Towards Scalable Sub-THz Massive MIMO: Beamforming ASICs and 3D Die-to-Die Interconnects

Harrison Liew

Towards Scalable Sub-THz Massive MIMO: Beamforming ASICs and 3D Die-to-Die
Interconnects

by

Harrison Liew

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Borivoje Nikolić, Chair
Professor Ali Niknejad
Assistant Professor Jessica Boles
External Member Farhana Sheikh

Summer 2024

Towards Scalable Sub-THz Massive MIMO: Beamforming ASICs and 3D Die-to-Die
Interconnects

Abstract

Towards Scalable Sub-THz Massive MIMO: Beamforming ASICs and 3D Die-to-Die Interconnects

by

Harrison Liew

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Borivoje Nikolić, Chair

The incessant growth of wireless communication demand is driving the desire for massive antenna arrays operating at mm-Wave to sub-THz carrier frequencies, where many GHz of bandwidth are readily available. However, this opportunity comes with an integration dilemma as antenna pitch falls below 1mm and digitized baseband data rates surpass a terabit per second. To address these and other design challenges, heterogeneous integration (HI) of radio, mixed-signal, baseband, and power delivery chiplets is a solution. Towards this goal, this dissertation presents a multi-part exploration into the technologies and design methodologies required for future sub-THz massive MIMO systems.

First, a pair of baseband ASICs for scalable linear massive MIMO arrays with digital beamforming is presented. The learnings from the design process of these chips is directly applied to the improvement of Hammer, a physical design flow generator, that has since accelerated the implementation of over ten research chips and is now used in multiple courses. Extrapolating from the integration limitations of the presented and other state-of-the-art chips, the scaling challenges and HI opportunities of sub-THz two-dimensional massive MIMO arrays is subsequently reviewed. The findings culminate in a case for 3D chiplet integration into "radio cubes", which are consequently tiled horizontally into a scalable array. To enable 3D integration, a comprehensive study of technology constraints for standardizing 3D die-to-die interconnects is performed, demonstrating how scaling roadmaps for process and packaging technologies converge towards a very different circuit architecture compared to existing die-to-die interconnects. Given the opportunities afforded by this architecture, a Chisel generator framework is presented that performs rapid design space exploration for a novel defect repair mechanism and validates a new standard (UCIe-3D™) in the Intel 16 process technology. Most importantly, this generator demonstrates a significant design effort reduction for exploring and implementing die-to-die interconnects–a key enabler for a future chiplet ecosystem. Finally, with solutions proposed for key technologies needed for

the envisioned heterogeneously-integrated system, a set of integration concepts is presented, detailing remaining technological feasibility tradeoffs.

To my partner, Helen Zhou; my parents, Shelley Mau and Seng Liew; and all my friends, family, and mentors who have supported me in my career.

Contents

List of Figures

List of Tables

List of Listings

Acknowledgments

I feel tremendously lucky to have been able to learn from so many talented people here at Berkeley and at my various internships, co-ops, and full-time positions before and during my Doctoral studies.

Most importantly, Bora Nikolić, my advisor, who has been instrumental in supporting me with a wealth of resources and helping me think critically about my own and others' research. Bora, you are particularly skilled in connecting academia with industry, which has been invaluable in allowing me to tape out such complex chips and work on 3D interconnect research. Though the way you set your expectations is simultaneously sky-high and frustratingly vague, I hope I was able to meet a sufficient number of them throughout my meandering journey. I will certainly miss your group meeting stories and industry gossip–I will have to find new sources of entertainment!

Secondly, I would like to thank Farhana Sheikh, my internship mentor turned committee member and now full-time manager. Your mentorship style is perfect, as you have stimulated my curiosity to dig ever deeper into problems; helped me think big picture and long-term; improved my communication and writing skills; and helped show me the ropes of the academic, industry, and government research landscape and politics. I am particularly grateful for your continuous support over the last two years and for the opportunity to continue working with you going forward.

I particularly enjoyed my internship experiences as they provided project diversity and new mentorship experiences, and am somewhat sad that I will not have the luxury of having these experiences going forward. Beyond Farhana, David Kehlet and others at Intel, including those in the Altera CTO Advanced Chiplet Technologies Group and the UCIe Standards Group: Jong-ru Guo, Zuoguo Wu, Gerald Pasdast, Zhiguo Qian, Sathya Tiagaraj, Stephen Wong, and Tanay Karnik, provided great feedback during the process of 3D interconnect design. Marcos Tavares at Nokia Bell Labs gave me a challenging project far outside my comfort zone on an analog implementation of for RNN-based eigenvalue decomposition that incredibly won the 2nd best internship project prize.

The other faculty members of my committee, Ali Niknejad and Jessica Boles were also instrumental in providing feedback and guidance. From my Hydra Spine tapeouts to my qualifying exam to a fascinating collaboration on piezoelectric resonator-based power conversion, you were patient and wonderful teachers. Multiple faculty at Columbia University, like Mingoo Seok, Peter Kinget, and David Vallancourt, put me on the path towards integrated circuits research. Finally, the team at Analog Bits taught me clocking and SerDes design, which proved to be surprisingly valuable throughout my time here despite my research being nominally unrelated.

Many senior graduate students helped me along the way, to whom I remain tremendously indebted. John Wright, Colin Schmidt, and Greg LaCaille showed me the ropes for how to tape out, spending countless hours teaching me physical design and analog integration. There is a long list of collaborators on the Hydra Spine tapeouts. Yue Dai was an amazing partner on the Hydra DSP project; James Dunn was the lead designer on the Hydra Spine

Funding

Chapter 1

Introduction

1.1 Motivation

The last few decades have witnessed the incessant growth of wireless communication traffic, driven by the rapid proliferation of internet-connected devices for applications ranging from mobile video to autonomous vehicles to augmented/virtual reality. An examination of the evolution of cellular (3GPP) and Wi-Fi (IEEE 802.11) standards reveals a steady march towards a target of *1 Tbps data rate* per-user equipment (UE) by the middle of the next decade to satisfy an annualized growth rate in mobile traffic of about 1.4× [1]. Compounding the problem, the peak *density* of these UEs is also increasing, such as in urban environments or venues like stadiums. To satisfy this need, standards like 6G are targeting a 10× connection density boost over 5G to a target of 10 million devices per square kilometer [2].

Unfortunately, the capabilities of wireless communication technologies are currently unable to reach those targets. Since Claude Shannon introduced the concept of channel capacity in 1948 [3], which states that a channel's capacity is a function of its bandwidth and signal-to-noise ratio (Eq. 1.1), numerous innovations such as LDPC [4] and OFDM [5] have already brought modern communications very close to the Shannon limit. Increasing capacity can be achieved by increasing bandwidth, which is available only in the higher-frequency mm-Wave and sub-THz bands [6], leading to significant opportunities and challenges in circuit design and system integration. Increasing spectral efficiency (capacity divided by bandwidth) by a single bit/s/Hz requires a doubling of signal power, clearly not a scalable solution as wireless networks already consume about 200TWh of electricity annually–about 1% of global electricity consumption [7]. This is worsened by the fact that noise power also includes interference from other UEs, which is a function of network density. On the topic of density, networks serve multiple UEs within a given budget of frequency, time, and power using multiple access technology to divide up those axes. These techniques are known as FDMA, TDMA, CDMA, and OFDMA (and variants), which form the basis of wireless communications standards. Fundamentally, however, the combination of capacity-boosting and multiple access techniques are merely knobs to trade off per-UE capacity versus network density, but do not

fundamentally increase the entire network's capacity within its frequency and power budgets.

$$C = B \log_2(1 + \frac{S}{N}) \tag{1.1}$$

where

$$\begin{aligned} C &= \text{Channel capacity (bits/s)} \\ B &= \text{Bandwidth (Hz)} \\ S &= \text{Signal power (W)} \\ N &= \text{Noise power (W)} \end{aligned}$$

In reality, wireless communications, by virtue of it occurring in three-dimensional space, has another dimension (i.e., budget) to divvy up: space. To achieve practical *spatial multiplexing*, **massive MU-MIMO** (multiple user, multiple-input, multiple-output) is the technique that utilizes a large number of antennas at a base station to communicate with cheap, MU-MIMO UEs [8]. This antenna array generates multiple, simultaneous, and non-overlapping beams of electromagnetic energy steered precisely to the direction of each UE (called **beamforming**), thereby forming distinct, nearly interference-free channels, upon which traditional capacity-boosting techniques can be applied. In this regime, increasing the number of antennas to infinity theoretically resolves to the asymptotes of:

- Infinitesimally thin beams, meaning infinite antenna array gain and minimal power per bit (only limited by intrinsic system noise)

- No more uncorrelated noise and interference, and no fading effects (e.g., multipath, reflections)

- Perfect channel reciprocity, meaning equal uplink and downlink channel characteristics, simplifying channel estimation

- Only contamination between cells, which can be mitigated by coordination and cell shrinkage

Ever since this concept was introduced, significant research and development has led to some small-scale deployments, but no demonstrations so far approach this asymptotic limit [9].

1.2 Background

1.2.1 Massive MIMO, in Hardware

Currently, practical beamforming requires applying coordinated phase shifts to an array of radio frequency (RF) front-ends to form constructive wavefront interference in the direction

of the desired beam. Phased array antennas must be spaced at half-wavelength ($\lambda/2$) to avoid grating lobes, which is spurious energy directed in undesired directions. These antennas are typically made as square metallic patches on a substrate, which are highly-manufacturable, have inherent directivity, and have a reasonably large steering angle [10]. Achieving higher fractional bandwidth (bandwidth divided by center frequency) is in active research, needed to take advantage of the 10's of GHz of bandwidth available at mm-Wave and sub-THz frequencies. Phase shifting can occur in multiple places in the signal chain, such as at the RF front-end, in the data converters, or in the digital baseband. The location and connectivity of the phase shifting block determines the number of signal chains after beamforming, which translates to the number of possible simultaneous beams that can be formed. There are additional considerations such as the supported dynamic range of received signals, beamforming performance, and power consumption when choosing between analog, hybrid, and digital beamforming architectures. To fully realize the massive MIMO paradigm, fully-digital beamforming with extremely large arrays allows for the most powerful and flexible method of implementing algorithms for channel estimation, detection, and nonlinearity correction. In this architecture, an entire signal chain (RF-to-bits) is dedicated to each antenna with all processing and phase shifting occurring in the digital domain. However, this requires an inordinate amount of data to be moved and processed within the system, leading it to be traditionally dismissed as too area- and power-hungry.

To begin to tackle the bandwidth limits of fully-digital beamforming, an architecture targeting scalable linear antenna arrays is presented in [11] and elaborated in [12]. This architecture, called Hydra, implements a two-stage beamforming algorithm, using interchangeable Head modules with radio frequency front-ends, Spine modules with digital maximum ratio combining (MRC) beamforming, and a Tail module with frequency-dependent zero-forcing (ZF) beamforming. Heads and Spines are paired together, and then Spines are daisy-chained together, solving the scalability problem. Bandwidth reduction is achieved in this algorithm by making the links between Spines and Tail proportional only to the number of users served in the system (K in Fig. 1.1), rather than the number of antennas (P in Fig. 1.1), which must be a multiple of the number of users for good per-user performance.

An E-band hardware implementation of this architecture is presented in [13]. Commercial off-the-shelf (COTS) components are used for the Head modules, and VCU118 field-programmable gate arrays (FPGAs) implement the logic for the Spine modules. Per-module beamformed data are aggregated on a PC that emulates the Tail logic. While this demonstrates the scalability of the architecture, it is still not practical for a real-world system, due to its high power consumption and latency compared to the low throughput (equivalent to only 10's of MHz of bandwidth). For comparison, numerous other testbeds are also limited in bandwidth and consume upwards of 1kW (if disclosed at all) [14]–[17].

Significant effort has been made in shrinking these testbeds with custom application-specific integrated circuits (ASICs). To-date, many published papers have claimed to support massive MIMO at a wide variety of frequencies, some well above 100GHz. However, they have generally come with significant caveats, such as only containing the RF-to-analog baseband chain (RF beamforming), only the analog baseband-to-bits chain, only generating a small
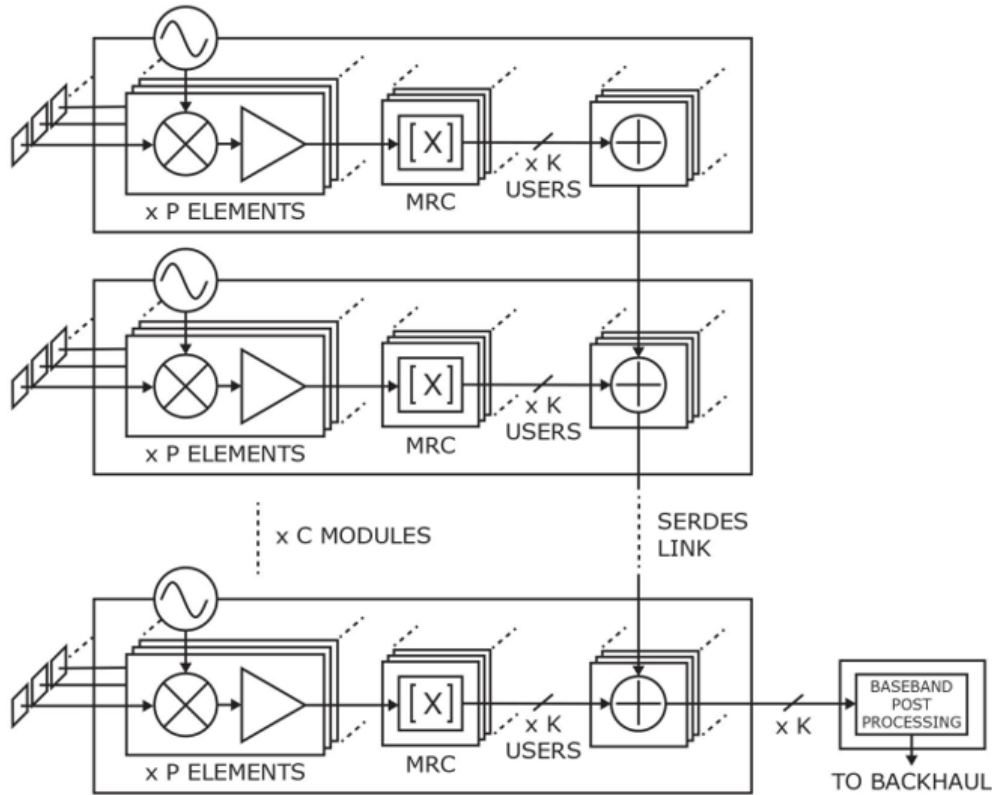
Figure 1.1: Hydra module architecture, reproduced from [13].

number of beams (e.g., only two afforded by H/V polarization), or supporting low bandwidth. Only [18] has demonstrated a fully-integrated (RF-to-bits) digital beamforming array for 16 antennas (in a 4×4 array) and 4 simultaneous beams at 28GHz for 1GHz bandwidth–albeit only as a receiver (i.e., uplink only). Clearly, there remains a significant gap between the state-of-the-art and the theoretical limits of massive MIMO.

1.2.2 Generator-based and Agile Design

With the complexity of the chips needed to support massive MIMO and the relatively small amount of design resources allocated to wireless circuits, it is critical to employ development methodologies that dramatically improve design productivity. To this end, generator-based design contrasts with traditional methodologies by developing a large class of highly-parameterized designs from which one can generate many distinct instances. This allows for rapid design iteration without locking-in final parameters until more information is available–a key principle in the agile software design methodology as described in the Agile Manifesto [19].

UC Berkeley has since developed a menu of generators and implementation tools that en-

able agile *hardware* development. Chisel [20] is a Scala-based hardware description language, upon which generators such as the RocketChip [21] RISC-V [22] processor core are written. Hardware generators like this are integrated into the Chipyard [23] platform, a one-stop shop for SoC designers to write generators, simulate them, and implement them on FPGAs (with FireSim [24]) or ASICs (with Hammer [25], described in detail in Chapter 3). Beyond digital design, Berkeley has also developed the Berkeley Analog Generator (BAG) [26], an analog circuit generator framework that dramatically reduces the time it takes to create analog IP and port them to new process technologies. The work described in Chapters 2, 4, and 5 all extensively employ combinations of these tools for significant design productivity gains.

1.2.3 Heterogeneous Integration

In Chapter 2, the case is made for heterogeneous integration (HI) to make massive MIMO a reality. HI, as a concept, especially applied to phased arrays, is nothing new. In fact, Gordon Moore, in a follow-up to his seminal Moore's Law paper [27], described:

"It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected."

and:

"The successful realization of such items as phased-array antennas, for example, using a multiplicy of integrated microwave power sources, could completely revolutionize radar."

Currently, industry is investing heavily into enabling HI for a wide variety of applications. Led by advancements in wafer-level processing and packaging, chiplets are now able to be placed and connected very closely together, unlocking significant performance boosts.

HI architectures can be categorized by the following paradigms [28]: scale-up, split, disaggregate, and aggregate. In the scale-up paradigm, capabilities can be expanded by tiling identical compute units into an array, as exemplified by [29] and [30]. In the split paradigm, an existing design is chopped up into separate chiplets, usually driven by the need to exceed the reticle limit or improve chip yield, as exemplified by [31] and [32]. In the disaggregate paradigm, different components of a design, such as cache memory or I/O, are split into their own chiplets, thereby optimizing for process technology and/or data movement, as exemplified by [33] and [34]. In the aggregate paradigm, a system is built-up from modular functional units to achieve maximum performance and to enable entirely new applications, as exemplified by [35]. If done correctly, the possibilities for IP reuse (in the form of chiplets in an ecosystem) and new capabilities are unbounded in the aggregate paradigm, limited only by resources and industry's will to coalesce around chiplet standards. This is explored in [36], whose radio cube concept forms a significant basis for this dissertation.

1.3 Dissertation Scope and Outline

Motivated by the combination of these disparate background areas, this dissertation considers an exploratory journey from the agile design of baseband processing ASICs to the design of

standardized 3D die-to-die interconnects as key technologies needed for the realization of two-dimensionally scalable sub-THz massive MIMO planar arrays. Chapter 2 describes the design of two baseband processing ASICs for a massive MIMO system and the challenges faced in their design and bringup. It concludes by rethinking the scaling and integration problem, motivating the work later in this dissertation. Chapter 3 describes the improvements made to a VLSI flow tool to enable the designs described in Chapters 2 and 5. Chapter 4 discusses understanding the fundamentals of standardizing vertical interconnect, which is implemented in a Chisel generator as described in Chapter 5. Finally, Chapter 6 presents a 3D-integrated massive MIMO system concept given tradeoffs between power delivery schemes and antenna placement, presents a summary of contributions, and discusses future work to make this vision a reality.

Chapter 2

A Tale of Two Chips

2.1 Introduction

In order to realize real-time, low-latency, and high-throughput operation, custom ASICs are required for the Head and Spine modules of the Hydra architecture [11]. Shrinking the components down dramatically reduces power consumption and allows for higher-frequency computation, which directly translates to higher bandwidth. The sections below describe the design of Spine ASICs in two different FinFET processes, and the challenges faced in their design and bringup. The intended system realization of the Head and Spine ASICs is shown in Figure 2.1, where an array of packaged Head ASICs, such as those in [37], are paired with an array of packaged Spine ASICs on a PCB. The daisy-chain links are implemented as high-speed serial links (SerDes), while the packages are arrayed precisely to satisfy the $\lambda/2$ spacing requirement for the antenna array.
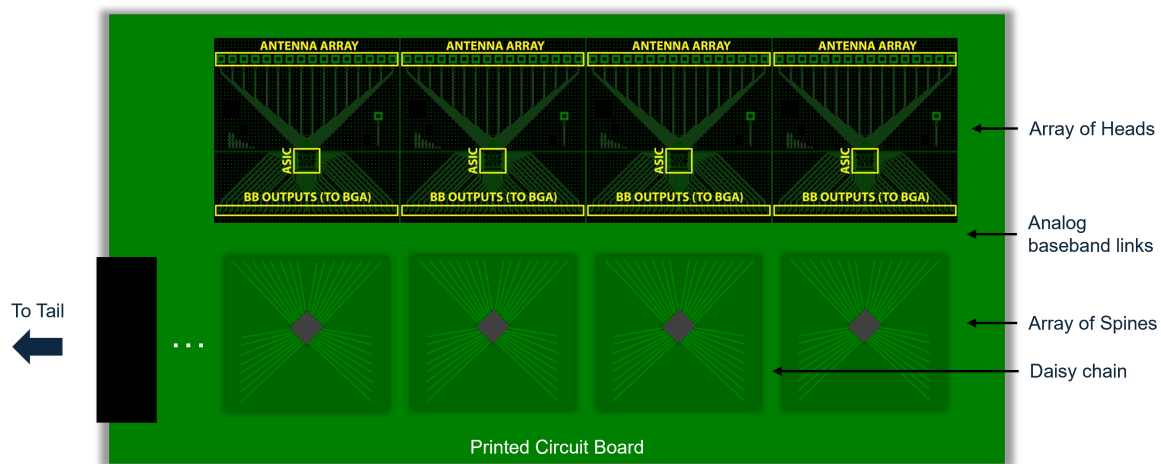
Figure 2.1: Head/Spine integration concept.
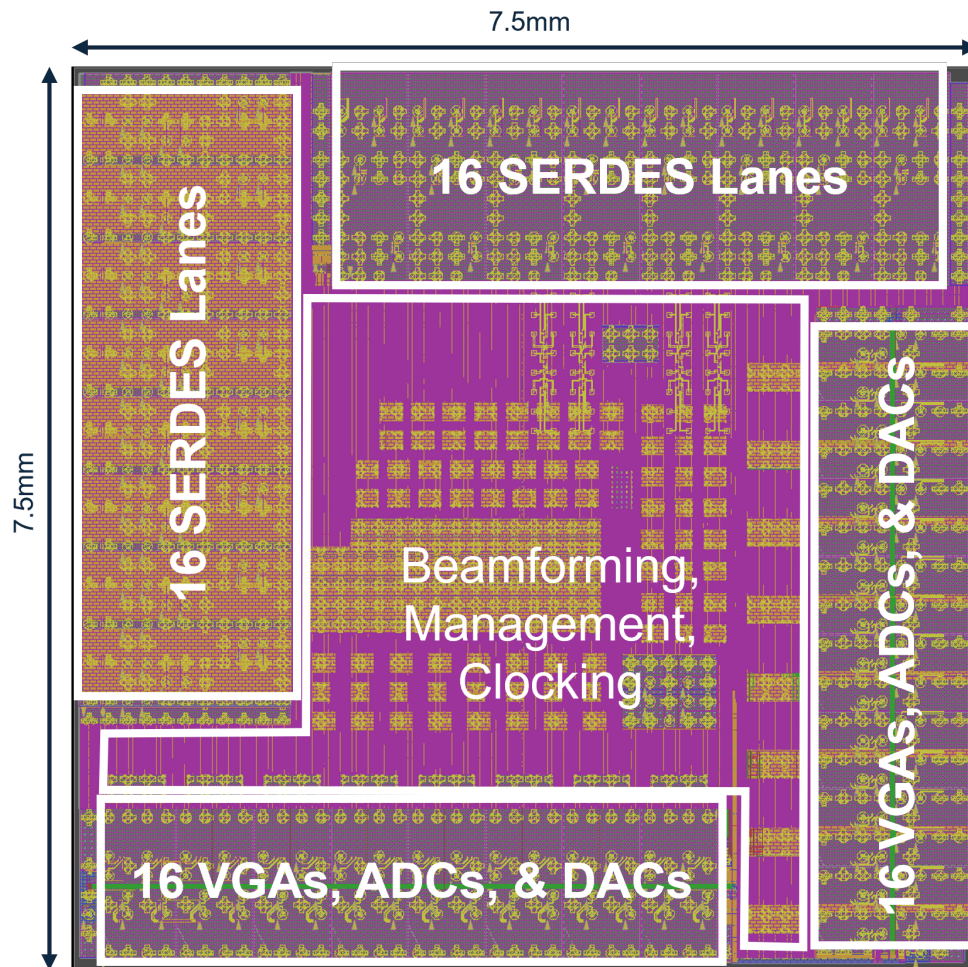
Figure 2.2: Hydra Spine I final annotated floorplan.

2.2 Hydra Spine I

The first Spine ASIC was an ambitious design in the TSMC 16FFC process technology on a 7.5mm × 7.5mm die. Targeting 16 antennas and 16 simultaneous users at 1 GHz of bandwidth, multiple innovations were implemented in the design of this ASIC given the limited personnel and time constraints (<8 months from conception to tapeout). Figure 2.2 shows a high-level floorplan of the ASIC, with the beamforming, management, and clocking circuitry [1] surrounded by slices of VGAs [2], ADCs [3], DACs [4], and SerDes lanes [5].

---

[1]Credit: Zhenghan Lin
[2]Credit: Ethan Chou
[3]Credit: Zhaokai Liu
[4]Credit: Amin Torabi
[5]Credit: Zhongkai Wang and Kevin Zheng, with a digital protocol wrapper adapted from [38]

This floorplan is required in the context of the integration concept in Fig. 2.1 (note the 45° rotated chips on the Spine package). The flow of data from the mixed-signal portion of the chip to/from the SerDes requires placing them on opposite sides of the chip, while the SerDes daisy chain drives the need to place them on adjacent edges. The remaining digital signal processing (DSP) logic can be placed in the remaining central portion of the chip, while low-speed I/Os can occupy available bumps in the internal area, as supported by packaging. Configuration of all the mixed-signal and DSP blocks is accomplished by connecting memory-mapped registers to the peripheral bus of a RocketChip subsystem [21].

2.2.1 Design of DSP Blocks in Chisel

The Spine module's primary functions are to perform channel estimation, maximum ratio combining (MRC) beamforming, and signal aggregation across the daisy chain. To implement this logic, DSP blocks were designed in the Chisel hardware description language [20] using special digital signal processing libraries [39]. Extensive parameterization with physical design in mind was encoded into the generators to allow for rapid iteration on the design, such as the ability to scale the number of antennas and users, datapath parallelization to hit throughput targets, and numerical precision (fixed point formats) to achieve timing while minimizing quantization errors. Towards this project, my block-level contributions were primarily for timing correction and inter-module synchronization. For fractional time correction, a Lagrange polynomial interpolator estimates the required time delay from the output of a Golay correlator, which is then fed into a finite impulse response (FIR) filter structure for iterative time shifting [11]. Integer time correction between Spine modules is implemented with a beacon signal-triggered buffer chain between the SerDes lanes on opposite banks. The rest of the block generators are written by other team members [6] and the open-source generator [7] is described in detail in [40]. Fig. 2.3 shows the major DSP blocks and its connection to the RocketChip system.

For orchestrating the streaming data in accordance with the test plan, a set of streaming memory and mux blocks is designed specifically to consume, generate, and redirect samples. One set of antenna-domain blocks are inserted between the ADC/DAC slices and DSP, while another set of user-domain blocks are inserted between the DSP and SerDes lanes. Figure 2.4 shows the block diagram of the streaming memory and mux block. The storage element is implemented as an array of dual-port SRAMs to support simultaneous reads and writes, needed for testing a subset of the DSP chain. Address and data orchestration, such as the ability to provide a starting and stopping address, repetition loops, and more is enabled by overlaying an AXI4-Stream [41] direct memory access (DMA) wrapper [8]. To load and read out the memory during testing, the memory is also connected to the higher-level Rocket subsystem via AXI4 to Tilelink [42] converters, which are not active while the block is operating in streaming mode.

