Abstract—FPGAs take advantage of 2.5D stacking technology to manufacture large capacity and high performance heterogenous devices at reasonable costs. EDA tools need to be aware of and exploit physical characteristics of such devices, for example the reduced connection count between SLRs, the infrequency of SLL channel occurence in the fabric, and the aspect ratios of individual SLRs. We implement a partition driven placer to explore various EDA options to take advantage of architectural features in 2.5D FPGAs. We improve the routability of designs by optimizing the placer for discrete SLL channels and reduced connection counts. We propose a cut schedule for the partitioner to orient the placement with awareness of the aspect ratio of SLRs to improve track demands within each SLR.

I. INTRODUCTION

2.5D stacking [1] enables FPGAs to meet the twin demands of higher logic capacity and heterogeneity. 2.5D stacking also permits lower latency communication between dies than competing technologies [2]. Devices with logic capacities that are impossible to build on a single die are made feasible by assembling multiple, better yielding, smaller dice on a passive interposer [3]. Market demands for heterogeneity and specialized functionality can be met by integrating application specific dies with FPGAs on a single package [4].

In this paper, we address the EDA challenges specific to implementing multi-die FPGA systems. Betz et al. [5] investigate the placement and routing challenges in multi-die FPGAs by enhancing the open-source academic VPR CAD tool to model and optimize for 2.5D FPGAs. In this paper, we consider current customer designs and synthetic designs to outline key EDA challenges beyond those studied in [5] and propose techniques to address them. We limit our study to manufacturable 2.5D FPGAs as constrained by current technology and economic factors.

A. Terminology

We refer to a single monolithic FPGA die as a Super Logic Region or SLR. A "2.5D" or multi-SLR device is assembled on a passive silicon interposer and connections are made through micro-bumps, or uBumps. The inter-SLR connections, called Super Long Lines (SLLs), are made on the silicon interposer.

In this paper, we refer to SLL capacity as a percentage of the capacity of tracks that exist within the SLR. For example, 25% SLL capacity means that the number of SLLs that cross the SLR boundary is 25% of the wires that exist if the cut was observed in an arbitrary region within the SLR.

B. 2.5D Stacking in FPGAs

Stacking technology is especially interesting for FPGAs due to the regularity in logic cells and interconnect, allowing identical arrays to be connected on an interposer with fine pitch wiring. The interposer consists of metal layers that enable wire traces that connect the individual FPGA SLRs.

We illustrate the physical limitations on 2.5D FPGAs by working through an example based on the 4-SLR 28nm test vehicle described in [6]. The number of uBumps available on each SLR limits SLL counts. Assuming a uBump pitch of 45um and an FPGA die size of 7mm × 12mm, we compute the maximum number of uBumps to be about 155 × 267 ≈ 41K uBumps per SLR. Assuming 30% of uBumps are unusable due to power and global signal considerations, and using half the uBump rows to communicate with adjacent SLRs, we can use (155 × 0.7) ÷ 2 ≈ 54 uBump rows. Assuming we meet the latency requirements and provision for sufficient number of metal layers on the interposer, we can achieve 54 × 267 ≈ 14.4K inter-SLR connections. Compared to a virtual monolithic device with identical logic capacity, this is about 25% of the vertical wires that would exist in the same region. Interfaces to the interposer (referred to as "SLL Channels") on the FPGA fabric need to appear at discrete intervals since the uBump pitch is coarser than fabric routing channels. The primary challenges in supporting placement and routing on multi-SLR devices arise from these two characteristics of SLLs - their relative infrequency and reduced count compared to traditional interconnect resources.
II. Evaluation Platform

A. Device Model and Designs

We generate device models for 2.5D FPGAs with similar feature mixes and logic counts as commercial FPGAs [7] and implement them in the Vivado® Design suite [8] modified to handle experimental architectures. We develop a global partition driven placer, described in section IV, combined with a packer and simple move based optimizer. We use the Vivado® router to implement designs.

