Leveraging Flexible Interconnect in Automated 2D and 2.5D System Design

By

Daniel P. Seemuth

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Electrical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2016

Date of final oral examination: 16 Feb. 2016

The dissertation is approved by the following members of the Final Oral Committee:

Katherine L. Morrow, Associate Professor, Electrical and Computer Engineering

Azadeh Davoodi, Associate Professor, Electrical and Computer Engineering

Jing Li, Assistant Professor, Electrical and Computer Engineering

Xinyu Zhang, Assistant Professor, Electrical and Computer Engineering

Jeffrey Linderoth, Professor, Industrial and Systems Engineering

Acknowledgments

My PhD work here at UW-Madison would not have been possible without the support of many people in my life. I especially thank Leanne, my wife, for all her love and support over the past years. She worked hard to provide for our family while I plodded through my PhD, and she is such a blessing and encouragement to me. Thanks also to Isaiah, our son, for his patience at home with Mom while Dad was working at school. I thank my parents, Dave and Karen, for their countless hours offering advice and wisdom, for helping take care of Isaiah, and for all their encouragement along the way.

I also thank Prof. Katherine Morrow and Prof. Azadeh Davoodi for directing, advising, editing, editing, editing, and for shaping my growth as an academic. I'm thankful to Michael Morrow and Joseph Krachey for helping me grow as an engineer. I enjoyed my time with my fellow students, including Ahmed Abulila, Paula Aguilera, Hadi Asgharimoghaddam, Daniel Chang, Amin Farmahini Farahani, Tony Gregerson, Zhenhong Liu, Felix Loh, David Palframan, and Hao Wang. I also enjoyed teaching, advising, and generally heckling the countless students in the many courses in which I served as a teaching assistant.

I thank God for His faithfulness to provide for all that we have needed: jobs, homes, friends, family, and joy.

Finally, thank you to everyone who has endured^W enjoyed my puns and sense of humor.

Abstract

Today's embedded systems and integrated circuits are too complex for any one person to be able to keep track of the entire design from a high conceptual level all the way down to the device level. Designing such systems and components can be a complex process that is often tedious, error-prone, and time-intensive. Because of these difficulties in system design, we present methods to perform aspects of component-level design automatically. As part of this work, we propose a new framework that allows designers to express connectivity requirements at a higher level of abstraction than explicitly placing components and enumerating or drawing each connection between those components. Our proposed framework accommodates design capture and layout generation for designs ranging from large printed-circuit boards (PCBs) down to 2.5D integrated circuits (ICs). We propose a simultaneous component placement, pin assignment, and I/O bank voltage supply assignment algorithm of particular utility for designs that include components with flexible I/O pins (e.g., Field-Programmable Gate Arrays (FPGAs) on a PCB or FPGA dies within a larger 2.5D IC). Exploiting flexible pins increases the size of the solution space, while the automated framework quickly explores the new design options to facilitate the design process. We then expand this framework to consider routability when placing components and assigning pins, and to perform 3D global routing and pin reassignment to minimize the number of metal layers required to route all connections. Thus our framework not only reduces design time and effort, but also directly reduces manufacturing costs of 2.5D ICs through decreased area and interposer metal layer count.

Table of Contents

1 Introduction and Motivation

Today's complex electronic designs are hierarchical; a large design is subdivided into multiple off-the shelf and/or custom components. Current computer-aided design (CAD) tools assist in designing individual components, but are less effective in determining how to physically arrange and connect those components within a larger design. The decision space of how to place and connect the components increases enormously when one or more of the components has *flexible interconnect*; in other words, when there are multiple possible pins on the component that could be used to make a required connection.

A very simple example of flexible interconnect is a set of digital output pins on a microprocessor. An indicator light-emitting diode (LED) may be connected to any of these pins, and the software that controls the light can be written accordingly. The Field-Programmable Gate Array (FPGA) is another common type of devices with flexible interconnect, but which takes the flexibility to an extreme level. FPGAs feature configurable internal logic and multiple banks of interchangeable input/output (I/O) pins; in fact nearly all of the FPGA pins are flexible in terms of how they can be used. A circuit implemented within an FPGA's internal logic can be placed and routed based on the chosen pin assignment, so as to make the required connections between the internal logic and the external system in which the FPGA is located. This means that there can be very many valid pin assignments for connecting the FPGA to other system components. During the system design process, the multiple valid solutions must be narrowed down to one particular solution that will actually be used to make the necessary connection.

We refer to the process of creating all required connections (including topology synthesis, routing, and wire sizing) as *interconnect synthesis*. The wide range of pin choices in components with flexible interconnect, such as FPGAs, provides a great deal of design freedom and can allow for improved solutions. However, it also expands the search space for finding a good solution, which complicates the design process.

To illustrate the scale of the decisions that must be made in systems with flexible interconnect, consider a simple design problem where two LEDs must be connected to a medium-to-high-end FPGA (Figure 1). This example FPGA is modeled on a Xilinx Virtex-7 485T in a 1157-pin BGA package [1], with 602 digital I/O pins (shown in green) that can be used to connect to a pair of LEDs. Any of these 602 pins can be connected to the LEDs; this means that there are $602 \times 601 = 361{,}802$ valid pin choices for this very small example system. A real system with far more connections and potentially multiple components that feature flexible interconnect

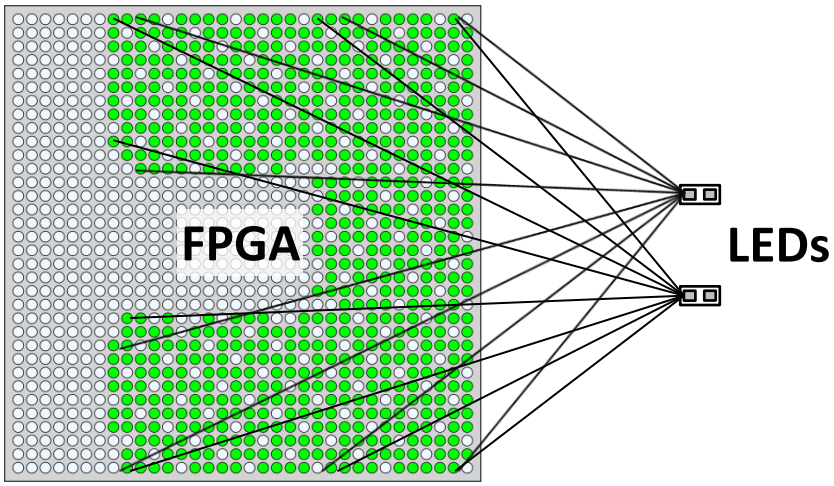


Figure 1: FPGA with many possible pin connections to two LEDs. Each black line represents one of the many possibilities for connecting each LED to the FPGA.

has an even far greater solution space to be searched.

The process of designing at the component level, whether on a printed circuit board (PCB) or for a 2.5D integrated circuit (IC), requires more than choosing a pin assignment; it also requires placing the components relative to one another. Current CAD tools assist in *separately* arranging components and then connecting them; however, finding a high-quality placement and choosing a high-quality pin assignment are actually **interdependent**.

Consider a trivial example of two rectangular components “A” and “B” that require a single connection between them. In Figure 2a, we show three possible placements for the case where the pin assignments were made prior to placement (the pins chosen for the required connection are highlighted in yellow). This figure demonstrates that the best placement depends on the chosen pin assignment. In Figure 2b we show the same placements as before, but this time allow any of the pins to be chosen for the connection between the two components. For each placement, a different pin assignment provides the shortest path of communication; in other words, the best pin assignment depends on the chosen placement.

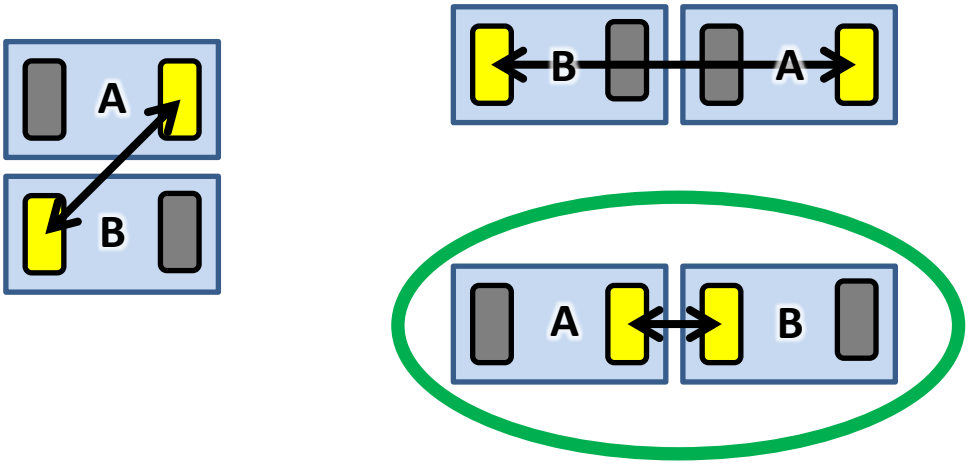


Figure 2a: Pin assignments (chosen pins shown in yellow) affect components' optimal positions. The circled placement provides the shortest connection given the chosen pin assignment.

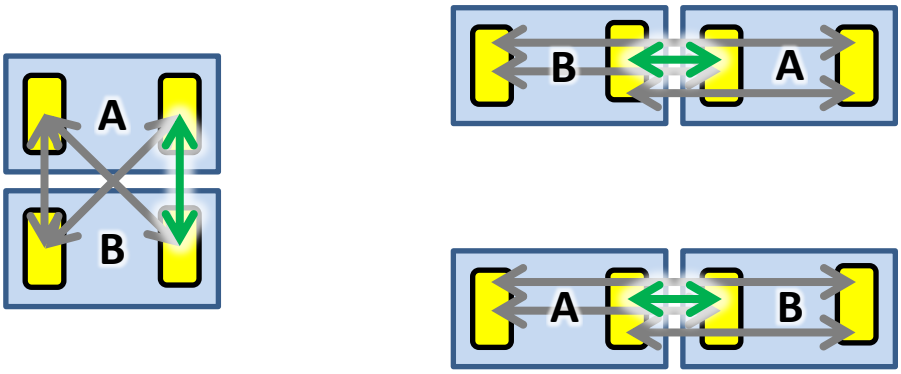


Figure 2b: Components' positions affect optimal pin assignments. For each placement, one pin assignment provides the shortest connection (green) out of the possibilities (grey).

Existing algorithms can place components given *fixed* connections between their pins, or assign signals to pins given *fixed* component positions. Flexible interconnect, however, demands new methods and algorithms that consider placement and pin assignment together to improve the

overall design solution. Interconnect synthesis with flexible interconnect presents several challenges:

- Design requirements must be provide enough freedom to properly leverage any applicable flexible interconnect.
- Due to the larger solution space, multiple valid solutions must be narrowed down to a particular solution that will be used in the final design (or in a particular design candidate during design exploration).
- Different interconnect solutions may dramatically affect later design decisions, so the chosen solution must be selected in an intelligent fashion. In order to guide this design process effectively, the design requirements must be precise enough to avoid poor or invalid solutions.

The work presented in this dissertation aims to enable designers to take advantage of flexible interconnect in PCB and IC designs. Using our methods, designers can specify connection requirements in terms of the requirements of the connections instead of specific pin assignments that not only would require more designer effort but also restrict later design steps. We provide an overview of these methods in section 1.1. The goal is to provide robust methods for electronic design automation (EDA) that exploit flexible interconnect to improve solution quality (e.g., to reduce manufacturing costs, total wirelength, and layout area).

1.1 Overview of Work

This section presents our overall focus in this dissertation and introduces our open-source electronic design automation framework that is discussed throughout this dissertation. In order to guide design decisions to leverage flexible interconnect, we explore the problems of component-level placement and interconnect synthesis for systems involving flexible interconnect. For example, Figure 3 shows component dies within an integrated circuit. In this example, each peripheral die will be connected to flexible I/O pins on the FPGA die. At the component placement level, an interesting problem is evaluating solution quality without necessarily knowing the exact details of the solution's interconnect (e.g., which pins on the FPGA die will be connected to the peripheral dies). The converse problem is evaluating a solution's interconnect quality without knowing the components' exact positions. We examine

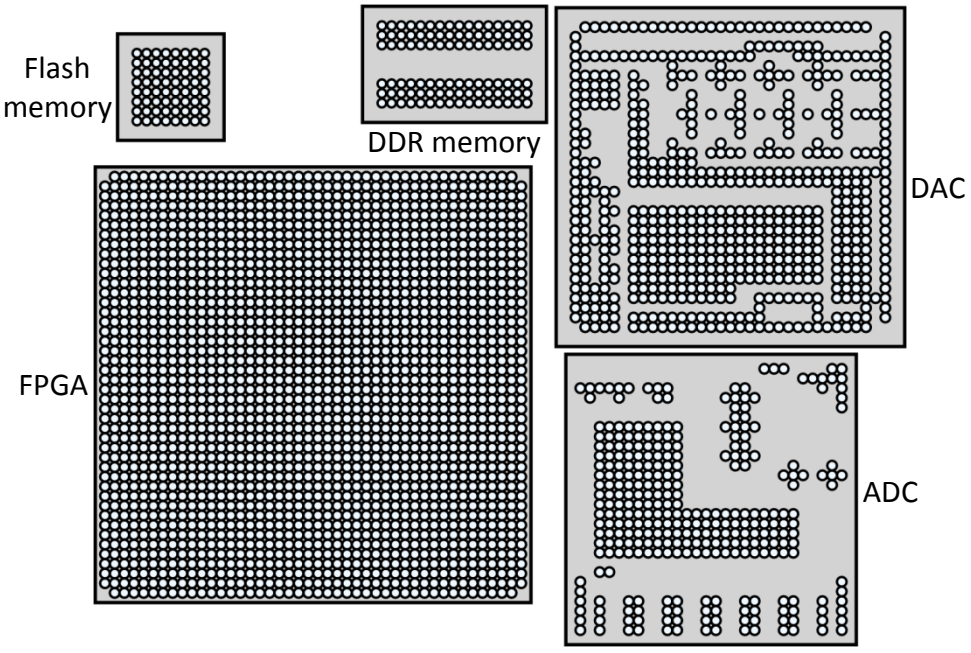


Figure 3: Example components in an integrated circuit (IC)

component-level placement and interconnect synthesis for both printed-circuit board designs and IC designs that include some components with flexible connectivity.

The process of designing such a system, whether it be on a printed circuit board or within an integrated circuit, can be complex, requiring many details to be specified. Some of the types of information that a designer must keep track of and decisions to be made include:

- Specific components that will be included in a design
- Power supply and supporting circuitry for each major component
- Connections that need to be made between the components
- Which pins on each component can be used to make the required connections
- Physical constraints on component placement
- Special considerations for sensitive components (e.g., low noise, high power, high frequency signals)

The large degree of required detail and manual decision-making can make the design process error-prone. Furthermore, design requirements are often interdependent, which can lead to an iterative design process that requires many revisions. Computer-aided design (CAD) tools (e.g., Cadence [2], [3]; Mentor [4]) assist somewhat in this design process, but they lack sufficient support for automated placement and taking advantage of flexible interconnect. We present an open-source electronic design automation (EDA) framework to facilitate easier system design that supports flexible connectivity. We enable users to more effectively manage a variety of design details, from capturing requirements, placing components, assigning pins to

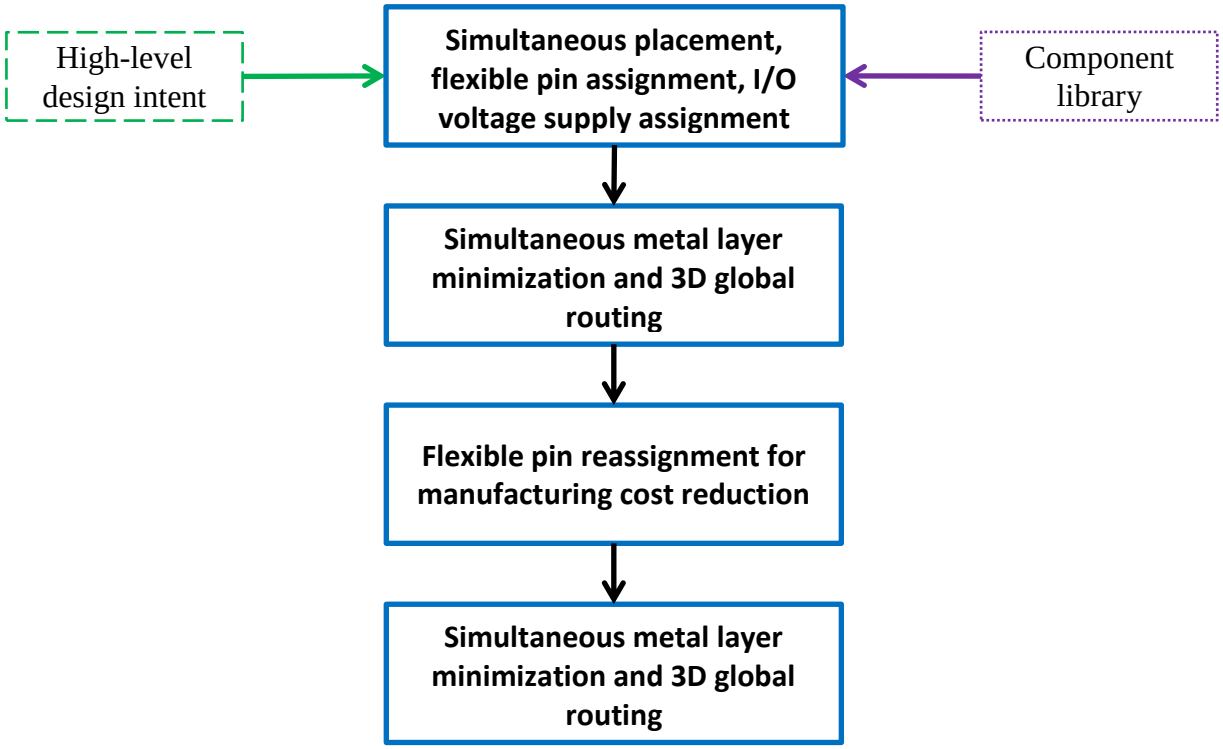


Figure 4: Overview of our electronic design automation framework

connections, routing each inter-component wire, and reassigning pins as needed for design solution improvements. Our framework, as shown in Figure 4, takes as input design intent at a higher level of abstraction than a schematic or pin-by-pin list of connections, and produces as output a variety of candidate layouts to meet the specified requirements.

This dissertation presents our research contributions for leveraging flexible interconnect in printed-circuit board design. Previous work in the literature, examined in chapter 2, lacked sufficient solutions to 2D and 2.5D design automation; as we will show, we extend the area significantly. In our initial work we perform simultaneous placement and bank assignment, whereas in later work we refine that process to perform actual pin assignment at the same time

as placement. Finally, we also present methods for manufacturing cost reduction through routability-driven placement and pin assignment, and through metal layer reduction and 3D global routing.

1.2 Objectives and Contributions

Our initial goal for this work was to improve the PCB design process by reducing the amount of low-level decisions a designer must make, reducing the effort and tedium of this process. These low-level decisions can have large-scale implications on the rest of the design, and such implications may be difficult for a designer to anticipate and visualize. The work grew into a framework capable of automatically performing a variety of PCB and 2.5D IC design tasks. Our work makes the following contributions:

- A novel method of capturing and representing PCB or 2.5D IC design intent
- A new adaptation of simulated annealing that assigns I/O voltage supplies and places components (or dies) to minimize estimated total trace length, while still supporting flexible pin assignment
- A novel method to estimate trace lengths for connections to devices with flexible pins prior to actual pin assignment
- An integer linear programming formulation that simultaneously generates pin assignments for all needed single-ended and differential connections

- A procedure to simultaneously place dies, assign pins to required (multi-terminal) connections, and assign compatible bank power supplies to each I/O bank within dies with flexible I/Os
- An integer linear programming formulation that simultaneously performs metal layer planning and 3D global routing for 2.5D interposer design
- An alternate integer linear programming formulation for metal layer planning and 3D global routing for 2.5D interposer design
- A pin reassignment algorithm for 2.5D ICs having flexible interconnect, to reduce interposer manufacturing costs
- A database of many components used in system design, which allows designers to quickly use these components in their PCB or 2.5D IC designs

Our framework can facilitate easier PCB and IC design by enabling rapid prototyping and design space exploration. This can be useful when the designer needs to gain a better sense of what could produce a better layout and overall design. By quickly trying out different possibilities, the designer can gain an intuitive understanding of what characteristics work well for the specific design. The best of the evaluated designs can then be chosen for production. This work includes automatic methods for component placement, flexible pin assignment, I/O voltage supply assignments, metal layer planning and 3D global routing, and flexible pin reassignment.

1.3 Document Organization

The remainder of this document is arranged as follows: In chapter 2, we introduce background information on printed-circuit board design, 2.5D IC design, and existing methods for solving some of the challenges in electronic design automation. In chapter 3, we present our work for facilitating automatic 2D printed-circuit board design. In chapter 4, we extend our 2D work to consider more design aspects within 2.5D IC design automation. In chapter 5, we modify our 2.5D IC design automation methods to also reduce manufacturing costs by minimizing the number of metal routing layers within a 2.5D IC. Last, we summarize the contributions of the presented work and discuss potential future directions.

2 Background and Related Work

Before we discuss our own contributions in detail, we first provide background information and related work on 2D and 2.5D circuit design and the supporting computer-aided design (CAD) algorithms. In section 2.1 we discuss the design process for 2D and 2.5D circuits and background information and related work for aiding this process. In section 2.2 we focus on the CAD algorithms used to offload some design tasks from the user. Together, these sections provide a foundation for our work presented in this dissertation.

2.1 Design Process for 2D and 2.5D Circuits

The general design process for these circuits is:

1. Choose components (whether packaged devices or bare dies) to be included in the design.
2. Define connections ("nets") between all components. This typically includes pin assignment – choosing specific electrical connection points on each component to use for making each net.
3. Place components – assign the relative positions and rotations for all components.
4. Route nets – determine the specific metal path that forms each electrical connection.
5. Simulate the design to evaluate key characteristics (e.g., speed, noise, power supply stability).
6. Iterate the above steps as necessary to produce a successful design.

This dissertation presents methods for steps 2–4: defining connections, placing components, and routing nets.

2.1.1 Printed Circuit Board Design

A number of commercial tools assist somewhat with the printed-circuit board design process. Cadence Allegro FPGA System Planner performs automatic pin assignment based on user-specified connectivity, but requires manual component placement [3]. Other PCB CAD tools require designers to specify connectivity on a net-by-net basis, though some subset of design steps may be automated [2], [5]–[7]. Mentor DxDesigner automatically assigns and displays net names for named buses and unnamed wires, but the latter are obscure indexed names that do not reflect the purpose or connections of the net [4]. Cadence Allegro's automatic power pin net assignment feature frees designers from specifying power, but not signal, connections on a pin-by-pin basis [2]. Although each of these tools is helpful, they still require the designer to specify most low-level details. On PCBs with tens of thousands of traces, this quickly becomes unwieldy. Commercial tools help designers to capture net-by-net connectivity between components and to implement this connectivity in a physical layout, but very few allow for higher-level design intent capture. This limits the tools' utility for system design specification and design space exploration. In contrast, our framework enables a designer to quickly explore, evaluate, and realize PCB designs.

PHDL is an academic project intended to address the issue of designer productivity using higher-level expression [8]. In particular, it seeks to give designers the ability to specify PCB

connectivity without needing to enumerate or draw hundreds or thousands of connections, but rather using a structural Hardware Description Language (HDL). Unfortunately, PHDL was designed to replace only typical, graphical schematic capture processes, so it does not include physical layout algorithms [8], it does not use a higher level of abstraction to define connectivity, and it does not include automatic component placement and I/O pin selection. In PHDL, although designers are no longer tied to legacy graphical schematic capture, they still must define each pin's connections. Thus PHDL is limited in its ability to streamline high-level design capture.

Other academic works have provided partial solutions to some of the concerns mentioned above. Xiong, et al., describe a multi-stage algorithm for placing I/O bump pads, I/O sites, and I/O cells to facilitate chip-package codesign [9]. This algorithm takes into account I/O voltage supply domains in I/O cell placement but is limited to I/O placement on a single IC. Ozdal and Wong provide a formal algorithm for trace escape routing between pairs of components on a PCB after placement and pin assignment is complete [10]. Lei and Mak present an algorithm for FPGA I/O pin assignment and escape routing based on FPGA bank I/O voltage standards [11]. This algorithm provides an alternate method of simultaneously assigning I/O voltage supplies and I/O pin assignment after all components have been placed. These academic works aid in parts of the design process, but they do not sufficiently take advantage of the increased freedom from flexible interconnect.

2.1.2 2.5D and 3D Integrated Circuits

As designs become more and more complicated, the increased connectivity requirements between packaged components become more and more difficult to satisfy due to slower scaling of I/O interconnect [12]. Silicon die feature sizes continue to shrink, which enables growth in component complexity, but the density of package interconnect – and consequently the amount of available I/O pins – grows more slowly [12]. As these feature sizes decrease, die yield tends to decrease due to having more transistors, etc. in the same die area. Also, larger dies typically have lower yield than smaller dies, all other factors being equal. Instead of producing more and more complicated, monolithic dies, large dies can be split into multiple smaller dies in one package atop a silicon interposer [13].

Two architectural paradigms have emerged that incorporate multiple dies in a single package. The first of these is 3D die stacking, which features dies connected vertically using through-silicon vias (TSVs) [14]. A number of applications have been shown to benefit from 3D integration, e.g., high-performance communication, high-capacity storage, low-power, and high-performance heterogeneous systems [14]. 3D integration allows for shorter interconnect, higher bandwidth between dies, and integration of dies produced using different processes [14]. Challenges to 3D integration include high heat densities, mechanical problems (cracks and stress), and testing of ICs and the component dies [14]. Casale-Rossi, et al, write: “Digital, [memory] and [analog] processes have not converged. For example, analog does not scale on advanced digital processes and cannot co-exist with large amounts of noisy digital circuitry. In addition, the cost of leading processes is increasing due to advanced lithography, complex

devices and a limited number of suppliers" [15]. 3D IC technology enables heterogeneous integration of these disparate processes [15]. Remaining challenges to 3D integration include management of thermal and mechanical stresses, I/O management, and testing [15]. 3D ICs can be designed with a variety of mechanical structures. Silicon interposers can be semi-embedded into an organic substrate; dies can be attached on both sides of an interposer; thermal paths can transfer heat upward through a heat sink and/or downward through the IC package and through a "heat slug" in a PCB; fluidic micro-channels embedded within interposers also can transfer heat away from high-power dies [16]. Marinissen discusses testing 3D ICs at various points throughout the manufacturing process: pre-bond tests (before stacking dies), mid-bond tests (on partial die stacks), post-bond tests (on complete die stacks), and packaged tests (on a complete, packaged IC) [17]. In addition to conventional wafer testing on interconnect and active circuitry, testing 3D ICs should identify faults specific to this new technology (e.g., TSV interconnects) [17]. Testing is complicated by physical probe access; microbumps are very difficult to probe, due to their small size and high density [17]. Despite current challenges, 3D integration provides a number of valuable benefits to IC design.

