Paper Title*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

par 1
what is floorplanning and how is it applied (what's its role) in the context of PR, what is the benefit of having a good floorplan or a floorplan at all

par 2
How is fp done currently and what are the shortcomings of those methods

par 3
In short what did I do and what are my contributions (clearly stated)

par 4
outline of the work

Floorplanning is one of the major challenges in the field of dynamic parital reconfiguration. The placement of the static and reconfigurable slots on the FPGA fabric must satisfy the application requirements set by the application designer while also respecting the technological constraints set by the manufacturer. The conventional approach for an automated generation of FPGA floorplans usually involves two steps. First for each slot, all the possible rectangular slots that satisfy the resources requirement of the slot are enumerated. This is done by starting a scan on the fpga fabric from the bottom left corner and lisitng all the rectangles that contain all the necessary resources for the respective slots. Then some sort of heuristics/optimization is applied to choose the optimal ones from the set of possible slots. This approach has many problems {*to be listed later*}

Our approach instead focuses on applying the optimization process on a lower level of abstraction of the fpga fabric i.e., rather than applying optimization to select the most optimal one from a set of pre-scanned slots, we modeled the different types of resources on the fpga and their distribution along with forbidden regions as a set of constraints and added these constraints to the predefined constraints related to dpr.

Identify applicable funding agency here. If none, delete this.

Let us consider a floorplanning example where we have to make a floorplan for two slots S_1 and S_2 on the FPGA fabric. Each slot has resource requirements denoted as $\{D_1, B_1, C_1\}$ and $\{D_2, B_2, C_2\}$ where D, B and C represent DSP, BRAM and CLB respectively.
Our proposed system takes as an input in the resource requirement of each slot and a description of the resource distribution of the FPGA fabric and it returns the placement coordinates of the slots on the fpga fabric.
A slot is represented using 4 parameters i.e. the two bottom left coordinates and the width and the height of the slot. In our considered example the slots S_1 and S_2 are represented as (x_1, y_1, w_1, h_1) and (x_2, y_2, w_2, h_2). A forbidden region is also represented as a slot hence a forbidden region F_i can be represented as fx_i, fy_k, fw_i, fy_i

II. Related Works

III. Background and Modeling

In this section we breifly describe the general architecture of FPGAs, the design flow in partial reconfiguration (PR) and the assumptions that led to the fromulation of PR floorplanning problem as a MILP problem. This work is based on the 7 series FPGA family from Xilinx.

A. FPGA Architecture and Partial-Reconfiguration

The configurable fabric of Xilinx FPGAs is divided into quadrants named clock regions. Within each clock region there are columns of different configurable resources with non uniform distribution. These resource can be CLBs, BRAMs or DSPs. Resources within a clock region share the same clock. A single column in a clock region is referred to as a tile. The number of resources in a tile varies depending on the device family. For example in Virtex 7z a CLB tile contains 50 clbs a BRAM tile contains 10 brams and a DSP tile contains 20 dsps. The functional logic compoenets (clbs, brams and dsps) and the routing logic components (switches, interconnects etc...) on the FPGA are configured based on a bit file stored in the configuration memory of an FPGA. This memory is organized into minimal configurable units called frames. A single frame in the configuration memory corresponds to a single tile on the fabric. In addition to the above mentioned resources, FPGAs also contain other components such as clock and clock modifying logic, I/O logic, configuration logic etc... Depending on the type of device family some of these components may or may not be included in a reconfigurable region. FPGAs, in particular 7 series devices, which are subject of this work, also contian routing resources called interconnect tiles. These tiles are placed back-to-back as shown in the **fig**. When floorplanning for partial reconfiguration, the position of these back-to-back boundaries must be known inorder not to split them and violate PR restriction.