Implementation tools trade-off the number of wires cut between each SLR and how balanced the utilization of each SLR is. If the utilization of each SLR is balanced, the probability of routing failure within each SLR is reduced while the number of inter-SLR cuts is increased. We use synthetic designs to understand this tradeoff because they allow us to incrementally control the design size, topology and complexity as described in [9]. Synthetic designs offer several benefits. They allow us to:

1) Analyze the incremental impact of utilization and design complexity.
2) Create designs with expected logic capacity and complexities that may not exist in current customer designs.
3) Identify the "breaking point" (i.e. point where designs become impossible to implement) of architectural decisions.

B. Estimated Channel Demand

To analyze routing demand, we compute the estimated channel demand (ECD) based on placement of designs. For each net, we look at the connectivity between various blocks and implement a stochastic model [10] to compute the probability of using horizontal or vertical tracks on a two dimensional grid. This metric lets us understand the placement quality of a design in terms of routing congestion independent of routing architecture.

The ECD computation is enhanced to be multi-SLR aware by identifying SLL channel locations in each SLR and splitting multi-SLR nets into subnets. We recursively partition the net for each SLR crossing and compute separate ECDs for each subnet.

III. EDA Issues for 2.5D FPGAs

A. Feasibility of Multi-SLR Devices

In this section we attempt to understand the feasibility and effects of implementing current customer designs on multi-SLR devices. These designs utilize 20% to 95% of all the various tile types available on modern FPGAs, including CLBs, RAM Blocks, and DSPs. The number of nets in these designs range from 500K to 3.6 million. We recursively bisect each design into 4 partitions, where each partition must fit in one SLR of a 4-SLR target device and compute the number of SLLs demanded across the SLR boundaries. The cut demand ranges from 0.5% to 10%, while physical limitations allow SLL capacity of upto 25% (refer section I-B). This is the primary result which motivates the possibility of multi-SLR devices. In our experience, current customer designs can always be partitioned such that the SLL demand is less than the SLL tracks that we can physically supply.

While the SLL demand is well below the supply, we still observe degradation in various design implementation metrics as the SLL supply is reduced. To illustrate this, we create 6 variants of a 4 SLR device, each having different SLL counts, and implement the partitioned customer designs on each variant. In fig 1, 100% SLL Capacity refers to a device where there is no reduction of tracks between the SLR boundaries and hence, the 4-SLR device can be treated as a large monolithic device. We arbitrarily trim the interconnect resources that cross SLR boundaries and create devices with SLL capacities ranging from 75% to 5%.

In fig 1, we show that all device variants with 12.5% SLL capacity or more can succesfully route the designs. As expected, with 5% supply, majority of designs fail due to oversubscription of SLLs. There is a 3-4% increase of routed wirelength, and a 2% impact to critical path delay as the SLL supply is reduced to 12.5% compared to the monolithic variant.

B. Inter-SLR Connections and SLL Channels

We now illustrate the impact of inter-SLR cuts on placement quality and routability by experimenting with synthetic designs with controlled SLL demand. We implement the benchmarks on a 2-SLR FPGA with 25% SLL capacity. We create a device model with realistic uBump pitches, resulting in relatively infrequent SLL channels on the FPGA fabric.

We generate the synthetic designs in the following manner:

1) Create 180 designs of varying utilization and routing complexity that are placeable in one SLR. These designs are neither too easy nor completely impossible to implement.
2) Create a duplicate instance of each design.
3) Connect the two design instances at the top level
4) Constrain each design instance to a single SLR

We control the SLL demand between SLRs by varying the top level IO ports on each design instance in step (3). Since we constrain the instances to SLRs in step (4), we are guaranteed to have SLL demand that is equal to the number of connections between the two design instances.

In fig 2a, we show the average ECD increase across the benchmark suite at various SLL counts. The plot shows that as the SLL demand increases, both horizontal and vertical ECD grows indicating an increase in routing congestion, resource usage and routability degradation within each SLR. This is because more tracks are consumed for routing to and from the SLL channels. In fig 2b, we illustrate the ECD heat map of a 80% resource utilized design with 80% SLL demand where we see that most of the nets demanding SLLs are concentrated in the middle. The bar chart shows that there are several SLL channels that are oversubscribed by more than 3x of the available SLLs in the channel. This is because the placer is unaware of the capacity and location of the SLL channels. In section IV-A, we explore strategies to make the placer SLL channel aware to improve SLL access and routability.