---

[6]Yue Dai, James Dunn, Victor Han
[7]`https://github.com/ucb-ee290c/fa18-mimo`
[8]Credit: Paul Rigge
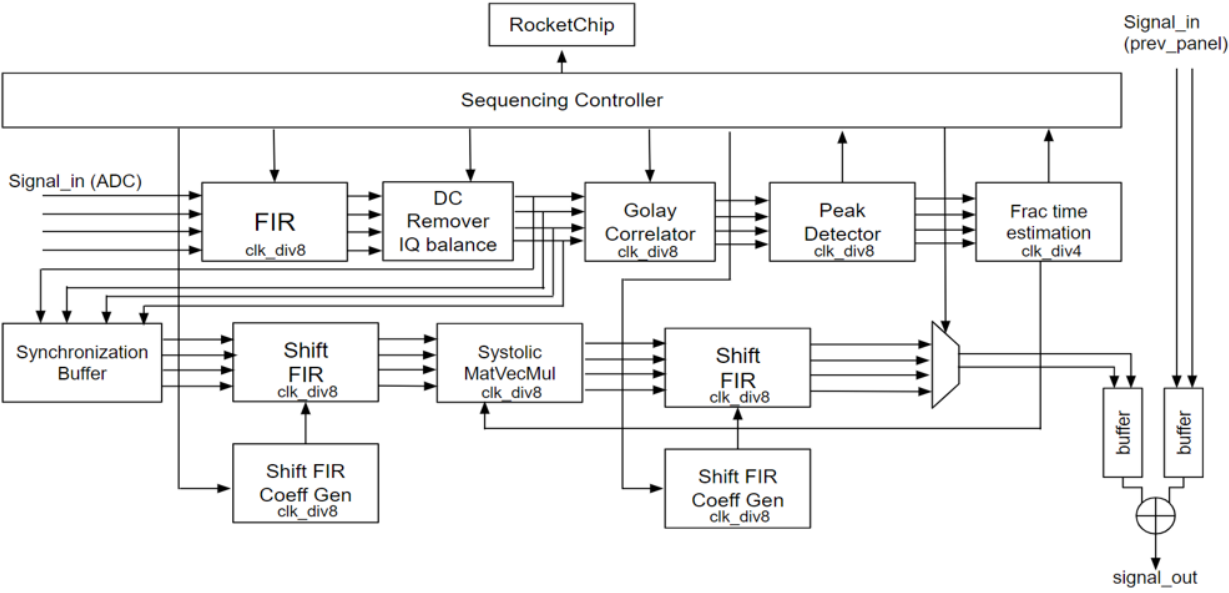
Figure 2.3: Hydra Spine I DSP architectural block diagram.
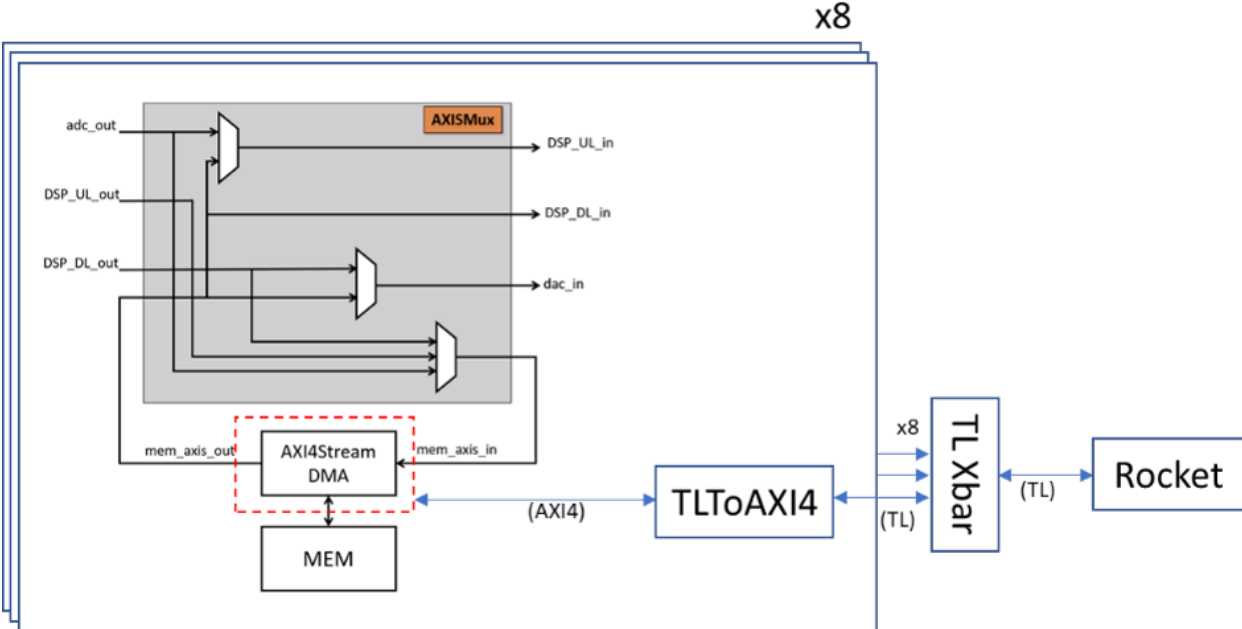
Figure 2.4: Streaming memory and mux block for sample-rate data orchestration.
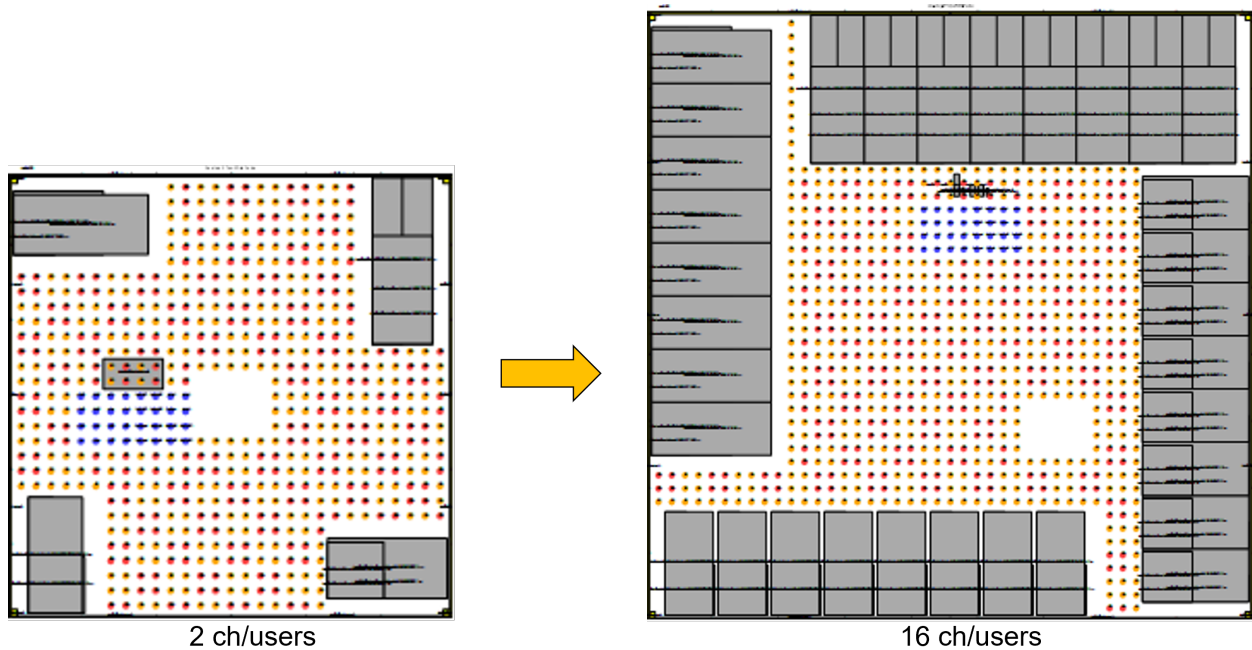
Figure 2.5: Visualized floorplan generator, scaling from 2×2 to 16×16 configuration.

2.2.2   Agile Physical Design

The physical design of this chip was performed in a highly-agile fashion using the Hammer physical design generator framework, described in Chapter 3 and [25]. Given the size of the chip relative to the design resources available, it is prohibitively time-consuming to perform a full-chip placement and routing for every iteration of pipe-cleaning the design and flow. In addition to using a hierarchical design methodology, iteration was performed on a chip of reduced size, targeting 2 antennas and 2 users. To enable this, a Python floorplan generator was developed to seamlessly generate scalable floorplans from this 2×2 configuration to the full 16×16 configuration. The generator outputs information for placement of ADC/DAC slices, SerDes slices, DSP block instances, SRAMs, bumps, and more. An example of the visualized output (see Section 3.4.1 for a description of this visualization tool) at the highest level of hierarchy for the 2×2 versus 16×16 floorplans is shown in Fig. 2.5.

Simultaneously, mixed-signal integration was performed in parallel and continuously while the analog blocks were being designed. Using the Berkeley Analog Generator (BAG) [26]—a Python-based analog generator—extended with custom SKILL and Python scripts, blackbox templates of the analog blocks were generated as a full set of abstracts (LEF, LIB, Verilog, etc.) for integration into the physical design flow [9]. In particular, an API for generating timing information allowed the designers to encode characterization intent by specifying the pin type, which sets up simulation sweeps and measurements for parameters

---

[9]Credit: Greg LaCaille and myself

such as output delay and setup/hold time, which are then passed to an internal Python tool called dotlibber[10] that writes the final Liberty files. Iteration on the top-level continuously informed the analog design, such as floorplan optimization to minimize parasitics given the constrained bump placement of differential input/output pairs and the placement of configuration and datapath pins for maximum routability. High-speed clock distribution was accomplished via transmission line abutment using custom routing DEFs instantiated at the hierarchical module wrapping each channel's group of ADCs/DACs, for example, thereby allowing the floorplan to scale up without redesigning the clock distribution network. Though this method results in phase offsets between each channel, the timing correction built-in to the DSP compensates fully for this. With this methodology, as the analog blocks were incrementally completed, integration became nearly seamless as updated layouts were drop-in replacements, thereby allowing the bulk of the physical design effort to be spent on optimizing the placement of the DSP logic and achieving a DRC/LVS clean flow.

2.2.3   Physical Design Challenges

The development of this chip significantly advanced methodologies for design space exploration (DSE), but, unfortunately, this chip was never taped out due to unresolved routing congestion. There were two primary contributors to this. First, the data movement problem never truly went away; instead, it was shifted to inside the chip, and made worse (in terms of required routing resources) by datapath parallelization. Figure 2.6 shows the floorplan of the chip with yellow arrows denoting the per-channel and per-user flow of data in the uplink direction from the ADCs to the SerDes lanes. Each of these arrows represents a 16-bit complex value, parallelized by a factor of 8 for throughput–in essence, a 128-bit bus per channel. Congested regions are shown in red, predominantly surrounding the beamformer and channel estimation logic. It is evident that compression of the routing around the beamformer requires very precise placement of the pins on the beamformer hierarchical sub-block, which was not optimized properly in the final stages when the floorplan was scaled up from 2×2 to 16×16 due to the use of automatic pin spreading algorithms which do not localize each 128-bit bus along the edge. Contributing to this problem is the granularity of the muxing to/from the streaming memory block in Fig. 2.4–nearly every block in the DSP chain had entry and exit points for maximum visibility, causing the high-speed routing resource requirement to more than double. Furthermore, the diagonal data traversing from the left bank of SerDes to the top bank of SerDes competes with the user-domain DSP logic for routing resources.

Second, a network on chip (NoC) generator was not available at the time of this design, but has since been developed and is described in [43]. Each pink block in Figure 2.6 represents a hierarchical sub-block with Tilelink interfaces for configuration on the peripheral bus of the RocketChip subsystem. Connecting all of these blocks with a single crossbar utilizes significant routing resources, regardless of the speed of the bus. Hierarchical crossbars are

---

[10]Credit: John Wright, `https://github.com/jwright6323/dotlibber`
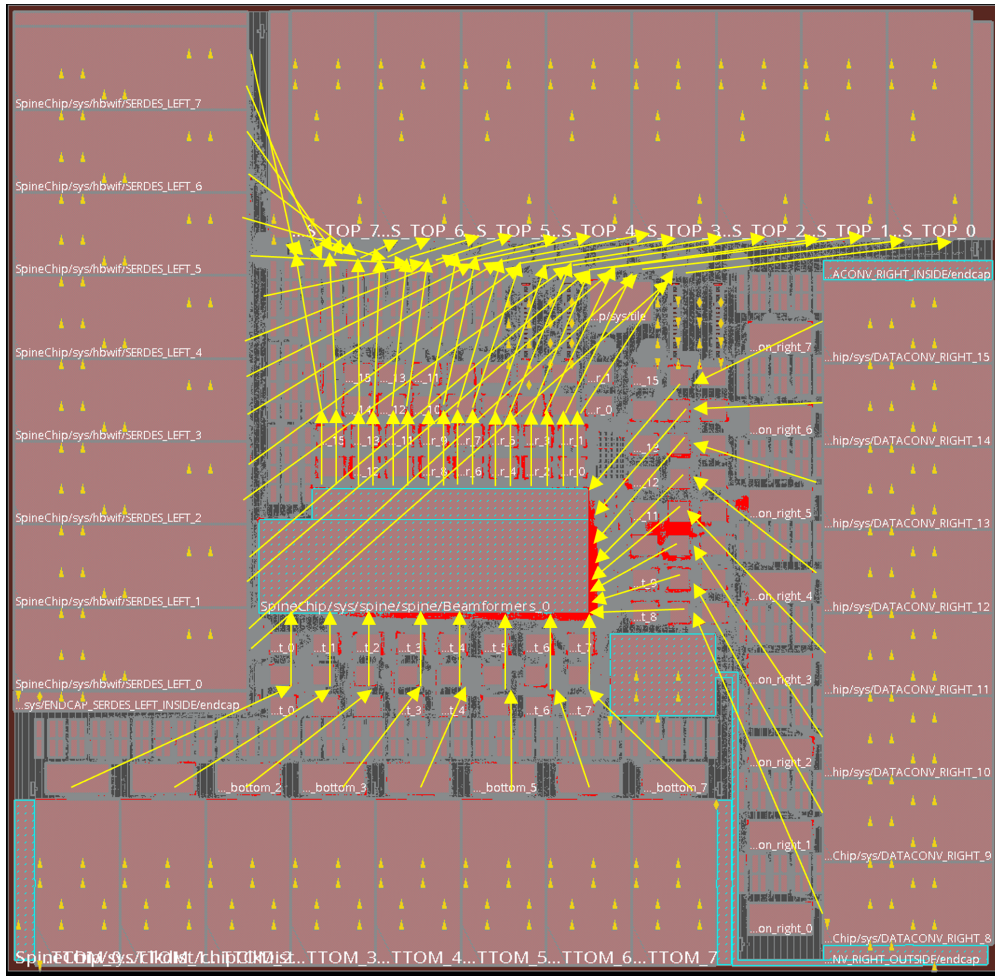
Figure 2.6: Congestion (red) and uplink dataflow (yellow).

insufficient to solve this problem, because their tree-like structure does not accurately reflect the physical layout of the bus' endpoints in the floorplan of this chip. Only a NoC with a predominantly mesh topology would have been able to confine the bus routing to point-to-point connections, significantly reducing the competition for routing resources between datapath and configuration buses.

2.3 Hydra Spine II

As a follow-on to the DSE experiments of Hydra Spine I, a second attempt was made to realize this chip. In this iteration, the process technology changed to Intel 22FFL [44] (see Sec. 3.4.2.1) and the die size was shrunken down to approximately 4mm × 8mm. Combined with a design team that was cut in more than half, this chip reduced the capability down to
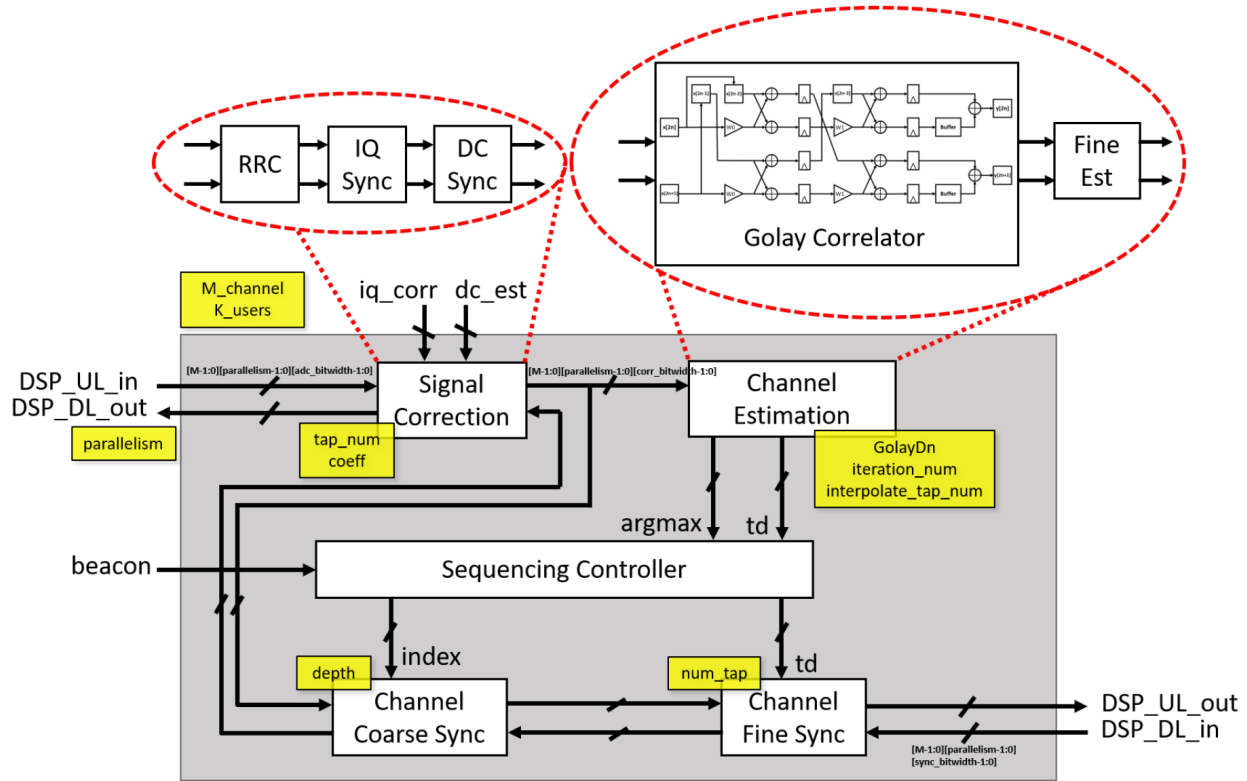
Figure 2.7: Hydra Spine II DSP architectural block diagram.

serving 8 antennas and 8 simultaneous users. The DSP block architecture and final floorplan are shown in Figs. 2.7 and 2.8.

The floorplan is arranged very differently from Hydra Spine I for a number of reasons: 1) the SerDes was unable to be ported in time nor was a 2.5D die-to-die link available; 2) the intent was to flip the chip directly onto a PCB instead of an organic substrate, thereby confining I/Os closer to the edge and precluding inclusion of a beamforming matrix; and 3) to prevent the congestion issues experienced in Hydra Spine I. Streaming memory was increased in size to account for the inability to stream data through the complete datapath and is paired with antenna-domain DSP (predominantly channel estimation and front-end correction) in each horizontal slice (one of which is highlighted in Fig. 2.8). The logical hierarchy of the chip had to be significantly revised to support this highly-channelized floorplan, in contrast to the previous chip where the DSP chain was all placed in the center of the chip. Thankfully, using custom FIRRTL [45] passes, the logical hierarchy can be rearranged at compile time, allowing the hierarchies in Chisel and physical design to diverge without significantly affecting verification difficulty. Beyond the agile design methodologies employed for Hydra Spine I, physical design lessons learned were also applied, such as more compact pin placement, better-partitioned hierarchical boundary timing constraints, and soft placement guiding for improved timing closure between the datapath and streaming memories.
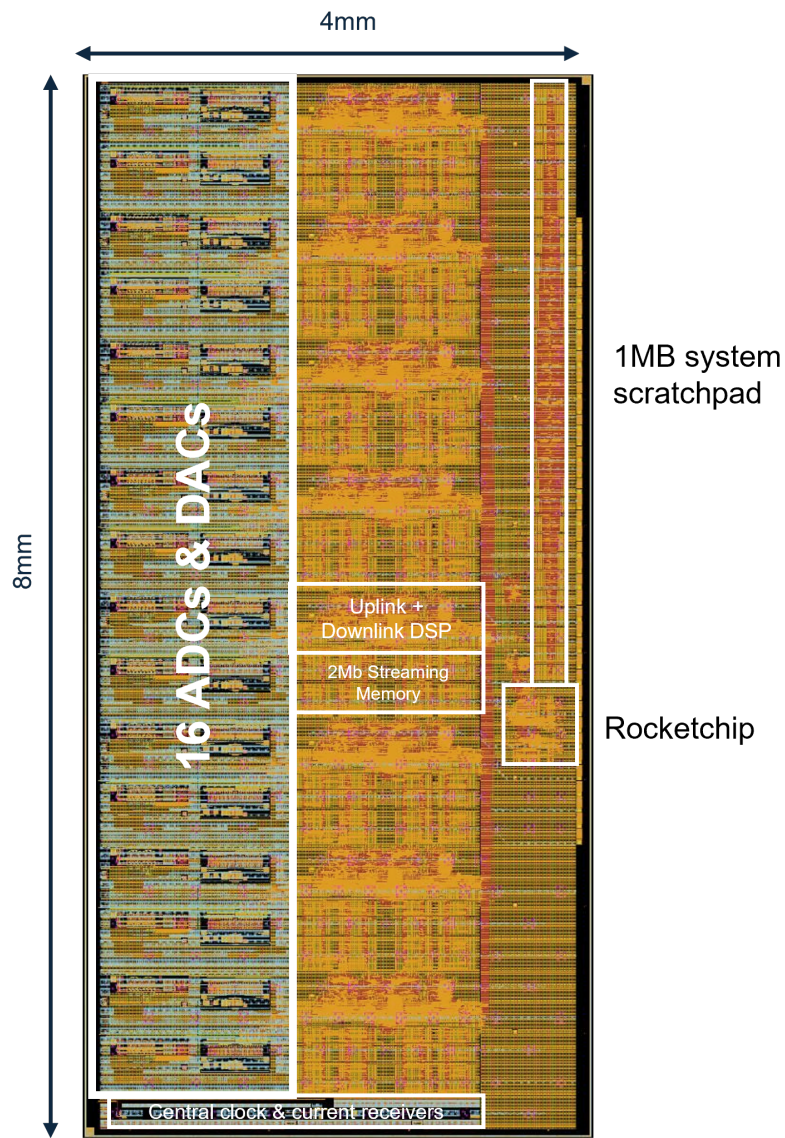
Figure 2.8: Hydra Spine II final die photo and annotated floorplan.
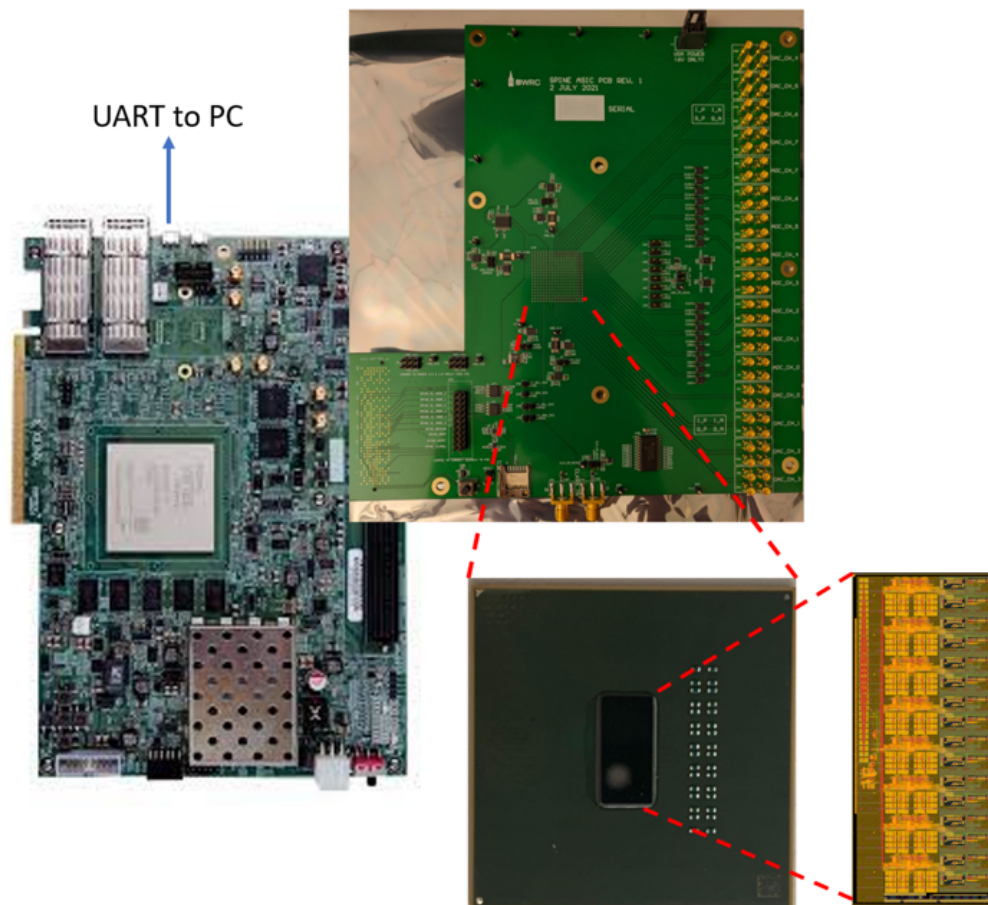
Figure 2.9: Hydra Spine II board-level test setup.

2.3.1 Results

The chip was mounted onto a 5mm × 5mm Kyocera organic substrate package, which was in turn mounted onto a daughter board. This daughter board connects to a VCU118 FPGA development board over an FMC+ connector, which contains a RocketChip soft core that communicates to the chip via a serialized Tilelink link. The daughter board has voltage regulators, current measurement, and VGA chips to support the various power domains and test plan cases. Significant custom software was written to sequence the board-level chips, set the on-chip configuration registers, and generate fixed-point test samples to load into the streaming memories. Fig. 2.9 shows how the board-level test setup was created.

Many system building blocks were functional; however, the complete system was not due to multiple bugs in the digital logic wrapping the mixed-signal blocks and DSP chain. Most egregiously, the asynchronous queues used for the clock crossings between the ADCs/DACs and the DSP chain were misconfigured, with a synchronization depth of 3 erroneously selected for a queue depth of 4. This means that it takes 3 cycles for a write pointer to propagate

to read pointer, causing the queue to fill up before the dequeue side can flush out. The consequence is that the queue applies undesired backpressure, causing the datapath to not be streaming. For the DACs, samples are held for 6 out of every 10 clock cycles, causing the output waveforms to be functional only when filtered down to $1/80^{\text{th}}$ the sample rate. Similarly, for the ADCs, 6 out of every 10 samples are entirely dropped. This, combined with a bug that inadvertently turned off every other slice of the time-interleaved SAR ADC structure means no possible valid stream of samples can be gathered, let alone to be processed by the channel estimation logic within the pre-determined pilot sequence interval.

At design time, this combination of parameters was not protected against in the asynchronous queue generator. A better design choice would have been to use simpler rational crossings because the ADCs/DACs' input/output data is at the same frequency as the DSP. Furthermore, due to time constraints, no behavioral models of the analog blocks were written to enable a full system simulation that exercised the queues, which would have caught these bugs at design time.