put picture detailing FPGA architecture with interconnect resources

Partially reconfigurable applications are often composed of $\boldsymbol{M_s}$ number of static and $\boldsymbol{M_r}$ number of reconfigurable modules that are to be placed in $\mathbf{N}_s$ static and $\mathbf{N}_r$ reconfigurable regions on the FPGA respectively. In PR applications the number of static modules equals to the number of static regions i.e., $M_s = N_s$ while the number of reconfigurable modules is always greater than reconfigurable regions i.e., $M_r > N_r$. Floorplanning in PR can then be defined as the process of allocating placement for N_r reconfigurable regions on the FPGA fabric.

state how PR-fp is currently done in vivado

B. Combining clock regions

The central clock column divides the FPGA into left and right regions as show on **fig**. But in our model we reduced the number of clock regions by combining all the horizontally adjacent clock regions into a single clock region. This simplifies our modeling without no penalty. As shown in the figure **fig**, a cartesian coordinate system can be overlayed on FPGAs to uniquely identify each resource on the logic fabric. The x axis represents each column of resources while the rows on the y axis represent fused horizontally adjacent clock regions. Combining the horizontally adjacent clock regions results in a low range of variables on the y axis. Added to that, organizing resources on the y axis on a per tile (per clock region) basis instead of as individual clbs, brams or dsps further contributes for a feasible floorplan as it constrains reconfigurable regions to be aligned to clock regions. Based on this abstraction the FPGA fabric is **W** columns wide and **H** clock regions high.

A reconfigurable rectangular region $R_i \in N_r$ is represented as

$\forall$ i = 1...,$N_r \wedge x_i, y_i, w_i, h_i \in \mathbb{Z}$

$$R_i = (x_i, y_i w_i, h_i) \mid x_i + w_i \leq W, y_i + h_i \leq H \qquad (1)$$

where x_i and y_i represent the bottom left coordinate and w_i and h_i represent the width and height of R_i resectively. A resource type *t* that is required by reconfigurable module M_r is denoted as c_{rt} while the same type of resource that is incorporated inside a reconfigurable region R_i is denoted as η_{it}.

talk about the modeling of forbidden regions

C. Discretization of the axis

Floorplanning is a two dimensional problem in that determining the number of resource contained in R_i invloves determining the resources on both axis i.e., determining the area of the rectangle. In formulating PR-floorplanning as a linear optimization problem the resource requirement constraints must also be linear. Hence to satisfy the condtion of modeling the resource requirement as a linear constraint either of the axis must be discretized. Deciding which axis to discritize is an important design decision since a reduced number of binary variables leads to an easily scalable model. In all the FPGA families that were chosen to be studied for this project, the number of rows (the number of clock regions on the y axis) was less than the number of columns on the x axis. For example in kintex xc7z045fbv676 there are 100 columns on the x axis as opposed to 7 rows (after combining all the horizontally adjacent clock regions separated by the central clock column) on the y axis. Hence we define

$\forall$ i = 1...,N_r, $\forall$ j = 1...,H
$\beta_{ij} \in$ [0,1] $\mid \beta_{ij}$ represents a clock region j in R_i.

D. FPGA resource finger-printing

As described in the previous section the proposed floorplanner takes as an input the resource description of the FPGA fabric and the resource requirement of each reconfigurable region and produces a feasible placement for each region. Resources in most FPGAs are distributed in a redundant manner this is to say that vertically adjacent clock regions have a fairly similar distribution of resources with the possibility of different forbidden clock regions being included in different clock regions. Hence describing the resources in a single clock region and the forbidden regions in all clock regions would be a fairly easy was of describing all the resources in all clock regions. The function f_t(x) is a piecewise function that can be used to describe the distribution of resource type *t* in the first clock region of the FPGA and μ_t is a constant that denotes the number resource *t* per tile. As an example f_t(x) for the clbs on **fig** can be described as

$$f_c(x) = \begin{cases} \mu_c * x, & \mathbf{0 \leq x < 4}, \\ \mu_c * (x-1), & \mathbf{4 \leq x < 7}, \end{cases} \tag{2}$$

fig depicts $f_t(x)$ for the first clock region zynq xc7z015.