The other major method to integrate multiple dies into a package is to use an interposer die, with component dies placed side-by-side on top of it [14], as shown in Figure 5. Connections between the dies are made through metal layers in the interposer, which allows for higher connection density between dies than between packaged components. This form of IC is commonly referred to as a 2.5D IC because it is "more 3D" than a traditional IC due to the interposer and the connections it allows between dies, yet "less 3D" than a die-stacked IC.

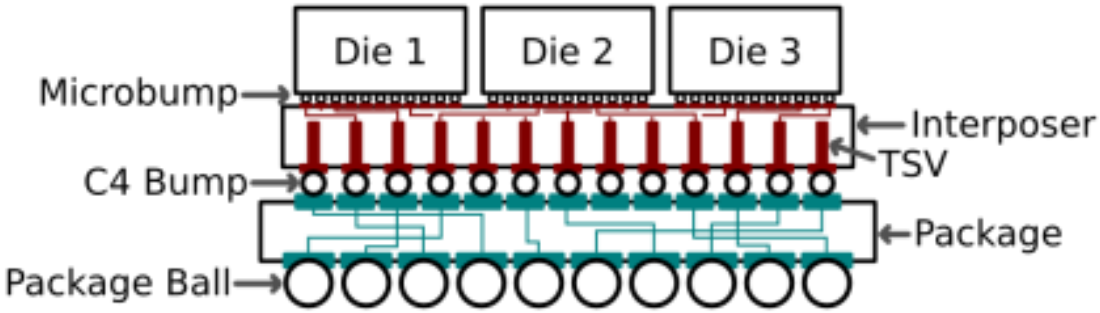


Figure 5: Illustration of 2.5D IC

A benefit of 2.5D interposer-based IC design (as well as 3D stacking) over monolithic low-yield, large dies is that multiple smaller, higher-yield dies can be connected via an interposer (or vertically in a 3D stack) to provide equivalent functionality [13], as illustrated in Figure 6. Very high-capacity FPGAs can be constructed from multiple, smaller FPGA dies placed side-by-side atop a silicon interposer [18], [19], as illustrated in Figure 7 [18]. Minimizing the number of inter-die signals can reduce both routing demand and interconnect delay [19]. CAD methods have been improved to model and optimize for 2.5D interposer-based FPGA ICs [19]. 2.5D and

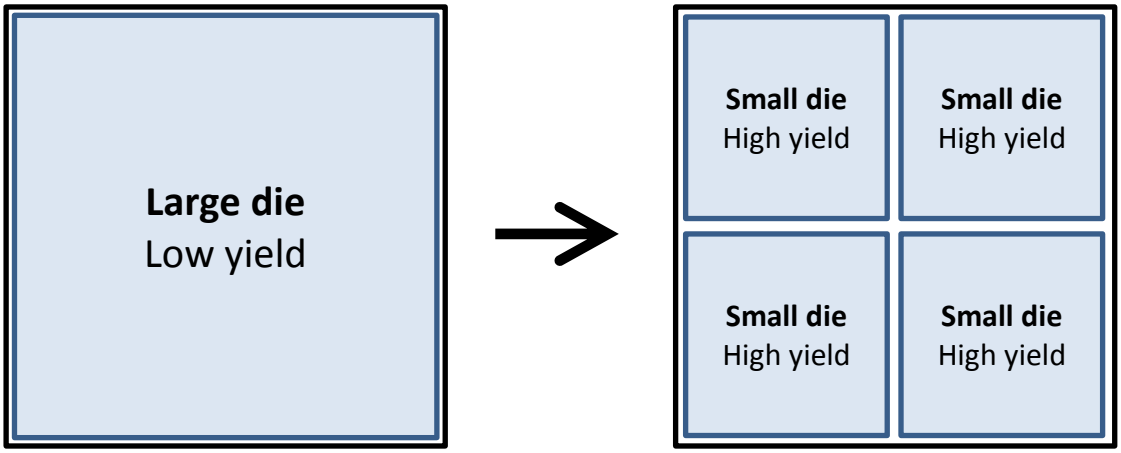


Figure 6: 2.5D designs enable composing a large IC out of multiple small dies instead of one large die

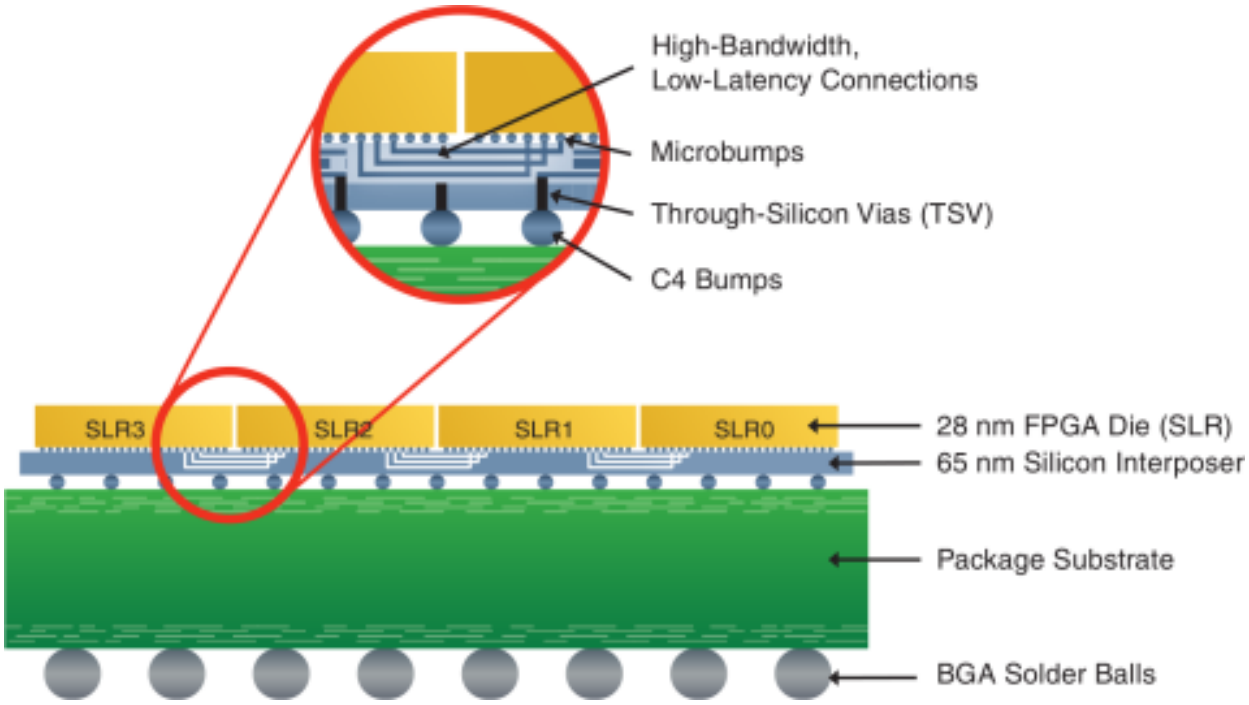


Figure 7: Four FPGA dies are combined side-by-side in the Xilinx Virtex-7 2000T FPGA [18]

3D design allows for integrating heterogeneous dies, each of which may be based on a different process [20]. 2.5D/3D technology can provide improved performance, power consumption, and bandwidth over traditional 2D design [20]. Partitioning a large die into multiple, smaller dies integrated on an interposer can enhance yield [20].

2.5D design overcomes some of the challenges of 3D die stacking. Analog, memory, and digital dies typically employ different technologies, complicating integration into a single IC [12][14]; 2.5D enables heterogeneity by allowing separately-manufactured dies to be connected with an interposer to create a single packaged IC [13]. Not only does 2.5D interposer-based design facilitate combining dies with different technologies, but also dies from multiple

manufacturers—without requiring coordination among those manufacturers as to how the constituent dies are designed.

Multiple demonstration platforms and commercial ICs have been produced that use 2.5D or 3D IC technology due to their many benefits. TSMC has targeted mobile applications by stacking memory dies atop logic dies to demonstrate 3D integration [20]. TSMC has integrated a N28HP ARM Cortex-A9 processor, a N65LP GPS, and a N50G wide-IO DRAM side by side on a silicon interposer [21]. Placing dies side by side provides many benefits over connecting components through a PCB: shorter interconnects, lower parasitic effects, and lower power [21]. Through-silicon via (TSV) -based interposers provide a good solution to challenges in traditional, monolithic chip packaging due to the interposers' high interconnect density, similar coefficient of thermal expansion to that of the connected dies, and shorter interconnects [22]. Microbumps connect dies to an interposer; these microbumps can have very fine pitch around 30–60 µm [22]. TSV interposer technology has enabled very high-capacity FPGA ICs to be built from multiple FPGA slices placed side-by-side on an interposer [22]. Xilinx has produced the Virtex 2000T, which is a high-capacity FPGA composed of multiple FPGA slices integrated onto a silicon interposer [12][18]. It features high interconnect density (>10K inter-slice connections) and low interconnect latency (around 1 ns) [12][18]. Microbumps with 40 µm pitch connect the FPGA slices to the interposer, and 200 µm pitch balls connect the interposer to the package substrate [12]. Xilinx's 7V580T integrates two FPGA slices with an 8x28Gb/s serializer/deserializer (SerDes) die [12][18]. Heterogeneity appears to be a large driving force for the adoption of 3D IC design techniques [13].

2.2 Computer-Aided Design Methods

In this section, we present common methods used for a variety of computer-aided design (CAD) tasks that aid in the 2D PCB and 2.5D IC design process. These tasks include component or cell placement (section 2.2.1), combined placement and signal assignment (section 2.2.2), routing and metal layer reduction (section 2.2.3), and FPGA-specific placement and routing (section 2.2.4).

2.2.1 Placement

Various works address assigning components to specific regions or positions within a layout. Cong and Luo propose a 3D floorplanner and an analytical method for placing differently-sized components in 3D IC design [23]. The placement objective in this work is to minimize total wirelength and number of TSVs [23]. Net wirelength is estimated using the log-sum-exp approximation to the half-perimeter wirelength (HPWL) model [23]. M.-K. Hsu, Y.-W. Chang, and V. Balabanov propose a three-stage 3D placement algorithm that analytically performs global placement including reserving space for TSVs, inserts TSVs and legalizes cell positions, and performs detailed placement on each layer [24]. The work includes an improved wirelength model featuring smaller estimation errors than the log-sum-exp wirelength model, and a 3D density model allowing for even cell distribution among the multiple layers [24]. TSV positions are determined during placement, so traditional 2D routers can provide routing solutions for each layer [24]. Fischbach, Knechtel, and Lienig propose a layout representation of 3D intellectual property (IP) blocks for floorplanning across multiple dies [25]. This representation can be used to floorplan 3D (or 2D) blocks with complex shapes, and it facilitates packing

multiple 3D blocks into small, dense layouts [25]. This method proved very effective in improving layout quality and consistency [25]. W. Sui, S. Dong, and J. Bian present a 3D FPGA placement algorithm composed of three steps: force-directed placement in 2D, placement legalization, and partitioning into multiple 3D layers [26]. Total wirelength is the minimization objective for 2D placement [26]. After 2D placement, all logic blocks' positions are legalized to discrete positions across the IC, in order according to each logic block's criticality for timing [26]. Finally, a simulated annealing-based algorithm partitions each logic block into a particular layer, attempting to minimize total wirelength and number of nets cut between layers [26]. Even though these works address component floorplanning or placement, they do not include any signal assignment techniques.

A number of works utilize congestion estimation to direct placement [27]–[34]. The goal in congestion estimation is to predict the density of interconnect in different sections of a layout and place components in such a way to reduce the interconnect density. As fewer connections pass through an area, fewer wire crossings likely will occur, and therefore fewer metal layers will be needed. We use this idea in our later work on routability-driven 2.5D placement and pin assignment, which we discuss in chapter 5 (starting on page 63).

2.2.2 Placement and Signal Assignment

Another set of works address both component placement and signal assignment. Ho and Chang use simulated annealing to place macros and I/O buffers within each die and to place each die on a silicon interposer with a minimized total wirelength and deviation from the desired layout

aspect ratio [35]. After placement, they use a bipartite matching algorithm to assign connections between microbumps and I/O buffers [35]. Kim, Athikulwongse, and Lim propose a force-directed 3D placement algorithm and two different schemes for handling TSV insertion: "TSV-site": place TSVs at fixed, regular positions and then place gate cells; or "TSV co-placement": place TSVs and gate cells simultaneously [36]. First, cells are distributed among dies using an algorithm based on FM partitioning, where the cutsize corresponds to number of TSVs [36]. Next, for the TSV-site method, TSVs are placed at fixed positions throughout the layout, gate cells are placed, and then nets are assigned to specific TSVs. For the TSV co-placement method, TSVs and gate cells are placed simultaneously [36]. After placement, existing 2D routing algorithms are used to route each die layer separately [36]. Tsai, Wang, and Hwang propose a two-stage algorithm to place macros and TSVs, then to reassign TSV signals as necessary to reduce total wirelength [37]. The first stage, based on simulated annealing, places macro and TSV blocks to minimize a weighted sum of a layout's area, total wirelength, and a penalty for a suboptimal layout aspect ratio [37]. The second stage, modeled as a minimum-cost maximum-flow problem, reassigns signals to TSVs to reduce the total wirelength [37]. These methods were shown to be more effective than performing placement under the assumption that any signal TSVs would fit within the bounding box of that signal's connected pins [37]. Zou, Xie, and Xie attempt to reduce metal layer costs by estimating the minimum number of metal routing layers based on models for interconnect and 3D fabrication costs [38]. They estimate routing demand based on interconnect length and the average number of pins per block, and they estimate the routing capacity based on routing area and metal wire pitches

[38]. Fabrication costs include factors such as die area, TSV area overhead, estimated number of TSVs, and TSV yield [38]. Their proposed method performs a binary search to determine the number of metal layers required for a design [38].

2.2.3 Routing and Metal Layer Reduction

Several prior works attempt to reduce the number of metal layers (and thus reduce interposer costs) in 2.5D IC design [38], [39]. These works are restricted to fixed pin assignments and do not consider flexible interconnect. Liu, Chien, and Wang present a global-routing-based method of reducing the number of metal layers in a 2.5D interposer [39]. They iteratively perform rip-up and reroute on an aggregate 2D graph and gradually reduce the number of metal layers, as

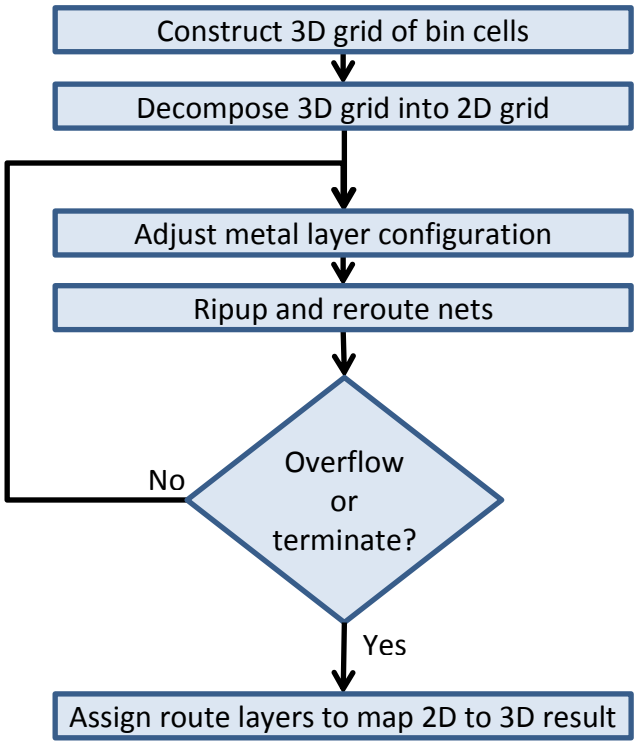


Figure 8: Metal layer planning and routing method from [39]

illustrated in Figure 8. As a post-processing step, their work maps the 2D routed results to 3D graphs to produce the final 3D routes. Zou, Xie, and Xie explore the tradeoffs between interposer area and number of routing metal layers [38]. They perform a binary search for the minimum number of routing layers, while balancing decrease in routing layer count against increase in area. Both of these works optimize input designs with *fixed* pin assignments, without considering flexible interconnect. Thus they are limited when optimizing systems with FPGA dies.

When designing the interposer in a 2.5D IC, it is essential that the number of metal layers allow routing all interconnects between the dies without exceeding the routing resources. To verify if a considered number of metal layers for the interposer is sufficient, various routing techniques such as [40]–[48] can be used. For example, NTHU-Route performs global routing by repeatedly ripping up and rerouting nets on a 2D grid [40]. It keeps a history of congested grid edges and includes a historical term in its cost function that gradually increases for frequently-congested grid edges [40]. Then NTHU-Route refines the layout to reroute nets to remove all overflow; finally it performs layer assignment to produce a 3D routed solution [40]. Liu, Kao, Li, and Chao present two bounded-length maze routing (BLMR) algorithms and a multi-threaded global router, along with supporting methods for improving solution quality [41]. Their bounded-length maze routing algorithms limit the maze routing search region, and the heuristic-based BLMR algorithm speeds up runtime by considering fewer paths than the optimal BLMR algorithm [41]. Hu, Roy, and Markov present a method that performs 2D global routing that then is projected to a 3D routed solution [42]. Their method produces an initial 2D grid routing

solution and repeatedly rips up and reroutes any nets that have design violations, until no violations remain or a maximum processing time is reached [42]. They then perform layer assignment to achieve 3D routes for all nets [42]. Xu and Chu present a multi-level 3D global router that uses a successive coarsening and decoarsening process to provide quality routing solutions [43]. Their work includes improvements in both the coarsening and decoarsening stages to improve route prediction and produce better solutions [43]. Cho, Lu, Yuan, and Pan present a method of 2D global routing and blockage-aware layer assignment [44], [45]. This method reduces routing congestion by shifting intermediate net nodes to less-congested areas and by considering historical cost measures as the routing algorithm progresses [45]. Though these methods are effective for routing designs with fixed connections, they do not consider flexible interconnect.

When routing multi-terminal nets, it is common to insert intermediate nodes, called Steiner nodes, to improve route quality; a method for generating these nodes is given in [49]. Cho, Lu, Yuan, and Pan present an improved method of shifting these Steiner nodes to avoid congested areas [45]. Zhang, Xu, and Chu developed a guided maze router based on their virtual capacity technique [46]. They extend this router to also reduce the number of vias needed to route all nets [47]. Liu, Li, and Koh present a hybrid unilateral monotonic router and a method of expanding the router's bounding box according to routing congestion [48]. This method can quickly produce high-quality routes while featuring fast runtimes [48]. Unfortunately, these routing algorithms do not support flexible interconnect, which means they cannot improve final layouts beyond what routes can be realized for the fixed connections.

2.2.4 Placement and Routing for FPGAs

One foundation of our work is the integration of placement and routing techniques with flexible devices such as FPGAs; a wide body of research has studied this subject [50], [51]. Versatile Place and Route (VPR) is an FPGA placer and global/detailed router that is effective for comparing different FPGA architectures [52]. It uses simulated annealing to iteratively swap the positions of logic blocks or I/O pads in order to minimize linear congestion or bounding-box wirelength [52]. After placement, VPR performs routing based on Pathfinder [53]: iteratively rip-up and re-route every net, increasingly penalizing routing resource overuse [52]. We adapt techniques from both VPR and Pathfinder throughout our work.

3 2D and Printed Circuit Board Design Automation

Our initial work focused on issues related to electronic design automation of systems featuring components with flexible interconnect. FPGAs fall into this category due to their flexible I/O pins. Other components also may have flexibility in how they are interconnected because it may not matter *which* pin within a group is connected to a particular net. In systems having programmable controllers or microprocessors, certain signals can be connected in an arbitrary order that is handled properly in software. For example[1], consider a 7-segment LED display connected to general-purpose input/output (GPIO) pins on a microcontroller. A schematic diagram for this example is shown in Figure 9, which was created using the open-source PCB

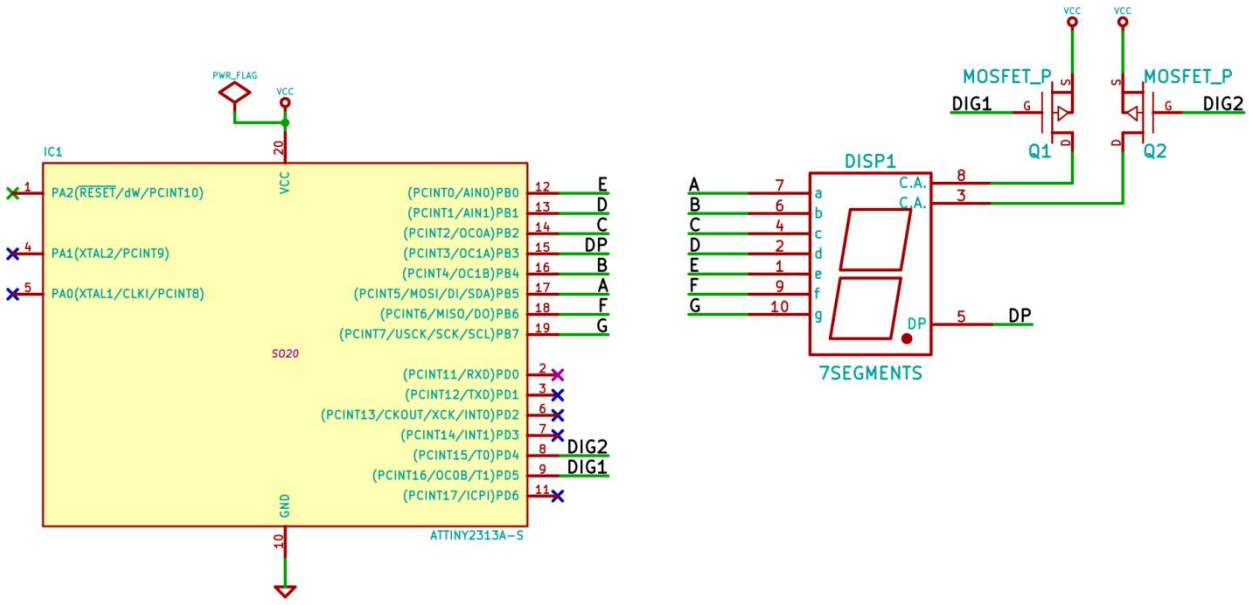


Figure 9: Schematic diagram of a 7-segment LED display connected to a microcontroller's GPIO pins. Diagram created in KiCad [54]

[1] No project within Electrical and Computer Engineering would be complete without including a decent number of LEDs.

computer-aided design tool KiCad [54]. Displaying numbers can be done easily using a lookup table in software; changing the connections' pin order simply requires re-generating that table.

Components with flexible interconnect include both packaged components connected via PCBs and dies connected via silicon interposers. Our work has focused on component placement, I/O bank voltage supply assignment, and pin assignment for both 2D PCB and 2.5D IC design. This chapter focuses on 2D PCB design automation.

3.1 Methodology

Printed-circuit board design involves placing components on a substrate, assigning connections between components' pins, and routing metal traces to implement these connections. Often there is freedom in assigning connections to pins; designers take into account flexible interconnect when performing PCB design. If during trace routing it becomes apparent that two nets cross (requiring additional layers), the designer can swap the pin assignments of these two nets if the assigned pins are flexible and can accommodate the adjusted assignment. This section presents our methodology to take advantage of the additional opportunities provided by flexible interconnect.

We incorporate capturing high-level design constraints (section 3.1.1), simultaneous component placement (section 3.1.3) and bank assignment (section 3.1.4), and ILP pin assignment (section 3.1.5) into one overall framework that is available under an open-source license. This framework includes tools for defining and viewing components, visualizing results, importing design requirements, importing and exporting PCB layouts, and executing the

algorithms described in the previous sections. Our framework is published at http://spcg.ece.wisc.edu .

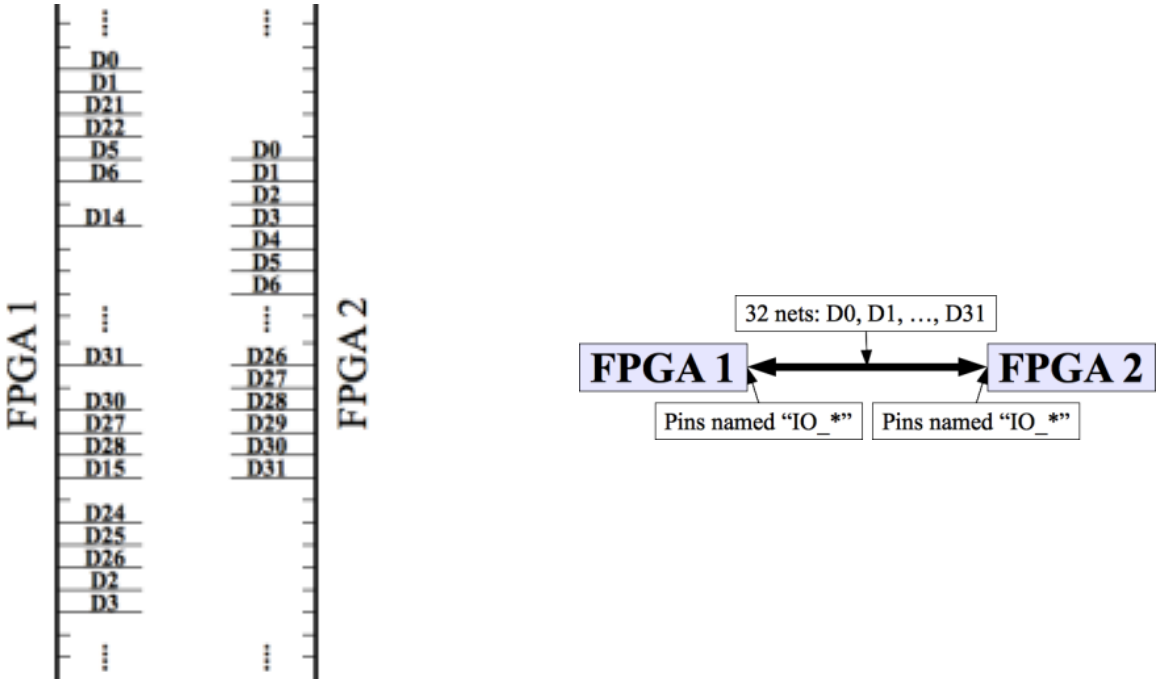
3.1.1 Capture of High-level Design Constraints

Printed circuit board (PCB) design is a tedious, time-intensive process, as a designer must specify many low-level details during design capture. In particular, these low-level details include connectivity requirements for all nets, including the pins that will be used on each device for these connections. To facilitate faster and easier PCB design, we have developed a method of design capture that works at a higher level of abstraction than traditional design methods. Using our method, designers use a scripting language to define the devices and connectivity required between them on the PCB. However, rather than specifying connectivity on a pin-by-pin basis, designers describe groups of connections with common requirements: components at each end of the connections, logic level standard, single-ended vs. differential, and pins eligible to make the connections. This provides the necessary information to produce a valid layout and pin assignment without overly constraining design decisions. After these design requirements are captured, our framework automatically generates a set of layouts to meet the given requirements. The designer may choose between these and/or “tweak” them to produce the desired final result. An overview of our framework was shown in section 1.1. The advantage to this high-level design capture is that designers can specify components and connection requirements without being required to manage many low-level details, plus the framework can leverage the opportunities provided by flexible pins in devices such as FPGAs.