2.4 Discussion: Reformulating the Integration Problem

The learnings from these two chips helps reformulate the fundamental problems at hand in the push towards scalable sub-THz massive MIMO systems. It turns out that the scale of integration and shrinking required far exceeds the blueprint of turning Heads and Spines into ASICs and performing board-level integration. Fundamentally, sub-THz massive MIMO has the following "wish list":

1. **Technology and frequency evolution**

   a) The desire for greater per-user capacity is driving the need for higher carrier frequencies at mm-Wave and sub-THz, where 10's of GHz of bandwidth are readily available. At these frequencies, the various components of the system (RF front-end, data conversion, digital baseband) become optimal in different process technologies. Similarly, antennas and antenna transition structures become highly dependent on package substrate materials [46], [47].

   b) Greater bandwidth increases data movement by an order of magnitude, rendering board-level integration increasingly impractical, as the power and area of the inter-module SerDes links would become dominant. See Table 2.1 for a rough calculation of the power required for a 16 antenna and 16 user Hydra Spine at a conservative 4GHz symmetrical uplink/downlink bandwidth at 6 bits for I/Q data in a system where the data converters must be placed on a separate die from the digital baseband. The benefits of using 3D interconnect is clear, and are explored and analyzed in Chapters 4 and 5.

2. **Scalability for different applications**

   a) Depending on the requirements of the cell, such as varying environments (indoor/outdoor), user density, mobility, and required throughput, network operators desire a tailored array deployment. Scalable two-dimensional antenna arrays made from modular sub-arrays can reduce the cost of this variety by enabling design reuse.

   b) The drive towards higher frequencies makes atmospheric attenuation worse, necessitating a denser deployment of base stations to maintain coverage [6]. Consequently, the lifecycle cost of the base station must be reduced, which is dominated by the power consumption of the system.

3. **High reliability and low latency**

   a) Reliability in wireless systems are becoming increasingly important as the applications served diversifies from consumer devices to industrial control, autonomous vehicles, and more [2]. Ensuring that more data packets can be transferred without errors involves increased interference mitigation in the channel via intelligent beamforming, decreased bit error rates in system interconnects, and support for higher bandwidth for packet redundancy.

   b) Low latency is demanded by real-time applications, but also from the shortened coherence time of the channel at higher frequencies at the same mobility speed. Consequently, beam search and tracking must be performed at a higher rate, which requires a more tightly integrated system.

Table 2.1: State-of-the-art interconnect comparison.

Reach (application)	Architecture	Total power (W)	Total area (mm²)
Long (backplane)	112Gb/s serial [48]	20.7	20
Extra-short (on-package)	112Gb/s serial [49]	5.7	6.1
2.5D (Die-to-die, Si bridge)	8Gb/s parallel [50]	1.7	2.6
3D (Die-to-die, stacked)	16Gb/s parallel [50]	1.38	0.26
3D (Die-to-die, stacked)	2.8Gb/s parallel [35]	0.92	2.3

2.4.1   The $\lambda/2$ Dilemma

In the pursuit of satisfying this "wish list", the most pressing limitation towards the construction of sub-THz wireless system comes from the $\lambda/2$ antenna spacing requirement as

enumerated from Sec. 1.2. For two-dimensional arrays to be modular and scalable, antenna-domain circuits and processing—from the RF front-end to the digital baseband—must be integrated within the $(\lambda/2)^2$ area. To date, fully-integrated arrays have been demonstrated for the FR2 bands of 5G NR [51] and implicitly show that the chip area is already pushing the $\lambda/2$ spacing limit. Arrays at higher frequencies have been demonstrated, but contain only the RF front-end and employ analog beamforming, thereby limiting the number of achievable beams. Beyond the 60GHz band, phased array and beamforming ICs have not been able to keep up with with $\lambda/2$ scaling without employing tricks such as dummy antenna insertion and antenna duplication, resulting in undesired effects like grating lobes and severely-limited beam scanning [9].

It is important to note that this $\lambda/2$ scaling requirement is only true for *phased arrays*; true time delay (TTD) arrays [52], [53] can theoretically place antennas in any pattern because the system manages wavefront alignment in the time domain, which is approximated by the phase shift at the carrier frequency in phased arrays. However, significant gaps still exist in compensating for clocking mismatches with sufficient resolution versus range and being able to steer more than one simultaneous beam [6].

Thus, as RF circuit research continues well into the 200GHz+ bands, where antenna spacing falls well below 1mm, an acute area constraint arises for the integration of all the domains needed in a wireless system. Therefore, towards the goal of scalable systems employing fully-digital beamforming in uniform arrays, it is instructive to examine the scaling roadmaps of these domains, summarized in Table 2.2.

2.4.2   A Solution: Tiled Radio Cubes

Beyond the inability for each domain in Table 2.2 to fit within the same $(\lambda/2)^2$ area, each domain is optimized for different process technologies, driving the need for disaggregation. One method of creating two-dimensional arrays is by stacking flexible PCBs [65] so as to maintain linear scaling on each individual PCBs, allowing for scaling perpendicular to the PCB plane. However, this method is impractical for scaling to large arrays as the flexure at the ends becomes too extreme and thermal dissipation becomes nearly impossible. Instead, the 3D-integrated radio cube concept from [36] is significantly more practical and lines up better with industry's drive towards HI. In this concept, the RFIC, mixed-signal IC, and basedband digital chips are 3D-stacked on top of a package substrate with a 2D array of embedded antennas. Stitching together these sub-arrays with 2.5D die-to-die interconnects achieves the required array scaling into a system-on-package, as conceptualized in Fig. 2.10.

Many key technologies are required for this to work. Thankfully, industry is on-track to enabling HI with novel devices, wafer-level processing, tool support, and a chiplet design ecosystem [66]. However, in order to integrate all the domains in the 3D stack, 3D interconnect must be standardized and easily constructable in the wide variety of process technologies required in this system. Towards this goal, Chapters 4 and 5 will describe the development of a standardized 3D interconnect generator from first principles analysis of technology scaling roadmaps.

Table 2.2: Roadmaps: Can Everything Fit Inside $(\lambda/2)^2$?

Domain	Yes	No	Outlook
Front-End (RF, Antennas)	• Lower output power with array scaling • Transmission line-based design [54], [55] • Significant antenna R&D [56], [57]	• Poor power amplifier efficiency [58] • Little benefit from technology scaling	OK for now, everybody is thinking about it
Mixed-signal (ADC/DAC)	• Converging with wireline requirements [59] • Increasingly digital architectures [59]	• Technology scaling $\neq$ area scaling (due to interleaving, pipelining)	OK for now, but few are thinking about it
Digital (Baseband DSP)	• Direct benefit from technology scaling • Increasingly-efficient and low-resolution decentralized algorithms [60], [61]	• ML-aided algorithms are increasingly memory-intensive [62]	Great!
Power Conversion	• Significant R&D into on-chip/in-package passives [63]	• Needs new conversion topologies and control schemes [64]	Barely existent now, but R&D is driven by warehouse-scale AI
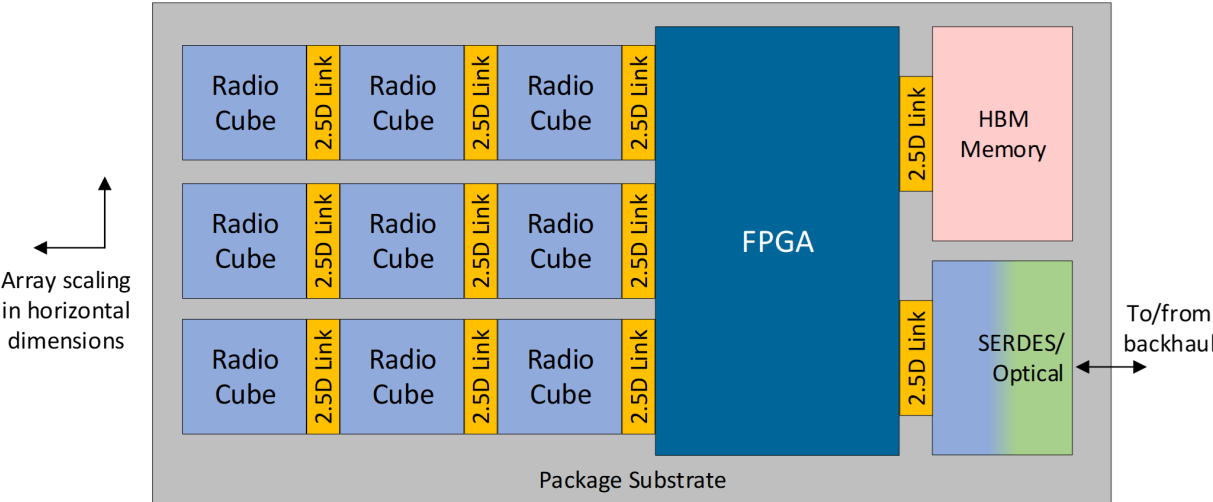
Figure 2.10: Concept of tiling the radio cubes from [36] into a scalable 2D array.

Chapter 3

Hammer: Enhancements to a Modular and Reusable VLSI Flow Tool

Process technology scaling and hardware architecture specialization have vastly increased the need for chip design space exploration, while optimizing for power, performance, and area. Hammer[1] is an open-source, reusable physical design (PD) flow generator that reduces design effort and increases portability by enforcing a separation among design-, tool-, and process technology-specific concerns with a modular software architecture. In this work, we outline Hammer's structure and highlight recent extensions that support both physical chip designers and hardware architects evaluating the merit and feasibility of their proposed designs. This is accomplished through the integration of more tools and process technologies—some open-source—and the designer-driven development of flow step generators. An evaluation of chip designs in process technologies ranging from 130nm down to 12nm across a series of RISC-V-based chips shows how Hammer-generated flows are reusable and enable efficient optimization for diverse applications.

3.1   Introduction

Demand for custom silicon has skyrocketed with the proliferation of specialized domains including AI/ML, IoT, AR/VR, and autonomous vehicles. Yet, non-recurring engineering (NRE) costs have exploded with every generation of chips due to technological advancements (e.g., transistor scaling, specialized architectures, and design heterogeneity) and global forces (e.g., trade restrictions and manufacturing shortages). It is therefore important for the semiconductor industry to adapt to these challenges by increasing the productivity of PD flows.

---

[1]`https://github.com/ucb-bar/hammer`

Towards this goal, we proposed Hammer [67], an open-source PD flow generator which demonstrates extensive flow reuse and large reductions in design effort. Since then, Hammer has focused on enhancing architectural design space exploration (DSE), teaching PD in more university courses, and encouraging development by its user base. Hammer's recent integration into the Chipyard framework [23] and expanded plugin library (Table 3.1) now enable full-stack implementation of SoCs. Due to its reusability, Hammer has been used in UC Berkeley for tens of fabricated chips, four student courses, and numerous hardware architecture explorations, some of which are evaluated in this work in Section 3.5.

3.2 Overview

Hammer abstracts away the intricacies of PD flows into a generic form compatible with many computer-aided design (CAD) tools and process technologies. It is a Python framework that invokes underlying tools with necessary options and generated Tcl scripts to perform actions such as logic synthesis (Fig. 3.1). Hammer lowers the barriers to learning and executing PD flows, while encouraging flow reusability across all kinds of hardware designs. To achieve this, Hammer's design principles are:

1. **Separation of concerns**: Hammer decouples concerns specific to tools, technologies, logical design, and physical design from the flow construction itself.

2. **Standardization**: Hammer codifies a data interchange schema through which design constraints, flow options, and database files can be specified and propagated.

3. **Modularity**: Hammer defines abstractions that are implemented by interchangeable and shareable plugins for specific tools and process technologies.

4. **Incremental adoption**: Hammer flows can mix reusable and custom solutions (e.g. from foundries or tool vendors) as needed to accelerate design and implementation.

These principles ensure that PD intent is encoded in a format that is understandable by other users and applicable to other designs, making Hammer-generated flows increasingly reusable over time. Key components in Hammer's software architecture (Fig. 3.1) are:

1. **Hammer intermediate representation (IR)** is the standard configuration data interchange schema. It is serializable in YAML for human readability and JSON for programmatic generation. It uses **metaprogramming**, wherein snippets of IR can use and modify other snippets, transclude files, and much more. Notably, these features expose Hammer IR as an annotation format for higher-level generators (see Sec. 3.7).

2. **Hammer tool & tech plugins** implement Hammer's abstractions that encapsulate CAD tool- and process tech-specific concerns. Plugins inherit common methods from core Hammer classes and supply default Hammer IR configurations. Tool plugins
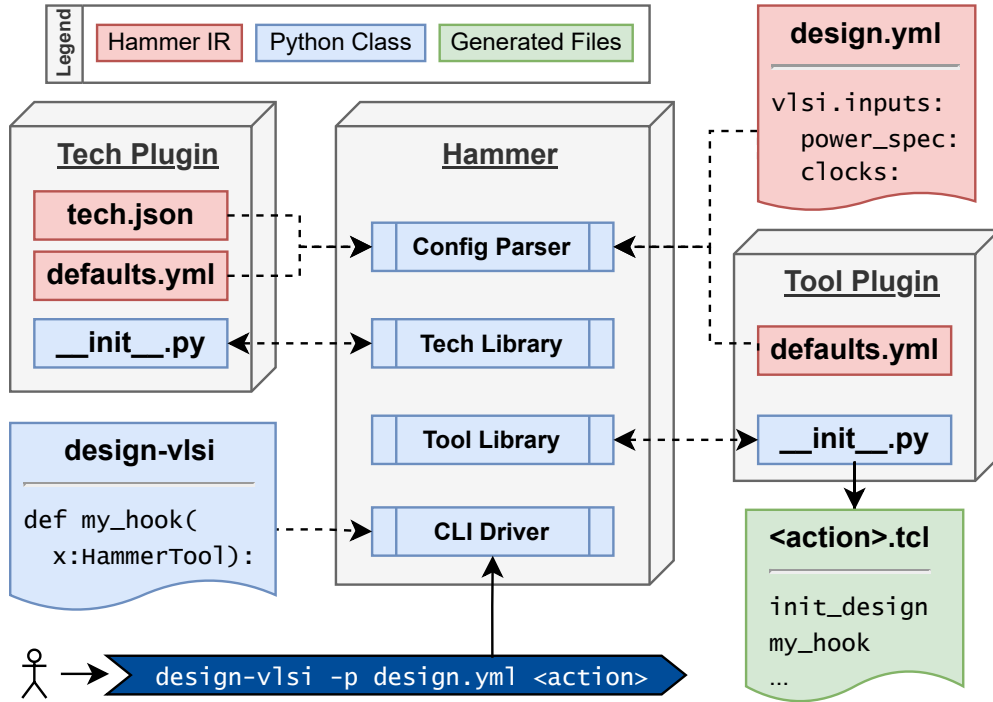
Figure 3.1: Hammer software architecture.

define PD flow steps and methods for generating tool-specific Tcl based on design configurations. Tech plugins enumerate PDK source files and also supply default IR. All supported plugins (Table 3.1) are interoperable, but note that some tech plugins are proprietary.

3. **Hammer hooks** are Python methods that replace, modify, or add to a tool plugin's default flow steps. Users write hooks to inject Tcl to customize the flow as required by a design. Tech plugins may also specify hooks to automatically include commands needed by a given process technology. Hooks promote agile Hammer development, since users write them to incrementally prototype features, before contributing them to plugins and/or core Hammer for reuse.

4. The **Hammer driver** is the command-line and Python interface through which a user orchestrates their flow graph. After the user selects plugins, specifies flow steps to execute, and inserts custom hooks, Hammer generates a set of build dependencies for the entire flow graph as a Make fragment. In a hierarchical flow, Hammer builds a flow graph for each submodule in the physical hierarchy and links them together in the correct order of assembly up to the top level.

Additional high-level APIs enable users to specify complex design-specific features with

Table 3.1: Supported tool (left) and tech (right) plugins.

Action	Tool	Foundry	Node
Logic Synthesis	Genus^C, Yosys, Vivado^X, DC^S	A	16nm FinFET 28nm Planar
Place and Route	Innovus^C, Vivado^X, OpenROAD [68], ICC^S	B	16nm FinFET* 22nm FinFET*
DRC/LVS	Calibre^M, ICV^{S,*}, Pegasus^{C,*}, Magic/Netgen*	C	12nm FinFET 14nm FinFET
Simulation	VCS^S, Xcelium^C	D	28nm SOI
Power, EM/IR	Joules^C, Voltus^{C,*}	Education	ASAP7* [69] FreePDK45
LEC	Conformal^{C,*}, Magic*	Skywater	130nm Planar
STA	Tempus^{C,*}		

^{C} Cadence ^{S} Synopsys ^{M} Siemens Mentor ^{X} Xilinx * Added by the author

simple Hammer IR inputs. Boundary timing constraints and power domains are generic inputs that Hammer translates into standard constraint file formats like Synopsys Design Constraints (SDC) and Universal Power Format (UPF). Power meshes are generated from simple target parameters such as track allocation and density. Hammer combines these with stackup information from the selected tech plugin to automatically calculate legal widths and pitches. These APIs are useful for reducing startup overhead and getting sane early physical feedback.

3.3 Architectural Improvements

The high degree of flow reuse described below in 3.5 was made possible by several architectural improvements to Hammer's internal infrastructure.

3.3.1 Flow Construction and Control

During the course of developing several new technology plugins, it was observed that the quality of the files delivered in the PDK, standard cell libraries, and hard IP libraries varied greatly between technologies. For example, some technology LEFs needed layers updated or changed, some technologies required particular tool-specific settings, and some technologies require post-processing of database files after they are written from a tool. Furthermore, several tools have settings that are not written to checkpoint databases, for example, setting multi-threading on a session-by-session basis. To accommodate all of these requirements, tool plugins were augmented with post-install scripting, all plugins were augmented with tool/tech-specific hooks, and the concept of persistent hooks was introduced.

Post-install scripting occurs after the technology plugin package is loaded, but before the library files are checked for consistency. Python methods can be added the tool plugin's `__init.py__` file to perform any manner of file manipulation, with the "hacked" output file(s) dumped to the technology cache directory. Subsequently, the library listings (TechJSON) in the `.tech.json` file can point to files/directories contained in that cache. New rules for how paths are resolved in TechJSON were created to resolve this and other new cases, and are described in Appendix A.1.

Tool/tech-specific hooks are written in the same manner as user hooks and are meant to encode tool/tech-specific settings/commands that are design-agnostic, and can be paired with options defined in the tool/tech plugin's `defaults.yml`. This way, each new project does not need to copy Tcl code needed to get DRC-clean designs, greatly improving flow reuse. The hooks are loaded and processed when the plugin is loaded, and subsequently become targetable by subsequent hooks. The resolution order for steps and hooks is: 1) tool steps, 2) tool hooks, 3) technology hooks, 4) user hooks, meaning user hooks have the broadest scope and can override any other hook. This order makes sense because tool steps contain the most generic (tech- and design-agnostic) commands while user hooks contain commands specific to the tool, technology, and design.

Persistent hooks are a new concept that lives alongside the step and hook sub-flow construction methodology. Persistent hooks run at specific points during every tool invocation, regardless of whether flow control is used to run a subset of steps and hooks for that tool. The three types are persistent, pre-persistent, and post-persistent. Persistent hooks always run at the start of tool invocation, regardless of what happens afterwards. Pre-persistent hooks always run at the start of tool invocation if the target step/hook is included in or after the resolved list of steps/hooks to run, and are run after persistent hooks. Post-persistent hooks are the inverse, in that they run if the target step/hook is located before the resolved list of steps/hooks to run, and are also run after persistent hooks. Certain tool plugins also write out separate scripts for manual debugging in a GUI window. In this case, persistent hooks are also written to the top of the script, before the checkpoint database is read. Figure 3.2 demonstrates how hooks are specified and resolved into a linear graph. Note that certain syntax structures are omitted for clarity.

Finally, flow control, i.e., the ability for the user to run a subset of the step/hook graph, was improved to accommodate tech-specific hooks and persistent hooks. Figure 3.3 in demonstrates how the resolved graph is re-processed in different cases of flow control for execution with normal and persistent hooks.

3.3.2 Multi-Library Support

In the course of developing the Skywater 130nm plugin, it was observed that the various libraries available (sky130a[2], s8[3], sky130_scl[4], to name a few) were all built to target the

---

[2]`https://github.com/google/skywater-pdk`
[3]SkyWater proprietary
[4]Obtainable from Cadence Support

Process	Resolved Graph

Load config

Load tool

Tool Plugin
```
steps = [step1, step2, step3, …, stepN]

hooks = [
    persistent_hook(toolhook1),
    pre_insertion_hook(step2, toolhook2),
    post_insertion_hook(step3, toolhook3)
]
```

```
toolhook1(p)
   step1
 toolhook2
   step2
   step3
 toolhook3
     ⋮
   stepN
```

Load tech

Technology Plugin
```
hooks = [
    pre_persistent_hook(step1, techhook1),
    replacement_hook(step2, techhook2),
    removal_hook(toolhook3)
]
```

```
toolhook1(p)
techhook1(p)
   step1
 toolhook2
 techhook2
   step3
     ⋮
   stepN
```

Get user hooks

Project Driver
```
hooks = [
    pre_insertion_hook(step1, userhook1),
    post_persistent_hook(techhook2, userhook2),
    pre_persistent_hook(stepN, userhook3)
]
```

```
toolhook1(p)
techhook1(p)
 userhook1
   step1
 toolhook2
 techhook2
userhook2(p)
   step3
     ⋮
userhook3(p)
   stepN
```

Figure 3.2: Resolution of tool, tech, and user hooks.

Command	Executed Graph
`hammer-vlsi … -start_before_step step2 …`	Error (step2 replaced)
`hammer-vlsi … -start_before_step step3 …`	`toolhook1(p)` `userhook2(p)` step3 ⋮ `userhook3(p)` stepN
`hammer-vlsi … -only_step step1 …`	`toolhook1(p)` `techhook1(p)` step1
`hammer-vlsi … -only_step stepN …`	`toolhook1(p)` `userhook2(p)` `userhook3(p)` stepN
`hammer-vlsi … -start_after_step step1` `-stop_before_step step3 …`	`toolhook1(p)` `toolhook2` `techhook2` `userhook2(p)`
`hammer-vlsi … -start_before_step toolhook2` `-stop_after_step techhook2 …`	`toolhook1(p)` `toolhook2` `techhook2` `userhook2(p)`

Figure 3.3: Graph execution with flow control.

same technology, but have different file and naming conventions as they come from different vendors. Other technologies have different "flavors", for example ones that are optimized for power, area, or speed. Only certain combinations of these are compatible, and the user must be given the choice of which libraries to use. To accommodate this, the data structures in the TechJSON were converted[5] to Pydantic models[6]. Not only does Pydantic now provide data validation, but it also allows for construction of the library definition via code instead of

---

[5]Credit: Vighnesh Iyer
[6]`https://pydantic.dev/`

as a static JSON file, while maintaining backwards compatibility with existing tech plugins written in JSON format. Selection of libraries is done via new keys in the tech plugin's `defaults.yml` file. The first example, converting the Sky130 plugin, resulted in a reduction of almost 2,000 lines of code while simultaneously supporting more library options. Listings A.1 and A.2 compare the library specificaition format for an example technology LEF and Verilog file selection.

3.4 Community Contributions

Many new plugins were added to Hammer, as indicated by the $\star$ symbols in Table 3.1. Certain critical plugins were also maintained continuously to keep up with new features, bugfixes, and deprecations. This is a summary of notable direct contributions by the author, in chronological order:

1. Floorplan visualizer

2. IC Validator DRC & LVS

3. ASAP7 and Chipyard demo/tutorial

4. Voltus EM/IR analysis

5. Issue packaging for OpenROAD plugin

6. Conformal logical equivalence checking

7. Magic DRC & Netgen LVS

8. Tempus static timing analysis (STA)

9. Pegasus DRC & LVS

Below are some expanded details about some of these contributions. The author also orchestrated many other contributions from the community and undergraduate mentees which are not listed here.

3.4.1 Floorplan Visualization

During the course of Hydra Spine I (Sec. 2.2) development, it was necessary to obtain quick visual feedback from the floorplan generator to ensure that the floorplan as given to Innovus was correct. To this end, a new SVG-based floorplan visualization tool was added to Hammer that emulates the style of the Innovus floorplan view. It can be inserted at any point in the flow, though the behavior is different for PCB deliverable tools. The tool reads in the placement constraints input, and constructs SVG shapes in this order:

1. Print the chip name over rectangles denoting the floorplan bounds and core area. An orientation marker is added to the reference corner.

2. Draw all macro, hierarchical, placement guide, obstruction, and overlap constraints. Dimensions of the macro rectangles are derived from the master definition or the macro's LEF definition. Text indicating the instance path is overlaid, and can be shortened to fit (e.g., `Top/path/to/inst` as `T/p/t/inst`) and a diagonal orientation line is drawn on top to indicate any rotation/mirroring. Macros take precedence over obstructions for clarity.

3. Draw all bumps and associated port names. Bump radius is derived from the bump's LEF. A grid indicating every 5th bump is added to the periphery to keep track of the bump coordinates. For PCB deliverable tools, pad designators are also overlaid.

Note that the coordinate system for SVG is different from chip design–the origin is at the top left, and the Y-axis increases downwards, requiring all Y-coordinates to be mirrored and translated accordingly. For PCB deliverable tools, it is assumed that this visualization is being used for flip-chips mirrored about the X-axis, hence X-coordinates also need to be mirrored and translated. Placement constraints are not relevant in PCB design, so only the chip name and bump locations are drawn. A sample output is shown in Figure 3.4a for the chip and Figure 3.4b for the PCB.

3.4.2 Technology Plugins

3.4.2.1 Intel 22nm FFL

Hydra Spine II (Sec. 2.3) was the chip that drove development of the Intel 22nm FFL plugin. Since this was the first widely available Intel Foundry process technology, the hook infrastructure described in 3.3.1 was directly improved during the development of this plugin. This plugin was used in the 2023 and 2024 versions of the EE194/290 tapeout course at UC Berkeley, where a total of 6 chips were taped out by predominantly undergraduate students. A handful of other universities have also used it for courses and research.

