatically for designers. In this experiment, ISBA tries to find the DPR scheme with the least reconfiguration delay within the resource limits as shown in Table II. Particularly, there are complex design constraints and the non-concurrency relations among candidate PR Modules. Since there are 88 vertices on the *Intersection Graph*, there is a large solution space which is impossible to explore all by hand. Our approach ISBA finds the optimal solution which consists of two groups of PR Modules with the total reconfiguration delay of 0.3646 ms during the whole execution period. Though there is no available automatic approach for PR Module generation to be compared with, we implement two heuristic algorithms to demonstrate the competence of ISBA. The first one (***RD* Greedy**) gives precedence to the vertex with the minimum *RD* among the vertices. Then the sorted vertices on the *Intersection Graph* are iteratively selected if the chosen vertex is not connected to any current selected ones. Once the area saved by increasing PR Module groups enables the DPR implementation to fit into the XC5VLX50T chip, the vertex selection stops immediately. While the second heuristic algorithm (**Area Greedy**) gives precedence to the vertex with the maximum saved area (see Eq. (3)) among the vertices which are not adjacent to any current selected vertices and have the *RD* cost no more than 1 ms. Experimental results of the comparison between ISBA and the two algorithms are shown in Table III. We can see that though the greedy approaches can also generate the feasible designs which can fit in the board in terms of resource utilization, the exploration ability of heuristic approaches including the reconfiguration cost aware heuristic approach is limited because the selected vertex will impose the no-intersection restriction on the later-on vertex selecting process which often leads to the local optimum. The

TABLE II. RESOURCE UTILISATION OF HARDWARE MODULES

Module	Slices	DSPs	BRAMs
Autocorrelation	5995	17	0
Quantization	192	0	0
Reflection	604	2	0
Transformation	156	0	0
Main	94	0	0
Gsm_mult	54	2	0
Gsm_div	47	0	0
other modules	627	0	0
#Total	7769	21	0
#Available	7200	48	60

TABLE III. THE COMPARISON BETWEEN ISBA AND TWO HEURISTIC ALGORITHMS

GSM	ISBA		RD Greedy		Area Greedy	
	Optimized RD (*ms*)	*Algor. Runtime*	*Optimized RD* (*ms*)	*Algor. Runtime*	*Optimized RD* (*ms*)	*Algor. Runtime*
Value	0.3646	< 10ms	1.2052	< 10ms	1.2651	< 10ms
Ratio	1		3.31 X		3.47 X	

Domino effect is hard to be overcome by greedy strategies and makes the PR Module scheme generated by the heuristic algorithms needs more than 3X reconfiguration overhead compared with our ISBA approach. By contrast, our approach provides an efficient way to optimally and automatically solve the PR module generation problem.

Taking the PR Modules design proposed by ISBA, we uniform the ports of each group of PR Modules and re-synthesize them separately with Precision Synthesis RTL 2011a.61. The generated netlist files can be fed to Xilinx's PlanAhead directly to build DPR implementation of GSM successfully. PlanAhead reports that the DPR implementation consumes about 6408 Slices and 16 DSPs on the XC5VLX50T chip. This means GSM can fit into XC5VLX50T by using DPR technology which saves area, power and cost. Fig.10 presents a portion of XC5VLX50T with the layout of PR Regions in PlanAhead. The successful design of GSM from advanced algorithm to DPR implementation demonstrates that our proposed ISBA can bridge the gap between high-level synthesis and current DPR design flow successfully.

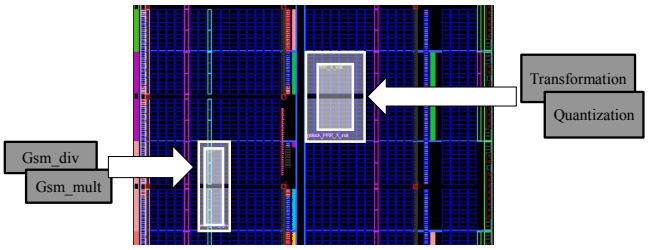


Figure 10. Part of the layout shown in PlanAhead with two PR Regions.

VIII. CONCLUSION AND FUTURE WORK

This paper proposed a novel algorithm (ISBA) for PR Module generation problem which is essential in modern DPR design flow. Firstly the PR Module generation problem is formally represented as a Partial-Vertex Coloring Problem (PVCP) with constraints and then an *independent set*-based model is proposed to solve this problem. ISBA formulates this PVCP as a Maximum-Weight Independent Set Problem (MWISP) with multiple objectives optimized and design constraints well considered. We also demonstrate the usefulness of our algorithm in modern FPGA design flow by showing the design process of an application in real world. As high-level synthesis tools are being popular and they are playing an increasingly important role in hardware design nowadays, our approach which can fill the gap between high-level synthesis of algorithm description and DPR implementations is very promising. In fact, HW/SW partitioning and task scheduling for DPR systems has been well studied by a lot of works such as [12-14]. The output of their works can be the input of ISBA if handled properly. We will work on the link between these works and ISBA in the future.