3.1.2 Design Requirement Specification

Using our proposed framework, a designer specifies the components that must be connected and the connectivity requirements for a design. Our framework includes an expandable library of predefined components, with annotations for component dimensions, pin coordinates, and pin functions. To specify connectivity requirements, a designer defines groups of connections (called "bundles"), including such details as net names, applicable I/O standards, and eligible pins for these connections on the devices that they connect. Eligible pins can be matched by a variety of search parameters, such as pin function (e.g., "has function named IO_*"), I/O bank (e.g., "in bank 32"), or specific pin sets (e.g., "is in the set {A1, A2, A3, A4, B1, B2, B3, B4}"). This provides the flexibility for specifying eligible pins according to the particular design's requirements.

For example, Figure 10 illustrates two different methods of specifying a 32-net connection between two FPGAs. Instead of choosing 32 I/O pins on each FPGA and enumerating 32 nets connecting the FPGAs (as in Figure 10a), the designer can indicate a requirement that 32 I/O pins on each FPGA be connected with nets having certain names (as in Figure 10b). Our framework compiles textual descriptions of all such requirements and then simultaneously chooses specific I/O pins to meet all the designer's requirements.



a. Defining connection requirements using low-level schematic design intent capture

b. Defining connection requirements using higher level of abstraction

Figure 10: Two methods of specifying connections between two FPGAs

3.1.3 Component Placement

After capturing design requirements, our PCB design framework assigns positions and rotations to all components. It uses a form of simulated annealing (SA) similar to TimberWolf's standard cell placement algorithm [55], as illustrated in Figure 11. In simulated annealing, the current solution is perturbed randomly, and the modified solution is accepted according to a probability based on the total cost increase or decrease. A perturbation that increases the total cost will be accepted as a function of SA's "temperature," which decreases as the algorithm progresses. Our SA placement algorithm places components in a PCB layout in order to minimize predicted total

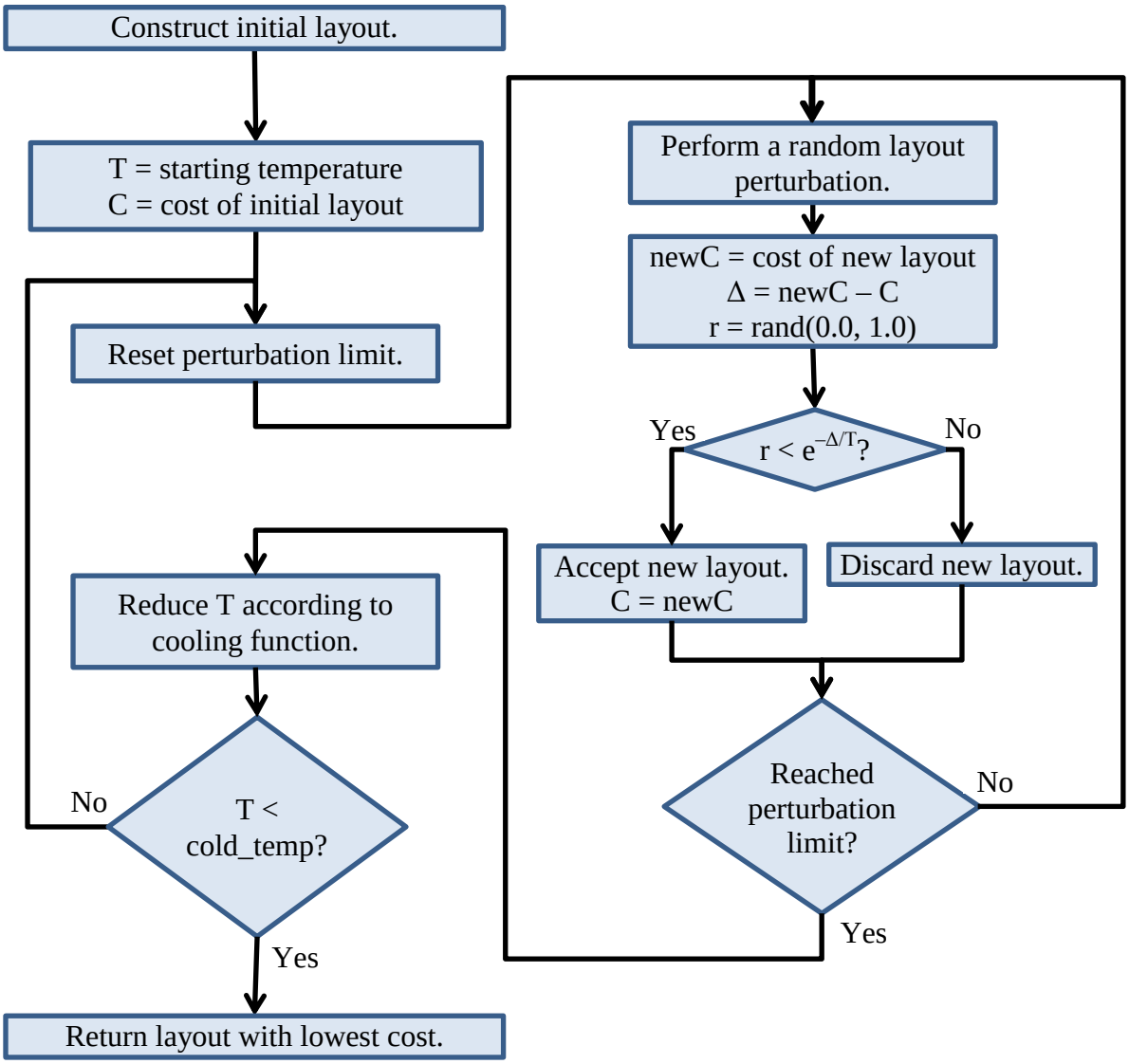


Figure 11: Overview of simulated annealing

trace length and component overlap. Our cost function includes a linear combination of two direct costs:

- The sum of estimated lengths of all connections
- The area of the bounding box containing all components

Our cost function also includes several weighted penalties in the cost function that represent hard design requirements:

- A penalty for board dimensions larger than the designer-specified limits
- A penalty for overlapping components
- A penalty for I/O voltage incompatibility, proportional to the number of connections having incompatible I/O voltage requirements
- A penalty for pin and differential pair overuse, proportional to the number of connections in excess of capacity for each bank of pins

The weights for these penalties are initially low, so that SA can freely explore a wide solution space. As the algorithm progresses, these weights increase gradually to relieve design violations while still achieving quality results [53].

We use an existing cooling schedule [52]: for a given SA temperature, compute the ratio R_{accept} of accepted perturbations to attempted perturbations at that temperature, then compute the new temperature as per Table I based on the current temperature and the computed R_{accept}.

Table I: Simulated annealing cooling schedule [52]

$R_{accept} > 0.96$	$T = 0.5 \cdot T_{old}$
$0.8 < R_{accept} \leq 0.96$	$T = 0.9 \cdot T_{old}$
$0.15 < R_{accept} \leq 0.8$	$T = 0.95 \cdot T_{old}$
$R_{accept} < 0.15$	$T = 0.8 \cdot T_{old}$

In this schedule, the temperature decreases quickly while most perturbations are accepted. While there is a good balance of accepted vs. rejected perturbations, the temperature

decreases slowly. Once most perturbations are rejected, the temperature decreases at a moderate rate to allow for solution improvement before terminating the algorithm.

To produce a new trial solution, our framework randomly chooses from five different perturbation types:

- Move one component to a new location within a certain range of its current position. This range is initially large, but decreases with temperature decrease [52].
- Exchange the positions of two random components regardless of range.
- Rotate one component by some multiple of either 45° or 90° (chosen by the designer).
- Move one terminal of one connection to a different bank of pins within the same component (bank assignment is further discussed in section 3.1.4).
- Exchange two connections' terminals between banks of the same component.

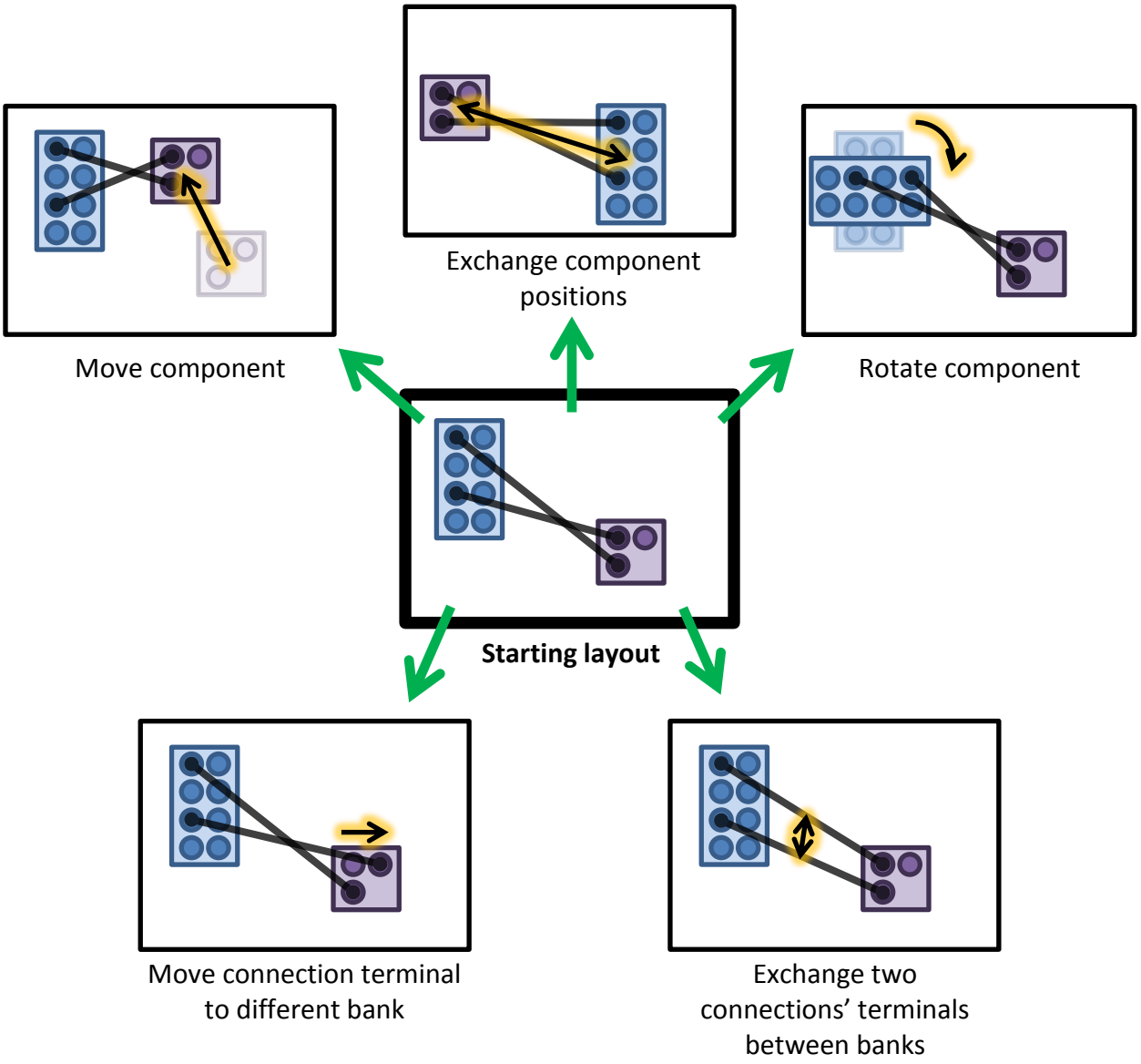


Figure 12: Perturbation types for 2D placement algorithm
Colored rectangles are components, and circles are banks (groups) of pins.

These perturbations are illustrated in Figure 12. In this figure, two components (colored rectangles) are arranged in a layout with connections between two banks (circles) of pins in each component. Components' positions can be changed (in the top three illustrated perturbations), and the connected banks can be changed (in the bottom two perturbations).

The default probabilities for these perturbation types are shown in Table II.

Table II: Default perturbation probabilities

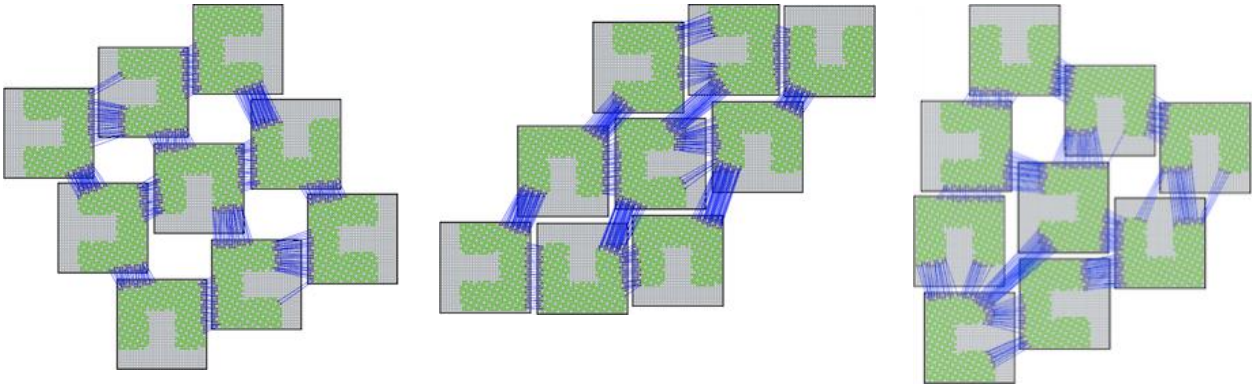
Perturbation Type	**Probability**
1: Move component	33.3%
2: Exchange components	10.0%
3: Rotate component	6.7%
4: Move connection terminal	38.5%
5: Exchange connection terminals	11.5%

3.1.4 Bank Assignment

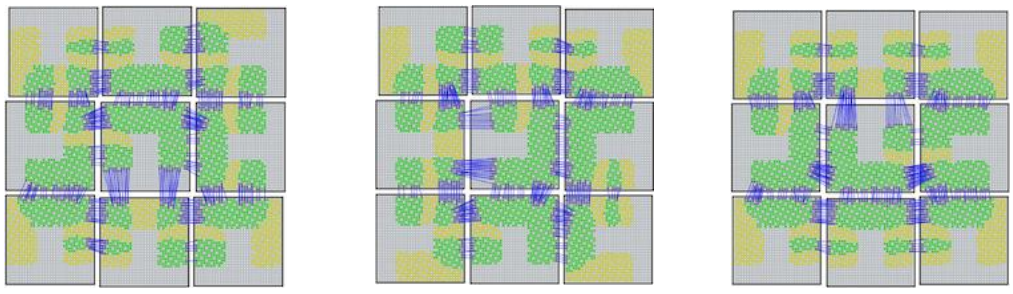
To improve estimation of connection lengths and to facilitate grouping of connections by compatible I/O levels, we include bank assignment as part of the same simulated annealing algorithm used for component placement described in section 3.1.3. In addition to assigning component positions, our SA algorithm also assigns all connections' terminals to banks of eligible pins. The bank assignments are required to heed the number of available pins in each bank, and no connections with incompatible I/O levels are allowed to be assigned to the same bank. Any I/O power supplies in a bank are set to the voltages required by the connections to that bank.

Because connections have not yet been assigned to specific pins, our placement algorithm requires an estimate for connection lengths. Using the overall center of mass (CoM) of a connected component is a common method for estimating the endpoint location of a connection. This approach can result in relatively accurate connection length estimates *for small components*, such as small lookup tables, because the computed CoM is near the actual

"pin" used for the connection. For large components, however, the assigned pin may be far from the computed overall CoM. In some cases the distance between the assigned pin and the CoM may be greater than the final trace length between the assigned pins. Therefore, we compute the centers of mass (CoM) of eligible pins in each component's banks for each group of related connections. We then estimate connection lengths as the distance between centers of mass of the connected component banks. This method proved more effective than estimating connection lengths using the CoM of each component overall. Figure 13 shows



a. Layouts produced for a 9-FPGA design when estimated trace lengths are calculated using the CoMs of *all* eligible pins on the relevant devices



b. Layouts produced for a 9-FPGA design when estimated trace lengths are calculated using the CoMs of *each bank's* eligible pins on the relevant devices

Figure 13: Layouts produced for the same 9-FPGA design description

layouts that were produced using these two different methods: device-level centers of mass (Figure 13a) and bank-level centers of mass (Figure 13b). Wirelength and area data for results using device-level CoMs are shown in Table III. This table indicates the wide range of wirelengths and data, indicating strong inconsistencies resulting from using the device-level CoM method of estimating connection lengths during placement. In contrast, the results using bank-level CoMs gave smaller and more consistent wirelengths and areas, as shown in Table IV.

Table III: Costs for 9-FPGA design using *device* CoMs

Seed	Wirelength (1-norm)	Wirelength (2-norm)	Area
1	7762.26	7019.41	14931.1
9	10517.42	9466.22	17332.0
5	5890.49	4846.64	19110.4
6	6047.78	4784.86	20095.8
4	3981.50	3383.57	21860.9
10	9464.26	8238.92	22607.7
3	7934.49	6261.58	23792.8
7	11949.51	10740.38	24193.7
8	13814.37	10483.47	27097.8
2	17886.40	13473.21	39780.3

Table IV: Costs for 9-FPGA design using *bank* CoMs: this method resulted in consistently higher-quality layouts than using *device* CoMs

Seed	Wirelength (1-norm)	Wirelength (2-norm)	Area
3	2793.39	2517.04	12117.0
1	2769.22	2501.69	12134.7
9	2773.78	2515.33	12143.7
2	2705.86	2459.08	12150.0
5	4999.50	3859.12	12160.2
6	2708.99	2450.49	12165.7
10	2630.41	2388.69	12170.2
8	2728.85	2487.26	12192.4
7	2947.45	2637.46	12305.2
4	2963.33	2631.39	12313.9

3.1.5 Pin Assignment

After simultaneously performing component placement and bank assignment, our framework assigns specific pins (or differential pairs) to each required connection. These pin assignments adhere to the bank assignments generated during simulated annealing and to applicable constraints. We automatically generate an integer linear programming (ILP) formulation to assign all pins for a PCB layout. We use GLPK [56] to solve the ILP formulation.

Our ILP formulation requires the minimization of the total connection length, i.e., the sum of all connections' pin-to-pin distances. We use Euclidean distance (2-norm) instead of Manhattan distance (1-norm) for connection length, because traces in PCB layouts are not restricted to only horizontal and vertical lines; 45° traces are quite common. The ILP minimization is subject to the following constraints:

- Each pin is limited to at most one connection.

- Each differential pair is limited to at most one differential connection.
- Each connection must be assigned to eligible pins in the banks that were automatically assigned during simulated annealing.

Because connections were already assigned to banks of pins in such a manner that all connections assigned to the same bank use compatible I/O levels, the specific pin assignments provided by this ILP formulation also will ensure compatible I/O levels.

3.2 Evaluation

To evaluate our methods and framework, we define a number of commercially-available components and example systems that use these components. For example, we define components including:

- Basic passive components, e.g., resistors and capacitors
- Over 70 Xilinx and Altera FPGA models, with multiple physical package variants for applicable models
- Multiple analog to digital converters (ADCs)
- Multiple digital to analog converters (DACs)
- Multiple DRAM and Flash memories
- Various logic, interface, and driver integrated circuits

Using our framework's high-level design capture, we describe example systems that use many of these components:

- Grids of FPGAs, to compare our automatic placement with specific intuitive layouts
- FPGAs connected via AC-coupling capacitors, to evaluate placement and bank assignment and pin assignment for designs containing components of very different sizes
- A signal-processing system with a modern FPGA, DDR3 memory modules, ADC and DACs, and fixed analog and high-speed connectors, to evaluate I/O level grouping and component placement with mixed moveable and fixed component positions
- A subset of the BEE3 hardware platform [57], to compare our automatically-generated layouts with existing, commercial designs

We apply our presented methods to each of these example systems in order to automatically generate a variety of layouts that satisfy the design requirements. These layouts include positions and rotations for all components, power supply assignments as applicable for banks of I/O pins, and specific pins to use for each connection in the design. We evaluate these layouts by measuring total wirelength and footprint area. We measure wirelength using 1-norm (Manhattan) distances and using 2-norm (Euclidean) distances; we focus on the 2-norm in PCB design automation, as PCBs more often include routing not restricted to 90° angles.

3.3 Results

Using a variety of example system requirements, we show the capability and utility of our methods for producing efficient layouts with much less effort than manual design. Some of these layouts are shown in Figure 14 through Figure 17.

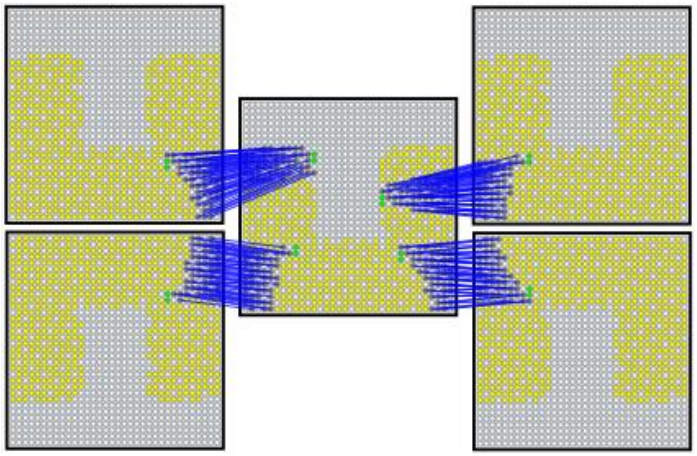


Figure 14: A generated layout for a system of a center and satellite FPGAs

Table V: Costs for system of center and satellite FPGAs

Seed	Wirelength (1-norm)	Wirelength (2-norm)	Area
1	6218.41	5997.21	6811.13
2	2388.70	2174.83	7912.34
4	2489.84	2190.44	7916.39
3	2472.03	2184.81	7918.34
6	2550.93	2262.30	7923.32
5	2643.19	2316.80	8019.10

Figure 14 shows an automatically-generated layout based on high-level requirements for four FPGAs each connected to a center FPGA. We generated a variety of different layouts much more quickly than designing these layouts manually. In so doing we can explore the design space for this benchmark, including the tradeoff of low area and high total wirelength (6811.13 and 5997.21) vs. higher area and lower total wirelength (7912.34 and 2174.83).

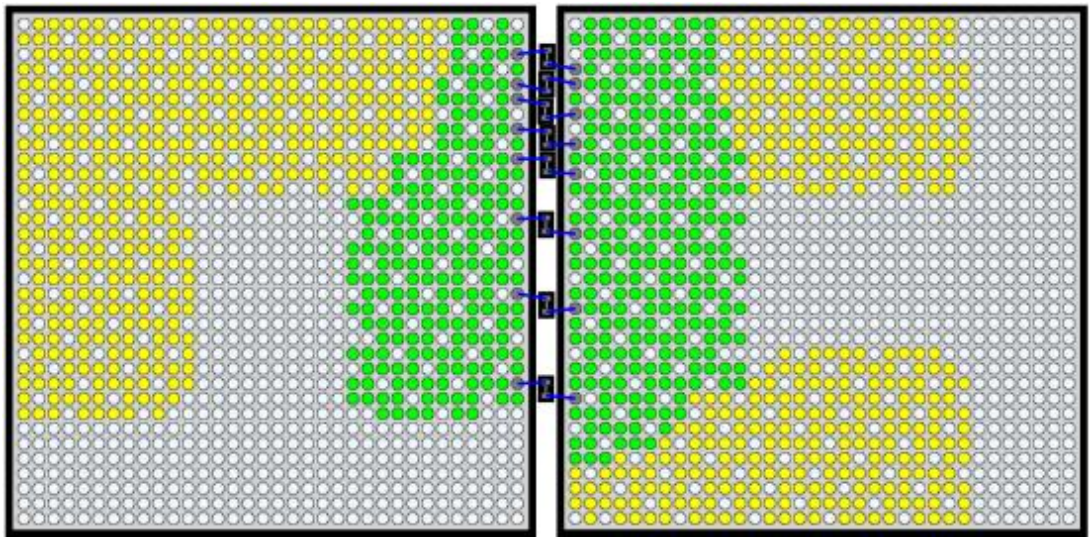


Figure 15: A generated layout for two FPGAs with
AC-coupling capacitors, enlarged to show detail

Table VI: Costs for FPGAs with AC-coupling capacitors

Seed	Wirelength (1-norm)	Wirelength (2-norm)	Area
2	34.591	31.305	2516.74
1	325.434	315.534	2555.61
3	124.325	107.293	2556.67
4	54.497	44.822	2602.55

Here again we generated multiple layouts for a benchmark system of two FPGAs connected via AC-coupling capacitors. We see fairly consistent areas (2516.74 – 2602.55) with a wide range of total wirelength (31.305 – 315.534). Again, our goal is to speed up design capture by requesting only the information actually needed, while allowing our automated methods to fill in design details.

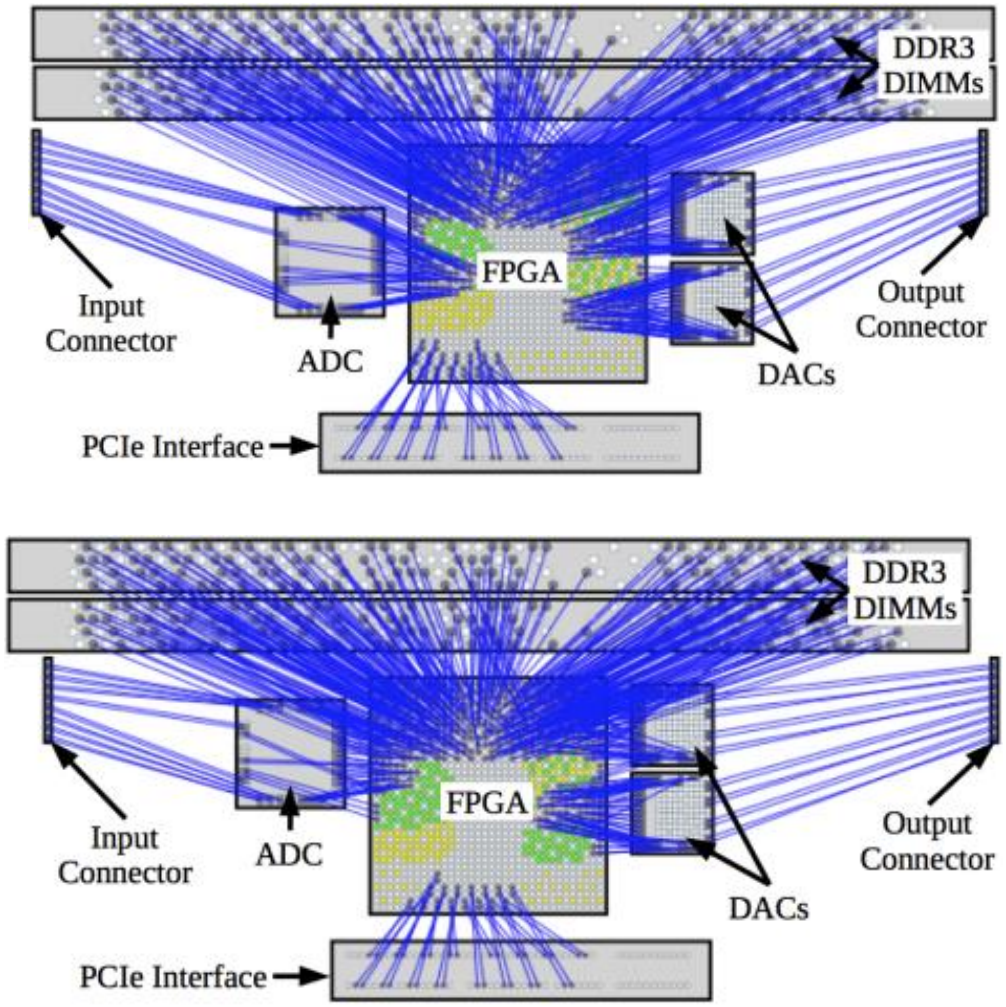


Figure 16: Two generated layouts for a signal-processing system with fixed-position external input and output connectors

Table VII: Costs for signal-processing system

Seed	Wirelength (1-norm)	Wirelength (2-norm)	Area
1	15867.7	12149.2	9764.39
2	15187.3	11737.0	9954.66

In Figure 16 we see two similar layouts that were automatically generated from the requirements of an ADC and two DACs connected to a controlling FPGA, with RAM and I/O connections to each IC. Performing even just the task of pin assignment would take much longer than specifying the connection requirements, we can allow our framework to automatically explore many options for placement and pin assignment.

