Congestion Prediction in FPGA Using Regression Based Learning Methods

Rahul Prakash
dept.of Electronics and Electrical Engineering
IIT GUWAHATI
Guwahati, India
p.rahul@iitg.ac.in

***Abstract*—Design closure in general VLSI physical design flows and FPGA physical design flows is an important and time-consuming problem. Routing itself can consume as much as 70total design time. Accurate congestion estimation during the early stages of the design flow can help alleviate last-minute routing-related surprises. This paper has described a methodology for a post-placement, machine learning-based routing congestion prediction model for FPGAs. Routing congestion is modeled as a regression problem. We have described the methods for generating training data, feature extractions, training, regression models, validation, and deployment approaches. We have tested our prediction model by using ISPD 2016 FPGA benchmarks. Our prediction method reports a very accurate localized congestion value in each channel around a configurable logic block (CLB). The localized congestion is predicted in both vertical and horizontal directions. We demonstrate the effectiveness of our model on completely unseen designs that are not initially part of the training data set. The generated results show significant improvement in terms of accuracy measured as mean absolute error.**

***Index Terms*—routing congestion; machine learning; EDA prediction**

I. Introduction

Machine Learning (ML) has found its application in various Electronic Design Automation (EDA) problems, and has been used to make early prediction on various quality related parameters. In EDA, ML finds its applications in yield prediction , timing and power optimization, auto-tuning of CAD tool parameters , and routing congestion estimation . Learning from prior designs and predicting performance and closure issues has immense value in the EDA community. A recent announcement from Xilinx about its ML-suite strongly affirms the idea that ML will play an important role in Electronic Design Automation. In FPGA physical design flow, routing is the most time-consuming task. It is prevalent for a router to either fail or take a long time to complete the routing, especially for very high logic utilization designs. Locally congested regions can result in failed routing or sub-optimal designs with problems in timing closure. Usually, a designer must replace the design in order to obtain better closure after a failed routing attempt. Routing Congestion is measured as the ratio of the number of used routing tracks to the number of available routing tracks. Routing congestion can be measured accurately only after the detailed routing. Traditionally, in ASIC design flows, congestion is estimated after the global routing using statistical models. Since Global Routing-based statistical and probabilistic prediction models do not work well on smaller technology nodes and FPGAs, alternate approaches are needed to predict the routing congestion before the router attempts a complete routing task.

II. Related Literature

A.

Technological advancements and pressures on time to close a design result in many approaches that predict design feasibility during early stages. Routing failures must be predicted before a design is routed in detail. In , Liu et al. discuss a Global Route (GR) based routing estimation tool where they take into consideration both local nets inside a cell as well as global nets connecting multiple cells. The authors in introduced a new routing congestion analysis tool called CGRIP that operates on a flexible model of global routing and models the congestion estimation using Integer Linear Programming.

III. FPGA Architecture Description

A heterogeneous FPGA device consists of various types of resources inside it, the most prominent of which are CLBs (Configuration Logic Blocks), BRAMs, DSPs, and IO blocks. Each CLB location is surrounded by vertical and horizontal wiring resources spread in the routing channels. Switchbox and the Connection Box allow the wires to connect between horizontal and vertical channels and to connect from the CLB to the wiring resources in the channels, respectively. We have modeled our routing congestion prediction model for the Xilinx Virtex Ultrascale+ architecture, which is based on TSMC 16nm process node.

A. Equations

Generally, more than 95LUTs, and Flip Flops present inside the CLB slices. CLBs are surrounded by a limited number of vertical and horizontal routing resources. Commonly, a CLB and its surrounding routing are also called a gcell. The Xilinx Vivado routing congestion reporting tool calculates and reports the vertical and horizontal congestion for each CLB block inside an FPGA. In order to correlate and compare the routing

congestion predicted by our model with the actual values reported by the Xilinx Vivado routing congestion tool after the detailed routing stage, we also measured the congestion values only for the CLB blocks. Finegrained prediction of congestion around each CLB is a precious tool for designers as it helps them improve the localized placement to locally adjust or remove over-congested regions. Xilinx Vivado reports horizontal and vertical congestion separately for each gcell. It calculates the congestion by taking the ratio of the number of routing tracks available to the number of tracks used. Mathematically, congestion is given by the following.

$$congestion = \frac{numberoftracksrequired}{numberoftracksavailable} \quad (1)$$

B. Prediction Framework Flow

- The proposed training and deployment model for our regression-based routing congestion estimation is illustrated in Figure 3. Two design flows are used for (i) creating the model for predicting the routing congestion and (ii) testing and predicting congestion using the trained model. It is significant to note that the testing flow uses never-before-seen designs and is, thus, utterly blind to any data properties that belong to the test design set.The training flow in takes a design netlist as an input. The design is placed and routed for a selected FPGA device using the Xilinx Vivado 2018.3 toolset. As we have stated later, in our experiments, we have used Xilinx Ultrascale Virtex (Xilinx XCVU095) devices for mapping the designs. This placement and routing step results in determining actual post-route horizontal and vertical congestion for each CLB site (gcell). These actual values of horizontal and vertical congestion around each used CLB are retained as a label for creating the model. we extract nine features for each of the utilized CLB sites for the mapped design mapped. The extracted features and corresponding labels are used for building regression-based models for the prediction of congestion. and we have used CometML machine learning platform for our experiments

[].