put a picture of piecewise graphs for zynq for bram, clb and dsp

The height h_i of a reconfigurable region R_i is the sum of all the clock regions included in the region
$\forall$ i = 1...,N_r, $\forall$ j = 1...,H

$$h_i = \sum_{j=1}^{H} \beta_{ij} \tag{3}$$

Accordingly, the amount of each type of resource included inside a region R_i can be defined as

$$\eta_{it} = \sum_{j=1}^{H} \beta_{ij} \cdot (f_c(x_i + w_i) - f_c(x_i)) \tag{4}$$

IV. Floorplanning problem formulation

put a picture of PR design flow in Xilinx

Partial-reconfiguration involves dynamically switching modules in a reconfigurable region whilst other reconfigurable and static regions continue to be operational. **Fig** depicts the PR design flow in Xilinx FPGAs as implemented in Xilinx Vivado. The major steps in the automated PR flow are

- *Synthesis*: At this level the behavioral description of modules written in hardware description langauges (Verilog or VHDL) is converted into a gate-level netlist. In PR design flow, floorplanning is mandatory (it is optional in the standard flow) and it must be done before the implementation step.
- *Implementation*: In this step the gate-level netlist output of the previous stage is functionally mapped to specific device resources on the FPGA and then placed and routed.
- *Bitstream generation*: In PR desgin flow, the final output of an FPGA design tool is a set of bit files which contain the configuration information for both the logic and routing resources of the static and reconfigurable regions.

In floorplanning for PR, each R_i, which are also named *Pblocks* in the Xilinx design flow, are subject to the following restrictions and requirements.

1) Reconfigurable regions must contain only valid reconfigurable resources (for example in 7 series FPGAs only CLBs, BRAMs and DSPs must be in R_i)
2) The minimum number of resources incorporated by a reconfigurable region $\mathbf{R}_i$ (Pblock) must at least be equal to the resource requirement of the largest module hosted in the region
3) Two reconfigurable regions must not overlap. Overlapping is equivalent to sharing at least a single tile
4) The left and right edges of R_i must not split interconnect columns
5) Reconfigurable regions(Pblocks) can span non reconfigurable components such as configuration blocks, central clock column etc... without considering them as part of the region(Pblocks)
6) The height of R_i must be aligned to clock regions. This is an optional requirement but enforcing it results in starting the reconfigurable module from a known initial state re-configuration

V. MILP formulation

In this section the MILP model of the PR floorplanning problem is presented. The section first summarizes the considered assumptions and abstractions related both to the FPGA and its resources as well as the floorplanning problem. The description of the model consists of defining variables, constraints and an objective function to be optimized.

A. definition of optimization variables

Variables denoting the relationship between two slots
For two slots S_i and S_k

- $\gamma_{ik} \in [0,1]$ is a binary variable used to identify whether S_i is found on the left or on the right of S_k
  γ_{ik} = 1 if $x_i \leq x_k$ [i.e. S_i is on the left of S_k]

B. Constraint definition

1) ***Semantics constraints****:* The following constraints ensure the soundness of some of the variables.

Constraint 1: $\forall$ $S_1 \in$ N, $\forall$ i = 1...,N^{max} , $\forall$ x_i = 0..., W, $\forall$ y_i = 0..., H $\forall$ w_i, $\forall$ h_i

$$\begin{aligned} x_i + w_i \leq W \\ y_i + h_i \leq H \end{aligned} \tag{5}$$

meaning: the right most x coordinate and the top y coordinates of S_i must not exceed the boundaries of the fabric

Constraint 2: $\forall$ i = 1...,N^{max}, $\forall$ j = 1...,clk_reg^{max}, $\forall$ k = 1...,r $\forall$ h_i

$$h_i = \sum_{j=1}^{clk\_reg} \sum_{k=1}^{r} \beta_{ijk} \tag{6}$$

meaning: The height of S_i must be the sum of binary rows in each clock region which are set to 1