3.4.2.2 ASAP7

Support for the ASAP7 predictive PDK [69] was added with the intention of demonstrating Hammer to the world via tutorials. At the time, ASAP7 was only delivered with a basic standard cell library–no IP, such as SRAM. In order to make a proper tutorial within Chipyard, a dummy SRAM compiler—a Python package called Phyrilog[7]—was adapted in order to deliver the required collateral files, such as LEF, GDS, LIB, Spice, and Verilog models. It begins by reading in the desired SRAM configuration (word size, number of words, ports, etc.) and Verilog model and outputs a width, height, and timing estimation roughly scaled

---

[7]Credit: Sean Huang, `https://github.com/sehuang/phyrilog`
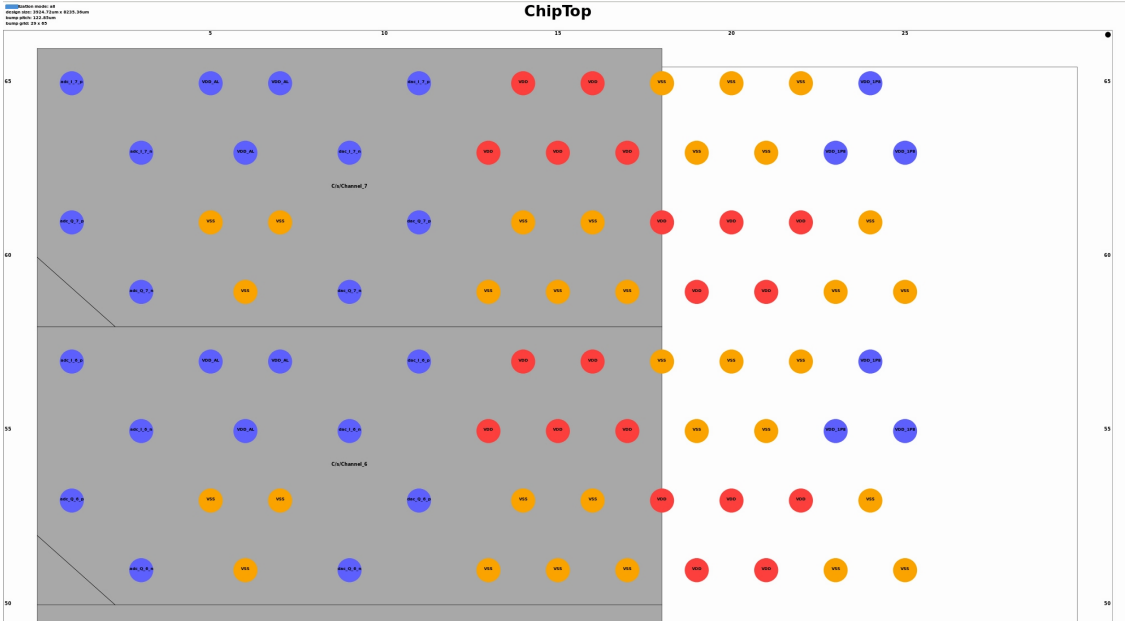
(a) Chip mode
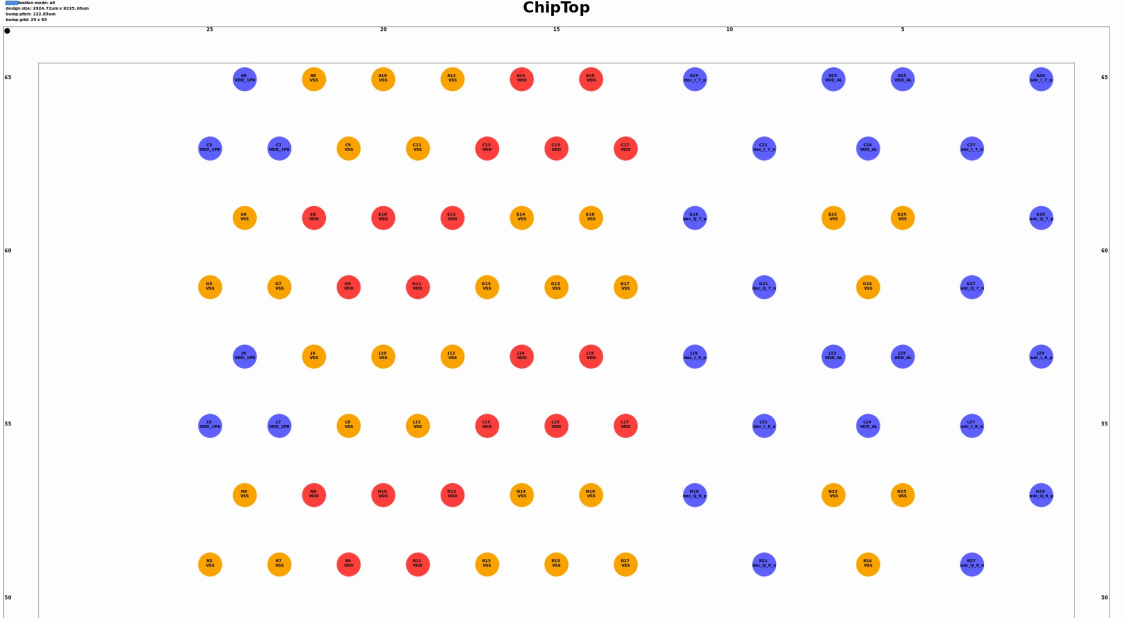
(b) PCB mode

Figure 3.4: Hammer floorplan and bumps SVG visualizer.

down to 7nm from the SAED32 memory compiler [70] using equation fits (Eqn. 3.1 for setup
and hold and 3.2 for output clk-q and transition). Note that these equations have not been

validated against any actual 7nm-class SRAMs, and integration of FinCACTI [71] timing models remains for future work. Timing parameters were written out in Liberty format using dotlibber[8]. Timing is calculation using the following parameters and formulas, which returns the same values for rise and fall and depends, to the first order, on only the number of words in the SRAM.

Table 3.2: Dummy SRAM timing equation parameters.

Parameter	Description	Provided by
t_{rp}	Related pin transition time	Template table
t_{cp}	Constrained pin transition time	Template table
t_{ip}	Input net transition time	Template table
C_o	Total output net capacitance	Template table
R	Global scaling factor	PVT corner
i_1	Template axis 1 scaling factor	PVT corner
i_2	Template axis 2 scaling factor	PVT corner
N	Number of words	SRAM config

$$t_{s,h} = R \cdot i_1^{\ln(10t_{rp})} \cdot i_2^{\ln(10t_{cp})} \left(1 + \frac{N}{10} \right) \tag{3.1}$$

$$t_{ckq,tr} = R \cdot i_1^{\ln(10t_{ip})} \cdot i_2^{\ln(10C_o)} \left(1 + \frac{N}{10} \right) \tag{3.2}$$

The generated SRAMs are mostly devoid of geometry inside; however, pins are spread out along the left edge and power stripes are drawn across the macro on the fourth metal layer, which are written out to the LEF with cover obstructions on layers below. Dimensions are also scaled down directly from SAED32; integration of area estimates from FinCACTI are left for future work.

Resources for the tutorials and labs created with ASAP7 are given in Appendix A.3.

3.4.3   Tool Plugins

3.4.3.1   Voltus IR Drop Analysis

While the Voltus plugin was already able of calculating post-place and route (P&R) power[9], Voltus' ability to analyze IR drop is particularly useful to ensure that the power grid is designed properly. To enable the IR drop flow, the technology library must be characterized to generate power grid verification (PGV) files by using more files such as such as Spice

---

[8]Credit: John Wright, `https://github.com/jwright6323/dotlibber`
[9]Credit: Daniel Grubb

models and macro GDSs. The flow also requires a separated-out listing of filler and decapacitance cells, which necessitated an augmentation to the special cells Hammer API that was doubly useful during filler insertion in P&R to hit decapacitance targets. Depending on the required results fidelity, the user can elect to use tech-only PGVs or also generate them for all standard cells and macros. PGVs are generated for every specified analysis corner and re-generated as required, e.g., if a macro has been updated. Finally, the PGVs are combined with input power information to obtain static and dynamic IR drop results. Early rail analysis (without PGVs) was also added to the Innovus plugin as an optional step to get early feedback on power grid design.

A sample output from both tools for a Chipyard TinyRocketConfig is shown in Appendix A.3. Figure A.1a is the Voltus-generated image, while Figure A.1b is early rail results as viewed in Innovus. Note that the macros on the left/right are the dummy ASAP7 SRAMs, which do not have power information, hence the IR drop result is not accurate for those blocks.

3.4.3.2   Other Signoff Tools

At first, DRC and LVS only supported Mentor Graphics (now Siemens) Calibre. However, as various technology plugins were added, delivered without Calibre decks or requiring final signoff with other tools, it became necessary to add more tools, which were Synopsys IC Validator, Cadence Pegasus, Magic, and Netgen. All of these tools have significantly different execution models in batch mode, as they rely less on Tcl scripts as other tools in a physical design flow. For example, IC Validator prefers to sets deck variables using command line flags, instead of as statements in deck files as in Calibre. Finally, Verilog to netlist conversion and connectors to GUI results viewers were added as required.

Tempus for static timing analysis (STA) and Conformal for logical equivalence checking (LEC) were also added to aid in signoff and verification and are available to run after synthesis and P&R tools. Tempus performs signoff timing analysis, which provides a more accurate calculation of timing paths than in synthesis and P&R. Conformal is used to verify that the synthesized and post-P&R netlists are functionally equivalent, which provides a good sanity check that the design has not been logically altered by certain optimization algorithms or manual changes before running gate-level simulation.

3.5   Evaluation: Flow Reuse Across Diverse Chips

Hammer has been used to tape out a diverse set of chips since those evaluated in [67]. Several noteworthy chips are compared in chronological order in Table 3.3 and Fig. 3.5. The most revealing metrics are 1) the portion of unique lines of code (LoC) in Fig. 3.5 (i.e. design-specific Hammer IR, hooks, and other scripts) and 2) the person-months spent on PD in Table 3.3, which is the number of designers multiplied by the equivalent number of months working full-time on PD. The drop in both metrics over time despite little application
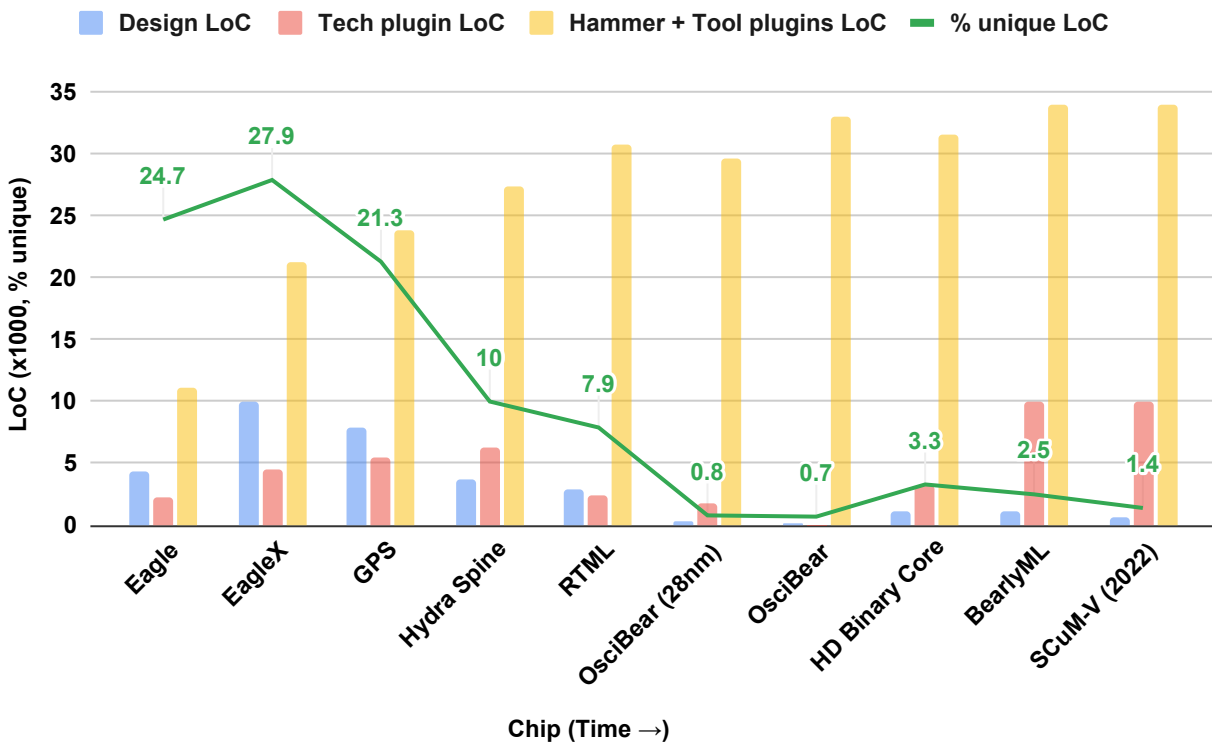
Figure 3.5: LoC and PD effort trends.

and floorplan commonality (Fig. 3.6) demonstrates Hammer's increasing reusability. Over the same period, Hammer's codebase—excluding proprietary tech plugins—grew by $3\times$ from 11,000 to 33,000 lines. Several clusters of chips designed simultaneously demonstrate effective reuse:

Eagle, EagleX, and GPS: EagleX is an evolution of Eagle [72] and was taped out together with the GPS chip one year later. Between Eagle and EagleX/GPS, 16,000 lines of reusable code were added to the Foundry A 16nm plugin and Hammer, demonstrating how PD intent from Eagle was made generic for subsequent chips.

Hydra Spine and ARGO: The Hydra Spine chip was taped out in Foundry B's 22nm (Hydra Spine II, see Sec. 2.3) and drove development of the IC Validator plugin. It was followed by ARGO, which started in Foundry B's 22nm node before switching to Foundry C's 14nm node and finally its 12nm node. This underscores how users drive Hammer plugin development and switching foundries and nodes is made seamless by plugin interchangeability. Both chips also utilized the hierarchical flow developed for the Eagle chip and were built primarily by one graduate student.

OsciBear/SCμM-V and HD Binary Core: The OsciBear/SCμM-V chips was designed by undergraduates as part of a course [73] and the HD Binary Core chip [74] was designed by a single graduate student with little prior experience in PD. Hammer was an

Table 3.3: Comparison of chips using Hammer.

	Eagle [72]	EagleX	GPS	Hydra Spine	ARGO	OsciBear / HD SCμM-V [73]	Binary Core [74]	BearlyML
Description	9-core RISC-V SoC	22-core RISC-V SoC	GPS receiver SoC	MU-MIMO baseband SoC	RISC-V SoC for ML	Bluetooth SoC	HDC processor	RISC-V SoC for ML
Foundry node	A 16nm	A 16nm	A 16nm	B 22nm	C 12nm	A 28nm, SKY130, B 16nm	A 28nm	B 16nm
Die Area (mm^2)	24	56.3	24	32	16	0.33, 7.6, 4	4	4
Signoff frequency	1.05 GHz	1.05 GHz	500 MHz	2 GHz	1.1 GHz	50-100 MHz	—	400 MHz
Hierarchy levels	3	3	1	3	2	1	1	1
% unique LoC	32.9	38.7	27.1	11.1	8.6	0.8, 0.7, 1.4	3.4	2.5
Person-months	22	10	6	5	4	8, 1, 10	8	10

★ Predominantly undergraduates (tapeout course)

(a) Eagle   (b) Hydra Spine   (c) ARGO
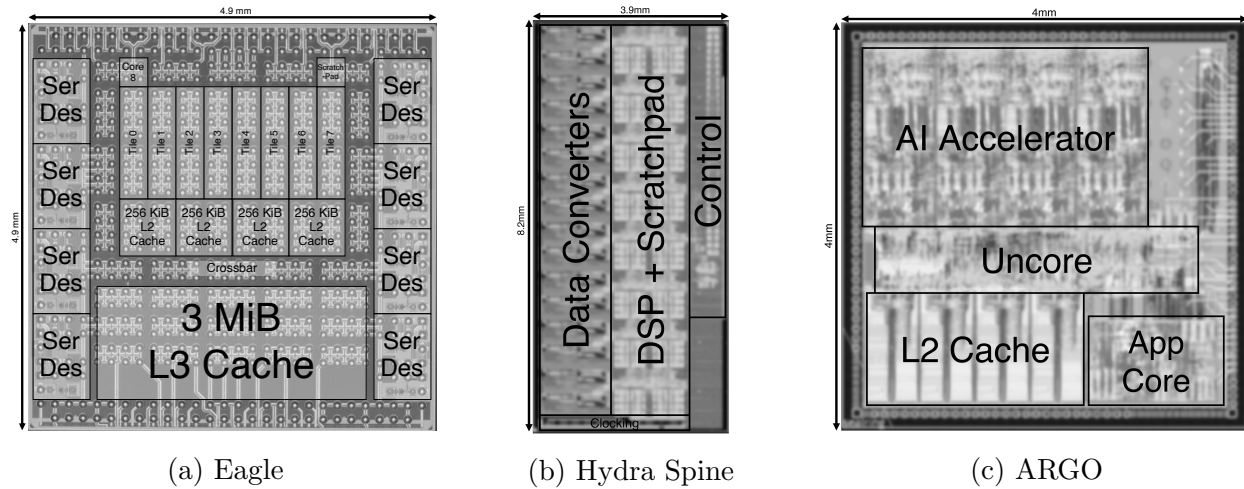
Figure 3.6: Diversity of Hammer-generated chips.

ideal tool in these chip designs because flows are easy to compose and already encode PD experience from advanced Hammer users. Again, due to plugin interchangeability, OsciBear was taped out in SKY130 only a few weeks later.

3.6 Related Work

3.6.1 Vendor/Foundry Reference Flows

Tool vendors and foundries provide reference flows for tools and process technologies, which typically consist of Tcl template scripts, but are not modular and require customization for any given design. They also lack a framework for applying relevant customizations to other designs, limiting reusability. In contrast, Hammer facilitates initial customization via hooks, which can be made reusable by integration into plugins in a design-agnostic way. Reference flows also do not include safety checks such as input checking, which can cause unexpected failures. Open-source flows like qFlow [75] and OpenLane [76] avoid some of these issues but are not designed to be modular and portable to other tools and technologies.

3.6.2 SiliconCompiler and mflowgen

SiliconCompiler[10] [77] and mflowgen[11] [78] are the most comparable open-source flow tools to Hammer. SiliconCompiler's strengths are in its API for its data schema and flow graph, parallelized cloud compute, and metrics extraction. However, its data schema does not enforce a separation of concerns and does not codify design inputs as clearly as Hammer

---

[10]https://github.com/siliconcompiler/siliconcompiler
[11]https://github.com/mflowgen/mflowgen

IR, which limits flow reusability between designs. mflowgen's strengths are in its flow graph management and modularity via tool step and PDK nodes. However, the modularity is so fine-grained that there is no method to inject common functionality between nodes of a similar flow step in different tools. Neither tool currently has features analogous to Hammer IR's metaprogramming or Hammer hooks.

3.7 Future Work

Hammer auto-generates a full flowgraph as Make dependencies, but future versions will include an interactive flow management interface and robust metrics extraction and post-processing to further enhance DSE. Hammer IR's metaprogramming, while powerful, should be traceable. Keys will be annotated with where they are set, modified, and consumed, and then checked for validity [12]. Hammer will continue adding support for more open-source tools and process technologies, additional PD analysis tools (e.g. aging and manufacturability), and other features such as ECO flows for late design closure. Continuous integration with sample designs will be setup to ensure that plugins remain compatible across all tool and PDK versions. Hammer will continue to deepen its integration within Chipyard, including compiling physical design annotations from Chisel generators, such as abstract floorplan constraints [79].

3.8 Conclusion

In this work, we demonstrate how Hammer helps hardware architects achieve design goals beyond what simulation and synthesis alone can inform, while vastly reducing the PD effort when taping out large SoCs in a variety of process technologies. Hammer's design principles of separation of concerns, standardization, modularity, and incremental adoption combined with open-source availability ensure that it can continuously be improved and maintained by its user base. We hope that this work will enhance a growing ecosystem of agile hardware generators that can lower the NRE cost of implementing increasingly specialized hardware.

---

[12]Flowgraph and IR tracing has been preliminarily implemented by Bryan Ngo

Chapter 4

Silicon Process Technology Constraints for 3D Die-to-Die Interconnects

As CMOS scaling has slowed down, high-performance computing has turned to 2.5D and 3D heterogeneous integration to simultaneously increase density and performance. Vertical die-to-die (D2D) interconnect standards are required to enable an ecosystem of 3D-stackable chiplets, as has been proposed in a draft AIB-3D specification [80]. However, an analysis of technology scaling reveals that vertical D2D interconnect standards must bridge a design space wider than has existed for horizontal interconnect, with wide-ranging implications on architectural specifications.

4.1 Introduction

Recent grand demonstrations of HI such as Ponte Vecchio (PVC) [35] have showcased how aggregation of chiplets, in varying technology nodes optimized for their functions, enable nearly limitless expansion of compute capability. Benefits come from reducing parasitics, decreasing latency by localizing data movement, shrinking form factors, improving yield, optimizing for technology nodes, and reusing IP in the form of chiplets. Some applications are mature now, such as HBM [81], stacked image sensors [82], and memory over logic [83]. These have worked well due to very clear benefits of partitioning across existing well-defined boundaries, such as between logic and memory.

In the future, it is envisioned that an ecosystem of chiplets could soon be available for HI of systems-in-package previously unrealizable in any technology and/or constrained by form factor and scalability reasons, such as intelligent radio cubes operating with sub-THz arrays [36]. In such a paradigm, the vertical D2D interconnect may vary widely in physical configuration (e.g., bond pitch) depending on technology node and the choice of face-to-face (F2F), face-to-back (F2B), or back-to-back (B2B) attachment. What is clear,

however, is that the landscape of 3D HI of modular chiplets is within the realm of system-level partitioning, wherein the bond pitch will range from the low 10's of μm down to approximately 1μm [84][50]. The upper end of this range is in-line with the lower end of the pitch range in horizontal interconnect standards such as UCIe™ Advanced Package [85] (25μm) and AIB high-density [86] (20μm).

Currently, 3D-ICs are designed with commercial 3D-IC tool suites[1], wherein a **single-cockpit** performs all design and signoff, such as inter-die timing analysis [87]. This methodology requires co-design of stacked dies, which is essentially an extension of the existing partition-based hierarchical design. To propel forward the move to a true chiplet ecosystem, interconnect standards are needed to remove the degree of freedom of interconnect design from this methodology with minimal performance impacts, while still being able to answer all the architectural concerns presented in Figure 4.1.
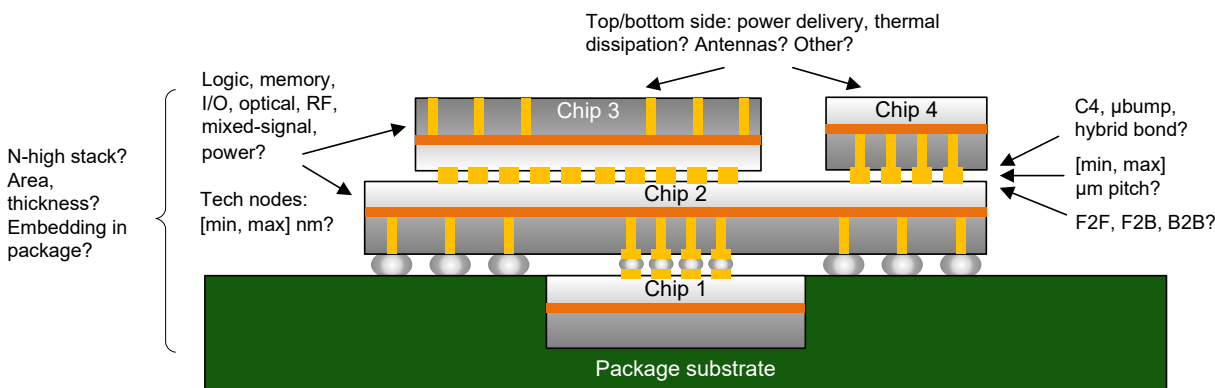
Figure 4.1: Architectural concerns in 3D-IC design.

At first, it may seem straightforward to extend existing horizontal interconnect standards to 3D. After all, they have matured rapidly due to their straightforward scaling from existing edge-launched buses such as DDR and HBM. However, vertical interconnect is shown to be constrained differently as compared to horizontal interconnect [50] and thus their circuit architectures and physical arrangements (i.e., bump maps) will not scale in the same manner. As an illustration of how these problems are yet to be solved, electrical or physical specifications for AIB-3D are still in progress.

In this chapter, the theoretical bounds of 3D D2D interconnect are probed over projected scaling trends, and its implications on the design of a standard are analyzed. First, the current roadmap of device and assembly scaling is described with a focus on interconnect. Next, key assumptions and choices in vertical D2D interconnect standard design are proposed. To test these, routing resource analysis shows how on-die routing resources, instead of bumps, limit achievable bandwidths and bump maps. I/O circuits are analyzed next, with particular emphasis on ESD and cell area, followed by a discussion of some tradeoffs regarding clocking

---

[1]Synopsys 3DIC Compiler, Cadence Integrity 3D-IC

and de-skewing strategies. Defect profile and repair is a critical concern for mixed-foundry
3D ICs, and a comprehensive analysis of repair strategies is analyzed. Finally, we conclude
with recommendations for vertical D2D interconnect standards and discussions of remaining
work in light of technology scaling trends.

4.1.1   3D-IC Scaling Trends

3D-ICs with enormous amounts of inter-chiplet interconnect bandwidth are being enabled
by two concurrent scaling trends: 1) assembly and packaging capability, specifically wafer-on-
wafer (WoW) and die-on-wafer (DoW) bonding at sub-μm tolerances, and 2) the continuous—
albeit slowing—shrinkage of lithography features and the march beyond traditional CMOS
device patterning. This delineation is somewhat blurred, as features that enable die stack-
ing, such as thru-silicon vias (TSVs), are lithographic but are essential for WoW and DoW
bonding and are therefore scaling in-step with assembly and packaging.

   PVC's adoption of Foveros technology [88] allows it to implement vertical D2D intercon-
nect achieving 0.2pJ/bit efficiency at a 36μm μbump pitch. However, with the advent of
direct Cu-to-Cu hybrid bonding, a 5$\times$ reduction in interconnect capacitance is achieved from
40μm μbump to 10μm hybrid bonds [89]. Scaling down to $\approx 1\mu$m pitch for F2F and F2B
hybrid bonding with extreme wafer thinning down to sub-μm thickness [90] is just over the
horizon. This level of assembly scaling over just the last decade is remarkably aggressive and
enabled new wafer processing technology, including enhancements in chemical-mechanical
polishing (CMP), wet etching, electrochemical deposition (ECD), and wafer alignment. The
reliability of hybrid bonding is improving but problems remain, such as bonding voids [91]
and particle defects [92], as well as various post-bond reliability threats [93]. TSVs have
similar and additional issues, such as possible shorts to the substrate. However, the more
alarming problem is that testability is getting worse to downright impossible. As bond
pitches shrink, we can no longer perform cost-efficient wafer-level or pre-bond test, such as
with probes. This process may even introduce damage via physical and electrical stress and
surface contamination, thus decreasing yield. Therefore, reliability improvements merely
change our redundancy calculations, but will never remove the need to have robust yet low-
overhead built-in self test and built-in self repair schemes. Nevertheless, at this level of
scaling, we can now assume that the range of bond pitch for vertical interconnect standards
is fully enabled in future nodes, and that future R&D will further improve yields and drive
down costs.

   On the lithography side, traditional Dennard's scaling of devices has been replaced by
a move into the third dimension to maintain density scaling. Gate-all-around (GAA) de-
vices (also called nano-wire or nano-sheet) move the transistor width scaling vertically and
are nearly ready for productization. Complementary FETs (CFET) are being developed,
wherein the NMOS device is stacked on top of the PMOS device, both as GAA devices.
Unfortunately, on-die interconnect is not scaling at the same rate, due to an increase in
resistance and electromigration effects without good metallurgical solutions. This imbalance

in scaling is shown in [94], where CFETs require 3.5-track standard cell construction, down from the roughly 6.5-track construction in today's FinFET technologies.

On the backside (i.e., in-substrate), density continues to scale with buried power rails (BPR) and nano-scale TSVs [95]. BPRs easily relieve the routing resource impact in the move towards CFETs, but TSVs remain challenging to design around due to device keep-out zones that are a very significant portion of the TSV pitch and lower reliability. These are caused by alignment tolerances, signal integrity impacts from parasitic capacitance, thermal and mechanical stresses, and more [96]. Improvements in TSV manufacturing and modeling promise to reduce this impact but will continue to have a sizeable impact on vertical interconnect design in backside attach scenarios.

4.1.2 Foundations of Vertical D2D Interconnect

Today, the space of D2D interconnect standards and solutions is exploding as 2.5D interconnect standards such as UCIe™ [85], AIB [86], and BoW [97] are maturing. We are witnessing the release of many D2D PHYs, both custom and standards-compliant, from many existing and new players in this industry. However, for vertical interconnect, we observe that existing 2.5D PHY architectures are being repurposed for 3D PHYs [50]. While this is nominally effective design re-use, it will become evident that these solutions do not address the scaling and applications space that 3D interconnect will need to occupy.