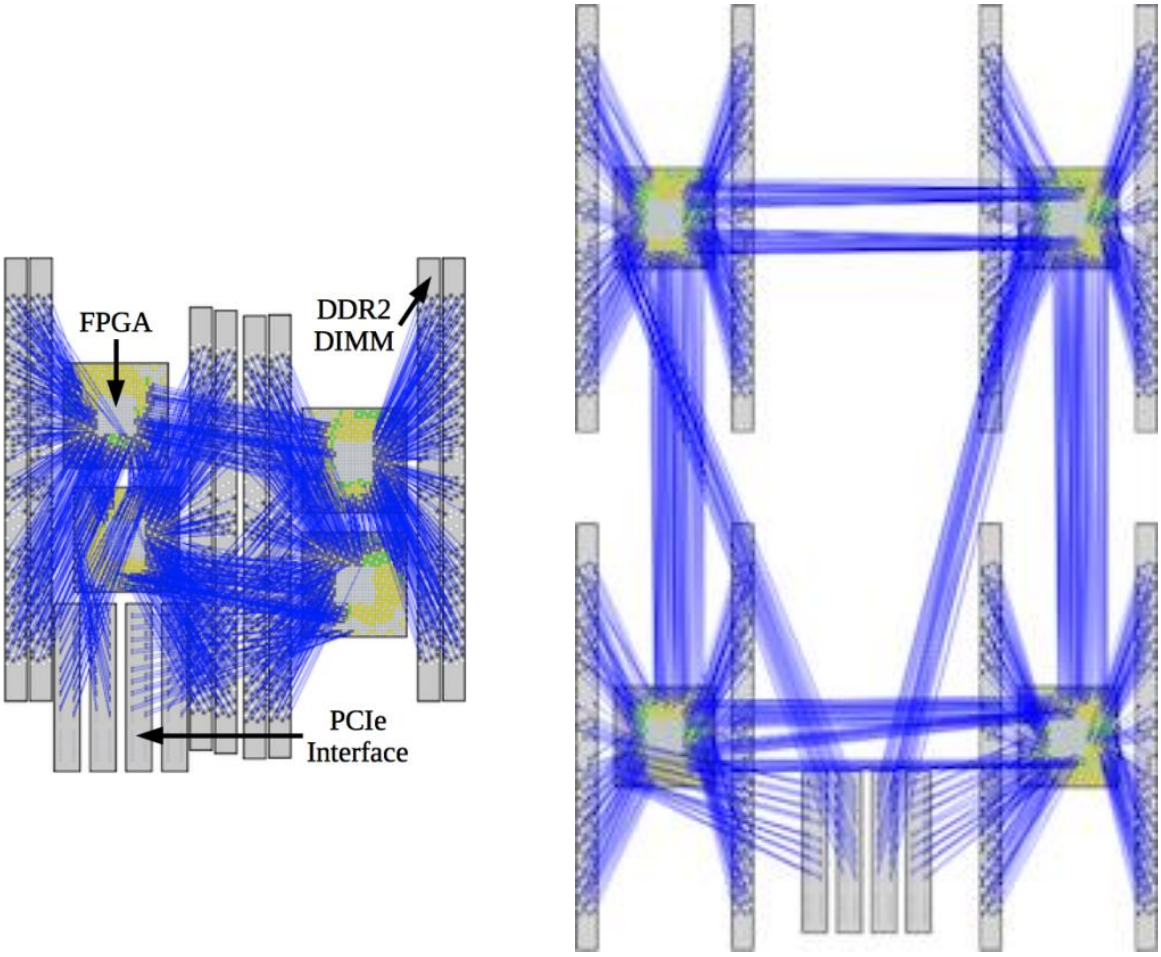


Figure 17: Automatically- and manually-placed layouts for a system similar to a subset of the BEE3 hardware platform [57]. The manual layout includes extra space to accommodate components not included in the automatically-placed subset of the design.

Figure 17 shows two layouts for a benchmark system based on a subset of the BEE3 hardware platform [57]. These layouts include four FPGA ICs connected in a ring arrangement. Each FPGA connects to two DDR2 DIMM sockets and to an 8-lane PCI-express connector. The layout on the left was automatically generated using our described methods. The layout on the right was manually placed based on the BEE3 board [57]. The automatically-generated layout is much more compressed than the manual layout because the manual layout includes space for components that are not modeled in this benchmark, but the overall arrangement of the components is very similar.

3.4 Summary

Our methods for automatic placement, bank assignment, and pin assignment can speed up the PCB design process by enabling design space exploration without requiring the designer to enumerate many mundane details. Instead, the designer specifies requirements for a design at a higher level, and our framework fills in the missing details.

In contrast to existing PCB computer-aided design tools, our framework abstracts the design problem to maximize designer productivity and ease of design entry and to facilitate rapid prototyping. Designers use a novel, scriptable PCB design requirement specification approach to describe required connectivity at a high level of abstraction. The framework then uses this description to guide automatic placement and signal bank, I/O voltage supply, and pin assignment of PCB components. A novel method of estimating trace lengths improves FPGA component placement, since exact pin assignments are not yet known. We show that our

framework is able to produce efficient layouts for a variety of designs with differing characteristics. Although an experienced designer may be able to create slightly better designs, our framework allows a person, experienced or not, to create reasonable layouts with much less effort.

4 2.5D Integrated Circuits

We next expended our work to focus on 2.5D integrated circuit design automation. Although by no means identical to 2D PCB design, this form of design also requires placement and connection of components, rather than transistors. As discussed in section 2.1.2, 2.5D ICs are composed of a set of multiple dies arranged side-by-side and connected via a silicon interposer [13]. The example 2.5D IC illustrated in section 2.1.2 is repeated here in Figure 18. This IC design method provides several benefits, including facilitating the integration of dies from multiple manufacturers and vendors without coordinating how those dies are designed.

2.5D integrated circuits can contain FPGA component dies or other dies having flexible interconnect. As we will show, we can exploit this flexibility to produce 2.5D IC designs of higher quality (wirelength, area, and/or manufacturing cost) than if all connections were fixed. We propose the first (to our knowledge) problem formulation and procedure for performing placement and pin assignment that supports 2.5D ICs containing one or more dies with flexible I/O (e.g., FPGA dies). We present this procedure and evaluate its utility for 2.5D IC design.

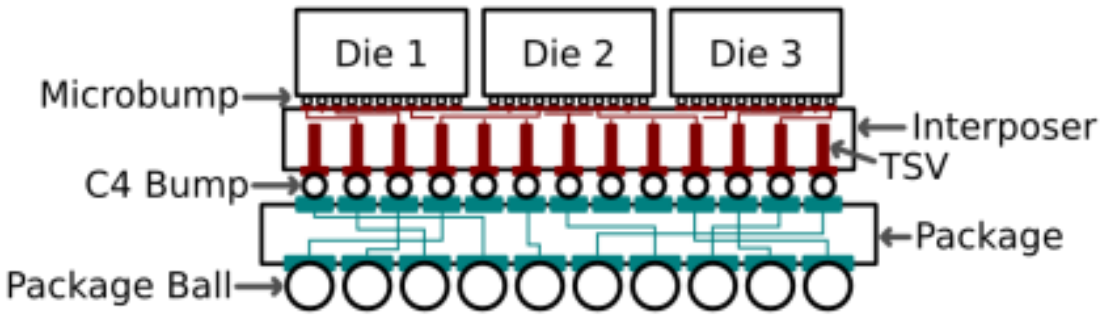


Figure 18: Illustration of 2.5D IC

4.1 Methodology

This chapter improves upon the methods we described in chapter 3 (2D PCB design automation). In particular, we present methods for simultaneously placing components, assigning power supplies, and assigning pins for each connection; this is in contrast to our initial work in chapter 3, which performed pin assignment separately, after placement and bank assignment. By considering placement and pin assignment simultaneously, our improved work can more effectively optimize both placement and pin assignment. Overall, our framework performs the following:

Given a set of dies to be connected, eligible pins for each connection, and I/O standard requirements (if any) for each connection, generate these features illustrated in Figure :

- Position and rotation angle for each die (positions are illustrated as offsets from the origin, and rotations as clockwise angles)
- Assignment of each connection to specific pins on the connected dies
- Power supply assignment for each bank of dies supporting this feature (e.g., FPGA dies)

This generation is subject to several hard constraints related to layout quality:

- There must be no overlapping dies.
- The layout must be no longer than the user-defined maximum sizes in both x and y dimensions.

- No pin may be assigned to more than one connection, and differential connections must connect to differential pair pins.
- Connections using pins within the same bank must use compatible I/O standards.

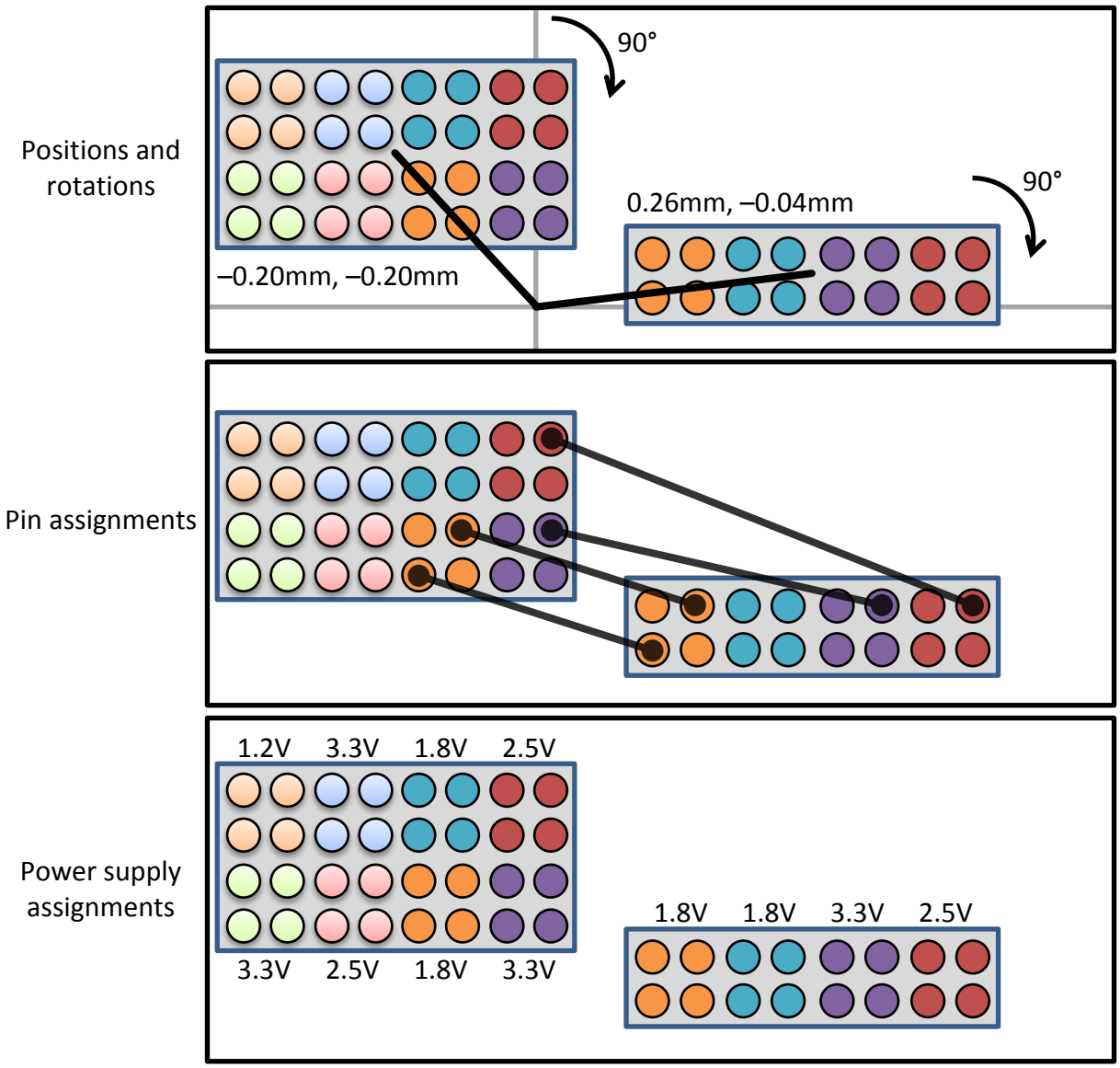


Figure 19: Generated layout features from 2.5D placement and pin assignment method

4.1.1 Improved Placement and Pin Assignment

The PCB design work described in chapter 3 simultaneously performed component placement, power supply assignment, and bank assignment for connections, but it waited until after the simulated annealing process to perform pin assignment. This early framework produced reasonable results but left room for improvement due to approximations in connections' terminal positions. During component placement and bank assignment, connections were assigned to *banks* of pins within the components based on the components' positions and the centers of mass of the banks of eligible pins. As a result, there is some error in the estimated connection lengths during bank assignment as compared to the estimated connection lengths after *pin* assignment. Our proposed 2.5D design procedure instead performs simultaneous die placement, pin (rather than merely bank) assignment, and power supply assignment. Performing these steps simultaneously avoids the case where a poor choice in an early design step then constrains later steps and the final result. Die placement and pin assignment are inter-related; the best placement depends on the assignment, and vice-versa.

We modify our framework presented in 1.1 to add this new capability. We also modify the cost function to include average and maximum estimated lengths of connections *from pin to pin* and to include a penalty for oversubscription *to each pin or differential pair*. Thus, the new cost function is a weighted linear combination of:

1. Average half-perimeter wirelength over all connections
2. Maximum half-perimeter wirelength over all connections
3. Area of bounding box containing all dies

4. Total interposer dimensions in excess of specified maximum
5. Total overlapping area between all pairs of dies
6. Number of pins assigned to connections incompatible with the assigned I/O bank power supply voltages
7. Number of connections over-subscribed to each pin (or differential pair)

Factors 4–7 are penalties on invalid layout characteristics; these weights are initially small to encourage freedom of movement at high temperatures, but increase as temperatures decrease. These weights are proportional to the inverse of the temperature. The penalties are analogous to the hard design requirements for the PCB work in section 3.1.3.

Layouts are iteratively altered by randomly-selected perturbation types:

- Move a die to a random position within a window about its current position; the window size decreases with the temperature [52].
- Exchange two dies' positions.
- Rotate a die by a multiple of 90°.
- Reassign one connection to a different eligible pin for that connection.

The last perturbation type (reassigning one connection's pin assignment) replaces the last two perturbation types related to bank assignments in the PCB framework's placement algorithm (reassigning one connection's *bank* assignment, and swapping two connections' bank assignments). Because swapping nets' assigned pins changes the current solution's cost by a relatively small amount, these net swaps were being accepted with higher probability than the

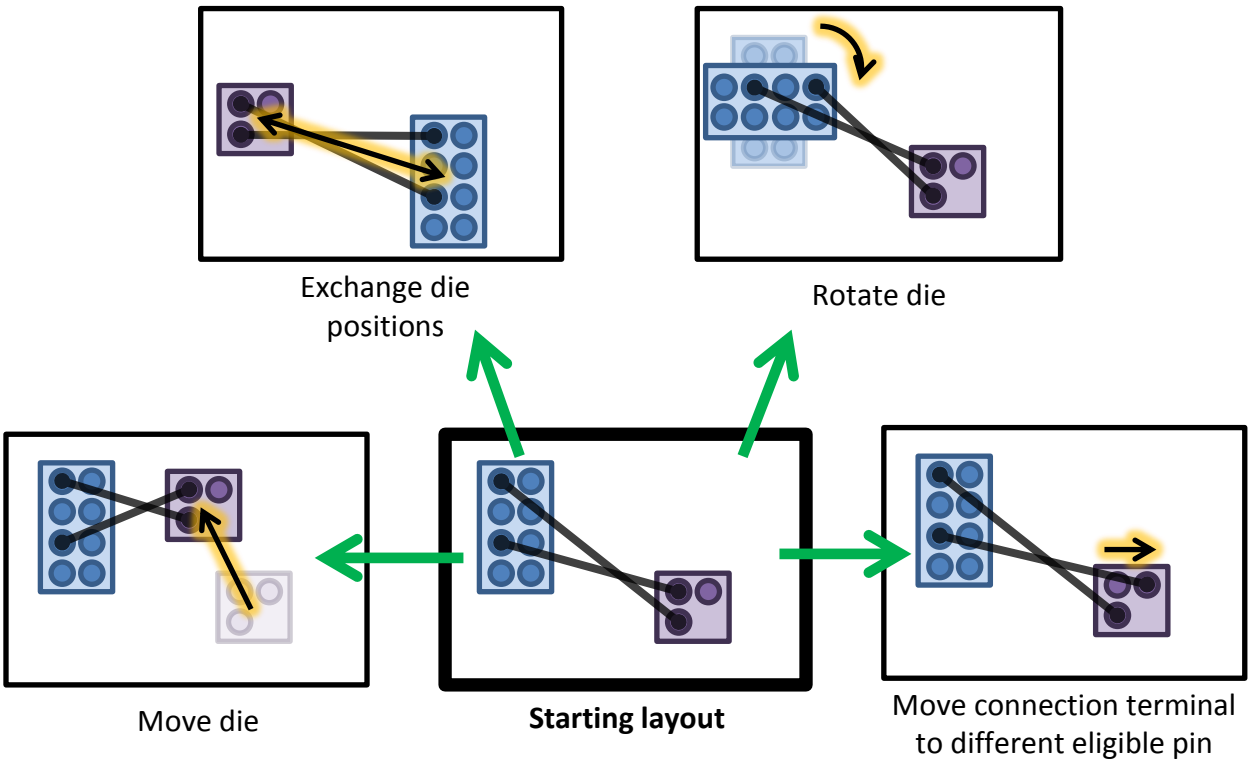


Figure 20: Perturbation types for 2.5D placement and pin assignment algorithm
Colored rectangles are component dies, and circles are individual pins.

other perturbation types. This affected other aspects of the placement algorithm, e.g., by slowing down simulated annealing cooling.

As part of adding support for 2.5D design, we also attempted to make the framework more flexible in terms of adding future support for a variety of other design automation tasks that process a variety of component types. This extension includes data structure expansions, algorithm changes, and die-level component definitions.

One algorithm change is that, rather than using static, parameterized SA initial and final temperatures, we adapt the SA temperature range to the system being processed. This allows

us to support automated placement and pin assignment for designs regardless of scale from PCB layout down to 2.5D IC design, and potentially even to handling placement and interconnection of large structures within a single die. To do this, our framework records the perturbation costs during one SA iteration at a fixed, high temperature. We compute the initial temperature as 20 times the standard deviation of these perturbation costs [52]. We also adapt an existing termination policy [52] to our needs: we terminate SA once the temperature drops below a fraction (K_{end}) of the current total cost. Unlike [52], our termination policy does not include a factor for number of required connections, as this factor is already a part of our total cost computation.

4.1.2 Pin Refinement

The difference in wirelength for modifying the pin assignment of an individual connection is small; thus simulated annealing may find a pin assignment that is close to the ideal for a given placement, but one that might be improved. We apply a post-SA pin assignment refinement step where we use an ILP formulation and solver adapted from our 2D pin assignment work from section 3.1.5.

Given positions and rotations for each die, and given bank assignments (as derived from the pin assignments) and eligible pins for each group of connections, generate refined pin assignments for each net in the groups of connections.

Define ep_i = (die, pin) to be a specific die pin. Define ep_i^D to be the die pin that is within the same differential pair as ep_i (if ep_i is part of a differential pair).

A group $g \in G$ of connections consists of $|g|$ single-ended or differential connections that connect from one set of eligible pins $E_g^L = \{ep_1^L, ep_2^L, \ldots, ep_n^L\}$ to another set of eligible pins $E_g^R = \{ep_1^R, ep_2^R, \ldots, ep_m^R\}$. The group and eligible pins are illustrated in Figure 21. We restrict $E_g^L \cap E_g^R = \emptyset$. (In practical examples this restriction is not troublesome.)

Define D_g to be 0 if group g consists of single-ended connections, or 1 if g consists of differential connections.

For a group g of differential connections, we assign each differential pair as a single unit, by including exactly one die pin within each eligible differential pair in the sets of eligible pins:

$$ep^{L^D} \notin E_g^L;\ \forall\ g \in G\ s.t.\ D_g = 1;\ \forall\ ep^L \in E_g^L$$

$$ep^{R^D} \notin E_g^R;\ \forall\ g \in G\ s.t.\ D_g = 1;\ \forall\ ep^R \in E_g^R$$

That is, for each group g of differential connections, if one die pin ep^L (ep^R) within a differential pair is eligible for a connection in that group g, then the paired pin ep^{L^D} (ep^{R^D}) is

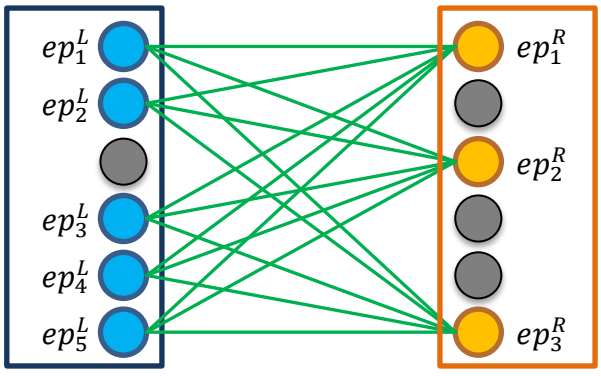


Figure 21: Example group of connections between eligible pins (blue and orange circles) on two components (blue and orange rectangles). Here $\left|E_g^L\right| = 5$ and $\left|E_g^R\right| = 3$. Green lines represent possible connections within this group.

not considered eligible. In this way, we can meet the differential connection requirements and remove redundant areas from the explored solution space.

Define E := $\bigcup_g E_g$ to be the set of all eligible pins over all groups of connections.

Define ILP variables: $X_{g,ep^L,ep^R} \in \{0,1\};\ \forall\ g \in G, \forall\ ep^L \in E_g^L, \forall\ ep^R \in E_g^R$. If $X_{g,ep^L,ep^R} = 1$, this means that a connection within *g* will connect between die pin ep^L and die pin ep^R. In Figure 21 above, each binary variable X_{g,ep^L,ep^R} is represented by a green line connecting eligible pins from each connected die in group *g*.

To more concisely describe the ILP constraints, define conflict sets CS_i to contain all ILP variables that assign a single-ended or differential (if applicable) connection to die pin ep_i:

$$CS_i := \bigcup_{\substack{g \\ ep_i \in E_g^L}} \bigcup_{ep^R \in E_g^R} X_{g,ep_i,ep^R} \cup \bigcup_{\substack{g \\ ep_i \in E_g^R}} \bigcup_{ep^L \in E_g^L} X_{g,ep^L,ep_i}$$

$$\cup \bigcup_{\substack{g \\ ep_i \in E_g^L \\ D_g = 1}} \bigcup_{ep^R \in E_g^R} X_{g,ep_i^D,ep^R} \cup \bigcup_{\substack{g \\ ep_i \in E_g^R \\ D_g = 1}} \bigcup_{ep^L \in E_g^L} X_{g,ep^L,ep_i^D}$$

Define $C_{epi} \in \mathbb{R}^2$ to be the absolute coordinates of the die pin ep_i.

Define round(*y*) to round *y* to *r* significant digits, where *r* is parameterized and defaults to 6.

Minimize

$$\sum_g (1 + D_g) \sum_{ep^L \in E_g^L} \sum_{ep^R \in E_g^R} X_{g,ep^L,ep^R} \cdot round\left(\left\| C_{ep^L} - C_{ep^R} \right\|_2\right)$$

such that

$$\sum_{x \in CS_i} x \leq 1 \ \forall ep_i \in \bigcup_g E_g^L \cup E_g^R$$

$$\sum_{ep^L \in E_g^L} \sum_{ep^R \in E_g^R} X_{g,ep^L,ep^R} = |g|; \ \forall g \in G$$

That is, minimize the total wirelength (Euclidean distance between two connected pins, over all connections), while restricting at most one connection (single-ended or differential) be made to each eligible pin, and exactly as many connections per group as required.

We found that the layouts produced by our simultaneous placement and pin assignment algorithm already had pin assignments allowing for near-minimum total wirelength. Our ILP formulation did improve the total wirelength by a small factor; this illustrates the effectiveness of performing pin assignment simultaneously with die placement.

4.2 Evaluation and Results

We scoured the available literature for any available information on existing, commercial die designs. We need realistic die dimensions (in particular, microbump pitch) and pinout information defining the function for each die pin. We found that a reasonable scale for microbump pitch was 40 microns and for interposer bumps 200 microns [12]. Based on conversations with industry experts, we determined that the pinout of a large device at the die level is very similar to the pinout of that device once packaged. In particular, a packaged FPGA IC has similar pinout to the FPGA die before packaging, simply from the perspective of limiting

wire length and crisscrossing within the package itself. We also were provided die pin arrangements for existing, commercial ADC and DAC dies. These example die models are shown in Figure 22.

4.2.1 Simultaneous Placement and Pin Assignment Algorithm Variations

We experimented with different simulated annealing cooling schedules and perturbation constraints in order to further improve our generated layouts' quality. As described (for 2D layouts) in section 3.1.3, one perturbation type moves a component elsewhere on the interposer, within a window about the component's current position. We evaluated two methods of shrinking this window based on the schedule in [52]. Both methods scale this window by the factor $(1 - 0.44 + R_{accept})$, with limits clamped to reasonable minimum and maximum distances. The window is updated once per SA temperature. Our two methods differ in computing R_{accept}: we use the overall acceptance ratio (total perturbations accepted vs. total perturbations attempted), or we use the average acceptance ratio (average of each perturbation type's acceptance ratio). We found that using the average acceptance ratio decreased overall area costs.