Constraint 3: $\forall$ i = 1...,N^{max}, $\forall$ j = 1...,clk_reg^{max}, $\forall$ k = 1...,r $\forall$ h_i

$$y_i \leq \sum_{j=1}^{clk\_reg} \sum_{k=1}^{r} H - \beta_{ijk} \cdot (H - (k + (r-1) \cdot j)) \tag{7}$$

meaning: y_i must be constrained not to be greater than the lowest chosen row

Constraint 4: $\forall$ i = 1...,N^{max}, $\forall$ j = 1...,clk_reg^{max}, $\forall$ k = 1...,r

$$\beta_{ij(k+1)} \geq \beta_{ijk} + \beta_{ij(k+2)} - 1 \tag{8}$$

meaning: rows in the same clock region in S_i must be contigious i.e., if β_{ij0} = 1 & β_{ij2} = 1 thenβ_{ij1} must also be equal to 1.

*2) **Resource constraints**:* These set of constraints ensure that each slot satisfies the resource requirements of the application. The total number of clbs in a slot S_i is expressed as

$$CLB(x_i, y_i, w_i, h_i) = \sum_{j=1}^{clk\_reg} \sum_{k=1}^{r} \beta_{ijk} \cdot clb_i \tag{9}$$

In the same way the amount of bram and dsp can also be expressed as

$$BRAM(x_i, y_i, w_i, h_i) = \sum_{j=1}^{clk\_reg} \sum_{k=1}^{r} \beta_{ijk} \cdot bram_i \tag{10}$$

$$DSP(x_i, y_i, w_i, h_i) = \sum_{j=1}^{clk\_reg} \sum_{k=1}^{r} \beta_{ijk} \cdot dsp_i \tag{11}$$

The resource constraint can simply be stated as the required number of clbs, brams and dsps must be greater than or equal to CLB(x_i,y_i,w_i,h_i), BRAM(x_i,y_i,w_i,h_i) and DSP(x_i,y_i,w_i,h_i) respectively. But the above functions are non linear and can not be used directly to formulate linear constraints.

linearization: In order to employ 9, 10 and 11 as linear constraint, they must first be linearized. To linearize these functions we define three auxilary real variables $\tau 1_{ijk}$, $\tau 2_{ijk}$ and $\tau 3_{ijk}$.

$\tau 1_{ijk} \in \mathbb{R} \mid \tau 1_{ijk} = \beta_{ijk} \cdot \text{clb}_i$

$\tau 2_{ijk} \in \mathbb{R} \mid \tau 2_{ijk} = \beta_{ijk} \cdot \text{bram}_i$

$\tau 3_{ijk} \in \mathbb{R} \mid \tau 3_{ijk} = \beta_{ijk} \cdot \text{dsp}_i$

Hence CLB(x_i,y_i,w_i,h_i), BRAM(x_i,y_i,w_i,h_i) and DSP(x_i,y_i,w_i,h_i) can be restated as

$$CLB(x_i, y_i, w_i, h_i) = \sum_{j=0}^{clk\_reg} \sum_{k=0}^{r} \tau 1_{ijk} \tag{12}$$

$$BRAM(x_i, y_i, w_i, h_i) = \sum_{j=0}^{clk\_reg} \sum_{k=0}^{r} \tau 2_{ijk} \tag{13}$$

$$DSP(x_i, y_i, w_i, h_i) = \sum_{j=0}^{clk\_reg} \sum_{k=0}^{r} \tau 3_{ijk} \tag{14}$$

Now the non linear expression is replaced by a linear one and to complete the linearlization a few constraints must be set. The following are constraints related to $\tau 1_{ijk}$ but similar constraints can be set for $\tau 2_{ijk}$ and $\tau 3_{ijk}$.