C. About Data features

In addition to these device specific features, we also explored features that we believed will have impact on routing congestion as a result of mapping of a design on a specific device. The following is the list of important features for each of the utilized CLBs/gcells described in Section 3. We have numbered the features as f1 to f9. Along with each feature's description, we also describe our intuition for selecting the feature.

- Number of Utilized Cells (f1): A Xilinx Ultrascale+ CLB slice consists of four different types of cells, viz., LUT, Flip-Flops, Multiplexers, and Carry Chains. One CLB has multiple units of each cell type. The number of cells utilized by a net inside each CLB tile plays an important role in measuring the congestion around each CLB block. begin
- Number of Pins (f2): Each cell in the CLB slice contains a certain number of pins. For example, a H6LUT contains eight pins (6 inputs and 2 outputs), a mux contains four pins, and a flip-flop contains five pins. When pins are connected to wires, they are an indicator of local congestion inside the CLB tile
  begin
- Number of incoming/outgoing nets (f3 and f4): If a net has pins spread across multiple CLB slices all over the FPGA, it can be considered as a multi-CLB net. Each multi-CLB net can be covered by a bounding box. The number of multi- CLB incoming and outgoing nets passing through a CLB slice provides us with the number of bounding boxes that crossed that particular CLB. The greater the number of bounding boxes passing through a CLB slice, the higher the congestion results. begin
- Number of Local/Buried nets (f5): Local nets or buried nets represent nets where all the cells are present in the same CLB slice. Even if all the cells are placed inside one CLB, the wires use the routing tracks present in the gcell to route the signals. Hence, this feature also contributes to routing congestion in the gcell.
  begin
- Location of the gcell inside the FPGA (f6 and f7): The absolute location of a gcell is not very relevant. When a group of cells is placed in a localized region, every cell in the group impacts congestion of other cells in the group. begin
- LUT (f8) and Flip Flop (f9) Utilization: Each CLB or gcell consists of 8 LUTs and 16 FFs. This feature represents the percentage of LUTs and FFs used inside a gcell. These features also contribute to congestion,
  All of the nine features were found to be important, and their importance in the decreasing order is listed as follows: f f5, f1, f6, f f , f2, f8, f9, f3, f4

IV. Methodology

Features f1 to f9 provide important data related to the characteristics of mapping of a design on an FPGA design. In order to create a model for predicting the local channel congestion, we need the data and corresponding labels. We have placed and routed the benchmark circuits by using Xilinx Vivado tools. the placement and routing step results in determining the actual post-route horizontal and vertical congestion for each CLB site (gcell). These actual values are used as training labels for the prediction models. We have trained our model by using four different regression models. The training data generated is divided into 70We also used four-fold cross validation by using all of the databins generated from four example designs used for training. Training is conducted locally on an Intel Core I7, 3.6 GHz octacore processor. We also trained and validated the model on Amazon AWS Sagemaker for cloud based training and deployment by

using T2 medium instance. We used Comet ML, an open source ML tool, for training and hyperparameter tuning. A brief theoretical description of the regression models used is described here.

A. Random Forest Regression:

Random forest is a supervised learning method that uses ensemble methods for learning. Ensemble learning is a technique that combines multiple weak learners to create a strong learner. In random forest, these weak learners are decision trees. The outputs from the individual learners are averaged in order to generate the final output. Random forest belongs to the bagging class of learners where, during training, data samples are selected at random without replacement. Bagging makes each model run independently and then aggregates the outputs at the end without preference to any model. Due to the introduction of the bagging method, the chances of overfit are lower in random forest. Random forest is a very fast method of training because each tree can be trained in parallel, and the inputs and outputs of each individual tree are not related to another. To summarize, the Random Forest Algorithm merges the output of multiple decision trees to generate the final output.

B. Linear Regression:

Linear regression is a method to find relationship between two continuous variables, where one is independent and the other is dependent. Linear regression tries to create a statistical relationship which may not be always deterministic. Linear regression tries to obtain a straight line which shows the relationship between the independent and the dependent variable. For most of the linear regression models, the error is measured by using least squared error. A linear regression line has an equation of the form Y = a + bX, where X is the independent variable and Y is the dependent variable. The slope of the line is b, and a is the intercept, which are both determined during training of the model.

CONCLUSION AND RESULT

R2 is defined below. N is the number of samples used in the regression, y is the actual value of sample i, ˆ y' is the predicted value using a regression model based on observations, and ¯ y" is the mean of the actual values. We measured the performance of our models by using R2 values. The R2 is always ranged between 0 and 1 i build model based on randomforest regression and model works nicely and i tested 4 different data and measure R2 value and i got.

$$R2 = 1 - \frac{\sum(y - y')2}{\sum(y - y")2}$$

We measured the performance of our models by using R2 values. The R2 is always ranged between 0 and 1, 0 being the worst fit and 1 being the best. Practically, a R2 value of 1 is impossible if we try to predict on completely unseen data. However, a R2 value of 1 is possible during training, but it may indicate a highly overfitted model.

- 1st 0.26.
- 2nd 0.51.
- 3rd 0.62.
- 4th 0.52