Table VIII: Simulated annealing cooling schedule [52]

$R_{accept} > 0.96$	$T = 0.5 \cdot T_{old}$
$0.8 < R_{accept} \leq 0.96$	$T = 0.9 \cdot T_{old}$
$0.15 < R_{accept} \leq 0.8$	$T = 0.95 \cdot T_{old}$
$R_{accept} < 0.15$	$T = 0.8 \cdot T_{old}$

We also evaluated four cooling schedules based on [52]. We compared the results of using 0.15 as the lowest R_{accept} threshold (last line of Table VIII) vs. using 0.10 as the lowest R_{accept}

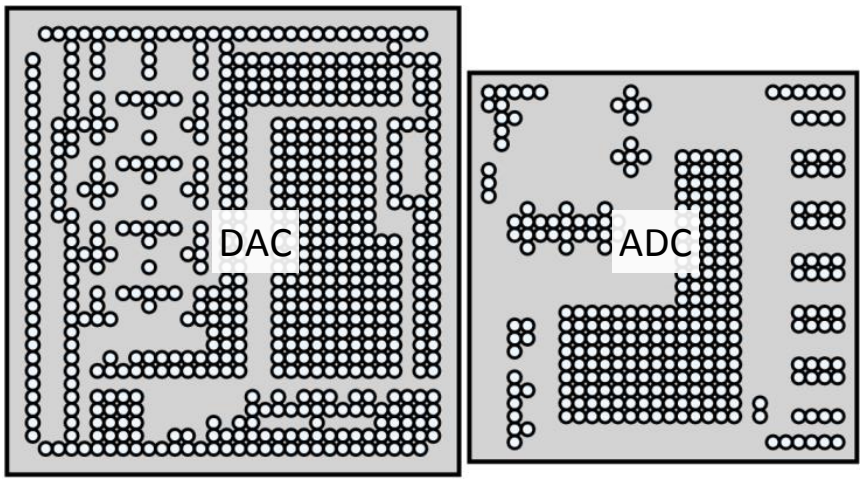


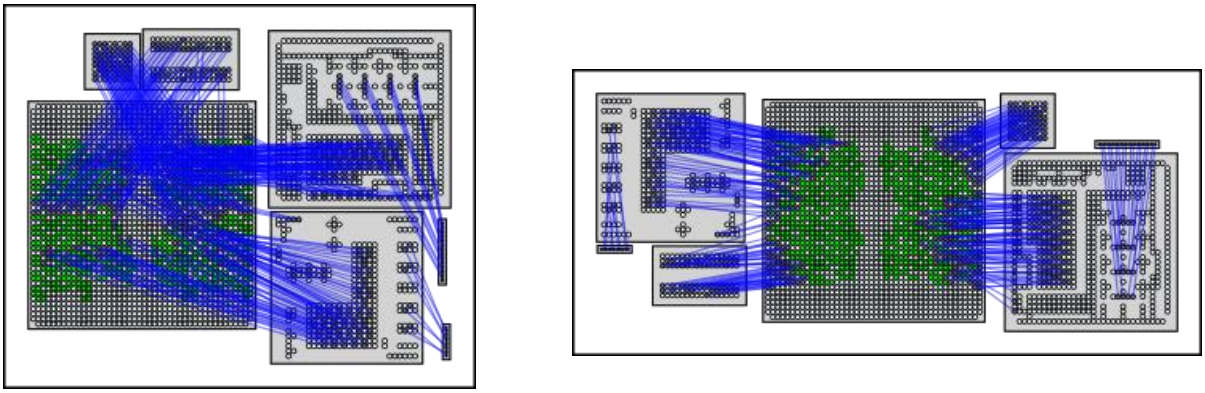
Figure 22: Die models for commercial ADC and DAC dies

threshold. We also compared the results of computing R_{accept} as the overall acceptance ratio vs. the average acceptance ratio. We test all four combinations of these two factors. Using the average acceptance ratio resulted in layouts with slightly reduced area, and using 0.10 as the lowest R_{accept} threshold resulted in layouts with reduced wirelength.

4.2.2 Sequential vs. Simultaneous Placement and Pin Assignment

Using our realistic die models, we defined example 2.5D systems for evaluating our methods. We used our framework to automatically generate layouts for systems with a variety of combinations of FPGA, ADC, DAC, Flash, and DRAM dies. In doing so, we showed that our methods can produce reasonable layouts that meet interrelated constraints. In these examples, we specified different I/O levels for connections, different numbers of dies, and different sizes of dies.

One example system we defined is a five-die system consisting of a modern FPGA, ADC, DAC, Flash, and DRAM dies. This system featured analog and digital differential I/O, and components varying in size from 4.84 mm^2 to 81 mm^2. Figure 23 shows two layouts for this system. The one shown on the left was generated by performing placement before pin assignment using a method similar to what was presented in section 3.1.3. That method performed bank assignment, but not pin assignment, during placement. Thus wirelength estimates during the placement process are based on assuming that the endpoint of a connection is located at the center of mass of pins eligible for that endpoint within the endpoint's assigned bank. The layout on the right was generated by performing placement simultaneously with actual pin (not just bank) assignment. This allows for more accurate wirelength estimates during the placement process. The costs for these layouts are shown in Table IX. Because of the higher accuracy in wirelength estimates, our simultaneous placement and pin assignment algorithm produced significantly lower final wirelength (1197.81 vs 2452.51) and marginally higher area (226.416 vs. 220.906) than performing placement followed by pin assignment.



a. Layout generated by performing placement before pin assignment

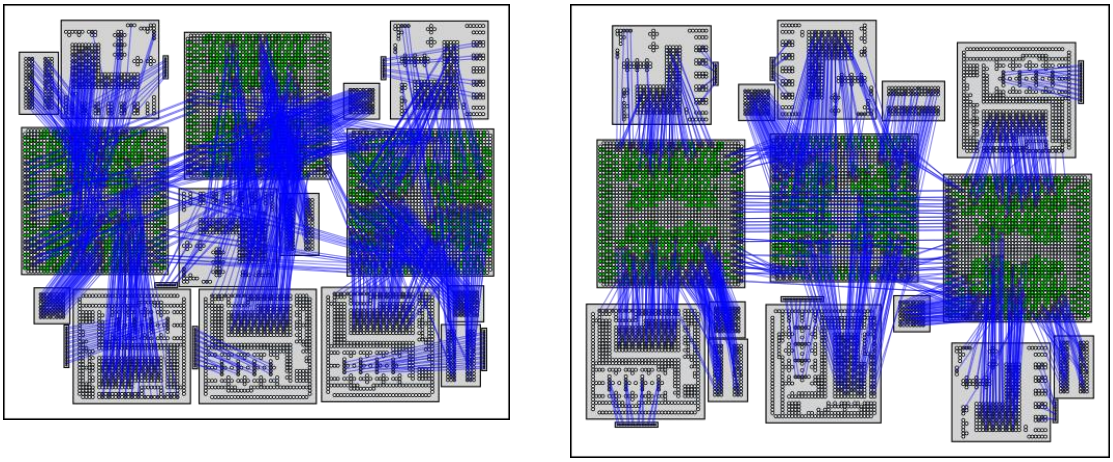
b. Layout generated by performing placement simultaneously with pin assignment

Figure 23: Automatically-generated layouts for five-die system

Table IX: Costs for five-die system

Method	Wirelength (1-norm)	Wirelength (2-norm)	Area
Sequential	2452.51	1959.22	220.906
Simultaneous	1197.81	1053.65	226.416

A larger example consisted of three copies of the five-die system, with the FPGAs connected in a ring of high-speed, differential connections. In addition to the features of each five-die system, this example also highlighted our methods' flexibility in connectivity requirements. We specified that any gigabit transmit differential pairs on one FPGA die were eligible to connect to any gigabit receive differential pairs on the connected FPGA dies. We made the same specification for the other dies in order to produce a high-speed, full-duplex, differential ring of connectivity. Figure 24 shows two layouts for this system: one generated by performing placement before pin assignment, and the other generated by performing placement simultaneously with pin assignment. The costs for these layouts are shown in Table X.



a. Layout generated by performing placement before pin assignment

b. Layout generated by performing placement simultaneously with pin assignment

Figure 24: Automatically-generated layouts for ring system

Table X: Costs for ring system

Method	Wirelength (1-norm)	Wirelength (2-norm)	Area
Sequential	9103.01	7565.65	683.390
Simultaneous	5470.91	4788.90	791.921

Here again we see that simultaneously placing dies and assigning pins allows for marked improvements in wirelength with moderately small area increases. Wirelength decreased from 9103.01 to 5470.91 (40% reduction), while area increased from 683.390 to 791.921 (16% increase). Our work allows designers to explore this trade-off between wirelength and area. A 40% decrease in wirelength may be worth only a 16% increase in area, depending on the specific design priorities. The decrease in wirelength could also lead to a decrease in the number of metal layers required to route the design. This area vs. metal layer count trade-off is an important design consideration; we examine this aspect in the next steps of our work.

4.3 Summary

Our automated 2.5D design algorithm enables the co-optimization of die placement and pin assignment by considering both of these design aspects simultaneously. We observe improvements in solution quality – significant decreases in wirelength with modest area increases – as compared to considering die placement and pin assignment independently. As we discuss in later chapters, we continue to improve our simultaneous placement and pin assignment algorithm by considering even more design aspects within the procedures.

5 Routability-Driven Placement and Pin Assignment

After our initial 2.5D design automation work, we extended our simultaneous placement and pin assignment algorithm to incorporate routability. Although this work could have targeted both PCB and 2.5D layouts, we focused on 2.5D IC design. Our goal is to reduce interposer costs by requiring less area and fewer metal routing layers in the interposer. According to Liu, et al, "[H]ow to achieve 100% routing completion rate in an interposer using a minimum number of metal layers plays a key role for the success of a 2.5D IC" [39]. A layout with poor placement or pin assignment can require more routing layers than a good layout requires; therefore we want to produce layouts with placements and pin assignments that are well-suited to routing to reduce the number of metal layers. This chapter describes our efforts toward adding this capability.

In order to produce layouts mindful of routing issues, we need some method to estimate how conducive a particular layout will be for routing the required connections. We present different methods of improving routability and compare them for suitability of use in this context. Especially important for our work is the trade-off between runtime and accuracy of the routability metrics. Because we incorporate measures of routability within our simultaneous placement and pin assignment algorithm, the routability metrics we use must be fast enough to compute within some loop of operations in our algorithm. As part of our evaluation, we explore this trade-off of runtime vs. accuracy.

5.1 Methodology

This chapter describes our expansion of our previous work to improve automatically-generated designs by considering manufacturing cost. In particular, we desire to decrease the number of metal layers required to route inter-die nets in 2.5D IC designs. By reducing the number of metal layers, we can directly reduce the design's manufacturing costs.

In section 5.1.1 we present our methodology for extending our simultaneous die placement and pin assignment algorithm to consider estimated routability. As part of this process, we must predict each net's route through the 2.5D interposer's metal layers; we discuss methods for estimating these routes in section 5.1.2. After performing our routability-aware placement and pin assignment algorithm, we continue the design process by minimizing the metal layers and routing all nets in 3D, as described in section 5.1.3. In section 5.1.4, we perform a sweep of possible parameter values that could be used in our metal layer minimization and 3D global routing methods: the granularity of the routing grid and the number of candidate routes considered for each net. Based on the results of 3D global routing and metal layer minimization from section 5.1.3, and guided by the results of section 5.1.4, we reassign flexible pins with the goal of further reducing the number of required metal layers (section 5.1.5). Finally, in section 5.1.6, we apply metal layer minimization and 3D global routing to the new pin assignment.

5.1.1 Integrating Routing Utilization-Based Cost

We extend our simultaneous die placement and pin assignment algorithm to reduce manufacturing costs by including measures of routability in the simulated annealing cost

function. In this section, we present a method of estimating routing utilization as a first step toward evaluating routability, which we improve upon in later sections. We first divide a layout into a 2D grid of equal-size rectangular “bins”, and create a 2D utilization matrix corresponding to these locations. Each entry in the matrix is initialized to zero. Then we route each net on this grid and increment by one (or two, for differential nets) each matrix location corresponding to a grid bin through which the net passes. These routes correspond directly to the 3D routes in the 2.5D interposer’s metal layers, but projected onto a 2D grid. As a result, we obtain a map of global routing utilization over our layout. As routing becomes more congested (i.e., as routing utilization increases), the design likely will require more metal routing layers. We experiment with various routing estimation methods, as described in section 5.1.2 (page 67). We later perform actual 3D global routing and metal layer minimization, as described in section 5.1.3.

To support our goal of improving routability, we use two metrics for estimating routing demand: the maximum routing utilization over all bins, and the standard deviation (or upside deviation, which considers only utilization values that are above the average) of utilization over all bins. Thus we can attempt to (1) reduce the maximum routing utilization in a layout, and (2) more evenly distribute routing utilization across a layout to decrease highly congested regions. To illustrate the utility of these metrics, consider the utilization maps in Figure 25. Maximum utilization alone does not always properly differentiate between layouts having routes more or less spread out across the layouts. Figure 25a and Figure 25b both have a maximum utilization of 3, but Figure 25b’s routing is more spread out across the layout. The standard deviation of utilization properly identifies Figure 25b as having routing more evenly spread out, though

standard deviation alone does not always properly identify layouts with lower routing utilization. Figure 25b and Figure 25c have the same standard deviation of utilization, but Figure 25c has a lower maximum utilization. By considering both total routing utilization and standard deviation (or upside deviation) of utilization, we can capture a clearer picture of a layout's routing qualities.

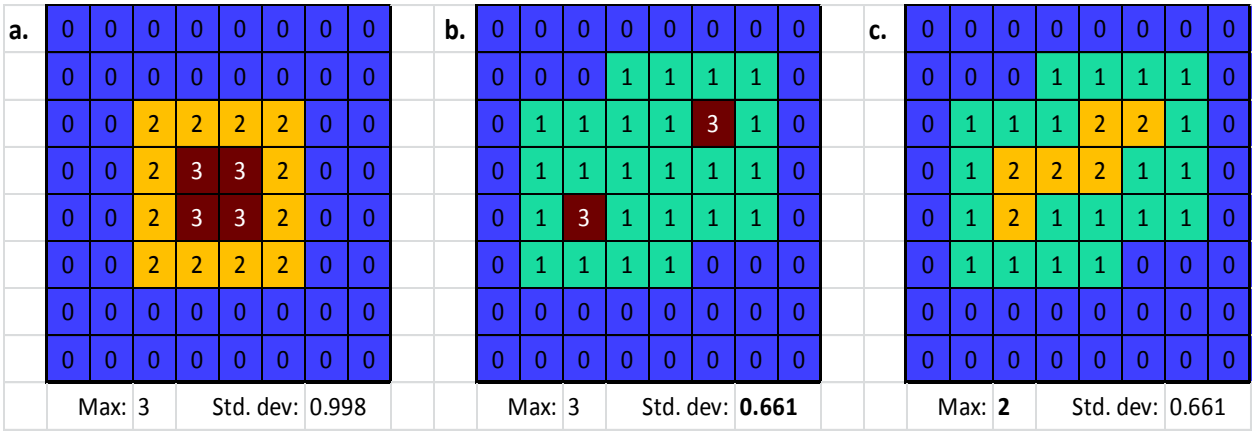


Figure 25: Routing Utilization Metrics Illustration

Next, we integrate this utilization estimation method into our simultaneous placement and pin assignment algorithm from section 4.1.1. After each layout perturbation, we update a utilization map and compute the maximum and standard (or upside) deviation of routing utilization. We add weighted terms to our simulated annealing cost function for these metrics. The default weights for the maximum and standard deviation are $\frac{10}{\#\ nets}$ and $\frac{40}{\#\ nets}$, respectively, which bring the weighted costs to approximately the same range as the other cost terms described in section 4.1.1.

Accordingly, we simultaneously place dies and assign pins while minimizing a linear combination of the following cost and penalty factors, where items 4 and 5 have been introduced specifically for routability-aware design optimization:

1. Average half-perimeter wirelength over all connections
2. Maximum half-perimeter wirelength over all connections
3. Area of bounding box containing all dies
4. **Maximum routing utilization across 2D grid**
5. **Standard (or upside) deviation of routing utilization across 2D grid**
6. Total interposer dimensions in excess of specified maximum
7. Total overlapping area between all pairs of dies
8. Number of pins assigned to connections incompatible with the assigned I/O bank power supply voltages
9. Number of connections over-subscribed to each pin (or differential pair)

We explore different methods for estimating net routing, which are used to evaluate routing utilization during simultaneous die placement and pin assignment. We discuss these methods in the following section.

5.1.2 Routing Estimation Methods

We test various methods for routing nets during simultaneous placement and pin assignment. These routing methods estimate the projection of the actual (3D) routes onto a 2D grid used in our simultaneous placement and pin assignment algorithm. Performing 3D global routing at

every step of our placement and pin assignment algorithm would be infeasible, as runtime would be unacceptably long. Instead, we estimate the 2D projection of each net's route in order to guide placement and pin assignment as described in section 5.1.2. After finishing placement and pin assignment, we perform metal layer minimization and 3D global routing (section 5.1.3) based on the resulting layout.

The first routing estimation method involves marking each bin in a straight line between terminals, using Bresenham's algorithm [58] for plotting this line. The second assumes a fixed "L" shape for each net between the source and sink. The third method chooses a route randomly from the set of pattern routes (defined below). The fourth method computes the lowest-cost pattern for each net based on the nets that already have been routed. The lowest-cost pattern is the one that contributes the least to the maximum and deviation of routing utilization. The final method finds the lowest-cost path between the net terminals based on the previously-routed nets, allowing for each net to be routed using any shape.We discuss these five routing methods in detail below, and we evaluate the effectiveness of the main routing methods in section 5.2.2.

Bresenham's Line Algorithm

This basic method plots a straight line from the net's source to the net's sink [58], and marks each 2D grid bin that the line passes through. This is a fast algorithm, but does not accurately reflect paths that actually will be used in routing. We initially explored this routing method, but we later replaced it with the more accurate methods described below.

Fixed-L Routing

This routing estimation method assumes that each net will be routed in an L-shaped pattern with one horizontal segment leaving the net's source followed by one vertical segment continuing to the net's sink. This method is likely to be very fast at the expense of some inaccuracy; the final path may differ from the predicted "L" shape.

Random-Pattern Routing

For each net, we construct a series of candidate pattern routes: L-shaped routes and Z-shaped routes [59], and U-shaped routes—which are L-shaped routes with detours. Figure 26 shows candidate routes for an example net, with the U-shaped routes along the bottom. The "depth" of the U-shaped routes (i.e., the length of the detour) is parameterized and defaults to 2 grid bin lengths.

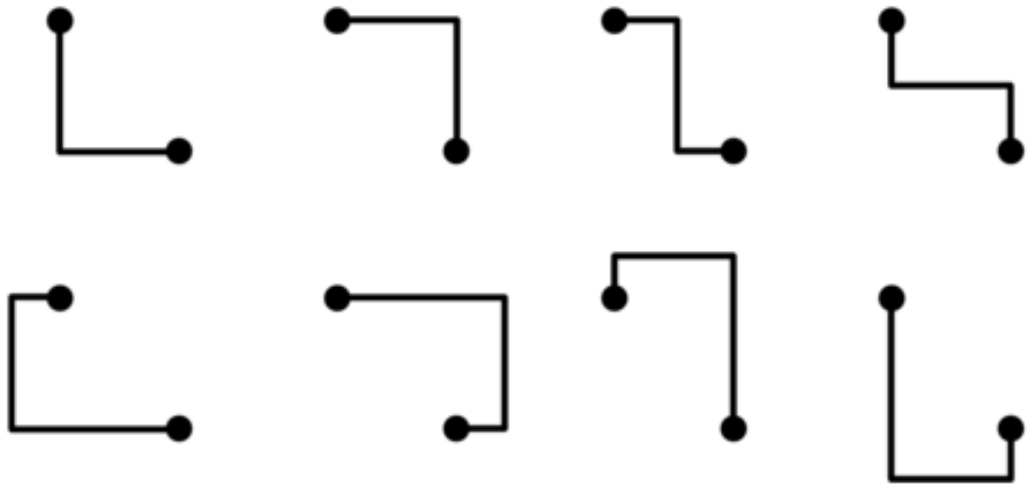


Figure 26: Candidate routes for an example net

For random-pattern routing, we construct these candidate patterns, but we omit any pattern that goes outside the layout region. We then randomly choose one of the remaining candidate patterns.

Best-Pattern Routing

In this method of route estimation, we iterate over all nets in random order. For each net we construct the same candidate routes as in random-pattern routing. We compute the total utilization and standard (or upside) deviation of utilization over all 2D grid bins that each candidate route passes through, based on the nets that so far have estimated routes. We then choose the candidate route that produces the least total utilization and deviation of utilization, and save the resulting layout and utilization matrix. In this way, we can quickly take into account other nets' routes when choosing a candidate route. When routing *all* nets, this is a greedy algorithm that minimizes routing utilization based on the nets that have been routed so far. However, when rerouting a single net (as when reassigning a net to a different flexible pin), this method does not affect other nets' routes. This may not find the *globally* optimal set of routes, but it allows for fast processing during placement and pin assignment.

Best-Path Routing

Instead of using pattern routing as in best-pattern routing, this method finds the lowest-cost path through the 2D routing grid based on minimizing the overlap with previously-routed nets. We use the A* search algorithm [60] to find the global route that has the lowest utilization

overlap with prior nets. As in best-pattern routing, this method may not find globally-optimal routes, but it allows for incremental routing updates during placement and pin assignment.

5.1.3 2.5D Metal Layer Minimization and 3D Global Routing

The techniques we presented in section 5.1.1 and 5.1.2 use 2D projections for all routes in the IC's interposer. These approximations allow for iterative refinement of die placement and pin assignment, but do not necessarily indicate the number of metal layers required to route all connections through the 2.5D IC's interposer. Restrictions in metal layer assignment may result in additional metal layers beyond what the 2D projections can predict. Therefore we extend our work to cover more of the IC design process: 3D global routing and direct metal layer minimization.

The cost of manufacturing an interposer in a 2.5D IC directly relates to the number of metal layers within it [39]. Therefore, CAD techniques that can reduce the number of metal layers can significantly help reduce interposer cost. To that end, we propose a simultaneous metal layer planning and 3D global routing method to minimize the number of metal layers required to route a given interposer. We present the problem formulation and an alternate but equivalent formulation. We employ this metal layer planning and 3D global routing method to further reduce the number of metal layers by exploiting component dies' flexible interconnect, as we discuss in section 5.1.5.

Problem Formulation

Our metal layer minimization and 3D global routing method takes as input:

- Position and rotation of each component die
- Pin assignments (and pin positions) for each inter-die connection

Our method minimizes the number of metal routing layers and simultaneously generates 3D routes and assigns metal layers for all inter-die connections.

First, we divide the interposer area into a grid of "bins" having uniform size. Then, for each inter-die net *i*, we generate a set T_i of 3D candidate pattern routes on each adjacent pair of metal layers (one horizontal layer, and one vertical). This approach assumes 2-pin nets; multi-pin nets can be broken down into 2-pin subnets. We include L-shaped routes, Z-shaped routes, and U-shaped routes as described in section 5.1.2 and shown in Figure 26 (page 69).

Next, we simultaneously minimize the number of metal layers and perform 3D global routing using the ILP formulation described below. We thus find the minimum number of metal layers required to route all connections within the interposer, and the corresponding 3D routes, given the original pin assignments. The problem is defined by the following:

- a set N of nets with flexible interconnect
- a set T_i candidate routes for each net $i \in N$, which include L-, Z-, and U-shaped routes on each adjacent pair of metal layers (e.g., Figure 27 shows the candidate routes generated for one L-shaped routing pattern on each adjacent pair of metal layers)
- a set E of 3D grid edges, which are the boundaries between adjacent grid "bins" on each metal layer

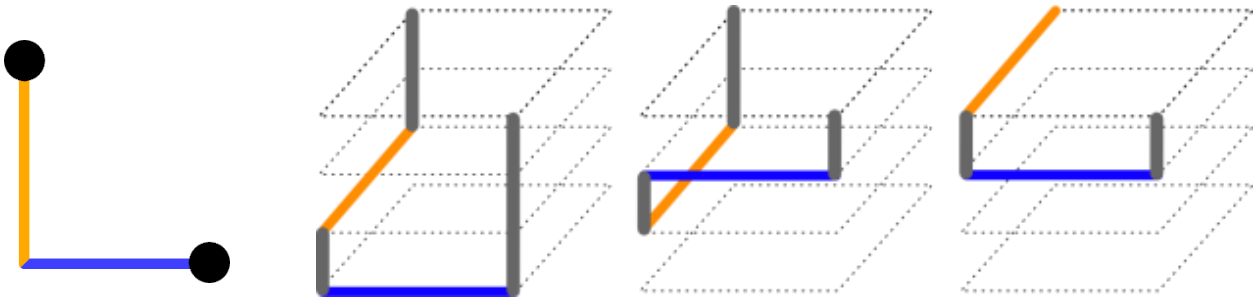


Figure 27: L-shaped routing pattern and generated 3D candidate routes

- a capacity limit C_e for each 3D grid edge $e \in E$; C_e is defined according to the bin size and manufacturing constraints on how densely wires can be routed in the 2.5D interposer
- an upper limit R for the number of metal layers, to restrict the ILP formulation to a finite number of variables

Based on the above, we generate an integer linear programming (ILP) formulation to simultaneously minimize the number of metal layers and assign routes and layers to all nets. For each grid edge, the edge overflow is defined to be the number of assigned routes passing through that edge, in excess of the edge's capacity. Whereas previous work minimizes total grid edge overflow, our formulation **directly** minimizes the number of metal layers while ensuring no edge overflow.

For each 3D grid edge $e \in E$, we define $t \ni e$ to mean that the candidate route t passes through grid edge e. For each net $i \in N$, for each candidate route $t \in T_i$, we define a binary variable:

$$X_{it} = \begin{cases} 1, & net\ i\ is\ assigned\ route\ t; \\ 0, & otherwise \end{cases}$$

For each metal layer $l = 1, \dots, R$, we define a binary variable:

$$R_l = \begin{cases} 1, & layer\ l\ is\ assigned\ to\ be\ used\ for\ routing; \\ 0, & otherwise \end{cases}$$

The ILP formulation is then given by:

$$\min \sum_{l=1}^{R} k^{l-1} \times R_l$$

$$\sum_{i \in N} \sum_{t \ni e} x_{it} \leq C_e \quad \forall e \in E$$

$$\sum_{t \in T_i} x_{it} = 1 \quad \forall i \in N$$

$$x_{it} \leq R_l \quad \forall l \in \{1, \dots, R\} \quad \forall i \in N \mid if\ t \ni e\ and\ e\ on\ layer\ l$$

In the above formulation, the objective expression minimizes the number of metal layers, where $k > 1$ is an arbitrary constant to ensure that lower-numbered metal layers are used first by assigning a higher penalty when using the higher metal layers. The first constraint restricts the number of routes through a grid edge to be less than or equal to the edge's capacity. Each net i is required to have only one route t, and each route t is permitted only if all layer(s) containing t are assigned to be used. The above formulation always generates a feasible routing solution. By solving this ILP formulation, we obtain the minimum number of metal layers and the corresponding route and layer assignments for all nets.