Constraint 5: $\forall$ i = 1...,N^{max}, $\forall$ j = 1...,clk_reg^{max}, $\forall$ k = 1...,r

$$\begin{aligned} \tau 1_{ijk} &\geq 0 \\ \tau 1_{ijk} &\leq BIG\_M \cdot \beta_{ijk} \\ \tau 1_{ijk} &\leq clb_i \\ \tau 1_{ijk} &\geq clb_i - (1 - \beta_{ijk}) \\ clb\_req_i &\geq \sum_{j=1}^{clk\_reg} \sum_{k=1}^{r} \tau 1_{ijk} \end{aligned} \tag{15}$$

*3) **Non-interference constraints**:*

Interference between two slots: Two slots S_i and S_k are said to be non interfering under the following conditions

```
**if** $x_i \leq x_k$ and $y_i \leq y_k$ **then**
    $x_i + w_i < x_k$ or $y_i + h_i < y_k$
**else if** $x_i \geq x_k$ and $y_i \geq y_k$ **then**
    $x_i + w_k < x_i$ or $y_k + h_k < y_i$
**else if** $x_1 < x_k$ and $y_i > y_k$ **then**
    $x_i + w_i < x_k$ or $y_k + h_k < y_i$
**else**
    $x_k + w_k < x_k$ or $y_i + h_i < y_k$
**end if**
```

This above condition can be encoded into a set of MILP constraints as follows

Constraint 6: $S_i \in$ N and $S_k \in$ N

$$\begin{aligned} \delta_{ik} &\geq \gamma_{ik} + \theta_{ik} + \Gamma_{ik} + \Omega_{ik} - 3 \\ \delta_{ik} &\geq (1 - \gamma_{ik}) + \theta_{ik} + \eta_{ik} + \Omega_{ik} - 3 \\ \delta_{ik} &\geq \gamma_{ik} + (1 - \theta_{ik}) + \Gamma_{ik} + \Psi_{ik} - 3 \\ \delta_{ik} &\geq (1 - \gamma_{ik}) + (1 - \theta_{ik}) + \eta_{ik} + \Psi_{ik} - 3 \\ \delta_{ik} &= 0 \end{aligned} \tag{16}$$

4) Interference with Forbidden slots: As stated before forbidden regions are also modeled as a normal slots hence the constraint for non intereference between a slot S_i and a forbidden region F_k can be set in the same way as done in the previous constraint formulation between two slots

Constraint 7: $\mathrm{S}_i \in \mathrm{N}$ and $\mathrm{F}_k \in \mathrm{F}$

$$\begin{aligned}
&\delta_{ik} \geq \mu_{ik} + \nu_{ik} + fbdn_1 + fbdn_3 - 3 \\
&\delta_{ik} \geq (1 - \mu_{ik}) + \nu_{ik} + fbdn_2 + fbdn_3 - 3 \\
&\delta_{ik} \geq \mu_{ik} + (1 - \nu_{ik}) + fbdn_1 + fbdn_4 - 3 \\
&\delta_{ik} \geq (1 - \mu_{ik}) + (1 - \nu_{ik}) + fbdn_2 + fbdn_4 - 3 \\
&\delta_{ik} = 0
\end{aligned} \tag{17}$$

VI. Experiemtnal Results

A. Experimental Setup

How was the experiment implmented i.e, with what kind of prog. langaguge, on what optimization tool, on what platforms... What is the system being tested for ? What is the compositon of the synthethic task suite used for testing. What type of FPGAs are modeled ? What are the challenges when switiching between models.

The system is tested for Exec time Vs Num of Reconfigurable regions and Exec time Vs %of resources used by the system on both zynq and virtex

The system is also tested for % of wasted resources (coeficients used to set allowed percentage of resources to be wasted %)VS Exec time on both virtex and zynq. The average wasted resources are also reported when not imposing these constraints and what effect it has on exec time and feasibility in general. my guess is that upto a certain point (upto a certain % of utilization of resources) not imposing these constraints improves the exec time but after the utilization increases (i.e., more applications require more resources) not imposing these coeffecients leads to infeasible constrants. This has to be tested with a carefully designed test suit which will test the limist of the platform.

Finally a case study on a real application on Zynq.