Fundamentally, we must not think about vertical and horizontal interconnect in the same way – there are a different set of limiting factors as shown in this comparison of 2.5D and 3D interconnects. In 2.5D, the reach is fixed at 1-20mm, whereas in 3D, the reach is up to an order of magnitude lower but potentially variable based on the die thickness and stack height. In 2.5D, bond pitch is dependent on the packaging technology, but in 3D, it is driven by lithography and assembly capability. Pitch mismatches in 2.5D are OK within reasonable routability in the package, but not OK in 3D. 2.5D PHYs are confined to the shoreline, whereas 3D PHYs could theoretically be placed anywhere in the die. This freedom can be hugely beneficial in terms of performance and power, but only if you can align them properly. Finally, 3D PHYs are more than just electrically coupled – they are thermally coupled too, which makes minimizing power consumption critical. Testability is also intimately coupled, so the more you can manage testing from one point in the stack, the better. Combined, these differences mean that 3D interconnect design is much less dictated by the process of transferring bits over a channel, where innovation has been directed towards overcoming the bandwidth limitations of long wires. Instead, system and physical design concerns will dominate the design process, so that we can ensure compatibility between stacked dies and account for all of the coupling effects. These differences are summarized in Figure 4.2.

In Figure 4.3, various roadmaps are plotted against the bond pitch on a log scale. Here, we can get a sense of the progression of enabling technology and its currently-envisaged applications all in one place. Pay particular attention to the regions which 3D interconnect standards may need to serve compared to 2.5D. Broadly speaking, 3D interconnect standards will target a large region of system-level partitioning applications and is not applicable where
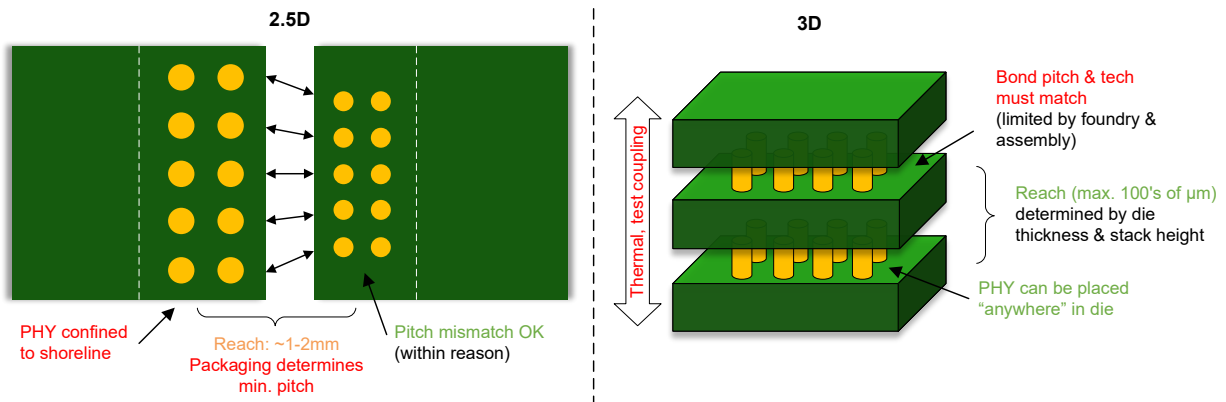
Figure 4.2: 2.5D $\neq$ 3D interconnect: different limiting factors.

you are partitioning functional units or cells using fully-custom 3D I/O or monolithic 3D fabrication. This means that the end user wants to aggregate relatively well-defined IP blocks together, rather than splitting arbitrary IP blocks into a stack of chiplets. The span of bond and assembly technologies to be supported contains the space of μbumps down to hybrid bonding, which is currently being enabled across a wide spectrum of CMOS device technologies. This is a large scaling range, but its impact on circuit design is much larger than it has been for existing interconnects. In light of this, a set of high-level assumptions and design choices must first be made, then tested in the following sections.

The first choice is that for a preliminary interconnect standard, F2F bonding using soldered μbumps or hybrid bonded pads should be prioritized, because nano-scale TSVs are not yet ready for productization and general die stacking more than 2-high, where F2B and B2B stacking is required, is also still under development outside of specific applications like image sensors and HBM. However, the same analytical principles apply in F2B/B2B cases—the only difference is that TSVs are quite lossy and require keep-out zones [96], affecting the I/O driver design as will be discussed below.

The second choice is that vertical interconnect signaling and circuits should be as simple as possible, extending to reliability and test needs. This is driven by two general priorities: 1) to optimize for per- channel energy efficiency and scale bandwidth with bond pitch, and 2) to minimize signaling and startup latency to reduce overhead compared to fully co-designed interconnect. Therefore, CMOS signaling must be chosen to remove any requirements for termination, passives, amplifiers, etc. The maximum serialization factor is chosen to be two (i.e., DDR), which is the highest factor that does not incur additional datapath latency via a serializer block. To maintain yield despite bonding errors, redundancy must be implemented as multiplexers at each I/O. Finally, built-in self-test by using loopback instead of full boundary scan is required since direct probing of pads is not possible at fine bond pitches [50].

The third choice is to locate, as much as possible, the circuits underneath the I/O "patch"
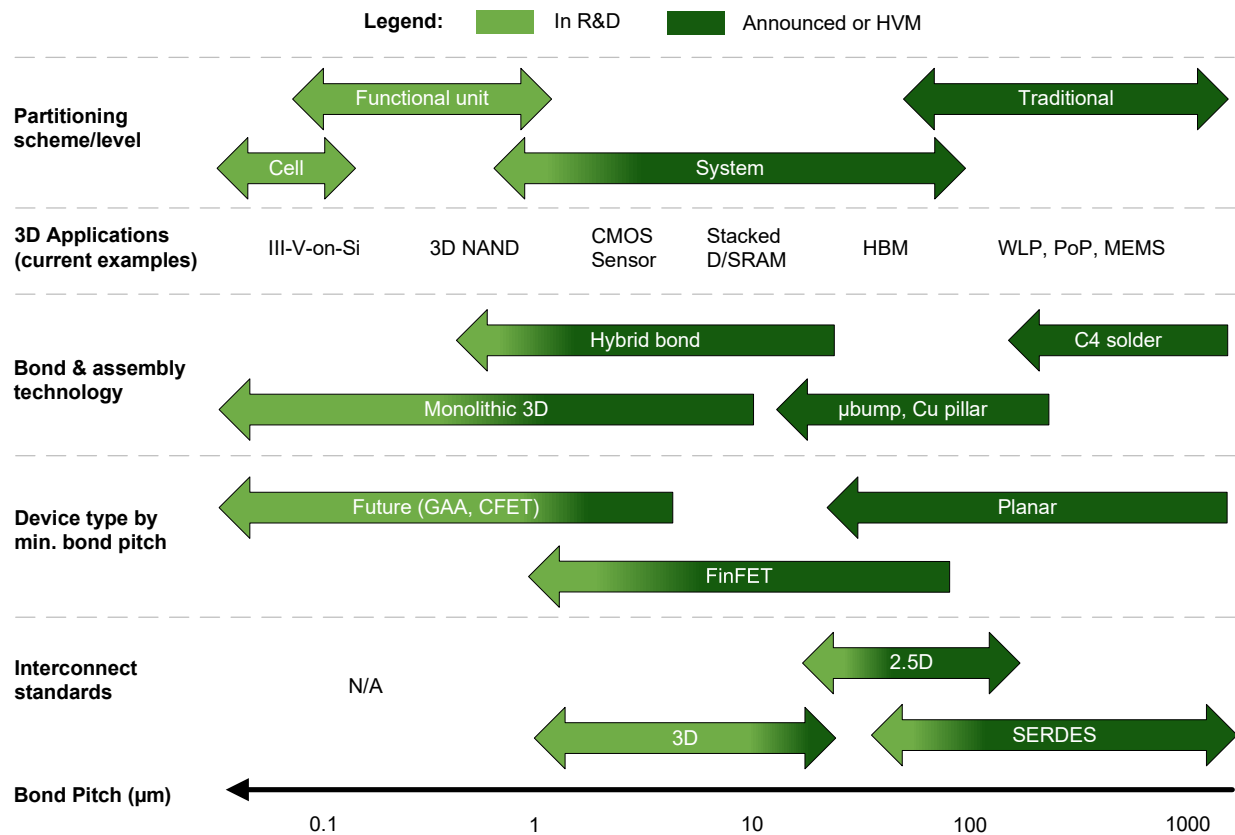
Figure 4.3: 2.5D ≠ 3D interconnect: scaling and applications. Adapted from [84].

area. This way, the area consumed by the interconnect macro does not spill out over the boundaries of the patch, the channel loss characteristics and thus timing margins are minimally dependent on the metallization of either stacked die, and potentially higher bandwidth is achievable with an auto place-and-route (APR) flow for patch construction instead of custom, length-matched routing to each pad. However, drawbacks include higher clock tree power that scales with the patch size, and a diminishing circuit area and feature budget especially at very fine bond pitch, resulting in limits on bandwidth scaling. From these three choices presented so far, it is evident that I/O circuit area must be optimized with respect to technology and bond pitch scaling.

The fourth choice is to use a source-synchronous clocking scheme. This means that the transmit clock is forwarded to the receive side, with phase alignment performed on either or both ends. By removing traditional SerDes blocks such as CDR and PLLs, circuit area is saved and very fast phase-locking schemes can reduce startup latency. This also provides for easier integration for common 3D partitioning use cases, such as memory stacked on top of compute cores. The most obvious limitation to this scheme is that per-I/O serialization and thus bandwidth is capped; however, this is both increasingly unnecessary and impossible

with aggressive bond pitch scaling, as will be proven later.

The fifth choice is to share the power supplies between the PHYs of stacked dies, like the "tightly-coupled" mode in UCIe™ [85]. In addition to enabling the aforementioned signaling choice, training complexity for clock alignment can also be simplified. This comes at the expense of level-shifting at the on-die interface and provisioning for power delivery up the entire stack. However, in the latter case, maximum power consumption can be specified and accounted for in the PHY design.

The last choice is to locate the on-die internal interface along one edge of the patch (i.e., one "shoreline"), like is done with existing interconnect standards. If more edges are required, the patch should be tiled with appropriate rotation/mirroring, much like how SRAM arrays are constructed. This minimizes the routing complexity in/out of the interconnect and prevents extra area consumption outside the bounds of the patch. This limits the interconnect's achievable bandwidth since it is bounded by one shoreline; however, we show that this bound is high enough to be worth the benefits.

A recap of the choices described above is shown in Figure 4.4.
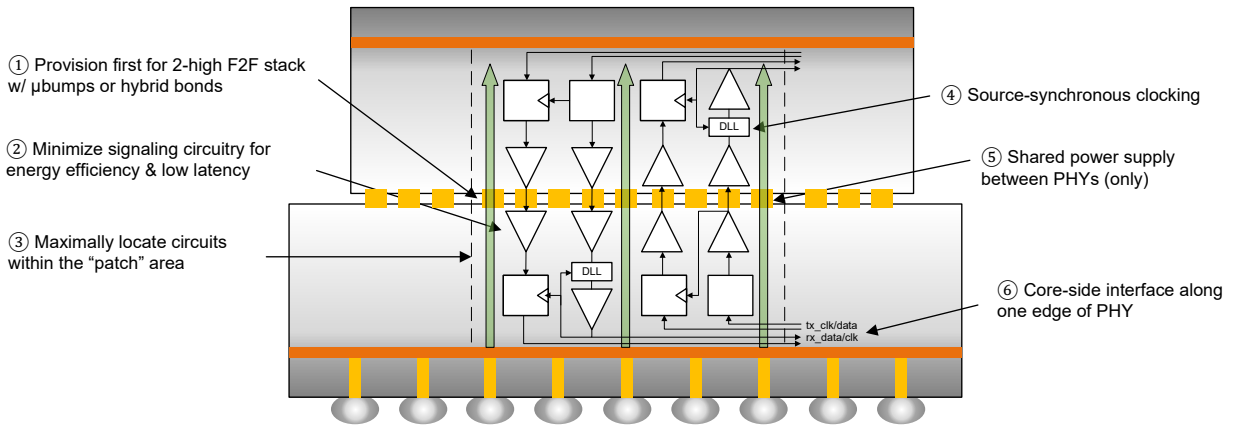
Figure 4.4: High-level architectural choices for standardized vertical D2D interconnect.

4.2 Scaling Constraint Analysis

4.2.1 On-Die Interconnect: Routing Resources

For this analysis, we first make some assumptions about the routing available for use over the patch. In a hypothetical process node, assume that there are 2 layers at the finest track pitch and 1 layer at $4\times$ the finest track pitch, in both horizontal and vertical directions. The coarser pitch layer is useful for routing further distances with less loss. Given the thousands of parallel wires we need to route into a patch, using more metal layers than this may cause excessive routing congestion. For every layer, assume the following penalties:

1. 20% of tracks are allocated for power routing, which is a reasonable estimate of power grid density

2. Signals traversing the patch is shielded for SI (50% hit)

3. Auxiliary interface signals consume 10% of tracks

4. APR achieves 80% congestion-free routing utilization

Of course, with BPR, the power routing penalty can be removed, and for small bond pitches, the routing length is short enough for shielding to be removed. In such cases, the following numbers can be scaled up by the corresponding factor.

Next, we must also consider routing through interface logic. For simplicity, assume that we can densely pack an array of flip-flops at the edge. Assume that D & Q share routing tracks and therefore disallow utilizing more than 1 fine-pitch routing track over the array. One routing track in the same direction as data is allocated for the clock and one is allocated for set/reset/enable. Finally, we sweep across a hypothetical series of process nodes that range roughly from a mature FinFET node "N" (corresponding to a 16nm-class technology [44]) which is used for analysis below down to the aforementioned CFET demonstration [94]. Table 4.1 shows a set of hypothetical process technology nodes with their finest track pitch and the D flip-flop track height and width. Since this analysis was performed, nodes such as Intel 4 [98] have been announced and confirm this roadmap.

Table 4.1: Hypothetical technology scaling roadmap.

Node	Track pitch (nm)	DFF track height	DFF track width
N	90	7	27
N+1	70	6	24
N+2	50	5	21
N+3	30	4	18

For node N, Figure 4.5 visualizes the achievable shoreline bandwidth given these routing constraints. Patch shoreline bandwidth considers the routing tracks at the interface edge, while core logic bandwidth considers the routing tracks over the flip-flop array. The transfer rate is assumed to be 4GT/s SDR, i.e. no serialization. The bond pitch is only for illustration and irrelevant to this preliminary calculation, however it is assumed at first that the entire patch is data signals. In all cases, the calculated shoreline bandwidth utilizing all available routing resources is in the range of 20Tb/s/mm.

Scaling down to node N+3 is shown in Figure 4.6, with various combinations of routing layer choices considered. The plot shows that at least two routing layers must be used, and that the shoreline bandwidth imposed by the interface logic becomes the limiting factor,
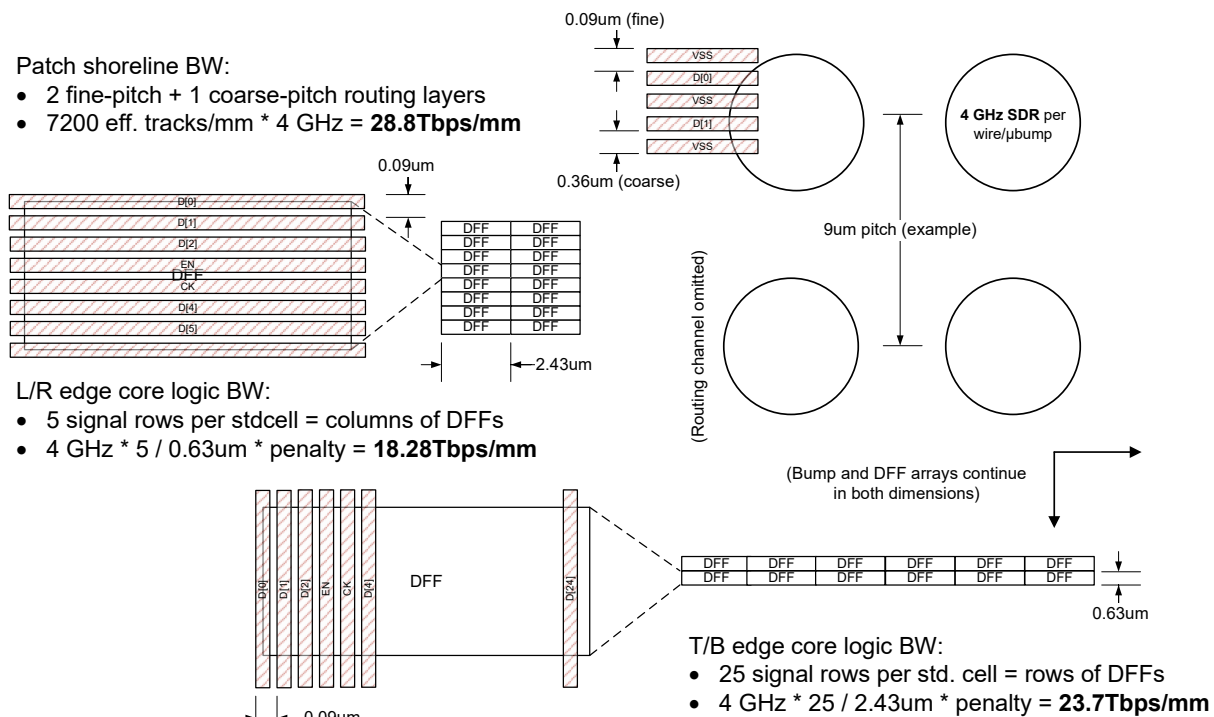
Figure 4.5: Shoreline bandwidth, visualized.

proving that using more layers does not buy you more bandwidth density. The difference between the blue and red line shows a large gap developing between core logic on the left or right edges versus the top or bottom edges. This is due to the shrinking track height of the standard cells with scaling. To solve this, the DFF array as assumed will need to be spaced out into multiple groups at the expense of timing closure and congestion. Accounting for power and ground bumps also theoretically helps, but we must not ignore the fully-dense DFF array assumption itself. One could argue that it is unrealistic because its power density at this transfer rate would create a hot spot. Ultimately, power density may become the limiting factor on increasing shoreline bandwidth. Regardless, shoreline bandwidths upwards of 40Tb/s/mm is theoretically possible with these assumptions.

Next, we analyze the patch depth: that is, the maximum numbers of rows/columns from the interface edge that is routable given the available resources. It is necessary to introduce here the concept of a via keep-out zone (due to the stack of metal from devices to pad, Figure 4.7a), which removes routing resources from the very edge: the first row or column of the patch. For our calculations, we will assume the keep-out zone is 0.5μm wide, regardless of technology node, for signal/power integrity (SI/PI) reasons. Figure 4.7b is a contour plot of the achievable depth and shows that with 2 or 3 routing layers used, at very fine bond pitch, the patch will not be more than approximately 10-15 bumps deep, below what is required to have a reasonable patch size in terms of number of parallel data signals, which directly relates
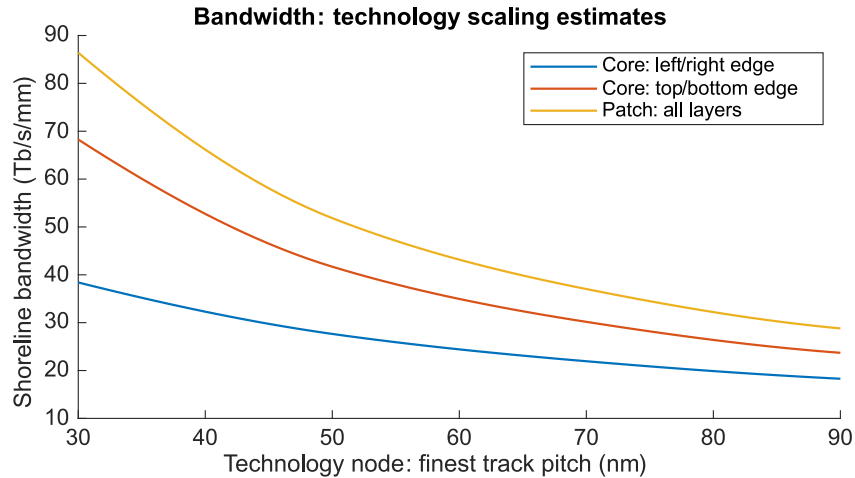
Figure 4.6: Shoreline bandwidth: technology scaling estimates.

to the available aggregate bandwidth within a given clock domain. Figure 4.7c visualizes the
effect of the keep-out zone in more detail – once it exceeds about 20% of the bond pitch, the
contours flatten. This means that the via keep-out zone dominates in capping the achievable
shoreline bandwidth, regardless of technology node and the only solution is to shrink it. It
is important to note that if we factor supply and auxiliary signal bumps into the patch area,
the patch depth may increase marginally while the shoreline bandwidth decreases.

A more realistic constraint on the patch, however, is that the patch is configured to sup-
port a desired aggregate bandwidth. This number is determined by the chiplet application.
For analysis, we sweep across 0.5 to 10Tb/s and find the aspect ratio, which is the ratio
of the dimension parallel to the interface edge to the patch depth, defined in 4.7a. Aspect
ratios larger than 1 imply the patch is depth limited. Figure 4.7d shows the aspect ratio
contours for the best case tech node (N+3). Scaling down towards 1μm bond pitches will
result in very "skinny" patches, especially in older nodes. Shielding could be removed at
such low bond pitches or more metal layers are used at the expense of congestion, but both
solutions require rigorous on-die SI analysis.

These results present a **strong case against serialization** in the I/O cells. Anything
that allocates additional wires to each I/O will result in a growth in the aspect ratio by
the **square** of the serialization factor: for example, 4× for DDR. This is not desirable
because patches that need to support such high aggregate bandwidths will create "walls"
in the chiplet's floorplan, diminishing any benefit from using existing edge-launched 2.5D
interfaces. Clock tree quality may also degrade as the patch geometry deviates far from a
square shape. Lastly, if the power supply comes through the package, it becomes difficult to
connect as the substrate/interposer pitch exceeds the patch depth.

In the sub-2/3μm bond pitch regime, it may be necessary to rethink the bump map
assumption from earlier. The relative size of clock conditioning circuits may instead warrant
a bump map in which a few rings of I/Os surround a central region where power and clocking

(a) Via KOZ, depth, and aspect ratio definition

(b) Patch depth

(c) Max shoreline BW

(d) Patch aspect ratio
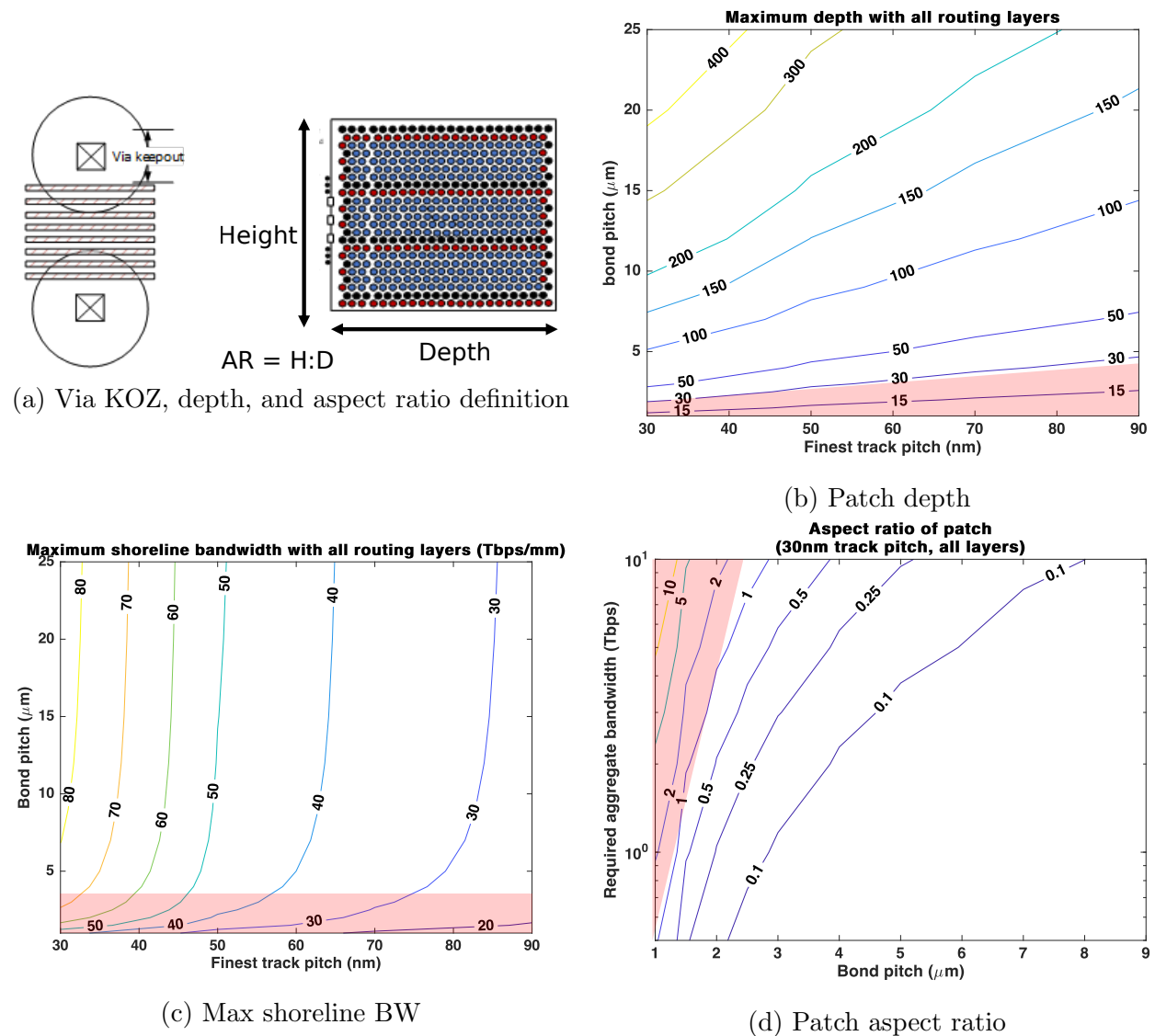
Figure 4.7: Via keep-out zone and its effect on patch depth and shoreline bandwidth vs. bond pitch and technology node.

are handled, which looks much like today's flip-chip bump maps. Unfortunately, serialization does not help, and such a bump map will set a lower limit on the areal bandwidth density as the clock circuitry dominates. Thus, there will likely need to be a bifurcation in the bump map specification dependent on parameters such as the bond pitch, technology node, routing layers, and serialization. This delineation point must be chosen carefully so that compatibility between chiplets in different technologies is maintained without wasting area in the most deeply scaled nodes.

In the opposite case, where the patch is not at all aspect ratio limited (e.g., $\geq 10\mu$m bond pitch), the tradeoff between specifying more serialization or porous soft macro construction must be addressed. Serialization will maintain bandwidth maximization as long as the rate is co-optimized with the channel [50], but introduces latency and decreases energy efficiency due to the additional circuitry needed. Increased complexity, such as in synchronously clocking the bus at high speeds, negates some cost benefit of vertical D2D interconnect. Porosity, on the other hand, preserves the I/O design and therefore maintains energy efficiency at the expense of bandwidth density. This requires a soft macro type of construction, requiring a high degree of design and verification automation due to the required co-design for power and thermal reasons. Otherwise, an "everybody loses" situation may arise with negative impacts on both core logic and the interconnect due to poor SI/PI and clocking, especially when TSVs are used [96]. Based on this, it is tempting to limit the supported bond pitch to approximately 10μm and below. However, if true HI is to be enabled with chiplets in nodes optimized for applications like RF, optical, or memory, support for higher pitches must be included. These points are summarized in Figure 4.8.

Figure 4.8: Alternate layouts and the scaling extremes.

4.2.2 I/O Circuit Architecture

At low bond pitches, the area of the I/O circuits will require significant optimization. Under the assumption of DDR signaling, the logic area can be very small in deeply scaled nodes. However, electrostatic discharge (ESD) protection structures will dominate the area. Thus, it is important to consider the range of allowable ESD specifications as a function of bond pitch.