C. SLR Aspect Ratios

To minimize development costs, commercial FPGA vendors normally design a single routing architecture for an entire family of devices. Modern FPGA families offer both monolithic and 2.5D FPGAs on the same package technology [7]. To maintain reliability of packages, they have to be of reasonable size and aspect ratios [2]. Hence, the aspect ratios of monolithic dies are relatively square as shown in fig 3a. In 2.5D stacking, as we add more SLRs to a device to increase logic capacity or heterogeneity, we increase size of the package in a single dimension (eg. height), while the other dimension (eg. width) remains constant. To enable flexibility in SLR integration, we naturally migrate towards SLR aspect ratios that are biased in one dimension. This can result in SLRs with noticeably different aspect ratios than monolithic dies.

In fig 3b, we illustrate the variance in aspect ratios between SLRs in a 4-SLR device compared to a monolithic die. As the ratio of \( \frac{w_{SLR}}{h_{SLR}} \) increases, the design placed in each SLR is forced in the horizontal orientation resulting in an increased demand for horizontal tracks. In section IV-B, we discuss strategies to optimize for different SLR aspect ratios by orienting the placement with awareness of total tracks available in each dimension.

IV. Partition-Driven Placer

We implemented a partition driven placer similar to one in [11]. We start by bi-partitioning the entire device into bins that have some capacity for placeable instances (eg. CLBs, DSPs, etc). We take the design and partition it into two such that the number of placeable instances in each partition is less than the bin’s capacity and the number of connections between the two partitions is minimized. Step (1) guarantees that the SLL demand between partitions are minimized. At this stage we can predict the number of SLLs that will be demanded. If this exceeds the capacity of the device, we terminate early.

We then continue by partitioning each bin and the design placed in that bin recursively using the partitioner described in [12] for this task. This technique of recursive partitioning is an effective way of reducing wirelength and thus reducing the expected routing resource utilization. The flow for multi-SLR placement is as follows:

1) For an \( n \)-SLR device, partition the design into \( n \) partitions and constrain each partition to an SLR.
2) Invoke the partition driven placer on each SLR
3) Perform local optimization and legalize the placement generated by the partitioner

Step (1) guarantees that the SLL demand between partitions are minimized. At this stage we can predict the number of SLLs that will be demanded. If this exceeds the capacity of the device, we terminate early.
A. SLL Channel Awareness

The placer as described in the previous section is unaware of the location and capacity of each SLL channel. As a result, even though the total cut demanded is less than the total SLL capacity of the device, it is possible that in a small region, more SLLs are demanded than are available. For example, fig 2b shows the high demand for SLLs near the middle of the device. Hence, it is important for the placer to be aware of the location and capacity of SLL channels to enable reasonable placements of logic with inter-SLR connections. In our flow, after the first partitioning step, we identify connections that cross the SLR boundary and modify them to go through placeable inter-SLR instances with fixed displacements corresponding to the length of SLL wires. The netlist is modified such that each inter-SLR net goes through an SLL instance and the SLL channel region in the device is added to the available capacity where these instances can be placed. Similar to the capacity of other placeable instances, the partitioner now considers the SLL instance demand and ensures that no partition bin overutilizes the available SLLs. Therefore, we guarantee that no SLL channel gets oversubscribed.

It is possible to constrain the router to use the SLL instances assigned by the placer. In our evaluation platform however, we allow the router to choose the SLL instances based on the larger routing picture. To enable this, once the placer completes, we undo our netlist modifications and restore the inter-SLR nets that were previously split.

Figure 4a shows the ECD heat map and SLL subscription of the same design shown in fig 2b placed with SLL awareness. The ECD of inter-SLR nets shows that the demand is more uniformly distributed along the width of the SLR boundary. Also, in the bar chart we see that no SLL channel is oversubscribed. The infrequent occurrence of SLL channels causes the placement to be spread along the width of each SLR. Since we guarantee no contention for each SLL resource, routing congestion due to inter-SLR nets is reduced. In fig 4b, we plot router congestion for both SLL unaware and SLL aware placements as it tries to resolve oversubscription of routing resources. Firstly, it is apparent that SLL aware placement poses much less challenge to the router than the SLL unaware placement. In this example, the SLL aware placement eventually resolves all contention, while the SLL unaware placement fails to route. Secondly, we see that the bulk of congestion in both placements is caused due to routing to/from SLL channels. We can further reduce this by decreasing the placer’s view of SLL capacity, effectively overconstrainting it as it legalizes the partition bins.