As mentioned before, R needs to be an upper bound on the number of metal layers to ensure a feasible solution is found when solving the ILP. To find an upper bound, one way is to visit each net sequentially. Each considered net will be routed using one of its candidate routes (as described before) with the fewest number of metal layers as identified up to that point. R will then be the number of metal layers after the last net is visited. This procedure is a very fast heuristic which generates a routable solution but will most likely result in a higher number of metal layers than the one found after solving the ILP formulation.

Alternate Problem Formulation

We present an alternate formulation that is equivalent to solving the ILP above but that produces results more quickly. The direct ILP formulation above is difficult for our ILP solver GLPK [56] to complete in a reasonable timeframe, likely due to the constraint interactions between the many candidate route variables x_{it} and the few routing layer eligibility variables R_l. To achieve our goal of minimizing metal layers and performing 3D routing, we implemented an equivalent method.

We first compute an upper-bound for the number of required metal layers. We assign random candidate routes to each net and determine the number of layers that provide enough routing capacity for these routes. This overestimates the number of layers required, but it runs very quickly.

We solve an alternate ILP formulation that minimizes the total 3D grid edge overflow, given a fixed number of metal layers. We decrease the number of metal layers by one, until our ILP

formulation reports a nonzero total 3D grid edge overflow. At that point, the previous result is returned, as the previous result reported no 3D grid edge overflow.

The alternate ILP formulation uses the same variable definitions as in the original formulation, except we exclude routing layer eligibility variables R_l but define additional integer variables. For each 3D grid edge $e \in E$, we define O_e to be the routing overflow for that edge. That is, O_e equals the number of routes passing through edge e in excess of that edge's capacity.

The ILP formulation is then given by:

$$\min \sum_{e \in E} O_e$$

$$\sum_{i \in N} \sum_{t \ni e} x_{it} \leq C_e + O_e \quad \forall e \in E$$

$$\sum_{t \in T_i} x_{it} = 1 \quad \forall i \in N$$

$$O_e \geq 0 \; \forall \; e \in E$$

In the above formulation, the objective expression minimizes the total 3D grid edge overflow. The first constraint restricts the number of routes through a grid edge to be less than or equal to the edge's capacity plus the overflow variable for that edge. This constraint, combined with the constraint that each overflow variable be nonnegative, causes each overflow variable to reflect the actual routing overflow. Without the nonnegative constraint, the total 3D grid edge overflow would reduce to a coarse measure of wirelength instead of a useful evaluation of routability. Each net i is required to have only one route t. By solving this ILP formulation, we

obtain the minimum total 3D grid edge overflow and the corresponding route and layer assignments for all nets. If the total 3D grid edge overflow is zero, then the connections are routable in the given number of metal layers.

5.1.4 Varying 3D Grid Size and Number of Candidate Routes

We experiment with varying the grid size for 3D routing and with varying the number of candidate routes considered. Both of these parameters affect the trade-off of accuracy of routing estimates vs. processing time. We use different values for grid size and for the number of candidate routes, in order to explore this trade-off. We discuss these results in section 5.2.6. To reduce the number of candidate routes to consider, we employ a heuristic method of route pruning.

Reducing Candidate Routes

We divide the interposer area into a 2D grid and initialized all the 2D grid edges' utilizations to zero. We enumerate a complete set of candidate routing patterns to consider. For each candidate pattern, we compute the total 2D grid edge utilization over all edges through which the pattern goes through, and we also compute the maximum 2D grid edge utilization over these edges. After generating these patterns, we sort the patterns by maximum edge utilization, then by total edge utilization. We then choose the N candidate routing patterns with the lowest maximum and total edge utilization.

5.1.5 2.5D Pin Reassignment

After obtaining the route and layer assignments for all nets, we leverage the component dies' flexible interconnect to reassign pins so that inter-die connections can be routed through less-congested areas of the interposer. This in turn enables a reduction in the number of required metal layers in the interposer. To our knowledge, this is a unique approach to reducing 2.5D IC interposer metal layer count by exploiting flexible interconnect pin reassignment. Because FPGAs have many I/O pins suitable for a wide variety of connections, and thus a net can be reassigned to different pins, FPGA dies provide ideal candidates for this pin reassignment stage.

We explore four different methods of choosing new pin assignments. In each method, we repeatedly reassign pins such that the new assignments decrease some measure of cost. The methods of evaluating cost are:

1. *Layer goal* method: define cost as total overflow, given a "goal" number of metal layers.
2. *Max utilization*: define cost as the maximum utilization over all edges.
3. *Max-stddev*: define cost as the maximum utilization over all edges and the standard deviation of edge utilization.
4. *Max-upside-deviation*: define cost as the maximum utilization over all edges and the upside deviation of edge utilization. As mentioned previously, the upside deviation is computed as for the standard deviation, except only data points above the mean are considered.

Each of these cost measures we explain in the following sections.

Problem Formulation and Layer Goal Method

For this stage, we again divide the interposer area into a 2D grid of uniform global cells of the same number, size and locations as the metal layer planning and 3D global routing stage (section 5.1.3). We then project the 3D routes created in the previous stage onto this 2D grid to compute the number of routes passing through each 2D grid edge. We use these measures of grid edge utilization to guide pin assignments and routes to less crowded interposer areas. We define a "goal" number of metal layers R^G of metal layers for this stage, which is computed as shown below based on the number of metal layers from the previous 3D global routing stage, which we call R^{3D}.

$$R^G = round(\alpha \times R^{3D})$$

In this equation for R^G, α is an arbitrary constant $0 < \alpha < 1$. Setting $\alpha = 0.5$ provides a reasonable target for reducing the number of metal layers. Interposer metal layers typically alternate between horizontal and vertical orientations; we restrict $R^G \geq 2$ because nontrivial routes are composed of both vertical and horizontal segments. We also want to reduce the number of metal layers by at least one horizontal and one vertical layer; therefore we also restrict $R^G \leq (R^{3D} - 2)$.

The 2D grid edge capacity limits are computed for horizontal grid edges and vertical grid edges based on the 3D grid edge capacity limits from the previous stage. We define for each metal layer l the layer capacity limit $C_e^l = c_e \ \ \forall e \in l$. We then define the 2D grid edge capacity limits:

$$C_e^{2D,h} = \sum_{l=1 \,\&\, l\ horizonal}^{R^G} C_e^l$$

$$C_e^{2D,v} = \sum_{l=1 \,\&\, l\ vertical}^{R^G} C_e^l$$

An exhaustive search of the solution space of all possible pin assignments would be prohibitively slow; 3D routing would have to be performed for each pin assignment in order to find a globally optimal solution. Thus, in order to quickly explore this solution space, we present a heuristic method of reassigning flexible pins (Figure 28) in order to reduce the total overflow, which we will define shortly. We assign new pins and choose new candidate routes for all flexible nets as long as these reassignments reduce the total overflow in the 2D grid. This search terminates once no improvements have been made in one pass over all flexible nets.

```
do:
  improved := false
  Randomly shuffle N^F
  For each net i ∈ N^F :
    Choose flexible pin p and candidate route t_i ∈ T_i^p such that:
      Pin p is not assigned to any net, or is already assigned to
        the current net i
      Pin p is within a bank that is compatible with net i
      Route t_i ∈ T_i^p minimizes total overflow in the 2D grid
    If new p, t_i were chosen:
      improved := true
      Update the 2D grid utilization according to choosing p, t_i
while improved is true
```

Figure 28: Pin reassignment heuristic

Total overflow is defined as the total number of nets routed through each 2D grid edge in excess of the edge's capacity (as determined using the equations above based on our goal number of metal layers). We define the set N^F of all flexible inter-die nets (e.g., nets connecting to I/O pins on FPGA dies). We define the set T_i^p of candidate routes for net i for the case when it is assigned to flexible pin p. These candidate routes are the same patterns used in the metal layer planning and 3D global routing stage (section 5.1.3), but mapped to a 2D grid instead of 3D.

When assigning pins on an FPGA die, the designer (or CAD algorithms) must consider the I/O power supplies required by each connection. FPGA I/O pins are typically grouped into "banks," each of which has its own power supply voltages. Accordingly, connections assigned to pins in the same FPGA I/O bank must support compatible power supplies. To accommodate these power supply requirements, we allow a net i to connect to a pin p only if p is within a bank whose power supplies are compatible with net i. If a net is assigned to a pin whose bank currently has no power supplies assigned, the power supply is assigned such that the bank is compatible with the desired net.

By performing this pin reassignment stage, we obtain pin assignments for all nets and the power supply assignments for all applicable banks of I/O pins. Our final pin assignment can produce fewer interposer metal layers for implementing all nets. We evaluate this final pin assignment as described in section 5.2.7.

Cost Measures Independent of the Number of Metal Layers

The *layer goal* method depends on choosing a goal for the number of metal layers, in order to compute the aggregate edge capacities for the projected 2D grid edges. However, the reduction in metal layers depends on this layer goal. We developed several alternative cost measures to guide pin reassignment, without depending on setting a "goal" for the number of metal layers.

When performing our pin assignment method from earlier in this section, instead of minimizing the total overflow across all 2D grid edges (which depends on the edge capacities that are calculated based on a layer goal), we can use alternative cost measures. We can minimize the maximum 2D grid edge *utilization*, which is the number of routes that pass through each edge. This is a function of the assigned routes, and it does not depend on any layer goal.

Another alternative is to minimize the maximum 2D grid edge utilization and the standard deviation of edge utilization. This allows for a more granular, less quantized, cost measure. We also experiment with replacing this *standard deviation* of edge utilization with the *upside deviation* of edge utilization, which considers only edge utilizations that are above the average. We discuss results from these experiments in section 5.2.7.

5.1.6 Metal Layer Minimization and 3D Global Routing using New Pin Assignments

The pin reassignment described in section 5.1.5 does not assign routes to metal layers; thus, we re-run the metal layer minimization and 3D global routing step described in section 5.1.3. In this final stage, we find a pattern-based 3D global routing solution using the minimum number of layers (based on our ILP formulation) for the improved pin assignment. The minimum number

of required metal layers may not be equal to the value of R^G chosen when using the *layer goal* pin reassignment method. This is due to the differences between using a heuristic in section 5.1.5 that allows non-zero overflow and performs 2D routing estimation, versus an ILP solver in section 5.1.3 that enforces a feasible (i.e., overflow-free) 3D global routing solution. If the solution found in this final stage results in fewer metal layers than the solution determined before pin re-assignment, we choose the new solution; otherwise, we return to the previous (before pin re-assignment) solution.

5.2 Evaluation and Results

In order to evaluate the effectiveness of the methods described in section 5.1, we generate many different placements and pin assignments using our routability-aware placement and pin assignment algorithm; initial results are shown in section 5.2.1. We explore our routing estimation methods and the trade-offs between runtime and routing utilization in section 5.2.2. In section 5.2.3 we ran our metal layer minimization and 3D global routing on these placements and pin assignments in order to compare the estimated routing utilization with the actual number of metal layers required. We consider using the standard deviation of routing utilization vs. using the upside deviation of routing utilization in section 5.2.4.

We evaluate our metal layer minimization and 3D global routing work in section 5.2.5 and test different values for 3D grid size and number of candidate routes considered in section 5.2.6. Finally, we present results from our 2.5D pin reassignment methods in section 5.2.7.

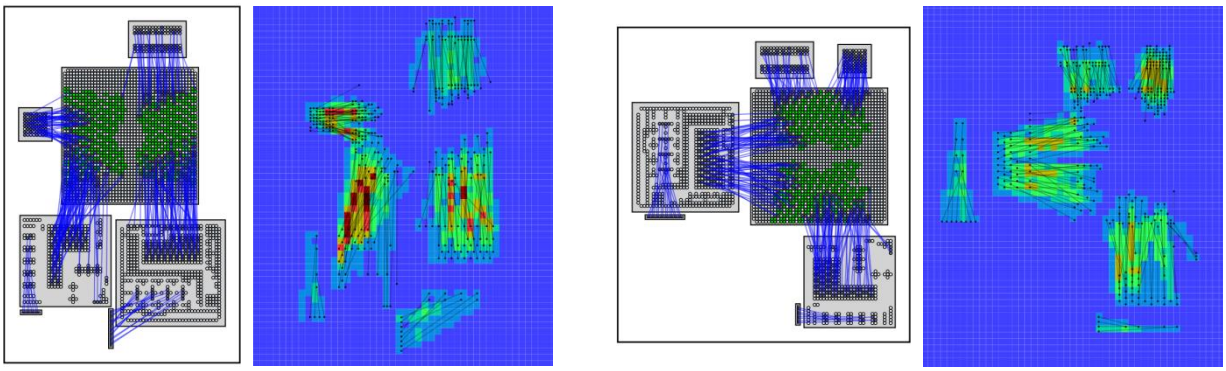
5.2.1 Incorporate Routability Cost into Placement and Pin Assignment Algorithm

We ran a series of experiments using our placement and pin assignment algorithm with integrated routing utilization estimation, on our five-die 2.5D IC benchmark from section 4.2. We used four distinct configurations and generated ten layouts for each configuration. We used the following configurations:

1. Include neither maximum nor standard deviation of utilization in SA cost function
2. Include maximum routing utilization in SA cost function
3. Include standard deviation of routing utilization in SA cost function
4. Include both maximum and standard deviation of routing utilization in SA cost function

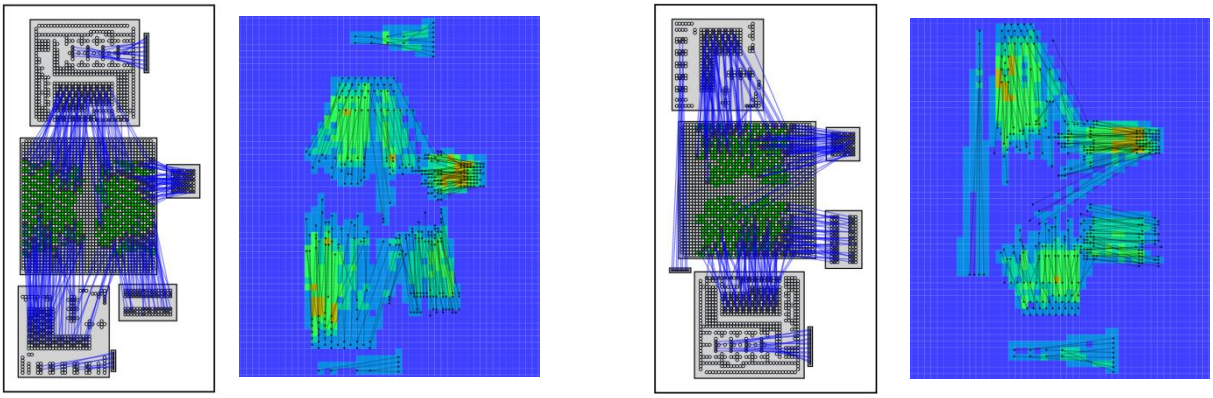
The results from these experiments are shown in Table XI, sorted by max utilization and standard deviation of utilization. Using the metrics from section 5.1.1 effectively reduced the estimated utilization as expected, and also is expected to improve the routability of the generated layouts. Figure 29a shows the top result using the configuration having neither the maximum nor the standard deviation of routing utilization in the SA cost function. Figure 29b shows the top result using the configuration having the maximum utilization in the SA cost function. Figure 29c shows the top result using the configuration having the standard deviation of routing utilization in the SA cost function. Figure 29d shows the top result using the configuration having both the maximum and the standard deviation of utilization in the SA cost function. For each result, the left image shows the component dies' positions and the selected

pins having connections indicated by blue lines. The right image shows the congestion map of that layout. All four routing utilization maps are shown using the same color scale.



a. Neither maximum nor standard

b. Maximum routing utilization



c. Standard deviation of utilization

d. Both maximum and standard deviation

Figure 29: Top layouts for utilization-driven routability estimation experiments

As shown in Figure 29, incorporating routing utilization into the simulated annealing cost function produced dramatic improvements in spreading out routes in the generated layouts. The layout in Figure 29a features multiple highly congested “hot spots” and has the highest maximum routing utilization of all four configurations. In contrast, Figure 29b–d show much lower utilization, more spread out across the layouts, with Figure 29d (the one generated using

a cost function that includes both the maximum and standard deviation of routing utilization) having the least overall utilization.

Table XI shows wirelength, area, maximum routing utilization, and standard deviation of routing utilization across the four cost function configurations. The best layout (i.e., the layout with lowest maximum routing utilization and standard deviation of utilization) had a maximum routing utilization of 22 and a standard deviation of 1.782 for the baseline configuration that did not consider routing utilization in the simulated annealing cost function. The configurations that included either maximum routing utilization or standard deviation of routing utilization (but not both) performed better. The best layouts for these had a maximum routing utilization of 14 and a standard deviation of utilization around 1.5. The best layout for the configuration including both maximum and standard deviation of routing utilization had a maximum routing utilization of 12 (a 45.5% reduction from the baseline) and a standard deviation of 1.505 (a reduction of 15.5% from the baseline). As expected, the best results for routing utilization occur when we include factors in our cost function for both the maximum routing utilization and standard deviation of utilization.

Table XI: Results for Initial Experiments of Utilization-aware Placement and Pin Assignment

Config	Seed	Wirelength (mm)	Area (mm²)	Max utilization	Std. dev. utilization
None	4	1425.8	282.2	**22**	**1.782**
None	9	1597.5	289.1	25	2.041
None	8	1512.4	249.1	26	1.999
None	1	1731.0	243.9	26	2.026
None	10	1502.7	245.8	26	2.083
None	6	1692.5	276.1	26	2.104
None	3	1371.8	246.3	27	1.849
None	2	1478.9	291.2	30	1.940
None	7	1616.7	241.3	30	2.169
None	5	1915.2	246.2	38	2.378
Config	**Seed**	**Wirelength (mm)**	**Area (mm²)**	**Max utilization**	**Std. dev. utilization**
Max	10	1216.1	333.2	**14**	**1.507**
Max	6	1312.8	332.9	**14**	1.575
Max	7	1588.0	299.6	**14**	1.710
Max	8	1489.9	268.5	**14**	1.722
Max	4	1532.4	303.7	**14**	1.738
Max	2	1663.4	262.8	**14**	1.845
Max	5	1947.3	250.9	**14**	2.009
Max	9	1640.0	288.9	16	1.915
Max	1	1718.2	256.8	18	1.992
Max	3	1893.1	290.2	18	2.151
Config	**Seed**	**Wirelength (mm)**	**Area (mm²)**	**Max utilization**	**Std. dev. utilization**
Stddev	8	1424.7	297.1	**14**	**1.499**
Stddev	2	1614.2	246.9	16	1.654
Stddev	7	1562.8	277.7	16	1.751
Stddev	5	1650.9	289.5	16	1.865
Stddev	1	1676.6	249.5	19	1.767
Stddev	9	1771.8	242.8	19	1.867
Stddev	10	1663.0	245.0	20	1.792
Stddev	3	1601.7	245.5	20	1.822
Stddev	4	1771.4	249.0	20	1.921
Stddev	6	1783.8	244.6	24	1.926
Config	**Seed**	**Wirelength (mm)**	**Area (mm²)**	**Max utilization**	**Std. dev. utilization**
Both	4	1455.2	304.2	**12**	1.505
Both	7	1460.3	303.3	**12**	1.511
Both	1	1534.9	245.9	**12**	1.630
Both	2	1838.1	252.6	**12**	1.689
Both	6	1363.5	301.7	13	**1.457**
Both	8	1404.0	298.3	13	1.539
Both	3	1501.0	341.4	14	1.574
Both	10	1471.4	295.0	14	1.666
Both	9	1708.3	243.7	16	1.870
Both	5	2007.2	257.5	16	2.041

5.2.2 Variations and Methods for Routability-Driven Placement and Pin Assignment

We performed a variety of experiments to explore the relationship between different variations and methods for our routability-driven placement and pin assignment algorithm. The variations we explored included:

A. Our "base" variation (A) was the placement and pin assignment algorithm described in section 5.1.1 above.
B. We added an "upside deviation" variation (B): a similar algorithm to (A), except we used the upside deviation instead of standard deviation of routing utilization in our simulated annealing cost function. The upside deviation is computed similarly to the standard deviation, except only values above the mean are considered in the upside deviation.
C. We added a "feasible neighborhood search" variation (C): a similar algorithm to (A), except once we reached a *feasible* layout (i.e., a layout that had all penalties equal to zero), we restarted simulated annealing. After restarting, we accepted perturbations only if the resulting layout also was feasible.

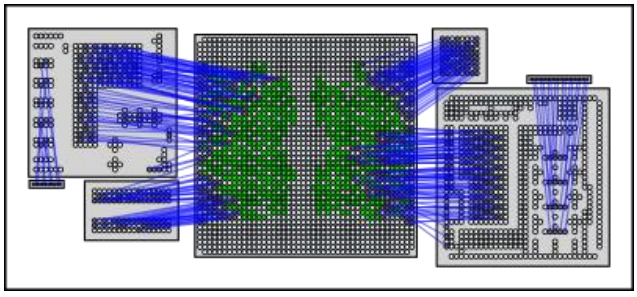
For each variation (A) – (C), we estimated routing utilization with the following methods described in section 5.1.2:

1. Fixed-L routing
2. Random-pattern routing

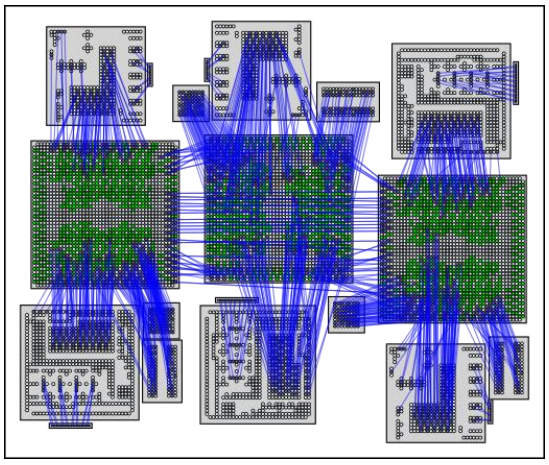
3. Best-pattern
4. Best-path routing

We ran these twelve variation/method combinations on up to four different benchmark designs. Two of these designs were described in section 4.2 (page 57): a five-die system consisting of a modern FPGA, ADC, DAC, Flash, and DRAM dies (Figure 30a, on page 90), and three copies of the five-die system, with the FPGAs connected in a ring of high-speed, differential connections (Figure 30b). Another benchmark (Figure 30c) consisted of an FPGA die connected to an analog-to-digital converter (ADC) die and a digital-to-analog converter (DAC). The other benchmark consisted of an FPGA die connected to two ADC dies and two DAC dies (Figure 30d). All four benchmarks included inter-die connections with multiple interconnect requirements (e.g., different I/O standards).

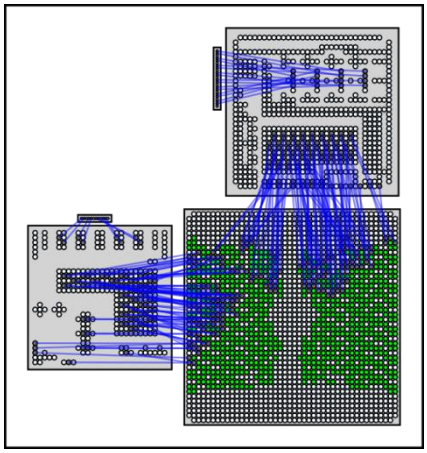
We compared the interposer area, total wirelength, area-wirelength product, routing utilization maximum, routing utilization deviation (standard or upside), and relative runtime for these experiments. The averages are shown in Table XII – Table XV below. For each benchmark and variation (A – C), the runtimes are normalized to the minimum across the routing utilization methods (1 – 4).



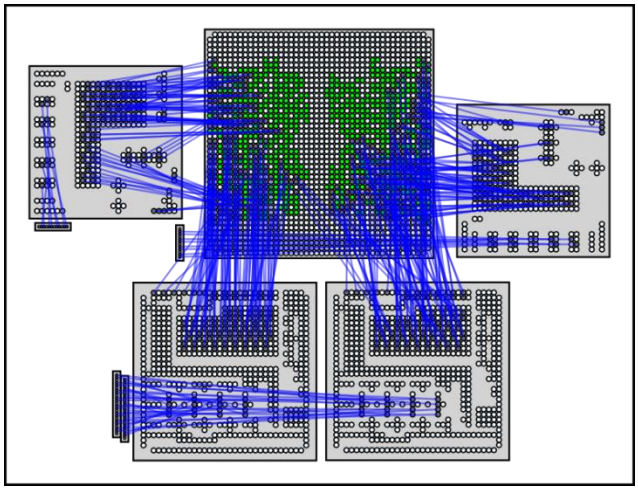
a. Example layout of FPGA+ADC+DAC+Flash+DRAM system



b. Example layout of ring system



c. Example layout of FPGA+ADC+DAC system



d. Example layout of FPGA+2xADC+2xDAC system

Figure 30: Example layouts for each tested benchmark system

Between the algorithm variations (A – C), we found that using the maximum and standard deviation of routing utilization gave comparable results, whether using the base simulated annealing algorithm (variation A) or using the feasible neighborhood search (variation C). We

found that using the maximum and upside deviation of routing utilization (variation B) gave slightly worse results than using the maximum and standard deviation (variation A), in terms of layout area and estimated wirelength.

Between the routing utilization methods (1 – 4), we found that best-pattern routing (method 3) and best-path routing (method 4) produced better area, wirelength, and area-wirelength product costs as compared to fixed-L routing (method 1) and random-pattern routing (method 2). These improved costs came at the expense of consistently longer runtimes (40% – 310% longer than the fastest method within a benchmark/variation group).

Table XII: Average results for FPGA + ADC + DAC Benchmark
In each group, the best results in each column are highlighted.

Algorithm Parameters		Area	Wirelength	A * WL	Util. Max	Util. Dev.	Time
Standard Deviation	Fixed-L	240.26	620.32	147,973	11.90	1.14	1.04
	Random-pattern	270.89	600.02	162,219	10.34	1.09	1.00
	Best-pattern	260.39	509.54	132,368	9.73	0.98	1.52
	Best-path	255.72	488.93	124,908	9.53	0.96	2.42
Upside Deviation	Fixed-L	250.30	753.69	189,038	9.03	5.01	1.05
	Random-pattern	284.44	767.59	221,279	9.47	4.10	1.00
	Best-pattern	263.30	667.89	177,072	8.73	3.83	1.50
	Best-path	256.51	636.97	163,832	8.50	3.68	2.54
Feasible Neighbor-hood	Fixed-L	240.81	618.29	147,785	11.54	1.13	1.00
	Random-pattern	271.55	564.59	153,247	9.98	1.04	1.02
	Best-pattern	261.39	516.32	134,576	9.88	1.00	1.75
	Best-path	257.27	535.28	137,002	9.47	1.01	3.74

Table XII shows the trade-offs between area, wirelength, maximum routing utilization, and deviation of routing utilization for the FPGA + ADC + DAC benchmark. For the standard

deviation “base” variation (A), we found that the best-path routing estimation method produced the lowest average maximum (9.53) and standard deviation (0.96) of routing utilization. For the upside deviation variation (B), we again found that the best-path routing estimation method produced the lowest average maximum (8.50) and standard deviation (3.68) of routing utilization. For the feasible neighborhood variation (C), we again found that the best-path routing estimation method produced the lowest average maximum (9.47) of routing utilization. The average standard deviation of routing utilization was comparable between the best-pattern estimation method (1.00) and the best-path estimation method (1.01). Between the algorithm variations (A)–(C), we found that the upside deviation variation (B) on average produced the lowest maximum routing utilization, but the highest deviation in routing utilization. Configuration methods that produced the lowest routing utilization typically incurred moderate increases in interposer area, since nets are more spread out over a larger area and are less congested.