REFERENCES

[1] Xilinx, Partial Reconfiguration User Guide, March 1, 2011.

[2] C. Conger, A. Gordon-Ross, and A.D. George, "Design framework for partial run-time FPGA reconfiguration", in Proc. ERSA, 2008, pp.122-128.

[3] P. Banerjee, M. Sangtani, and S. Sur-Kolay, "Floorplanning for partially reconfigurable FPGAs", presented at IEEE Trans. on CAD of Integrated Circuits and Systems, 2011, pp.8-17.

[4] L. Singhal and E. Bozorgzadeh, "Multi-layer floorplanning on a sequence of reconfigurable designs", FPL, 2006.

[5] K. Bazargan, R. Kastner, et al., "Fast template placement for reconfigurable computing systems", presented at IEEE Design & Test of Computers, 2000, pp.68-83.

[6] A. Ahmadinia and J. Teich, "Speeding up Online Placement for XILINX FPGAs by Reducing Configuration Overhead", in Proc. VLSI- SOC, 2003, pp.118-122.

[7] H. Walder, C. Steiger, et al., "Fast Online Task Placement on FPGAs: Free Space Partitioning and 2D-Hashing", in Proc. IPDPS, 2003.

[8] A. Ahmadinia, C. Bobda, et al., "Optimal free-space management and routing-conscious dynamic placement for reconfigurable devices", IEEE Trans. Comput., 56(5):673–680, 2007.

[9] L. Cheng and M. D. F. Wong, "Floorplan design for multimillion gate FPGAs", IEEE Trans. on CAD of Integrated Circuits and Systems, 25(12):2795–2805, 2006.

[10] R. Tessier, S. Swaminathan, et al., "A reconfigurable, power-efficient adaptive Viterbi decoder", presented at IEEE Trans. VLSI Syst., 2005, pp.484-488.

[11] Y. Hara, H. Tomiyama, et al., "Proposal and Quantitative Analysis of the CHStone Benchmark Program Suite for Practical C-based High-level Synthesis",Journal of Information Processing, Vol. 17, pp.242-254, 2009.

[12] S. Banerjee, E. Bozorgzadeh, and N.D. Dutt, "Physically-aware HW-SW partitioning for reconfigurable architectures with partial dynamic reconfiguration", in Proc. DAC, 2005, pp.335-340.

[13] Y. Zhang, X. Hu, and D. Z. Chen, "Task Scheduling and Voltage Selection for Energy Minimization", in Proc. Of Design Automation Conference, pp.183-188, 2002.

[14] M.D. Santambrogio, M. Redaelli, and M. Maggioni, "Task graph scheduling for reconfigurable architectures driven by reconfigurations hiding and resources reuse", in Proc. ACM Great Lakes Symposium on VLSI, 2009, pp.21-26.

[15] S. Ghiasi and M. Sarrafzadeh, "Optimal reconfiguration sequence management", Proceedings of the 2003 conference on Asia South Pacific design automation, 2003, pp.359-365.

[16] D. Koch, C. Beckhoff, and J. Torrison, "Fine-grained partial runtime reconfiguration on Virtex-5 FPGAs," in Proc. FCCM, 2010.

[17] P.M. Pardalos and J. Xue, "The maximum clique problem", J. Global Optim. 4 (1994), no. 3, pp.301–328.

[18] D. Warrier, W.E. Wilhelm, J.S. Warren, and I.V. Hicks, "A branch-and-price approach for the maximum weight independent set problem", presented at Networks, 2005, pp.198-209.

[19] P.M. Pardalos and N. Desai, "An algorithm for finding a Maximum Weighted Independent Set in an arbitrary graph", *Int. J. Comput. Math.*, Vol. 38: 163-175, 1991.

[20] M. Pelillo, "Heuristics for Maximum Clique and Independent Set", Encyclopedia of Optimization, 2009: 1508~1520.

[21] I.R. Bomze, M. Pelillo, and V. Stix, "Approximating the Maximum Weight Clique using replicator dynamics", IEEE Trans. Neural Networks, 11(6):1228-1241, 2000.

[22] S. Busygin, S. Butenko, and P.M. Pardalos, "A heuristic for the Maximum Independent Set Problem based on optimization of a quadratic over a sphere", presented at *J. Comb. Optim.*, 2002, pp.287-297.

[23] S. Yousuf and A. Gordon-Ross, "DAPR: Design Automation for Partially Reconfigurable FPGAs", in Proc. ERSA, 2010, pp.97-103.

[24] http://en.wikipedia.org

[25] K. Vipin and S. Fahmy, "Efficient region allocation for adaptive partial reconfiguration", in Proc. FPT, 2011.