In fig 5, we report on the full suite of benchmarks at various SLL demands. We show the normalized number of successfully routed designs in each suite. Not surprisingly, we see a decline in routing success as we increase the SLL demand in both SLL unaware and SLL aware flows. However, we are able to route more designs with SLL aware placements. As SLL demand increases, the need for SLL channel awareness and legalization becomes more important. Therefore, we see that as SLL demand grows, we see more benefit of doing SLL aware placements. In the 96% SLL demand suite, we are able to route almost 2x the number of designs compared to the SLL unaware flow.

We also see an increase in wirelength as SLL demand increases for both flows. The degradation is higher for SLL aware placements because SLL legalization causes the placer to spread the design resulting in an overall increase in wirelength to route the average net.

B. Cut Schedule

We explore the impact of the variation in SLR aspect ratio in 2.5D FPGAs caused due to the reasons described in section III-C. We create 7 device models with identical resource capacities and varying aspect ratios. These include devices which are “tall and narrow” (h ≫ w) on
one end of the spectrum and "short and wide" ($h \ll w$) on the other. We take a 75% utilized synthetic design with reasonable routing complexity and place it on each device with the partition driven placer. In fig 6a, we show the horizontal and vertical max ECD for each design placement. We vary the $h/w$ ratios from 1/8 to 8/1 and the ECD is normalized to ECD of placement on a device with 1/1 aspect ratio. As the device becomes "shorter and wider", horizontal ECD increases indicating the increase in horizontal routing tracks required to route the design. As the device becomes "taller and narrower", horizontal demand reduces, while the vertical ECD increases.

We consider aspect ratio in the partition driven placer by modifying the orientation of cuts at each partition step. We refer to the sequence of partitioning cut orientations as the "cut schedule" of the partitioner. At each iteration, the orientation of the cut is determined based on algorithm 1

**Algorithm 1:** Aspect Ratio aware Cut schedule

**Input:** partition area ($P$), tracks per channel ($T$)

**Output:** Partition Orientation OR Stop Partition

1. if $P < \text{min} \_ \text{area}$ then
2. return Stop Partition
3. end if
4. if ($P.width \times T.vertical) < (P.height \times T.horizontal)$ then
5. return Horizontal Cut
6. else
7. return Vertical Cut
8. end if

This algorithm chooses cut orientation such that we minimize for vertical (horizontal) track demand if the number of vertical (horizontal) tracks over the width (height) of the partition bin is less than the number of horizontal (vertical) tracks over the height (width) of the partition bin. The partitioner as a result, minimizes demand in the direction that is most resource scarce. For example, for the 1/8 AR device, the first four cuts made are vertical, which creates partition bins that are 1/1. The schedule then alternates between horizontal and vertical cuts. The algorithm is therefore able to naturally respond to device aspect ratio. It also handles architectures where we supply different number of vertical and horizontal tracks per routing channel.

Fig 6b shows the resulting ECD for the same design implemented with the aspect ratio aware cut schedule. As the device becomes "shorter and wider", the partitioner makes more vertical cuts to manage the horizontal track demand. At 1/8 AR, we are able to reduce horizontal ECD by 71% while increasing vertical ECD by 7%. On the other end of the spectrum, we reduce the vertical ECD by 20% while increasing horizontal ECD by 7%.

**V. Conclusion**

2.5D stacking is a promising technology for FPGAs, and the architectural decisions made to implement such devices give rise to several challenges and opportunities in EDA. The placer must specifically consider SLL capacity, SLL channel locations on fabric, as well as the SLR aspect ratios in order to improve wirelength and routability. In this paper, we implemented and enhanced a partition driven placer for multi-SLR FPGAs and showed that we can improve results by considering the specific architectural features of such devices.

**References**