Table XIII: Average results for FPGA + 2xADC + 2xDAC Benchmark
In each group, the best results in each column are highlighted.

Algorithm Parameters		Area	Wirelength	A * WL	Util. Max	Util. Dev.	Time
Standard Deviation	Fixed-L	416.90	1399.33	581,637	20.20	2.11	1.20
	Random-pattern	418.34	1540.93	644,927	13.83	1.90	1.00
	Best-pattern	404.81	1377.21	556,005	13.36	1.82	1.59
	Best-path	387.39	1399.60	541,720	12.60	1.78	2.67
Upside Deviation	Fixed-L	436.73	1714.27	747,329	19.67	7.98	1.19
	Random-pattern	435.30	1734.70	753,684	12.37	5.96	1.00
	Best-pattern	416.93	1531.90	639,112	12.23	5.78	1.49
	Best-path	411.55	1498.00	616,282	12.14	5.43	2.29
Feasible Neighbor-hood	Fixed-L	424.23	1353.08	572,951	20.92	2.07	1.01
	Random-pattern	418.19	1534.44	641,007	13.88	1.90	1.00
	Best-pattern	400.77	1374.33	549,251	13.33	1.80	1.64
	Best-path	392.35	1391.82	544,733	12.76	1.80	4.10

Table XIII shows the trade-offs between area, wirelength, maximum routing utilization, and deviation of routing utilization for the FPGA + 2xADC + 2xDAC benchmark. For the standard deviation “base” variation (A), we found that the best-path routing estimation method produced the lowest average maximum (12.60) and standard deviation (1.78) of routing utilization. For the upside deviation variation (B), we again found that the best-path routing estimation method produced the lowest average maximum (12.14) and standard deviation (5.43) of routing utilization. For the feasible neighborhood variation (C), we again found that the best-path routing estimation method produced the lowest average maximum (12.76) of routing utilization. The average standard deviation of routing utilization was the same (1.80) for the best-pattern estimation method and the best-path estimation method. Between the algorithm variations (A)–(C), we found that the upside deviation variation (B) on average

produced the lowest maximum routing utilization, but the highest deviation in routing utilization. Configuration methods that produced the lowest routing utilization for this benchmark incurred *decreases* in interposer area, in contrast to the earlier benchmark. This may be because including cost factors for routing utilization also encouraged layouts with smaller areas.

Table XIV: Average results for FPGA + ADC + DAC + Flash + DDR Benchmark
In each group, the best results in each column are highlighted.

Algorithm Parameters		Area	Wirelength	A * WL	Util. Max	Util. Dev.	Time
Standard Deviation	Fixed-L	289.00	1057.73	303,765	17.79	1.61	1.26
	Random-pattern	285.02	1111.39	316,244	13.27	1.49	1.00
	Best-pattern	273.76	1007.74	275,061	12.10	1.39	1.74
	Best-path	271.24	1018.20	276,198	11.93	1.39	2.02
Upside Deviation	Fixed-L	302.65	1221.47	367,541	15.47	6.80	1.21
	Random-pattern	306.95	1250.87	383,917	11.70	5.29	1.00
	Best-pattern	301.30	1115.21	334,378	11.00	5.09	1.43
	Best-path	295.51	1126.40	332,676	10.50	4.89	1.92
Feasible Neighbor-hood	Fixed-L	289.06	991.68	284,792	18.49	1.58	1.00
	Random-pattern	282.16	1119.15	315,851	12.90	1.47	1.01
	Best-pattern	281.08	1021.85	286,132	12.30	1.40	1.43
	Best-path	272.96	1018.47	277,160	11.96	1.40	2.39

Table XIII shows the trade-offs between area, wirelength, maximum routing utilization, and deviation of routing utilization for the FPGA + ADC + DAC + Flash + DDR benchmark. For the standard deviation “base” variation (A), we found that the best-path routing estimation method produced the lowest average maximum (11.93) of routing utilization. The average standard deviation of routing utilization was the same (1.39) for the best-pattern estimation method and the best-path estimation method. For the upside deviation variation (B), we again found that

the best-path routing estimation method produced the lowest average maximum (10.50) and standard deviation (4.89) of routing utilization. For the feasible neighborhood variation (C), we again found that the best-path routing estimation method produced the lowest average maximum (11.96) of routing utilization. The average standard deviation of routing utilization was the same (1.40) for the best-pattern estimation method and the best-path estimation method. Between the algorithm variations (A)–(C), we found that the upside deviation variation (B) on average produced the lowest maximum routing utilization, but the highest deviation in routing utilization. Configuration methods that produced the lowest routing utilization for this benchmark incurred *decreases* in interposer area, in contrast to the first benchmark. This may be because including cost factors for routing utilization also encouraged layouts with smaller areas.

Table XV: Average results for 3x Ring of FPGA + ADC + DAC + Flash + DDR Benchmark
In each group, the best results in each column are highlighted.

Algorithm Parameters		Area	Wirelength	A * WL	Util. Max	Util. Dev.	Time
Standard Deviation	Fixed-L	892.08	9586.67	8,542,945	183.53	12.81	1.13
	Random-pattern	996.14	9719.24	9,695,022	192.00	12.26	1.00
	Best-pattern	874.77	8485.86	7,433,895	187.59	10.97	1.41
	Best-path	877.05	9079.54	7,977,227	179.69	11.12	1.60
Upside Deviation	Fixed-L	1000.51	11145.60	11,157,703	171.73	31.46	1.08
	Random-pattern	1127.80	11793.10	13,326,572	192.00	25.90	1.00
	Best-pattern	987.23	10595.83	10,473,306	187.73	24.43	1.43
	Best-path	1008.81	10751.21	10,850,141	176.55	23.32	1.71
Feasible Neighbor-hood	Fixed-L	899.84	9278.44	8,348,712	179.44	12.69	1.16
	Random-pattern	1038.45	9735.75	10,120,990	192.00	12.21	1.00
	Best-pattern	861.60	7639.00	6,581,762	192.00	10.59	1.40

Table XIII shows the trade-offs between area, wirelength, maximum routing utilization, and deviation of routing utilization for the 3x ring of FPGA + ADC + DAC + Flash + DDR benchmark. For the standard deviation “base” variation (A), we found that the best-path routing estimation method produced the lowest average maximum (179.69) of routing utilization, whereas the best-pattern estimation method produced the lowest average standard deviation of routing utilization (10.97). For the upside deviation variation (B), we found that the fixed-L routing estimation method produced the lowest average maximum utilization (171.73), and the best-path method produced the lowest standard deviation (23.32) of routing utilization. For the feasible neighborhood variation (C), we found that the fixed-L routing estimation method produced the lowest average maximum utilization (179.44), and the best-path method produced the lowest standard deviation (10.59) of routing utilization. Between the algorithm

variations (A)–(C), we found that the maximum routing utilizations were comparable, but the upside deviation variation (B) produced highest deviation in routing utilization.

5.2.3 2.5D Metal Layer Minimization and 3D Global Routing

We performed 3D routing on automatically-generated layouts from our routability-driven simultaneous placement and pin assignment algorithm. Our 3D routing method is described in section 5.1.3. We compare the results from 3D routing with the estimated routability within placement, for multiple routability estimation methods (section 5.1.2). Our goal is to create an algorithm that automatically produces 2.5D layouts that require as few metal routing layers as possible.

Figure 31 – Figure 34 show scatter plots for the five-die benchmark design, for each of the four routing estimation methods described in section 5.1.2. The vertical axis is the maximum routing utilization estimated during simulated annealing, and the horizontal axis is the number of 3D layers required to route each layout.

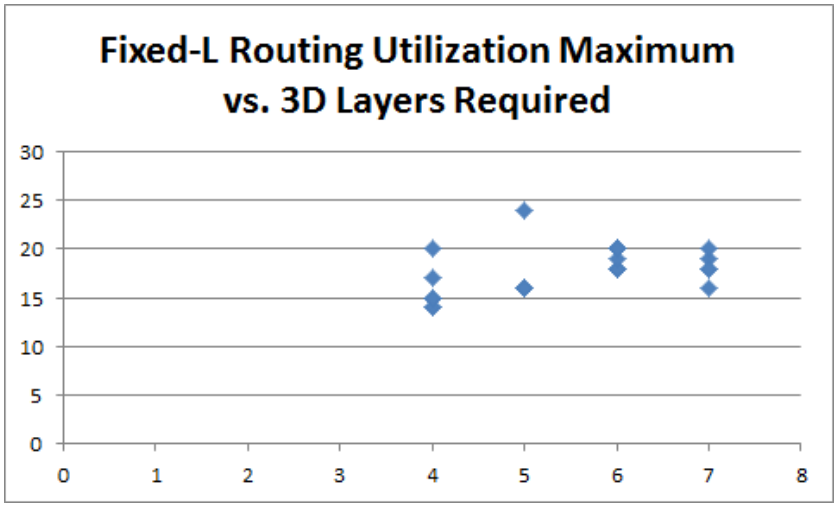


Figure 31: **Fixed-L Method**: Routing utilization maximum during placement and pin assignment vs. number of 3D routing layers required (p = 0.49)

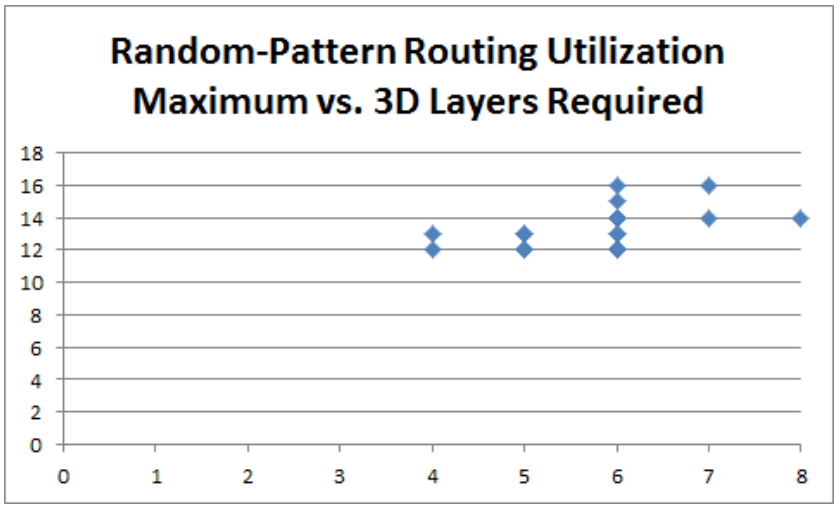


Figure 32: **Random-Pattern Method**: Routing utilization maximum during placement and pin assignment vs. number of 3D routing layers required (p = 0.45)

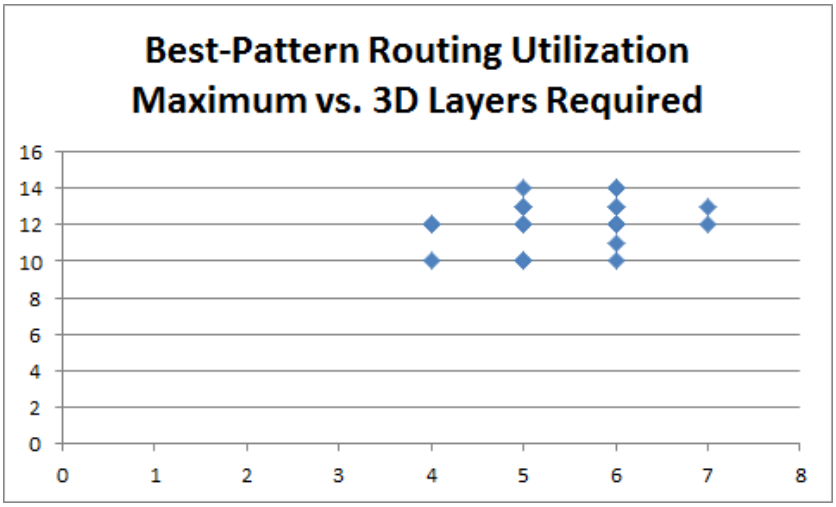


Figure 33: **Best-Pattern Method**: Routing utilization maximum during placement and pin assignment vs. number of 3D routing layers required (p = 0.26)

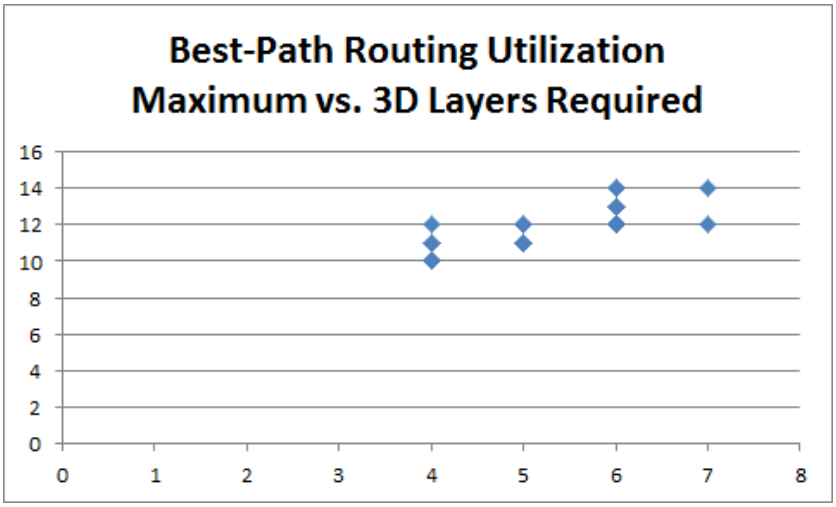


Figure 34: **Best-Path Method**: Routing utilization maximum during placement and pin assignment vs. number of 3D routing layers required (p = 0.73)

Because the fixed-L method of routing estimation is one of the faster methods, this method can be used for design space exploration, where speed is more important than accuracy of layout

predictions. However, for detailed layout realization, the best-path method should be used for routing prediction, as it correlates more strongly with number of layers required for routing.

Table XVI: Problem Size and Corresponding Average Number of ILP Variables

Benchmark design	Single-ended nets	Differential nets	Eligible pins	Avg. ILP variables
FPGA + ADC + DAC	11	85	948	9685
FPGA + 2xADC + 2xDAC	22	170	1056	28840
FPGA + ADC + DAC + Flash + DDR	99	88	1039	18368

Table XVI shows the number of single-ended nets, differential nets, total unique eligible pins across all components, and the average number of ILP variables for benchmarks used in our work. The number of ILP variables changes depending on different layout characteristics. For example, candidate routes are automatically excluded if they would extend beyond the boundaries of the layout. If a candidate route is excluded, then no ILP variables are created for that route.

5.2.4 Actual Routability for Standard Deviation vs. Upside Deviation

Because our goal is to produce layouts that require fewer metal routing layers, we desire accurate estimations of the number of required metal layers during placement and pin assignment. To further evaluate our layer estimation methods, we examined the effectiveness of using standard deviation vs. upside deviation during utilization estimation in our placement and pin assignment algorithm. As in section 5.2.1, we estimated routing utilization with the following methods from section 5.1.2:

1. Fixed-L routing
2. Random-pattern routing
3. Best-pattern routing
4. Best-path routing

We used these methods while performing our modified placement and pin assignment algorithm from section 5.1.1, which included cost factors for maximum routing utilization and (standard or upside) deviation of routing utilization. We performed 3D routing on the resulting layouts (as described in section 5.1.3), and we compared the effectiveness of using standard deviation vs. upside deviation of routing utilization in our placement and pin assignment cost function. A method (or variation) is effective if it results in fewer metal layers needed to complete routing. Table XVII and Table XVIII show the average and variance in the number of metal layers required when using standard deviation vs. upside deviation cost factors.

Table XVII: Effectiveness of Standard Deviation vs. Upside Deviation on Reducing the Routing Layers Required for FPGA + ADC + DAC + Flash + DDR Benchmark (Using Two Candidate Patterns Per Net)

	Standard Deviation		**Upside Deviation**	
Utilization Method	**Average**	**Variance**	**Average**	**Variance**
Fixed-L	6.72	1.60	6.05	0.58
Random-pattern	7.00	1.35	6.29	0.68
Best-pattern	6.30	1.16	6.48	0.81
Best-path	6.56	1.08	6.22	0.82

Table XVII shows the average and variance of the number of metal layers required when using the standard deviation or upside deviation of routing utilization in the cost function within our

placement and pin assignment algorithm (section 5.1.1), when considering two candidate routes per net (as discussed in section 5.1.4). Regardless of what method was used for estimating routing utilization (fixed-L, random-pattern, best-pattern, or best-path), using the upside deviation of routing utilization resulted in significantly lower variance in the number of routing layers required. The upside deviation also reduced the average number of layers in three of the four routing utilization methods (6.05 vs. 6.72, 6.29 vs. 7.00, and 6.22 vs. 6.56), as compared to using the standard deviation of routing utilization.

Table XVIII shows the average and variance of the number of metal layers required when using the standard deviation or upside deviation of routing utilization in the cost function within our placement and pin assignment algorithm, when considering *four* candidate routes per net. In three of the four methods used for estimating routing utilization (random-pattern, best-pattern, or best-path), using the upside deviation of routing utilization resulted in lower variance in the number of routing layers required. The upside deviation produced about the same average number of layers as using the standard deviation of routing utilization, providing slightly lower average layers in two routing utilization methods and slightly higher average layers in the other two methods.

Table XVIII: Effectiveness of Standard Deviation vs. Upside Deviation on Reducing the Routing Layers Required for FPGA + ADC + DAC + Flash + DDR Benchmark (Using Four Candidate Patterns Per Net)

	Standard Deviation		Upside Deviation	
Utilization Method	**Average**	**Variance**	**Average**	**Variance**
Fixed-L	5.55	1.44	5.23	1.61
Random-pattern	5.89	0.54	5.58	0.48
Best-pattern	5.62	0.70	5.75	0.58
Best-path	5.45	0.68	5.53	0.52

These results show that including the upside deviation of routing utilization typically is more effective in reducing the number of metal routing layers when candidate routes are reduced as described in section 5.1.4. Thus, using upside deviation would prove particularly useful in design space exploration for 2.5D IC designs having flexible interconnect.

5.2.5 Metal Layer Planning and 3D Global Routing

For three benchmark designs, we generated dozens of input layouts using the placement and pin assignment algorithm described in section 5.1.1. We then performed simultaneous metal layer minimization and 3D global routing, as described in section 5.1.3. Figure 35 shows histograms of the number of metal layers required to route each layout. The vertical axes represent the number of layouts, and the horizontal axes represent the number of metal layers required for routing.

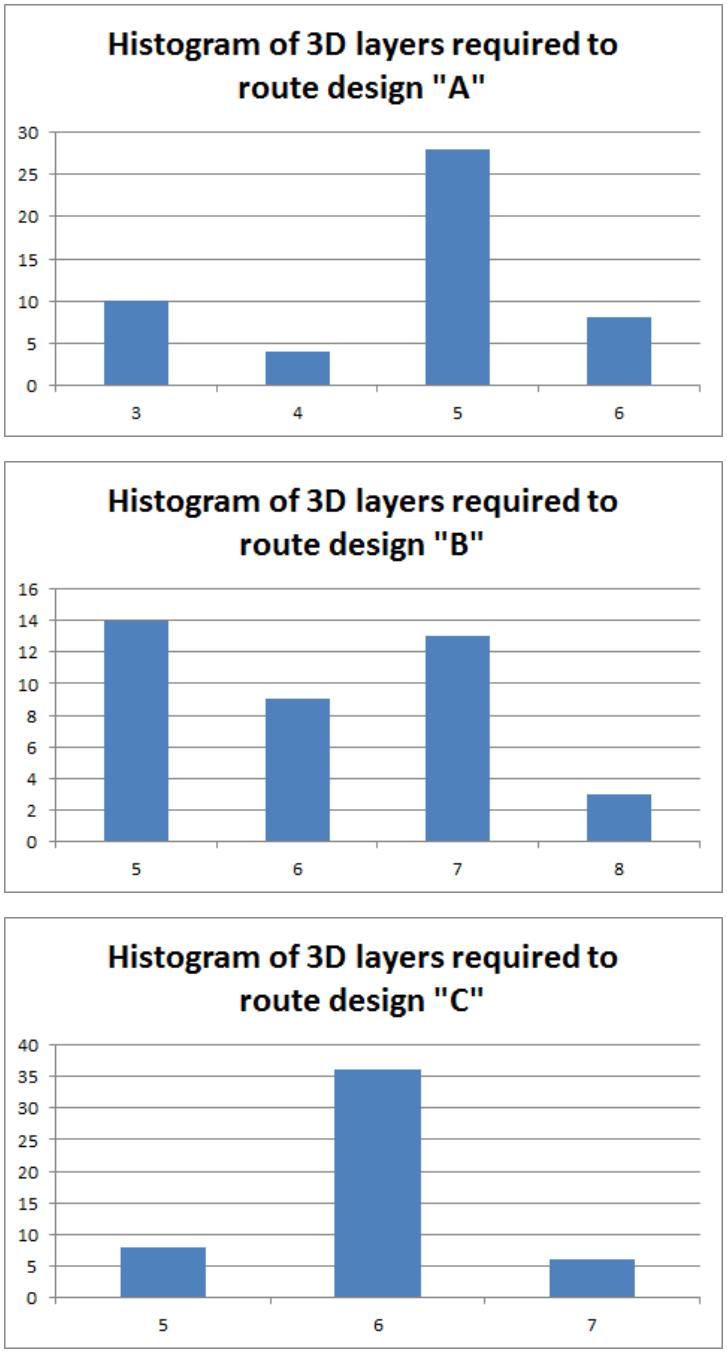


Figure 35: Histograms of metal layers (X axis) required to route each benchmark design

Figure 36 – Figure 38 show layouts with component die locations and pin assignments (left) and 3D routes for all nets (right). Blue areas have no routes passing through grid edges; yellow and red show increasing numbers of routes. As expected, the areas with many routes correspond to areas having many connected pins.

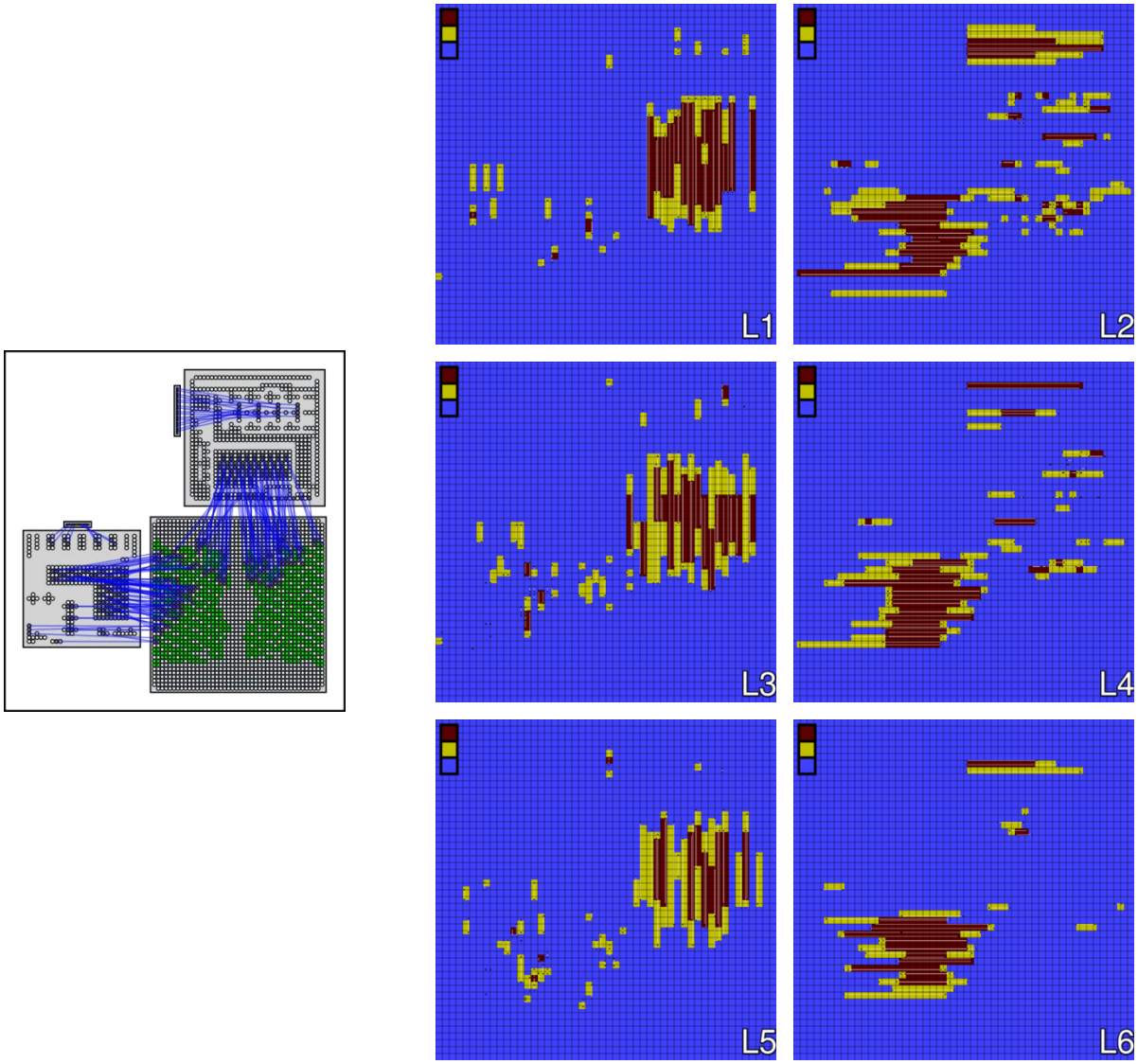


Figure 36: Die layout and metal routing layers for FPGA + ADC + DAC design. Blue areas have no routes passing through grid edges; yellow and red show increasing numbers of routes.