In stacked dies, the ESD phenomenon that matters for circuit design is the charged device model (CDM). Human body model (HBM) is of no concern since the vertical interconnect

should not be externally exposed through the package and dies should not be handled by humans prior to stacking. Without HBM, testing against the machine model (MM) could be revisited; however, CDM is still tougher to protect against due to the very narrow width and higher magnitude of its current pulse. Both CDM and MM voltage specifications are also lowered with improved assembly machinery. For our analysis, we are only concerned with pre-package CDM, which has a maximum of approximately 35V. The worst-case CDM ESD current profile used for analysis (35V @ 625mm^2 die area) and the peak current contours as a function of CDM voltage and die area are shown in Figures 4.9a and 4.9b, respectively.

(a) Current profile

(b) Peak current contours
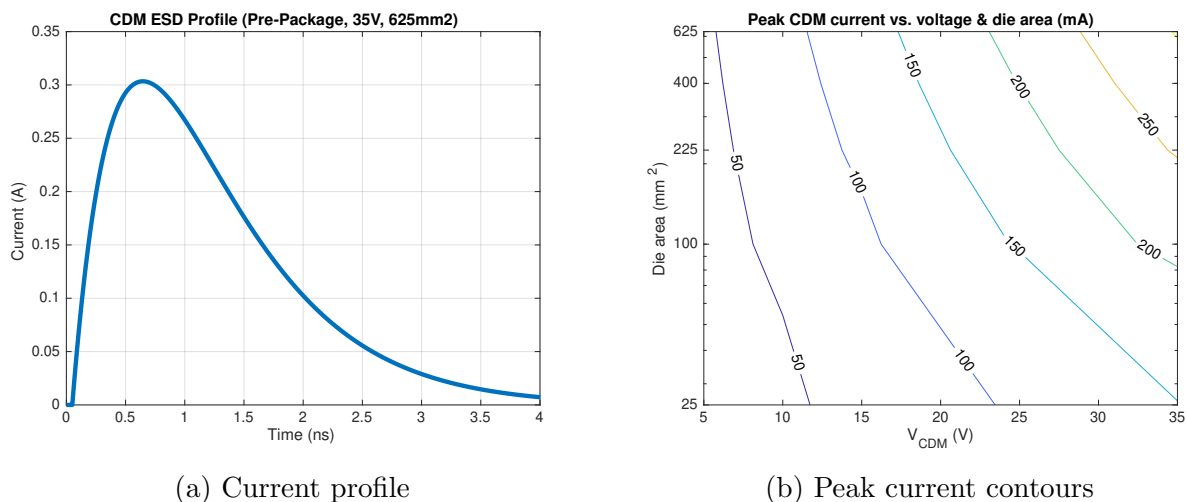
Figure 4.9: CDM ESD current profile and peak currents.

The first protection structure analyzed resembles the traditional primary + secondary diode with ballast resistor scheme (Figure 4.10a). Instead, the primary diodes can be as small as the technology's smallest unit diode in secondary diode arrays (D3/D4), and the output driver itself forms secondary self-protection due to the parasitic diodes between the body and drain terminals (Figure 4.10b). A ballast resistor in the 10's of Ohms can be implemented with very small ($<1\mu$m wide) thin-film resistors due to the low currents remaining after the diodes. A schematic simulation is performed in node N using standard device models, which is sufficient under the choice of the I/O circuit located directly under the pad due to the low parasitics and low voltages involved. An approximately 1μm wide output driver is chosen and ballast resistance swept from 10 to 200Ω, which is allowable due to the choice that no termination impedance is to be specified. As the CDM ESD voltage is swept along the axes in Figure 4.9b (5-35V CDM, 25-625mm^2 die area), the output transistors do not exceed their breakdown voltages (approx. 3.5V in node N) except in the lowest ballast resistance cases, as shown in Figure 4.11a.

To gauge the lower limit of this protection scheme, it is most instructive to observe the effective diode resistance at the peak of the CDM profile. As shown in Figure 4.11b, the resistance shoots up below approximately 100mA of peak current regardless of chosen ballast

(a) Traditional
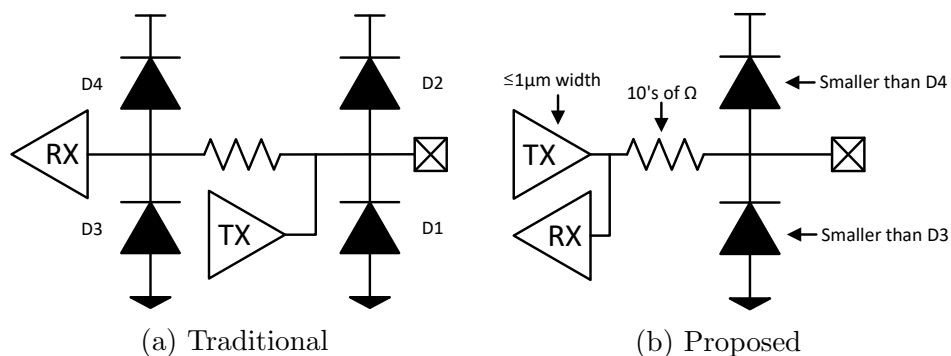
(b) Proposed

Figure 4.10: Traditional and proposed low-voltage ESD protection structures.

(a) Driver voltage

(b) Diode effective resistance
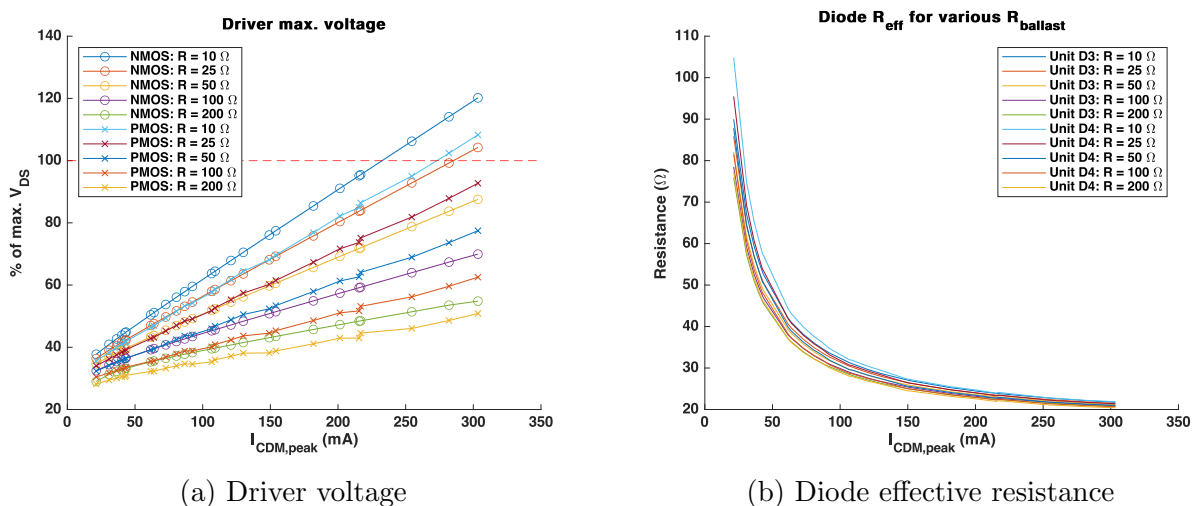
Figure 4.11: Driver voltage and diode effective resistance as a function of the peak CDM
ESD current.

resistance, which corresponds roughly to a 15V CDM spec. Thus, it is likely not area-efficient
to utilize ESD diodes below a 15V CDM spec. If diodes are used, the ballast resistance and
output driver width should be tuned in conjunction, such that the driver width is minimized
for low power consumption and the ballast resistance is chosen so that SI is maintained given
the RC filter between the ballast resistor and ESD + pad capacitance as seen by the driver.
A resistance of less than 100Ω is likely required for a channel capacitance on the order of
100fF.

The second protection scheme that should be considered below the 15V CDM spec is pure
self-protection. In this scheme, up to 100mA of peak current will be sunk via the drain-body
diodes, which must be absorbed by guard rings. In node N, schematic simulation results
in Figure 4.12a and 4.12b show that the NMOS transistor is the limiting one, requiring a
maximum width of approx. 2μm to stay under breakdown voltage limits (3.5V in node N)

at approx. 15V CDM, and decreasing as expected with lower voltage. However, for self-protection, TCAD simulations are likely required for proper sizing. This is because latch-up rule-compliant guard rings may be insufficient, requiring optimization of per-finger resistance against drain area to ensure that all fingers trigger during the ESD event and no current crowding occurs [99].

(a) NMOS   (b) PMOS
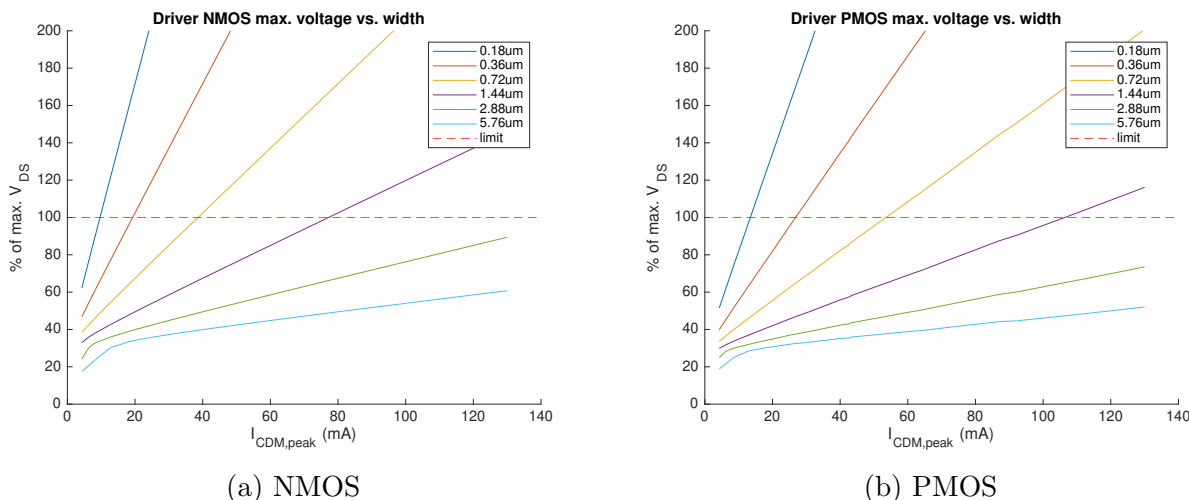
Figure 4.12: Minimum NMOS and PMOS transistor widths for self-protection.

Unfortunately, relying on the substrate and guard rings for ESD is not possible in all technologies. In the most obvious case, silicon-on-insulator (SOI) technologies do not have an accessible body terminal without drilling through the buried oxide. Less obvious, but perhaps more important, is the roadmap towards extreme wafer thinning and buried power rails, which significantly crowd substrate currents. Lastly, CFET technologies will also have no accessible substrate. The solution is to utilize specially constructed grounded-gate NMOS (ggNMOS) devices, such as the one presented in [100] for SOI technologies. These operate in principle similarly to the antenna diodes needed in all technologies. While silicon-controlled rectifiers (SCR) are significantly more area-efficient compared to ggNMOS [101], they are designed for high-voltage I/Os and require complex trigger circuits to reduce their triggering points and do not yet have a scaling roadmap for very low voltage CDM due to their use of BJTs.

Lastly, the very nature of CDM as a charge buildup phenomenon may warrant a rethink of CDM protection, from both the industrial test standard and models to the protection schemes. Internally distributed protection in a mesh network, perhaps embedded in the substrate using novel TSV structures or with a crossbar of phase-change materials is in active research [101]. Finally, with improving assembly machinery, the roadmap for CDM protection is trending rapidly downwards [102]. With roadmaps of 5, 3, and even 0V specifications, it is likely that ESD will cease to dominate the I/O cell area if transistor breakdown voltages do not significantly drop further. Thus, the logic and driver area must be analyzed next.

For a basic bidirectional DDR I/O cell with redundancy multiplexing but no circuits
for test as shown in Figure 4.13, logic implementation with standard cells and a 5V CDM-
compliant self-protected driver in node N consumes roughly $20\mu m^2$, meaning a bond pitch
of 5μm is possible with only F2F bonding. Naively using the scaling roadmap in Table 4.1,
this area would shrink down to $1\mu m^2$, but since the driver cannot scale directly, this logic
implementation will not fit. Furthermore, there are 4 data wires assigned per I/O, which
will exacerbate the depth issues described in the routing resource analysis. Multiplexing of
the Tx and Rx datapaths by reusing flip-flops at the expense of in-cell loopback capability
is proposed in Figure 4.14, conceptually even sharing a clock tree for a cluster of I/Os and
reducing data wires down to two in/out ones. This saves area at the expense of integration
complexity, for example due to loopback using the redundancy multiplexer paths instead of
occurring in-cell.
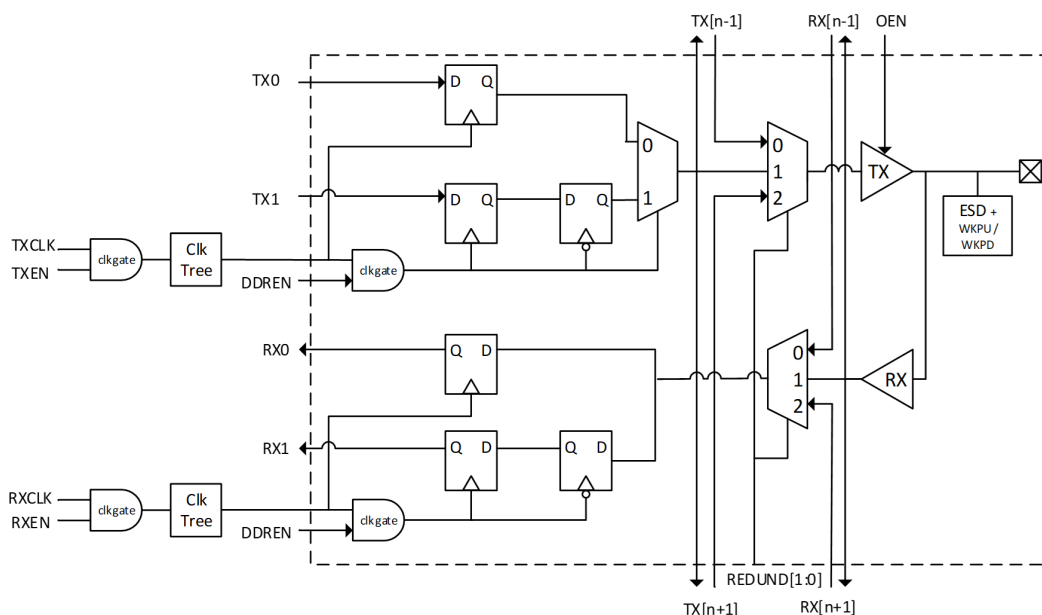
Figure 4.13: Basic DDR I/O cell.

In conclusion, the I/O cell area limitations may result in a specification with two regions
is roughly proposed in Figure 4.15. Region I has no restrictions on the I/O functionality,
whereas Region II will require removing I/O bidirectionality, moving the DDR retiming flops
to the interface edge and retaining only buffers under the pad, and/or requiring little-to-no
ESD protection. The exact bond pitch bifurcation point is dependent on the roadmap of
TSV keep-out zone reduction.
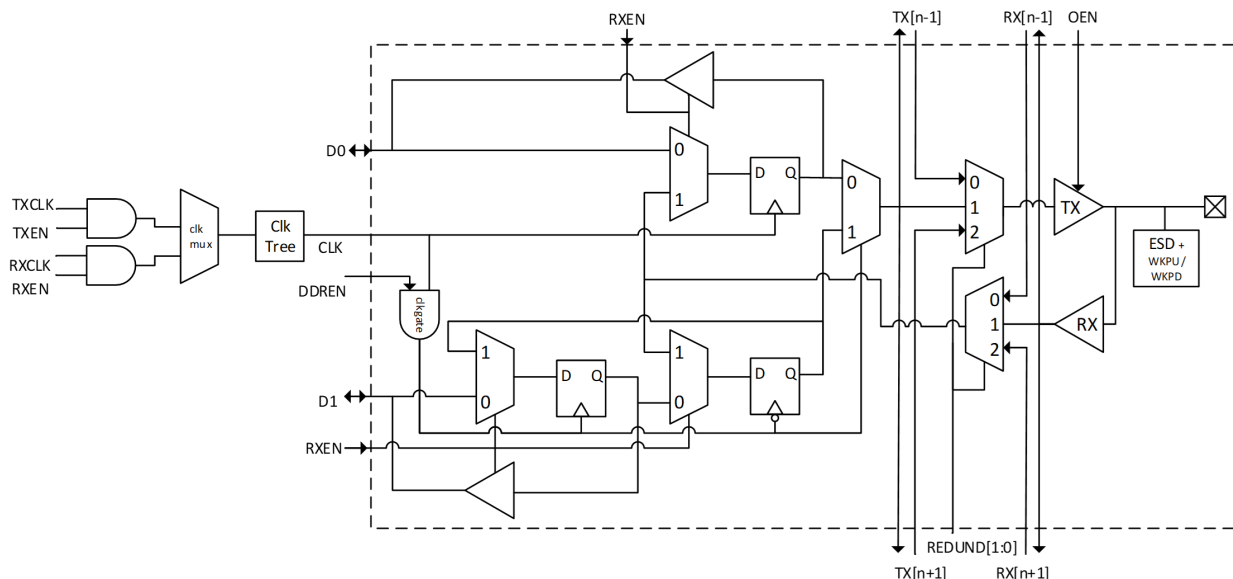
Figure 4.14: Reduced area DDR I/O cell.
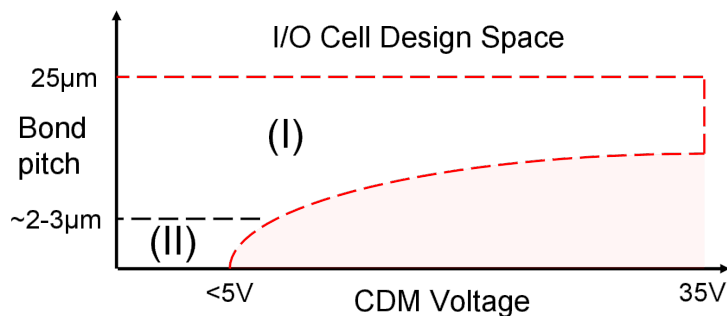
Figure 4.15: Proposed I/O cell specification vs. ESD & pitch.

4.2.3   Clocking and De-skew

As alluded to in the routing resource analysis, the area of clocking and de-skew circuits will
become dominant as the bond pitch scales down to 1μm. Thus, it is instructive to evaluate
how scaling will affect the clocking tradeoffs. Fundamentally, in a source-synchronous clock-
ing topology without de-skewing, the transmit and receive clock sinks (i.e., retimers) are at
unknown phases relative to each other and the logic beyond the interconnect interface due
to mismatches in clock tree insertion and channel delays. Clock insertion delay is a strong
function of bond pitch – at 25μm for a cluster of 512 I/Os per clock, it is certainly greater
than 1UI at 4GT/s and dominates over channel delay, which will depend mostly on metal
stack-up + TSV heights. However, at 1μm bond pitch and with extreme wafer thinning, they
will be comparable in magnitude. There shall be freedom in de-skew circuit implementation,

with variations of digitally controlled delay lines, phase generators + rotators, and/or delay locked loops.

In a simplistic architecture, de-skew is performed solely in the Rx side at the root of its clock tree. This is by far the most area efficient and lowest start-up latency architecture because it requires no training procedure. With tightly coupled supplies, the Rx de-skew only needs to correct for relatively slow temperature drift between the stacked chiplets, relaxing its specifications and allowing for fast-lock architectures. Unfortunately, this will require a clock crossing on the Tx side, adding latency, and is insufficient for applications requiring deterministic round-trip latency such as memory over logic. Instead, this may be a candidate for latency-insensitive applications such as extending bus protocols across chiplet boundaries or low duty cycle peripherals if fast-locking is achieved. In these cases, this is an efficient configuration at lower bond pitch and transfer rates.
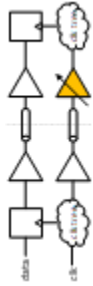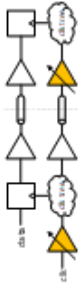
In the architecture encouraged by most source-synchronous interconnect standards, both Tx and Rx de-skew is used. While this solves the Tx clock crossing issue and minimizes latency, it requires a training procedure using sideband signaling, adding to start-up latency and decreased areal bandwidth efficiency. Furthermore, the Tx de-skew circuit must also adapt to voltage drift between the interconnect's supply and the core logic if the supply is not forwarded from the latter's domain. This requires a high loop bandwidth—even higher if DVFS is desired—and consumes higher power and area. This is likely the most flexible architecture for most tighter bond pitch and higher transfer rate configurations.

In an alternate architecture, de-skew is performed solely on the Tx side based on data alignment measurements returned from the Rx side. The forwarded clock arrives ahead of the data, in essence trying to synchronize the Tx and Rx clock tree roots, which removes the jitter amplification from the 1UI delay necessary in traditional de-skew loops. Continuous training/adaptation is required to correct for inter-chiplet temperature drift, requiring a return clock or extra sideband signaling. This may be an energy-efficient candidate for high bond pitch configurations where the transfer rate is not high.

Finally, variable clock phase forwarding and per-I/O de-skew will be required if the specification requires higher transfer rates at the larger bond pitches. This would be an extension of the UCIe™ clock architecture [85], for example, at the expense of lower energy efficiency.

These tradeoffs are summarized in Table 4.2. In conclusion, it is broadly desirable to increase the cluster size (I/Os per clock) as hybrid bonding yield improves to minimize the clocking power and area overhead. Clock tree quality is therefore very important, so clock SI/PI must be maximized. Furthermore, synchronizing clock tree trunks instead of the roots would dramatically reduce clock tree skew and power, shifting the timing margins to APR instead. In this scheme, de-skewing may not even be needed except to synchronize the Rx and Tx clocks for a roundtrip transaction. This topology is much closer to how 3D-partitioned logic is clocked when the stacked dies are co-designed; however, many new issues will be introduced, especially regarding testability. It remains for future exploration to study the optimal sub-clustering of Rx bumps to tap clocks from the Tx trunks using the algorithms reviewed in [96]. Separately, some concepts of inductive or transmission-line clock coupling

Table 4.2: Source-Synchronous clocking topology tradeoffs.

De-skew Method		Pros	Cons	Application
Rx-only		• Rx only corrects temp drift • Lowest area (no sideband) • Fast-lock = low startup latency	• May need Tx clock crossing • Round trip latency not deterministic	• Bus protocols, low duty cycle peripherals • Lower bond pitch & transfer rate
Tx + Rx		• No Tx clock crossing • Training only at startup • Adaptable from existing standards	• Sideband for startup • Tx de-skew needs higher bandwidth (V drift/noise)	• Most flexible for latency-sensitive applications • Lower bond pitch, higher transfer rate
Tx only		• No Tx clock crossing • Low Rx circuit area • Less jitter (clock ahead of data)	• Sideband continuously active (correct T drift) • Requires tracking clock	• Max. power efficiency applications • Higher bond pitch, lower transfer rate
Tx + per-I/O		• Max de-skew control • Can support large cluster size	• More circuitry per-I/O • Requires tracking clock • Lowest power/area efficiency	• Max. performance (aggregate BW) • Higher bond pitch & transfer rate

are being studied [103], which also promise to sidestep the many issues around TSVs.

4.3   Discussion

This work has addressed the design of the interface, I/O, and clocking mechanisms of vertical D2D interconnect. However, considering other 3D-IC and interconnect design issues, which include power delivery, thermal dissipation, and testing for known good die (KGD) [50], will impose additional constraints. For example, the logic densities at the tightest bond pitches here will be extremely high, resulting in high power densities and poor on-chip decoupling. As a quick calculation, if the dense flip-flop array used in the core logic bandwidth calculation is actually constructed in node N, the worst-case scenario of 100% switching activity at 4GHz in the corner used for hold analysis results in a power density of $8500\mathrm{W/cm^2}$, creating a hot spot. Bleeding-edge cooling technology like in-substrate microfluidics [104] may be required to dissipate this level of heat flux. Thus, further capping the maximum transfer rate and circuit/signaling voltage as the bond pitch scales will also need to be in the final interconnect specification.

Nonetheless, technology and assembly scaling combined with the $>20\times$ range of potential bond pitch support is simultaneously enabling over an order of magnitude increase in bandwidth potential while imposing strict constraints on PHY implementation. It will be tough to discretize this vast design space to balance the ability to create stackable chiplets (e.g., pitches must match) with the flexibility of optimizing parameters like cluster/patch size, pitch, transfer rate, etc. for each intended use case. Going too far in either direction risks rendering any vertical interconnect standard irrelevant. To prevent this, it will be necessary to adopt a **generator-based soft macro methodology** to PHY construction, akin to memory compilers, so that logic and circuits can be reused for multiple variants of bond pitch across multiple technology nodes. These generators must employ a **correct-by-construction** methodology, fully automating tasks such as bump assignment, I/O cell and interface pin placement, clock and timing constraints, and abstracts and model generation. Ideally, such a generator should also participate in the process of 3D logic partitioning, so that the required PHY configuration, mapping, and model insertion can happen in a synthesis-like manner.

A simplistic view of this methodology is shown here in Figure 4.16, with 3 separate designers of the system and the PHYs on different chiplets. The generator shown here is common to the designers of a leader and follower PHY. In addition to the RTL, the generator should output physical design constraints, filter out the relevant global parameters that need to be passed onto the designer of the follower interface, and insert an interface model at the system partition boundary to help with architectural design and verification. Overall, with the flexibility of a generator, interconnect instances can be better optimized for the application. IP reuse in this context will then be at the granularity of building blocks, instead of as finished PHY hardmacros. Beyond this, there will also need to be enhancements to EDA tools and physical design flows, which is ongoing work at many entities [105].
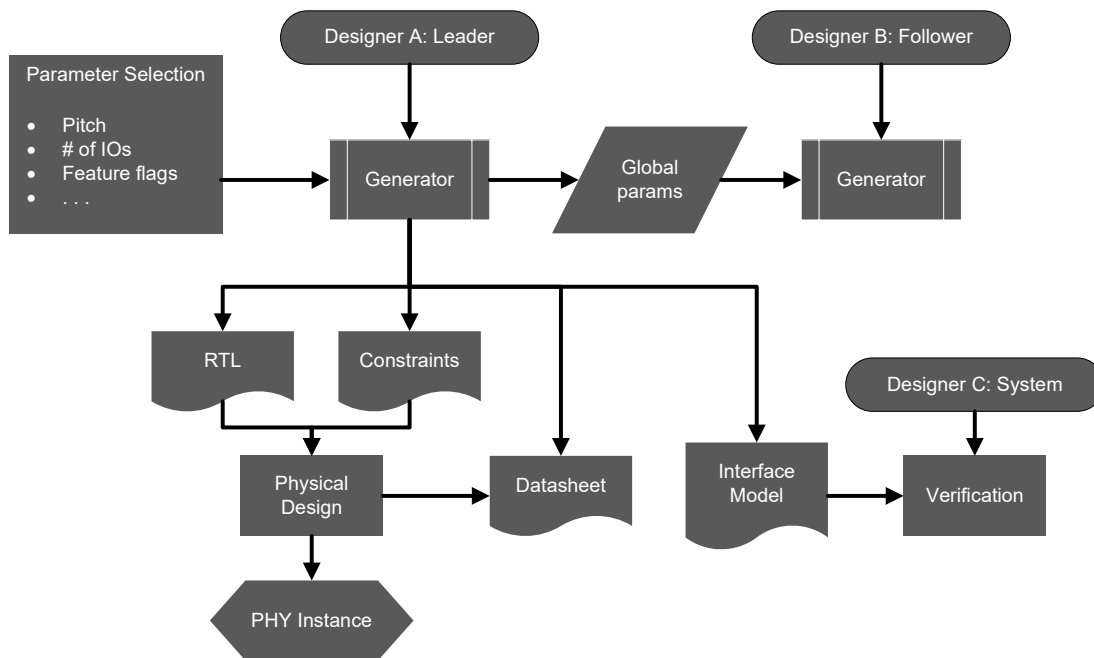
Figure 4.16: Generator-based soft macro methodology.