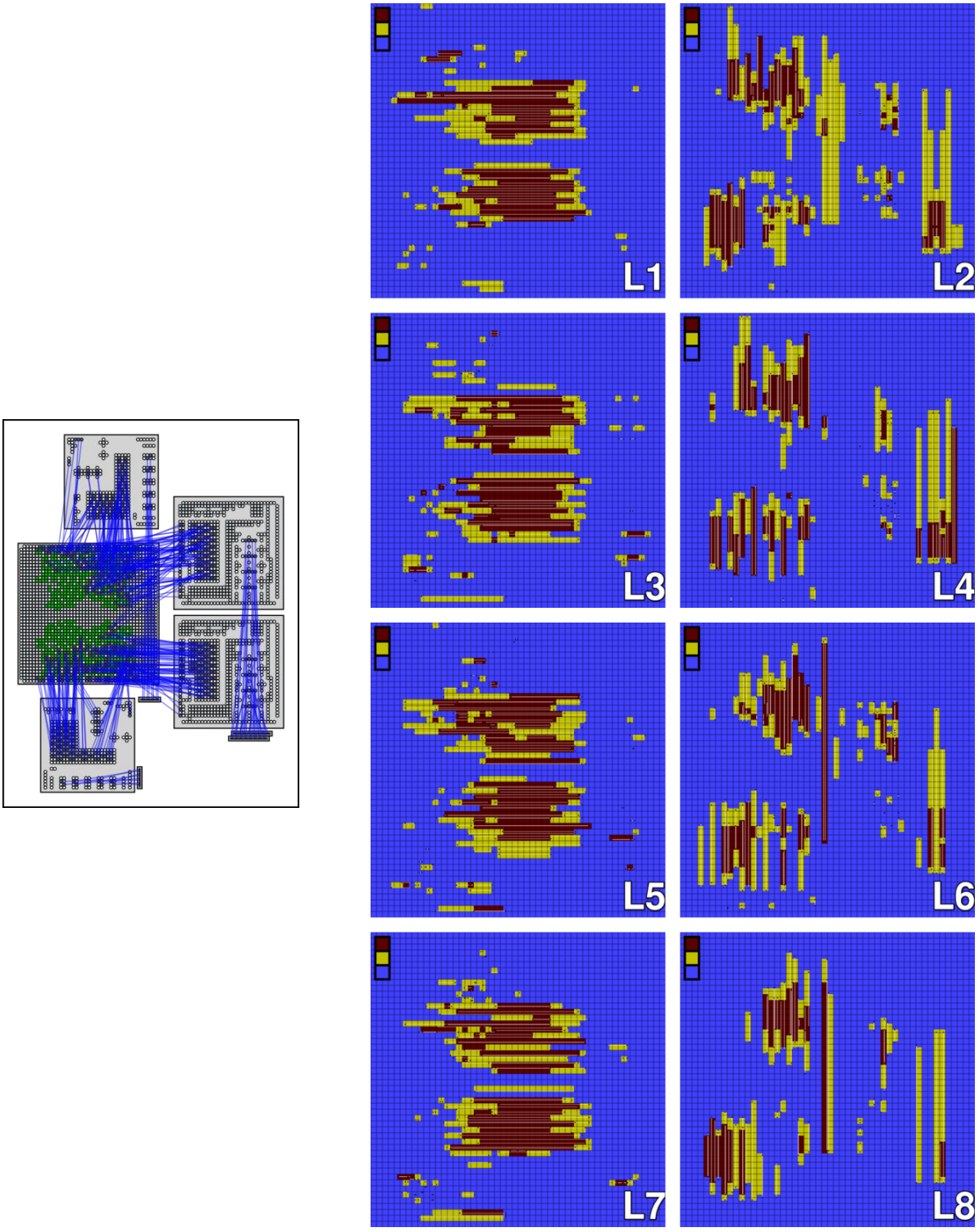


Figure 37: Die layout and metal routing layers for FPGA + 2xADC + 2xDAC design. Blue areas have no routes passing through grid edges; yellow and red show increasing numbers of routes.

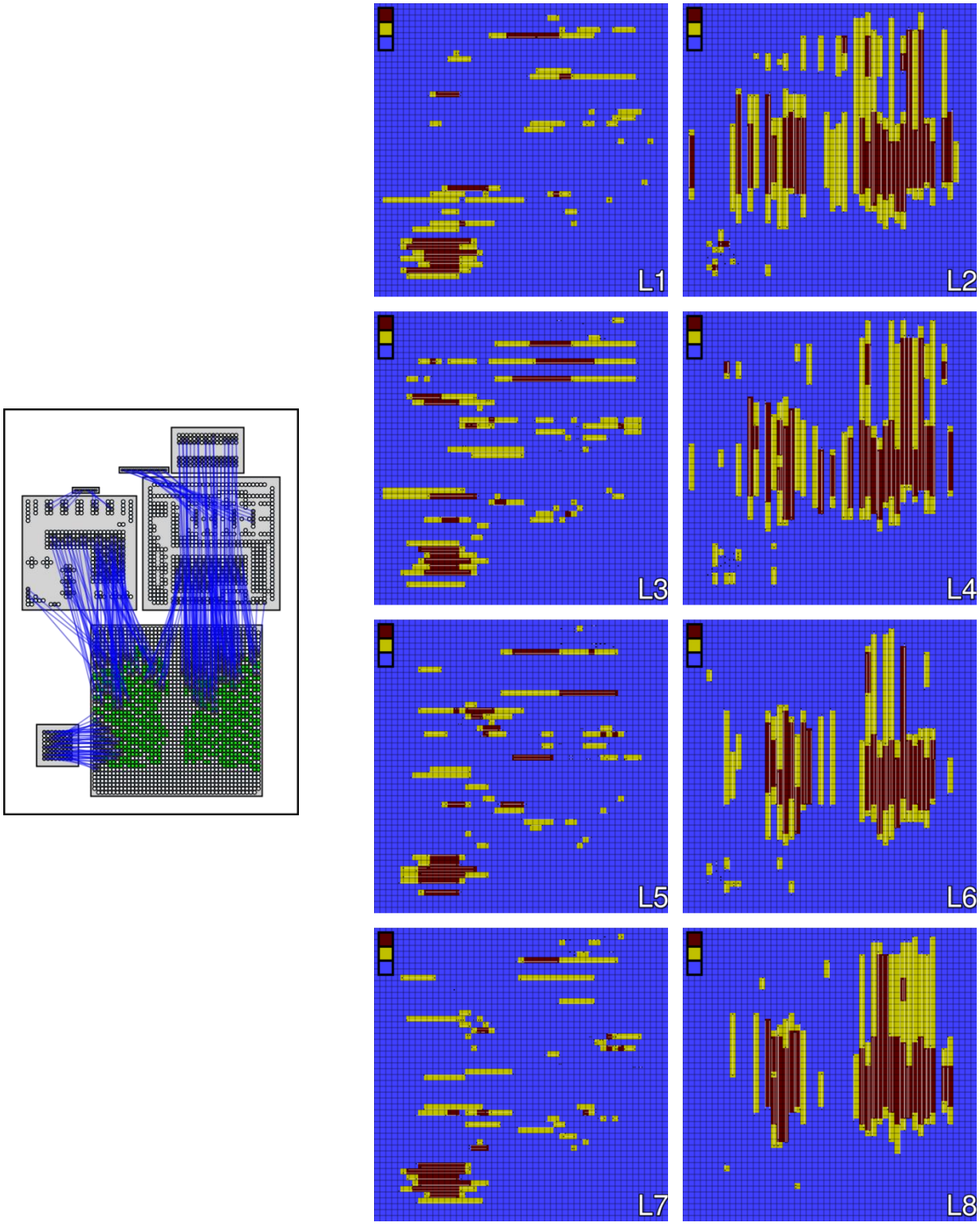


Figure 38: Die layout and routing layers for FPGA + ADC + DAC + Flash + DDR design. Blue areas have no routes passing through grid edges; yellow and red show increasing numbers of routes

5.2.6 Varying 3D Grid Size and Number of Candidate Routes

We experimented with three grid sizes, expressed as the number of bins across the interposer area: 30x30, 50x50, and 70x70. As expected, the larger grid sizes resulted in increased runtime, but improved granularity of 3D routes. Including all candidate routing patterns also resulted in significantly increased runtime as compared to reducing the number of candidate patterns. Table XIX – Table XXI show the average and variance of the number of metal routing layers across different grid sizes and different numbers of candidate routing patterns.

Table XIX: Number of metal layers required for 30x30 grid size, according to number of candidate patterns

		Candidate	Number of Layers	
Grid Size	**Design**	**Patterns**	**Average**	**Variance**
30x30	A	All	3.61	0.31
30x30	A	2	5.28	1.20
30x30	A	4	4.58	1.04
30x30	B	All	5.69	0.71
30x30	B	2	8.24	1.55
30x30	B	4	7.00	1.30

Table XX: Number of metal layers required for 50x50 grid size, according to number of candidate patterns

Grid Size	Design	Candidate Patterns	Number of Layers	
			Average	Variance
50x50	A	All	3.52	0.25
50x50	A	2	5.45	1.18
50x50	A	4	4.50	0.95
50x50	B	All	5.53	0.65
50x50	B	2	8.31	1.21
50x50	B	4	6.90	1.37
50x50	C	2	6.86	1.36
50x50	C	4	5.68	0.86

Table XXI: Number of metal layers required for 70x70 grid size, according to number of candidate patterns

Grid Size	Design	Candidate Patterns	Number of Layers	
			Average	Variance
70x70	A	All	2.86	0.78
70x70	A	2	4.41	0.51
70x70	A	4	3.61	0.34
70x70	B	All	4.00	0.27
70x70	B	2	6.27	0.74
70x70	B	4	5.44	0.52

As shown in these tables, a consistent trend is that when we considered more candidate routing patterns, we enabled our ILP formulation to explore a wider variety of solutions, and as a result produced layouts with fewer metal layers. However, reducing the number of candidate patterns improved runtime noticeably. When we considered 4 candidate routing patterns (results shown in yellow), we produced layouts with somewhat worse results in terms of number of routing layers, but with improved runtime.

When using our presented methods for design space exploration, it is more important to produce a variety of layouts quickly. In that case, considering 4 candidate routing patterns provides a reasonable trade-off between solution quality and runtime. On the other hand, when performing detailed design, considering all candidate routing patterns (results shown in green) allows for high-quality results at the expense of longer runtimes.

Our work in metal layer minimization and 3D global routing initially assumed fixed connections between dies. However, we can use these methods to even further reduce the number of metal layers. When we exploit flexible interconnect, we can change nets' connections to die pins in such a way as to enable designs more suited to routing using fewer layers. We explored this opportunity in section 5.1.5 and present results in section 5.2.7.

5.2.7 2.5D Pin Reassignment

To evaluate our pin reassignment methods, we started with 3D layouts generated by our metal layer minimization and 3D global routing stage described in section 5.1.3, using the same benchmark designs with a variety of placements and pin assignments. We applied our pin reassignment methods to these layouts and performed metal layer minimization and 3D global routing again. We compiled statistics on metal layer reduction enabled by our pin reassignment methods. By performing flexible pin reassignment, we were able to produce layouts that require fewer metal layers than would be required if the pin assignments were fixed (i.e., not flexible).

Table XXII: Number of metal routing layers required before and after pin reassignment
The best results are highlighted in green, and results very close to the best are highlighted in yellow. Identical results before and after pin reassignment are not highlighted.

		Metal Routing Layers Required					
		Before Pin Reassignment			After Pin Reassignment		
Design	Method	Average	Variance	Minimum	Average	Variance	Minimum
A	Layer Goal	4.65	1.11	3	4.21	0.63	3
B		5.94	0.93	5	5.88	0.92	4
C		5.93	0.20	5	5.80	0.25	5
A	Max Utilization	4.66	1.09	3	4.20	0.62	3
B		6.03	0.94	5	5.97	0.77	5
C		5.93	0.20	5	5.78	0.26	5
A	Max-std dev	4.70	1.05	3	4.23	0.60	3
B		5.87	0.72	5	5.87	0.72	5
C		5.93	0.20	5	5.80	0.25	5
A	Max-upside dev	5.32	0.22	5	4.86	0.12	4
B		6.10	1.09	5	6.00	0.80	5

Table XXII shows statistics on the number of metal layers required before and after pin reassignment, across multiple benchmark designs and pin reassignment methods. Our pin reassignment methods were able to reduce the average number of metal routing layers required for our benchmark designs. The *layer goal* and *max-upside-deviation* methods also were able to find pin assignments that reduced the number of metal layers even lower than the minimum from any design before performing pin reassignment. This shows that exploiting flexible interconnect can enable metal layer reduction that may be infeasible when treating all connections as fixed nets.

5.3 Summary

We presented several complementary methods for reducing the interposer manufacturing costs in 2.5D ICs by minimizing the number of metal layers required to route all inter-die connections within the IC. We estimate routing utilization during simultaneous die placement and pin assignment in order to guide the placement and pin assignment toward layouts that are likely to require fewer metal layers. After performing placement and pin assignment, we use our metal layer planning and 3D global routing stage to determine 3D routes for all nets and the number of layers required. Based on this information, we refine the pin assignment with the goal of further reducing the number of layers. We once again perform metal layer planning and 3D global routing based on the new pin assignment. Our results showed that these steps are effective in minimizing the number of metal layers in a 2.5D IC's interposer, and thus in reducing manufacturing costs.

6 Conclusions and Future Work

We explored a variety of computer-aided design methods for exploiting *flexible interconnect* in 2D and 2.5D systems. Our work included an open-source framework that accommodates design capture and layout generation for printed-circuit boards (2D designs) and 2.5D integrated circuits. Our framework captures connectivity requirements at a higher level of abstraction than explicitly or manually defining each connection. We presented methods that, based on these requirements, perform simultaneous component placement, pin assignment, and power supply assignment that can use flexible interconnect to improve design quality. We also presented our simultaneous metal layer minimization and 3D global routing methods and flexible pin reassignment methods that can produce layouts with fewer metal layers than would be required if all connections were fixed (i.e., not flexible). Our work's focus on flexible interconnect enables design improvements that would not be possible without taking advantage of the freedom that flexible interconnect can provide.

Our work can be extended to encompass more of the 2D or 2.5D design process, and could also be extended to consider 3D die-stacked architectures. We could also explore automatic co-design between 2D/2.5D hardware and internal FPGA logic and routing. This would allow the 2D/2.5D pin assignment to guide connections toward pins that would improve internal FPGA placement and routing, and guide the FPGA design process based on hardware considerations such as reduced manufacturing costs.

Another potential avenue for extended work would be to explore the interaction between our simultaneous metal layer minimization and 3D global routing process (section 5.1.3) and our routability-aware pin reassignment process (section 5.1.5). Iterating between 3D global routing and flexible pin reassignment could hone the pin reassignment process in our framework. This extension has the potential to further decrease the number of metal layers required to route all signals within a design.

Finally, our simultaneous metal layer minimization and 3D global routing process (section 5.1.3) can be expanded to consider via area constraints on assigned 3D routes. This likely would improve detail routing by modeling more detailed characteristics of practical routing. Our ILP formulation already supports edge capacities and edge utilization, as these can be modeled as part of our 3D graph.

Our work is the first in the literature to our knowledge to consider flexible interconnect as part of placement and pin assignment for 2D and 2.5D designs. We leverage the freedoms enabled by flexible interconnect to choose placements and pin assignments that can result in improved designs – in our case, designs with reduced wirelength, area, and/or number of metal routing layers. Through a variety of experiments, we show that leveraging flexible interconnect can ultimately produce designs that can be routed using fewer metal layers, saving significant manufacturing costs.

7 Publications

1. **D. Seemuth**, A. Davoodi, and K. Morrow. “Planning Flexible Interconnect in 2.5D ICs to Minimize the Interposer’s Metal Layers,” Under review for ICCAD 2016.

2. **D. Seemuth**, A. Davoodi, and K. Morrow. “Automatic Die Placement and Flexible I/O Assignment in 2.5D IC Design,” *Quality Electronic Design (ISQED), 2015 16th International Symposium on*, p. 524–527.

3. **D. Seemuth** and K. Morrow, “Automated multi-device placement, I/O voltage supply assignment, and pin assignment in circuit board design,” *Field-Programmable Technology (FPT), 2013 International Conference on*, p. 262–269.

4. P. Klabbers, M. Bachtis, J. Brooke, M. Cepeda Hermida, K. Compton, S. Dasu, A. Farmahni-Farahani, S. Fayer, R. Fobes, R. Frazier, C. Ghabrous, T. Gorski, A. Gregerson, G. Hall, C. Hunt, G. Iles, J. Jones , C. Lucas, R. Lucas, M. Magrans, D. Newbold, I. Oljavo, A. Perugupalli, M.Pioppi, A. Rose, I. Ross, D. Sankey, M. Schulte, **D. Seemuth**, W.H. Smith, J.Tikalsky, A. Tapper, and T. Williams, “CMS level-1 upgrade calorimeter trigger prototype development,” *J. of Instrumentation*, vol. 8, no. 2, Feb. 2013.

5. K. Compton, S. Dasu, A. Farmahini-Farahani, S. Fayer, R. Fobes, R. Frazier, T. Gorski, G. Hall, G. Iles, J. Jones, P. Klabbers, D. Newbold, A. Perugupalli, S. Rader, A. Rose, **D. Seemuth**, W. Smith, and J. Tikalsky, “The MP7 and CTP-6: multi-hundred Gbps processing boards for calorimeter trigger upgrades at CMS,” *J. of Instrumentation*, vol. 7, no. 12, Dec. 2012.

6. P. Klabbers, T. Gorski, M. Bachtis, K. Compton, S. Dasu, A. Farmahini-Farahani, R. Fobes, A. Gregerson, M. Grothe, I. Ross, **D. Seemuth**, M. Schulte, and W.H. Smith, “CMS Calorimeter Trigger Phase I upgrade,” *J. of Instrumentation*, vol. 7, no. 1, Jan. 2012.

8 References

[1] Xilinx, "7 Series FPGAs Packaging and Pinout Product Specifications User Guide." 13-Nov-2014. Available: http://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf

[2] Cadence Design Systems, "Allegro Design Authoring." Available: http://www.cadence.com/rl/Resources/datasheets/7433_Design_Creation_DS_FINAL.pdf

[3] Cadence Design Systems, "Allegro FPGA System Planner." Available: http://www.cadence.com/ rl/Resources/datasheets/pcb_fpga_ds.pdf

[4] Mentor Graphics, "DxDesigner PCB Design Definition Environment - Mentor Graphics." Available: http://www.mentor.com/products/pcb-system-design/ design-creation/dxdesigner/

[5] Cadence Design Systems, "Cadence OrCAD Capture." Available: http://www.cadence.com/rl/Resources/datasheets/pcb_orcad_capture_ds.pdf

[6] CadSoft Computer, "New in V6." Available: http://www.cadsoftusa.com/eagle-pcb-design-software/new-in-v6/

[7] Westdev, "Pulsonix V7.6 Brochure." Available: http://www.pulsonix.com/downloads/datasheets/Pulsonix%20V7.6%20Brochure.pdf

[8] B. Nelson, "PHDL / Wiki / Motivation." Available: http://sourceforge.net/p/phdl/wiki/Motivation/

[9] J. Xiong, Y.-C. Wong, E. Sarto, and L. He, "Constraint Driven I/O Planning and Placement for Chip-package Co-design," Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 207–212, 2006.

[10] M. M. Ozdal and M. D. F. Wong, "Simultaneous Escape Routing and Layer Assignment for Dense PCBs," International Conference on Computer-Aided Design (ICCAD), pp. 822–829, 2004.

[11] S.-I. Lei and W.-K. Mak, "Simultaneous Constrained Pin Assignment and Escape Routing for FPGA-PCB Codesign," International Conference on Field Programmable Logic and Applications (FPL), pp. 435–440, 2011.

[12] L. Madden, "Heterogeneous 3-D Stacking, Can We Have the Best of Both (Technology) Worlds?," International Symposium on Physical Design (ISPD), pp. 1–2, 2013.

[13] P. Doe, "2.5D interposers look increasingly like the near term, high performance solution," *3D Packaging*, no. 23, pp. 6–9, May-2012.

[14] R. Topaloglu, "Applications Driving 3D Integration and Corresponding Manufacturing Challenges," Design Automation Conference (DAC), pp. 220–223, 2011.

[15] M. Casale-Rossi, P. Leduc, G. De Micheli, P. Blouet, B. Farley, A. Fontanelli, D. Milojevic, and S. Smith, "Panel: 'Will 3D-IC Remain a Technology of the Future... Even in the Future?,'" Design, Automation and Test in Europe (DATE), pp. 1526–1530, 2013.

[16] J. Lau, "TSV Interposer: The Most Cost-Effective Integrator for 3D IC Integration," SEMATECH Symposium Taiwan, 13-Sep-2011. Available: http://www.sematech.org/meetings/archives/symposia/10187/Session2/04_Lau.pdf

[17] E. J. Marinissen, "Challenges and Emerging Solutions in Testing TSV-based 2 1/2D- and 3D-stacked ICs," Design, Automation and Test in Europe (DATE), pp. 1277–1282, 2012.

[18] K. Saban, "Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency." Xilinx, Inc., 11-Dec-2012. Available: http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf

[19] A. Hahn Pereira and V. Betz, "CAD and Routing Architecture for Interposer-based multi-FPGA Systems," International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 75–84, 2014.

[20]William Wu Shen, "3DIC System Design Impact, Challenge and Solutions," International Symposium on Physical Design (ISPD), pp. 63–64, 2014.

[21] F. Lee, B. Shen, W. Chen, and S. Lee, "3DIC from concept to reality," Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 394–398, 2013.

[22] R. Chaware, K. Nagarajan, and S. Ramalingam, "Assembly and reliability challenges in 3D integration of 28nm FPGA die on a large high density 65nm passive interposer," Electronic Components and Technology Conference (ECTC), pp. 279–283, 2012.

[23] J. Cong and G. Luo, "An Analytical Placer for Mixed-size 3D Placement," International Symposium on Physical Design (ISPD), pp. 61–66, 2010.

[24] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "TSV-aware Analytical Placement for 3D IC Designs," Design Automation Conference, pp. 664–669, 2011.

[25] R. Fischbach, J. Knechtel, and J. Lienig, "Utilizing 2D and 3D Rectilinear Blocks for Efficient IP Reuse and Floorplanning of 3D-integrated Systems," International Symposium on Physical Design (ISPD), pp. 11–16, 2013.

[26] W. Sui, S. Dong, and J. Bian, "Wirelength-driven Force-directed 3D FPGA Placement," Great Lakes Symposium on VLSI (GLSVLSI), pp. 435–440, 2010.

[27] Y.-L. Hsieh and T.-M. Hsieh, "A New Effective Congestion Model in Floorplan Design," Design, Automation and Test in Europe (DATE), p. 21204, 2004.

[28] B. Hu and M. Marek-Sadowska, "Congestion Minimization During Placement Without Estimation," International Conference on Computer-aided Design (ICCAD), pp. 739–745, 2002.

[29] Z. Li, W. Wu, and X. Hong, "Congestion Driven Incremental Placement Algorithm for Standard Cell Layout," Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 723–728, 2003.

[30] M. Saeedi, M. S. Zamani, and A. Jahanian, "Evaluation, Prediction and Reduction of Routing Congestion," *Microelectronics*, vol. 38, no. 8–9, pp. 942–958, Aug. 2007.

[31] C.-W. Sham, E. F. Y. Young, and J. Lu, "Congestion Prediction in Early Stages of Physical Design," *Design Automation of Electronic Systems*, vol. 14, no. 1, pp. 12:1–12:18, Jan. 2009.

[32] M. Wang, X. Yang, and M. Sarrafzadeh, "Congestion minimization during placement," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 10, pp. 1140–1148, Oct. 2000.

[33] M. Wang and M. Sarrafzadeh, "Modeling and Minimization of Routing Congestion," Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 185–190, 2000.

[34] K. Tsota, C.-K. Koh, and V. Balakrishnan, "Guiding Global Placement with Wire Density," International Conference on Computer-Aided Design (ICCAD), pp. 212–217, 2008.

[35] Y.-K. Ho and Y.-W. Chang, “Multiple Chip Planning for Chip-interposer Codesign,” Design Automation Conference (DAC), pp. 27:1–27:6, 2013.
[36] D. H. Kim, K. Athikulwongse, and S. K. Lim, “A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout,” International Conference on Computer-Aided Design (ICCAD), pp. 674–680, 2009.
[37] M.-C. Tsai, T.-C. Wang, and T. Hwang, “Through-Silicon Via Planning in 3-D Floorplanning,” *Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 8, pp. 1448–1457, Aug. 2011.
[38] Q. Zou, J. Xie, and Y. Xie, “Cost-driven 3D design optimization with metal layer reduction technique,” International Symposium on Quality Electronic Design (ISQED), pp. 294–299, 2013.
[39] W.-H. Liu, T.-K. Chien, and T.-C. Wang, “Metal Layer Planning for Silicon Interposers with Consideration of Routability and Manufacturing Cost,” Design Automation and Test in Europe (DATE), pp. 359:1–359:6, 2014.
[40] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, “NTHU-Route 2.0: A fast and stable global router,” International Conference on Computer-Aided Design (ICCAD), pp. 338–343, 2008.
[41] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, “NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing With Bounded-Length Maze Routing,” *Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 709–722, May 2013.
[42] J. Hu, J. A. Roy, and I. L. Markov, “Completing High-quality Global Routes,” International Symposium on Physical Design (ISPD), pp. 35–41, 2010.
[43] Y. Xu and C. Chu, “MGR: Multi-level Global Router,” International Conference on Computer-Aided Design (ICCAD), pp. 250–255, 2011.
[44] M. Cho and D. Z. Pan, “BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP,” Design Automation Conference (DAC), pp. 373–378, 2006.
[45] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, “BoxRouter 2.0: A Hybrid and Robust Global Router with Layer Assignment for Routability,” *Design Automation of Electronic Systems*, vol. 14, no. 2, pp. 32:1–32:21, Apr. 2009.
[46] Y. Zhang, Y. Xu, and C. Chu, “FastRoute3.0: A Fast and High Quality Global Router Based on Virtual Capacity,” International Conference on Computer-Aided Design (ICCAD), pp. 344–349, 2008.
[47] Y. Xu, Y. Zhang, and C. Chu, “FastRoute 4.0: Global Router with Efficient via Minimization,” Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 576–581, 2009.
[48] W.-H. Liu, Y.-L. Li, and C.-K. Koh, “A Fast Maze-free Routing Congestion Estimator with Hybrid Unilateral Monotonic Routing,” International Conference on Computer-Aided Design (ICCAD), pp. 713–719, 2012.
[49] C. Chu, “FLUTE: Fast Lookup Table Based Wirelength Estimation Technique,” International Conference on Computer-Aided Design (ICCAD), pp. 696–701, 2004.
[50] S.-J. Lee and K. Raahemifar, “FPGA placement optimization methodology survey,” Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 001981–001986, 2008.

[51] G. Haiyun and X. Juyan, "Research on the routing algorithm of SRAM-based FPGA," International Conference on Solid-State and Integrated Circuits Technology, pp. 1964–1966 vol.3, 2004.

[52] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," International Workshop on Field Programmable Logic and Applications (FPL), pp. 213–222, 1997.

[53] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-based Performance-driven Router for FPGAs," International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 111–117, 1995.

[54]KiCad, *KiCad EDA*. Available: http://kicad-pcb.org/.

[55] M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design*, 1st ed. McGraw-Hill Higher Education, 1996.

[56] A. Makhorin, *GLPK - GNU Linear Programming Kit*. Available: http://www.gnu.org/ software/glpk/, 2012.

[57] J. D. Davis, C. P. Thacker, and C. Chang, "BEE3: Revitalizing computer architecture research." Microsoft Research, Tech Rep., 2009.

[58] Joy, Kenneth I., "Breshenham's Algorithm," University of California, Davis. Available: http://graphics.idav.ucdavis.edu/education/GraphicsNotes/Bresenhams-Algorithm.pdf. 1999.

[59] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable Routing," International Conference on Computer-Aided Design (ICCAD), pp. 110–114, 2000.

[60] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[61] X. Wu, G. Sun, X. Dong, R. Das, Y. Xie, C. Das, and J. Li, "Cost-driven 3D Integration with Interconnect Layers," Design Automation Conference (DAC), pp. 150–155, 2010.