4.4 Conclusion

In conclusion, we have seen how vertical interconnect moves from a channel-limited regime, in which we try to extract more bandwidth over wires with varying degrees of loss, to one where we must have a holistic view of constraints from many sources. This necessitates that we do not reuse 2.5D interconnect standards and PHYs to build scalable 3D interconnect ones. This is due to the confluence of CMOS and assembly scaling, which are the dominant constraining factors for interconnect design from a standards perspective. To visualize this, Figure 4.17 shows a rough partitioning of the potential specification space along the axes of CMOS technology and bond pitch. The boundary contours are not shown to be linear with technology scaling due to the expectation that driver area will not scale with logic density due to SI through TSVs and lower bounds of ESD. The upper limit on technology is also arbitrary will change if older nodes are updated to support 3D stacking. Region I is optimized under the assumptions presented in the Foundations section, whereas Region II is constrained by circuit area as described in the I/O circuit section. Region III has a lot of remaining freedom to maximize bandwidth, albeit with significant tradeoffs as described at the end of routing resources section, but this suboptimality may be acceptable. Discretizing over this design space is therefore a fine balancing act because interoperability is so much more constrained than existing interconnect standards. Too little discretization will render some implementations as too low performance, while too much discretization will cause excessive incompatibility between chiplets. Therefore, going too far in either direction threatens to

make any standard irrelevant on arrival. Going forward, these partition boundaries will need to be adjusted to maximize interoperability between chiplets in different technologies, while also considering long-running 3D IC issues such as power delivery, thermal dissipation, and testability, are are exacerbated by scaling.
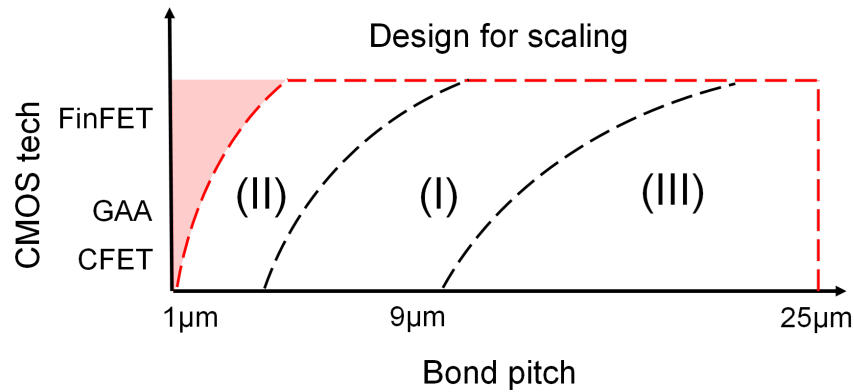
Figure 4.17: Vertical interconnect specification vs. scaling.

So, now that we are presented with the potential for an order of magnitude higher bandwidth density by moving to 3D, we must now achieve this through circuit simplification and design methodology optimizations. In effect, the physical specification such as the bump map must encode nearly all the solutions to the effects of scaling. This includes everything from signal and power integrity to built-in self-test and self-repair schemes. In the reverse direction, we hope that a standardization effort should drive some roadmaps, such as further ESD reductions and bond pitch scaling in older process nodes.

Chapter 5

A Chisel Generator for Standardized 3D Die-to-Die Interconnects

Technology-driven constraints highlighted in published works and the previous chapter demonstrate that a unique approach to 3D D2D interconnect design and implementation is required, while preserving the ability to customize the interconnect to accommodate future technology concerns and applications with minimal overhead. To this end, this chapter presents a framework to generate customized 3D D2D interconnect physical layers (PHYs) that are simultaneously standard-compliant, physical-aware, and can be automatically integrated into all stacked chiplets. The generator framework leverages the Chisel hardware description language to allow designers to 1) compile a port list directly into a PHY; 2) automate design and physical design; and 3) perform design space exploration of interconnect features (e.g., bump map pitch, clocking architecture, and others). The 3D PHY generator framework and features detailed in this work can be used to produce a reference implementation for a standard like UCIe-3D™, representing a significant paradigm shift from current specification and design methodologies for 2.5D D2D interconnect (e.g., UCIe™) implementations. This work concludes with the results of a redundancy design space exploration tradeoff study, showing the benefits of a proposed spatial coding redundancy scheme in an example PHY using emulated 9μm hybrid bonding for a 4 TX / 4 RX module array with 4:1 coding redundancy ratio.

5.1   Introduction

The power, performance, area, cost, and time-to-market (PPACT) benefits of 3D-HI are only realizable with a robust ecosystem of 3D-stackable chiplets mixed and matched to build next-generation 3D integrated systems [36]. The envisioned 3D chiplet ecosystem can only be realized by standardizing 3D die-to-die (D2D) interconnects [106]. Unfortunately, current "single-cockpit" tools[1] have significant methodology gaps, especially in creating 3D D2D

---

[1]Synopsys 3DIC Compiler, Cadence Integrity 3D-IC

interconnects [105]. These gaps lead to sub-optimal repurposing of 2.5D D2D interconnects for disaggregated 3D systems [50], creating highly customized chiplets that cannot be reused across different 3D platforms.

Unfortunately, standardizing 3D D2D interconnects is harder than 2.5D interconnects due to a slew of new concerns spanning the full stack as described in the previous chapter, from system-level (freedom of floorplanning, technology compatibility) to manufacturing (bonding defects, electrostatic discharge (ESD) levels) to multi-physics (power integrity, thermal coupling) [107]. Moreover, they must target an equal or lower PPACT than even planar on-die interconnects to make them attractive to use. This means that the 3D D2D interconnect physical layers (PHYs) should incur minimal overhead, both in circuit design (power, area, latency) and engineering (non-recurring expenses) terms. These concerns and requirements are not addressed by existing interconnect standards and their methods of specification and implementation.

A new automated approach is needed to capture and abstract these challenges into a common tool used to obtain the collateral needed to create spec-compliant and near-optimal PHYs while maximally preserving intellectual property (IP). Towards this goal, this work proposes an open-source generator of 3D D2D interconnect PHYs that aims to encode the standard's specification, be a design space exploration tool, and serve as a reproducible reference implementation, thereby replacing the status quo of specification documents interpreted by engineering teams.

5.2 Background

5.2.1 Methodology

Traditionally, interconnect standards are delivered as text documents that specify timing requirements, electrical characteristics, data protocols, and suggested bump maps. These are then interpreted by large teams of engineers spanning multiple disciplines, from system architects to analog circuit designers. Beyond the level of expertise required, optimizing a PHY and verifying it against the standard is a highly iterative and even error-prone process. Standards can also fall behind technology advancements due to slow industry consensus processes, potentially rendering them irrelevant. The amount of design effort and iteration is visualized in Fig. 5.1.

This proposed generator aims to solve these issues by automating the process of interpreting and optimizing to the standard, thereby lowering the barrier to creating the envisioned 3D-stackable chiplet ecosystem. To achieve this, the following principles guide the generator's design:

1. **Separable parameterization**: A robust set of parameters must abstractly describe technology and design concerns and be separated into shared global parameters and private instance parameters.
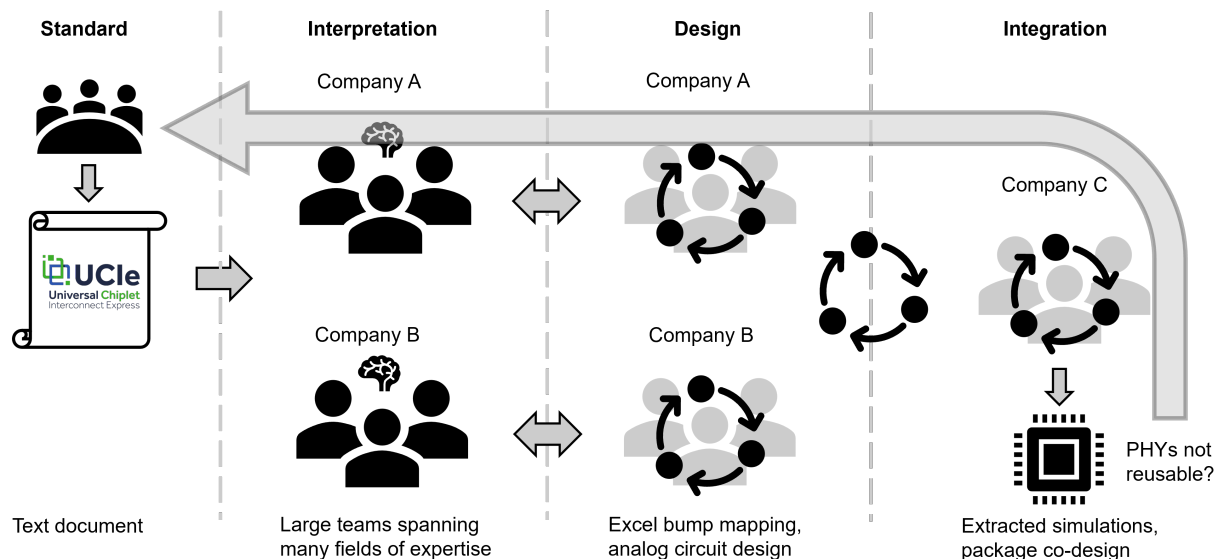
Figure 5.1: Typical 2.5D D2D interconnect design and integration process.

2. **Physical-aware**: Parameter combinations must be checked against technology capabilities to generate generally-feasible instances, thereby being technology-agnostic and flexible enough for fine-tuning.

3. **Data-aware**: The PHY must be minimally invasive to the data path and use data switching statistics to optimize the bump map for signal integrity and power.

4. **Correct-by-construction**: The generator's outputs (e.g., bump map and PD collateral) must be derived programmatically from the standard.

5. **Composability**: There must be sufficient options and modularity to add and select features and use specific hard IPs needed for their application.

6. **Open-source**: The framework must be easy to discover, use, and update by standards bodies and implementers to rapidly advance adoption, adapt to technology evolution, and proliferate a chiplet ecosystem.

By adhering to these principles, the issues outlined above are addressed in the following ways.

Principle #1, separable parameterization, is required so that a common framework can be used by all implementers of 3D interconnect PHYs in the envisioned chiplet ecosystem. As these teams are likely to be from separate organizations, only the minimal information needed to generate matching PHYs should be distributed (global parameters) while technology-specific information must be kept private (instance parameters). While both design- and technology-relevant parameters fall under both categories, the global parameters are used to

calculate common structures such as the bump map while the instance parameters will be used to fine-tune floorplan and timing constraints, for example.

Principle #2, physical awareness, is required due to the technology and bond pitch scaling analysis from Chapter 4. As will be described below, some parameters describe abstract process technology and packaging technology capabilities. As a result, certain parameter combinations may not be physically feasible, so the generator must be able to validate these and provide user feedback. However, override parameters are available in case the user desires to go beyond those bounds. Some checks are obvious, such as not allowing pitches outside the specification's bounds, while other checks are more subtle, such as limiting the number of bumps between power and ground for signal and power integrity (SI/PI) or limiting the number of signal bumps in a module to meet timing. This constitutes a significant advancement in clarity over the occasional ambiguity of static specification tables and example bump maps. This principle, when combined with Principle #1, ensures every implementer and integrator of generated 3D D2D PHYs work from the same technology constraint-driven rules, while abstracting away proprietary and instance-specific IP (technology-specific rules, IO cells, orientation, etc.).

Principle #3, data awareness, is required to minimize the overhead of the generated PHY. Since the use case of 3D interconnect is often folded logic or NoC-to-NoC communication [108], the interconnect is not serialized, meaning we can replace the concept of data transfer *protocols* with data-aware signal-to-bump assignment. The user is expected to provide the **data bundle** (a port list in Verilog) as a global parameter to the generator, along with the most common data switching statistic within the bundle. Prior work has shown that assignment of bumps in particular orders can result in meaningful reductions in power from coupling capacitances, especially when TSVs are used [109]. The generator can also combine this with coding redundancy schemes to further reduce power using techniques such as bus inversion. In aggregate, intelligent signal-to-bump assignment should enable the interconnect to appear to the system as merely a data shim with incremental power, area, and latency, making it ideal for chiplet partitioning.

Principle #4, correct-by-construction methodology, is required to ensure that implementers can trust the generator to produce a spec-compliant PHY that will work with the matching PHY on the other die with minimal manual intervention and verification. This means that the PD collateral must be generated in a way that is a 1:1 match with the logical implementation and constrains the design within the bounds of the specification, i.e., from a single source of truth. This is a different approach from existing interconnect standards, which provide no guidance on how to map signals to bumps, how to floorplan the PHY, or how to generate timing constraints. As a result, bump mapping is often done by hand using Excel sheets, and a separate script generates the physical design collateral. These tools are often unable to share information, leading to a higher error probability and a longer time to market if any parameters change during the design process.

Principle #5, composability, is required to allow implementers freedom to choose relevant features and go beyond a fully-digital PHY with hard IP building blocks for further optimization. For example, the generator includes options for the redundancy architecture,

configuration protocol, and insertion of hardened IO cells. As the generator's feature set
becomes richer, it can serve as a better reference implementation for more implementers of
3D D2D PHYs, significantly reducing cost and time-to-market across the industry.

Principle #6, open sourcing, is highly desired to make the generator continuously rele-
vant as the envisioned chiplet ecosystem is built and evolves. A standards body could adopt
this generator repository as the method of disseminating the standard with frequent updates
and public-facing discussion. Similarly, implementers can contribute better ways to design
the interconnect with other PD tools, or have private forks containing their own proprietary
optimizations. Lastly, open-source projects can lower the barrier for academics and research
institutions to adopt 3D-HI, further stimulating research. This is a different approach from
existing interconnect standards, which require agreement between representatives of consor-
tium companies to make infrequent changes to a relatively-static document.

Figure 5.2 visualizes the design process after adhering to these principles and Table 5.1
offers a high-level, qualitative summary of the contrast between the existing and proposed
methods of building 3D D2D interconnects, loosely adapted from [105] and consultation with
industry experts.
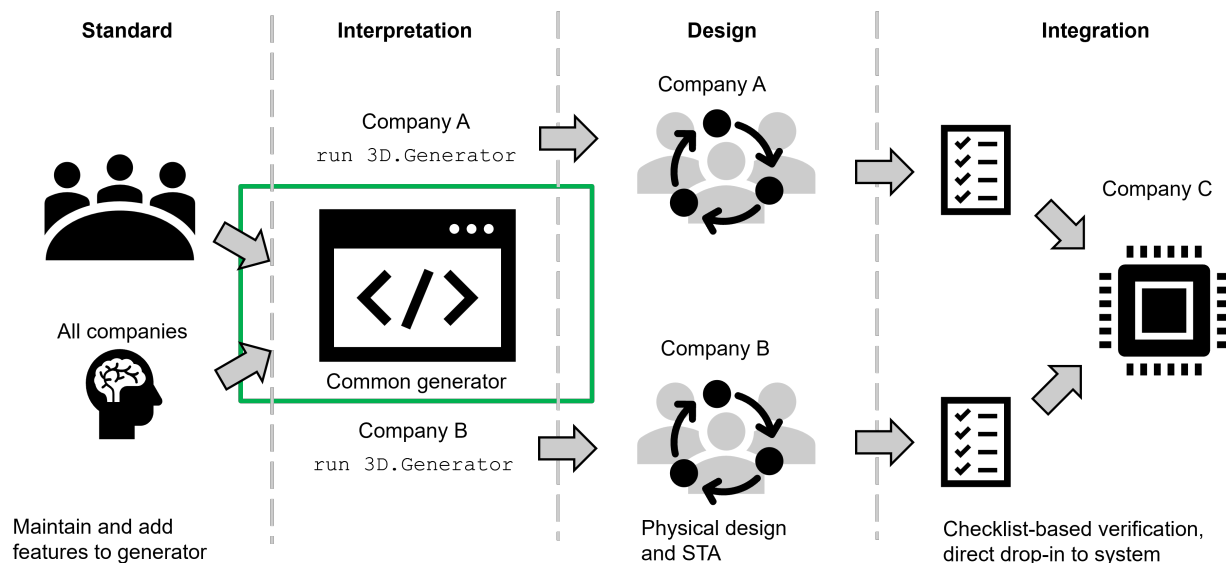
Figure 5.2: Proposed 3D D2D interconnect design and integration process.

5.2.2   Circuit Architecture

This generator is designed to align with UCIe-3D™'s [108] preliminary circuit architecture,
timing margins, and performance targets. Single-ended, single data rate (SDR) data and a
single-ended clock for transfer rates up to 4GT/s are forwarded to the receiver. In contrast
with traditional source-synchronous interconnects, the received clock negative-edge samples

Table 5.1: Qualitative comparison of D2D interconnect construction methodologies.

Metric	Current	Proposed
Tasks (per iteration)	• Spec'ing of PHY protocol, HAS • Spreadsheet-based signal ↔ bump mapping • MAS → RTL, manual constraints & PD implementation	• Define/get leader's parameters • Define instance parameters • Compile → PD from generated constraints
Pros for 3D interconnect	• Optimized for a specific configuration • Can incorporate proprietary features	• RTL/collateral generation separable between entities • Open-source Chisel = extensible by users
Cons for 3D interconnect	• Hard to parameterize, very slow, not scalable • Very error-prone (human input & feedback, uncommon code)	• Does not address/integrate with internal DV/DFT/security flows • Coarser granularity of design parameters & optimization
Codebase & No. of Users	30K LoC, 10 integrators	2.4K+ LoC, 1 integrator
Compilation time	Days	Minutes
Integration time	Weeks	Days

forwarded data with a variable clock-to-data differential delay. This delay skew—D_{tx} and D_{rx} for transmit (Tx) and receive (Rx) modules, respectively—is defined as a curve as a function of a shared supply voltage and budgets for a clock tree and clk-to-q delay on the transmit side and a clock tree and setup/hold time on the receive side. Unlike the 3D D2D interconnect in [110], the clock for both directions is not forwarded from only one die, as this architecture limits the maximum transfer rate due to additive jitter from the clock trees on each die. The timing parameters and timing closure equations are specified below. The UCIe-3D™ clock and data architecture is summarized in Fig. 5.3. The equations for deriving

D_{tx} and D_{rx} are given in Eqns. 5.1[2]. Please refer to the specification document for the rest of the timing specifications.
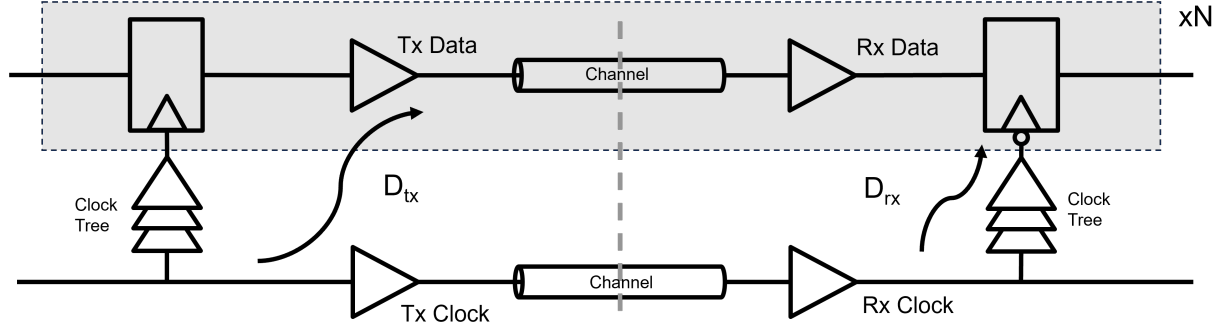
Figure 5.3: UCIe-3D™ circuit architecture.

$$
\begin{aligned}
D_{tx,typ} = D_{rx,typ} &= \frac{V_{cc}}{0.0153V_{cc}^2 + 0.0188V_{cc} - 0.0084} \\
D_{tx,min} &= \max(D_{tx,typ} - 0.08\text{UI}, 0) \\
D_{rx,min} &= \max(D_{rx,typ} - 0.08\text{UI}, 0) \\
D_{tx,max} &= D_{tx,typ} + 0.12\text{UI} \\
D_{rx,max} &= D_{rx,typ} + 0.12\text{UI}
\end{aligned}
\tag{5.1}
$$

5.2.3 Choosing Chisel

In aggregate, the shared power supply, clocking and datapath architecture, and timing targets encourage a near fully-digital PHY, meaning that timing specification compliance is achievable via multi-mode/multi-corner PD and static timing analysis (STA) in lieu of analog circuit layout and simulation. This also opens up the opportunity of designing the PHY to appear merely as a data shim with incremental power, area, and latency–a boon to SoC architects looking to partition existing designs. Accomplishing this requires a fully-automated signal to bump mapping flow originating from the data (e.g., a Verilog port list) to be transferred. Consequently, Chisel, as a domain-specific language overlaid on Scala, is well-suited to this task, due to Scala's strong emphasis on data structures and functional programming.

At its core, using an object-oriented language with strong typing means that hardware descriptions no longer need to be standalone. The code that generates hardware constructs (registers and modules) can live alongside code that generates software constructs (data structures and algorithms). For this application, because the logic is so simple, Chisel's verification issues are also of little concern. Finally, Chisel benefits from its existing integration

---

[2]Credit: Jong-ru Guo

with Chipyard [23], an open-source SoC construction framework, and Hammer [25], a PD flow generator, both of which are employed in the evaluation of this work (Sec. 5.5).

5.3   Defect Repair and Redundancy

One of the challenges identified in UCIe-3D™ is the need for a new defect repair strategy. As observed in [92], a particularly problematic source of hybrid bonding defects is surface contamination. Though wafer washing processes are improving, they are not improving fast enough—i.e., reducing remaining particle sizes at an rate exceeding bond pitch scaling—to decrease defect densities low enough to ignore, especially in a future chiplet ecosystem where chiplets will be diced separately, shipped, and stacked into large systems without the ability for probe testing. These particle defects blow out an entire region of bonds, which is not repairable using the same signal shifting techniques used in 2.5D interconnect [85], [86] that only able to repair isolated defects. The following subsections survey past approaches to repairing through-silicon via (TSV) defects and propose a physical implementation of coding redundancy [109] that is also able to repair particle defects with low overhead. In Sec. 5.6, the proposed redundancy scheme is compared against UCIe-3D™'s shifting redundancy scheme, shown in Fig. 5.4. Note that module-shift redundancy requires interleaving Tx and Rx modules, thereby requiring full-duplex signaling, while spatial coding does not. There must also be a spare Tx and Rx module for each row or column of modules shifted, which can result in significant area overhead for small PHYs.
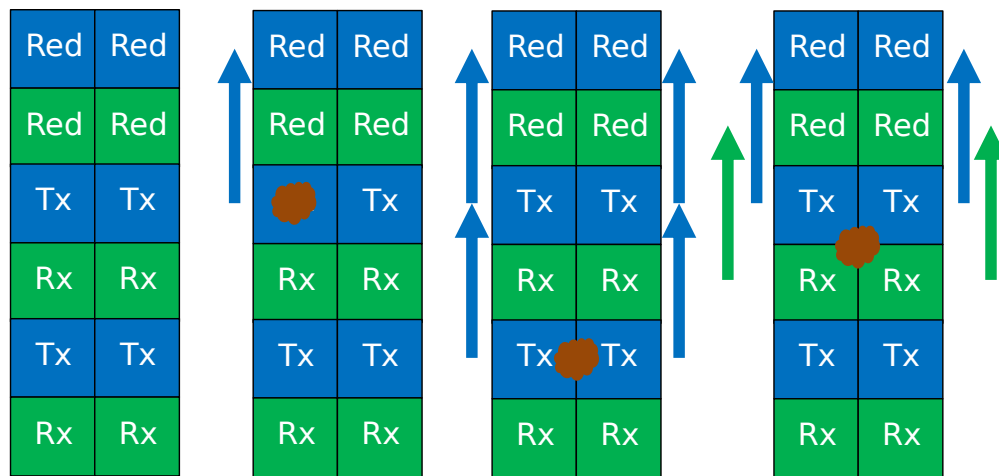
Figure 5.4: Module-shift redundancy scheme, repairing a defect spanning 1, 2, and 4 modules.

5.3.1   Previously Proposed Redundancy Schemes

Many schemes have been proposed for repairing clustered TSV defects, generally caused by bonding imperfections such as surface roughness and flatness, and modeled with a Poisson

distribution [111]. In general, the prior work can be divided into two categories: symmetric
and asymmetric. Symmetric schemes re-route data to spare TSVs and back to the original
TSV positions with complementary muxing in the Tx and Rx modules. Asymmetric schemes
use alternative logic (e.g., time multiplexing, coding) that requires configuration in only the
Tx or Rx module.

Common evaluation metrics include the repair rate for clustered failures within a 4×4
to 6×6 window size for a range of α (the distribution's clustering factor), logic area, and
added delay. Note that none of these schemes have been validated in silicon, though many
are supported by the high-bandwidth memory (HBM) industry, which currently implements
redundancy in the memory array and controller.

5.3.1.1 Symmetric Schemes

The most simplistic symmetric schemes are the signal-shift and the switch. Signal-shift moves
signals in the direction of spare TSVs, while the switch muxes any TSV within a group to
the spare TSV. However, neither has good repairability for multiple, especially clustered,
defects. To overcome this limitation, many topologies have been proposed, including router
[112][113], ring [114], cellular [115][116], grouping [117], 3D rotation [118], honeycomb [119],
circular [120], herringbone [121], and switch matrix [122]. All of these topologies use the
principle of maximum flow [123] to calculate the optimal rerouting paths from the faulty
TSVs to the spare TSVs, which are located either within or around the edge of the TSV
array. While this algorithm has a linear time complexity, the hardware required to do so is
prohibitive, hence re-routing is calculated offline, potentially imposing significant software
overhead on system bringup. Note that a few works also constrain re-routing paths using
total routing delay at the expense of repair rate [113], [115], [116], important for meeting
timing and latency requirements in real links.

Unfortunately, the particle defect condition is a 100% failure in windows exceeding the
array sizes evaluated in these topologies, of which only 3D rotation [118] and TRUST [122]
demonstrate full repairability at high area and delay cost. Considering that these topolo-
gies require multiple N-to-1 muxes for each TSV, the area of content-addressable memories
(CAMs) to encode the mux selects quickly dominates over the I/O cell area, thereby limit-
ing bond pitch scaling [107]. Here are general observations for the suitability of the various
topologies in this context:

1. Honeycomb, 3D rotation, circular, and grouping topologies are not suitable because
   they do not map well to scalable rectangular arrays due to the rigid placement of the
   spare TSV within sub-groups.

2. Router and 3D rotation topologies have excessive hardware and delay overhead given
   their achievable repair rates.

3. Ring, cellular, and TRUST have low enough overhead but have limits on the possible
   cluster size because of the spare TSV location around the perimeter of the TSV array.

4. Herringbone has a decently tileable and scalable architecture if defect areas are relatively small. It also addresses aging-related failures with a grouping option, but suffers in repairability.

5.3.1.2   Asymmetric Schemes

A tree-based scheme is proposed in [124], consisting of a two-stage structure with dedicated and shared spare TSVs in the transmit die followed by a simple set of muxes in the receive die. A time-division multiplexing access (TDMA) scheme is proposed in [125] that assigns time slots for each TSV and forwards a selection signal, negating the need for dedicated spare TSVs. A coding-based scheme is proposed in [109], pairing a coding bit with a group of TSVs and two complimentary implementations by making either the encoder or decoder fixed or configurable. All of these schemes promise lower hardware overhead because configurability is confined to either the Tx or Rx module, with the complementary module implementing simple, fixed logic.

Unfortunately, several of these schemes are also impractical in a predominantly-digital PHY implementation. The tree topology in [124] results in a wide variation in transition timing over the TSV array, degrading timing margin and generating excess switching activity. Using counter values for TDMA orchestration and data synchronization in [125] is not amenable to STA and requires manual layout to meet timing. Only [109] is attractive for the range of bond pitch and technology node scaling described in the previous chapter [107]. Low logic area and small variation in logic depth (only two stages of muxes and XOR/XNOR gates) helps timing closure at 4GT/s and fits within the bump array area, even at tight pitches. Furthermore, one of the two coding topologies (fixed encoder/configurable decoder) removes the need for a training procedure, thereby potentially eliminating sideband signaling like that in [86] and [85], while the other (configurable encoder/fixed decoder) allows for data-aware power reduction by adding classical bus inversion–both attractive features for 3D D2D interconnects.

5.3.2   Spatial Coding Redundancy

The coding scheme in [109] does not account for the clustered and particle defect condition in its analysis; however, there is no fundamental limitation to overlaying a bump mapping scheme that strategically places spare bumps or TSVs. Consequently, a spatial coding redundancy scheme is proposed as a physical implementation of [109]. The premise is to distribute the ***coding groups*** (i.e., signals coded together + coded signal) such that each signal in a group is sufficiently far enough away from each other, i.e., at least a maximum defect diameter away. By spreading the group out across the bump array, clustered and particle defects can be repaired because they affect only one signal per coding group, as long as another defect does not affect another signal in the same group. For μbump and hybrid bonding, the bump array pattern is hexagonal and square, respectively. By knowing the diameter of the

largest tolerable defect area[3], the maximum number of signal bumps that can fit inside that diameter can be deduced by **hexagonal or square packing in a circle** [126]. Each signal in this circle, called a ***coding cluster***, belongs to a distinct coding group, and all clusters have the same signal ordering. Given the number of signals in a group, we can arrange these circles into the overall bump array area, which is ideally square at the module level [107]. Hence, **circle in square packing** [127] can be used to find the optimal arrangement of coding clusters to minimize the required area. Figure 5.5a shows the square-in-circle packing concept. Figure 5.5b shows the circle-in-square packing concept, with the members of a single coding group highlighted, demonstrating how a particle up to the size of the coding cluster is unable to affect multiple signals in the same coding group.

It is also evident that spatial coding redundancy has a much higher repair rate than module-shift redundancy, because more than one defect can be simultaneously repaired within the PHY area. This comes at the cost of slightly higher logic and CAM overhead, which requires $n \times log_2(c)$ bits, where n is the cluster size (squares per circle) and c is the number of clusters per module (circles per square)–significantly less than prior work (Sec. 5.3.1 above), but higher than module-shift redundancy (one bit per module).

(a) Square in circle   (b) Circle in square and demonstration of resistance to particle defects
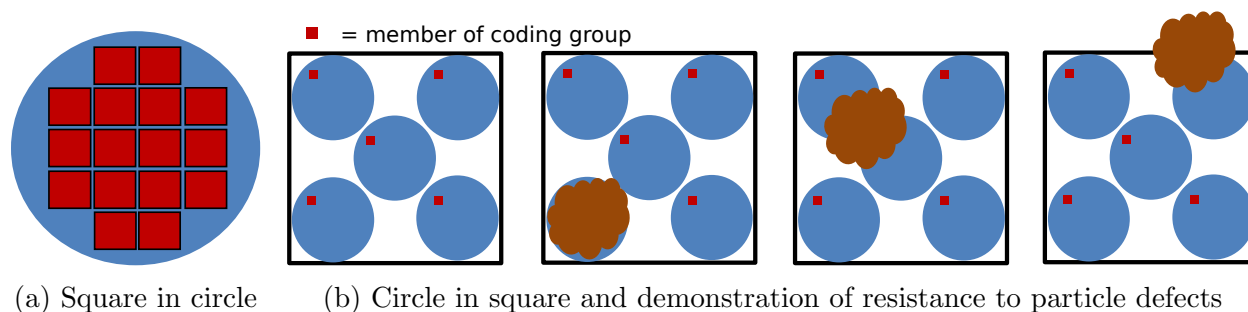
Figure 5.5: Spatial coding redundancy packing concept.

With this physical mapping, the critical path delay is increased proportionally to the distance between the signals in a group, i.e., the bond pitch multiplied by the particle diameter, and the number of bits in a coding group. In the configurable encoder/fixed decoder option, the coded signal is generated with a N-input mux, and then broadcast to XORs in the decoder. In the fixed encoder/configurable decoder option, the coded signal is a copy of the last signal in the coding group, and decoding is done with a forwards/backwards chain of XNORs. Given this difference, the former option generally incurs less routing-induced delay than the latter once the signals in the coding group are spread out in the configuration of Fig. 5.5b, as long as the coded signal is kept as close to the center of the coding group as possible. One final choice is to decide where the coding logic goes–on the core-facing or bump-facing side of the I/O retimer flop, shown in Figure 5.6. In a real implementation, the clock tree insertion delay to the flops will likely exceed the critical

---

[3]This diameter is derived from a system-level calculation of technology-specific defect densities and target yield. This information is proprietary and beyond the scope of this dissertation.

path delay through the coding logic, and constraining the location of the coding logic to the I/O cell area is detrimental given how the signals in a coding group are interconnected. Furthermore, the 0.2UI delay spread allowable for D_{tx} and D_{rx} demands as few cells as possible between the flops and channel for maximum timing margin. Hence, the lower implementation (logic on core-facing side) should be chosen at the expense of slightly higher switching power. A similar argument holds for placing the muxes on the core-facing side of the logic in module-shift redundancy.
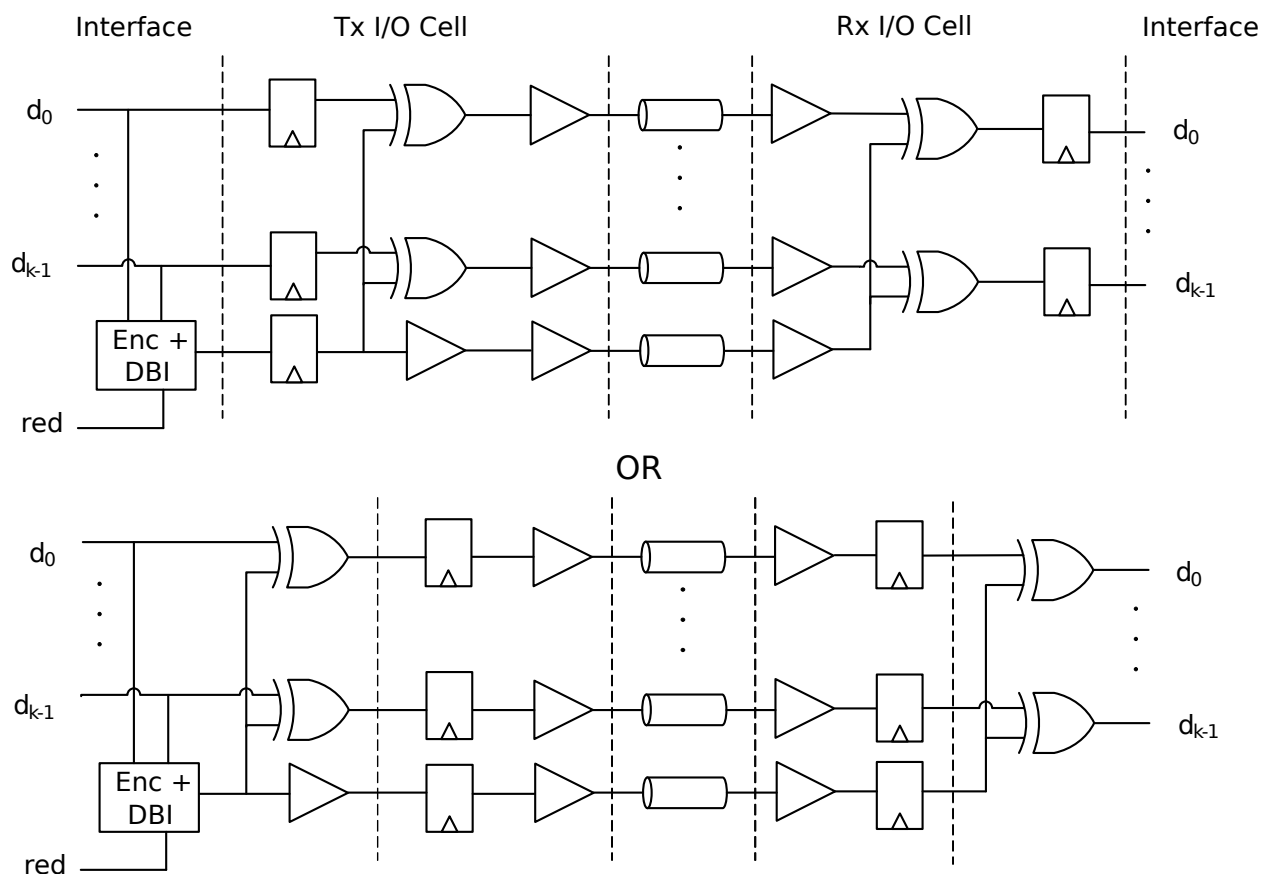
Figure 5.6: Logic/flop location in the spatial coding redundancy scheme. Bump-facing side (top) versus core-facing side (bottom).

5.4 Implementation

Signal-to-bump assignment is fundamentally different for each available redundancy scheme, requiring a program to perform redundancy calculations to generate a bump map. To accomplish this, the generator is architected to treat bump mapping as the primary task from which all other information (logic, collateral) is derived. Fig. 5.7 highlights the major inputs

(parameters), compilation tasks, and outputs (PD collateral) of this generator, each of which
are elaborated on below.
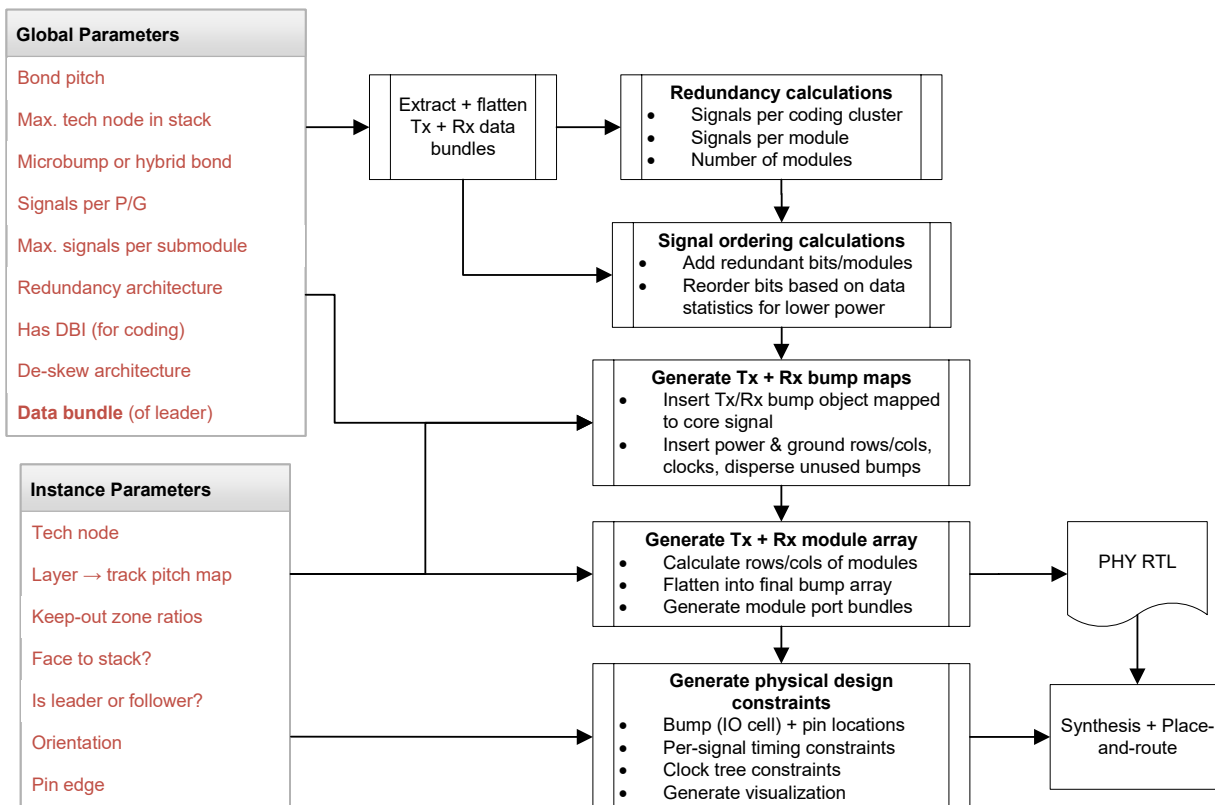
Figure 5.7: Generator architecture: parameters, compilation, and outputs.

5.4.1   Parameters

The separate Global (shared) and Instance (private) parameters are sub-divided into Tech-
nology and Design parameters. Some parameters are required and some are optional, and
users only need to explicitly specify parameter values that override the defaults. Global tech-
nology parameters include constants related to timing, bond pitch and type, and particle
defect size. Signal/power integrity (SI/PI) rules are also encoded globally as signals between
power/ground (P/G) and signals per module parameters. Global design parameters include
redundancy scheme, pin side, data statistics, and most importantly, the **data bundle** (the
Verilog port list of data to transfer). Instance technology parameters include keep-out zones
(KOZs), pin layers, front/backside power, on-chip delay model constants, and nominal volt-
age. Instance design parameters include stacking orientation, origin offset, and names of
certain cells. While none of the instance parameters encode proprietary technology process

information, they are still design-specific and should be kept private. Refer to Appendix B.1 for a detailed description of the available parameters and their descriptions.

5.4.2   Bump Mapping

The bump map is calculated in multiple steps from the input parameter set, as summarized in Figure 5.7. First, an ordered list of Tx and Rx signal bits is extracted from the global parameters by filtering the data bundle parameter for directionality (input or output), and then blasting apart the buses into individual bits while copying the signal name and bit index. Next, the status of the PHY instance as a leader or follower must be considered. Because the data bundle is defined globally with respect to the leader PHY, the direction of the data bits in follower PHY instances must be flipped. Finally, information about the data, such as the name, bit index, direction, related clock, pin location, and more, is containerized into a core signal data structure (`I3DCore` - see Appendix B.2).

Next, the procedure diverges depending on redundancy architecture (Sections 5.4.2.1 and 5.4.2.2 below), culminating in a bump map that is in the form of a three-dimensional array of bump container instances (`I3DBump` - see Appendix B.2). This container encodes information such as the bump name, connected core signal instance, bump position, module index, and Synopsys Design Constraints (SDC) timing constraint commands. Two bump maps are constructed in parallel for the Tx and Rx signals, then interleaved into a final bump map representing the entire full-duplex interconnect. Depending on additional information such as the leader/follower status, front/back attach configuration, and the intended flipping axis, this bump map array is mirrored accordingly to ensure alignment. This bump map is flattened further for use by internal data bundle generators, described in Sec. 5.4.3.

5.4.2.1   Coding Redundancy Calculation

Constructing the coding clusters in the spatial coding redundancy scheme described in Section 5.3.2 to sufficiently protect against particle defects while maintaining a modularized approach to the array construction is particularly suited for a generator. Particles are assumed to blow out a circular region of the bump array, hence it is specified as a maximum diameter in number of bumps (which, in this case, are required to be squarely gridded, i.e, have the same pitch in both dimensions). From the input parameters, the calculation procedure proceeds as follows:

1. Compute the coding cluster properties, which are the cluster diameter, number of clusters, cluster packing pattern, and signals per cluster. Packing algorithms are approximated with look-up tables because the bump array is not freeform (i.e., bumps must be on a fixed grid). A subset of the allowed square packing arrangements and their diameters is shown in Figure 5.8. Note that arrangements containing more than the largest cluster (max. 37) bumps is not very useful, as the number of signals in the cluster becomes exceedingly large and the design will likely run into clock tree delay
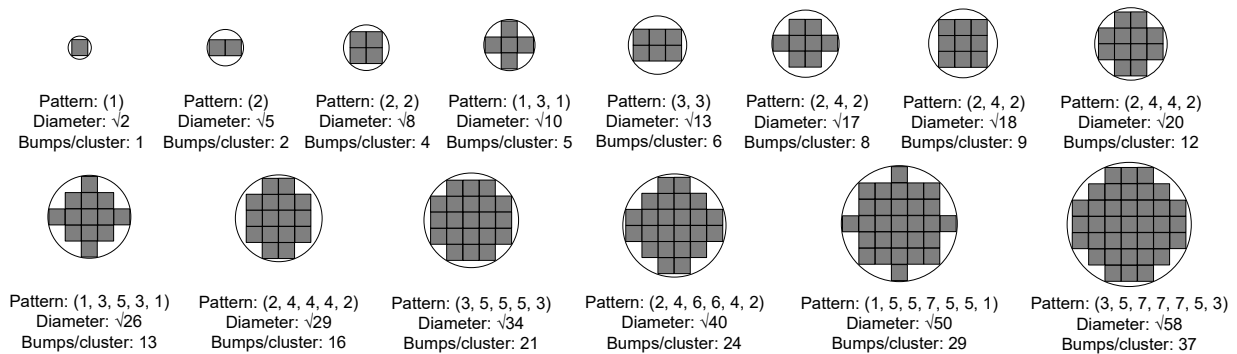
Figure 5.8: Square in circle packing arrangements.
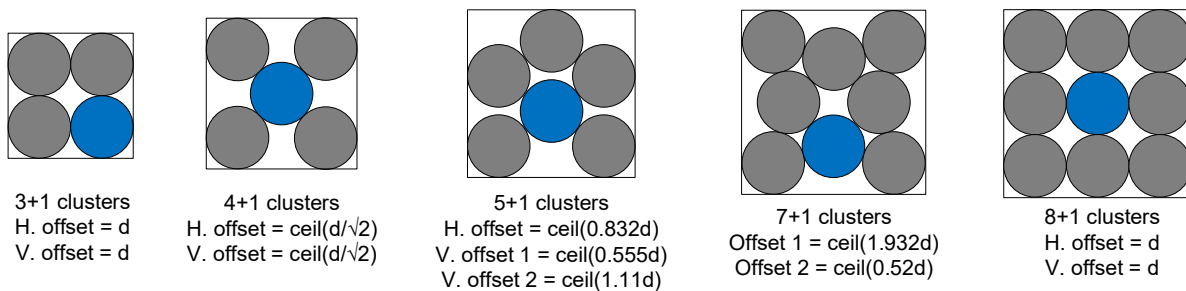
Figure 5.9: Circle in square packing arrangements.

and signal integrity issues. To ensure full repairability with a maximum diameter, the pattern that has a radius just over the maximum particle diameter is chosen. For circle packing, coding group sizes are kept within 4 to 9 to best balance area overhead and coding logic delay and are shown in in Figure 5.9. Note that the offsets given in the Figure are exact; subsequent snapping of the circles so the bumps are on-grid results in slightly lower packing density. SI/PI parameters are also used to calculate the signals per cluster.

2. Generate the bump assignment order and coordinates within and around each cluster using recursive algorithms. Inside the cluster, bus switching statistics are used to calculate the assignment order to minimize power due to coupling capacitance [109], such as spiral assignment for sequential data and sawtooth assignment for normally-distributed data. Note that the edge effects as described in [109] are not fully-present because clusters are surrounded by power/ground and modules are abutted. Only clusters adjacent to the patch's perimeter may experience edge effects, depending on how the patch is integrated.

3. Compute data, P/G, and clock bump coordinates across the module. The cluster containing the coded bits (highlighted in blue in Figure 5.9) is placed nearest to the center

of the module to minimize routing delay. Ground is filled into the ring surrounding
clusters and power is filled into extra space inside and between clusters. Finally, the
two clock bumps are assigned nearest to the center of the module but spaced at least
a particle diameter apart, so that they are closest to the root of the clock tree and
satisfy the particle defect repairability criterion.

Debug information is given following this calculation procedure. For example, a configu-
ration with 72 signals per module, 4 modules per Tx/Rx, and a maximum particle diameter
of 5 will result in the following:

```
Calculating Patch parameters...
    Patch dimensions: 2 rows by 2 columns of modules
    Circles per module: 5 with diameter: 5.830951894845301
    Signals per circle: 18 in pattern: List(3, 5, 5, 5, 3)
    Module dimensions: 15 rows by 15 columns of bumps
    Signal to P/G ratio: 90:260
```

5.4.2.2   Default Calculation

The calculation for shifting and no redundancy is much simpler than for coding redundancy,
as there is no clustering calculation; however, SI/PI and data statistic-dependent signal
assignment is still used. From the input parameters, the calculation procedure proceeds as
follows:

1. Calculate the rows and columns in each module, prioritizing the least number of bumps
   and square aspect ratio after accounting for the required P/G rows/columns for SI/PI.

2. Calculate the P/G bump row and column indices using a binary recursive algorithm
   that evenly diffuses the signals in each dimension, taking into account sharing of P/G
   over adjacent module boundaries.

3. Order the signal bumps row-by-row over the remaining bumps, using the same switch-
   ing statistic-based rules. Remaining unassigned bumps are set to ground using a re-
   cursive algorithm that traverses around the module from the corners inwards, so that
   the radius of the clock tree is minimized.

This calculation outputs debug information as well, for example for a configuration with
72 signals per module, 4 signals between ground, and 3 signals between power:

```
Calculating Patch parameters...
    Patch dimensions: 2 rows by 3 columns of modules
    Module dimensions: 12 rows by 10 columns of bumps
    Row pattern (sig/pwr): S S P S S S P S S S P S
    Column pattern (sig/gnd): S S G S S S S G S S
    Signal to P/G ratio: 72:54
```

5.4.3   Logic Generation
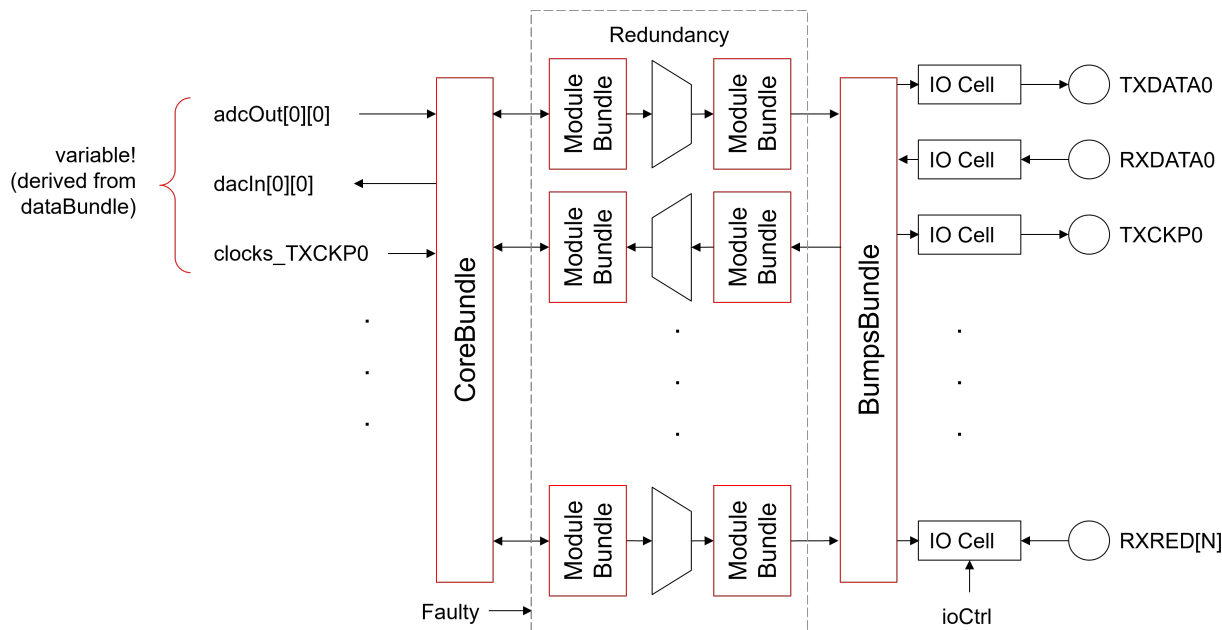
Figure 5.10: Programmatic bundle generation and connection.

Because the bump map is generated from the data bundle and the redundancy calculations are variable, the ports of the modules inside the PHY—for example, the module-level coding or muxing block—are not known ahead of time. This means that along with the logic, every module's port bundle must also be generated and connected programmatically at compile time from the resolved bump map.

Pattern matching is first applied on the flattened bump map to filter out P/G bump instances. Ordering is preserved because the signal name/type is mapped into bundles representing the core-facing signals and the obfuscated bump-facing signals (e.g., `TXDATA0`, `RXCKR1`). Additional filtering based on the module index is used to create the intermediate bundles surrounding each module-level redundancy block. Bulk connections between generated bundles is done with order- and name-based matching and bus reconstruction, thereby avoiding all manual signal-to-bump mapping and removing a significant source of design error. A visualization of the generated bundles and connections is shown in in Figure 5.10 for an exemplar ADC and DAC array data bundle. Further demonstration of how pattern matching and data structure manipulation is used to create and connect these bundles is shown in the annotated code of Listing B.1 for creating a `ModuleBundle` and Listing B.2 for bundle connection.

At the top-level, the interconnect is constructed as a core-facing to bump-facing bundle connection with optional redundancy blocks in between the core bundle and an array of IO cells. Depending on the redundancy architecture, module-level redundancy blocks (either

`RedundancyMux`, `Encoder`, or `Decoder`) are generated with `ModuleBundle`s exposed. Listing
B.3 demonstrates how coding groups are created completely from the bump map data structure. By default, the IO cells are behavioral but can optionally point to a custom I/O cell,
such as in cases where custom drivers with ESD protection structures are needed. Finally,
configuration logic is overlaid differently depending on the selected protocol. At this time, a
choice of bare wires or memory-mapped registers on a Tilelink [42] or AXI bus is available.
All the logic is emitted as a set of Verilog files.

5.4.4 Collateral Generation

In order to implement the PHY with digital PD tools, multiple collateral data structures must
be emitted from the generator in sync with the logic: 1) floorplan, 2) bump assignment, 3)
pin assignment, and 4) timing constraints. The requirements and procedures for generating
each of these items is detailed in the following subsections.

5.4.4.1 Floorplan Constraints

Floorplan constraints are emitted to a Design Exchange Format (DEF) or Hammer IR [25]
file. Bump coordinates and port assignments are derived from the bump map data structure.
The coordinates are also used to generate IO cell placement guides and KOZ blockages,
unless custom IO cells are used. Pins are then placed along the core-facing edge in a fashion
that simultaneously minimizes the pin to IO cell routing distance and inter-signal coupling
capacitance. First, a check for sufficient track resources on each layer is performed using
a 20% power strap density and 50% signal density estimate. Next, pins are then assigned
starting in the center of each row or column of the bump array on the lowest layer and
assigned to the signal connected to the bump nearest to the pin edge. Assignment proceeds
in an alternating pattern around the center for successive signal bumps, going up the specified
pin layers as routing tracks are exhausted within the bump pitch. This ensures that bumps
furthest away from the pin edge are assigned the higher, less lossy layers. Though redundancy
adds logic in between the pins and IO cells, this pin placement still functionally guides the
placement of the redundancy logic for improved timing closure.

5.4.4.2 Visualization

A floorplan visualization using the Scala Doodle library [128] is also generated in PNG, PDF,
and live preview format. At this time, the visualization only supports square bump patterns
for now due to the way Doodle handles shape tiling using `.above` and `.beside` functions.
Bumps are drawn as color-coded (by bump class) circles with a diameter of half bump pitch
on an invisible square. Inside each bump, the bump name and core signal name are printed.
These objects are tiled row-by-row and column-by-column as derived directly from the bump
map data structure. Next, dotted grid lines denoting the boundaries between modules is
overlaid on top of the bump map, encoded as a separate picture with tiled rectangles with
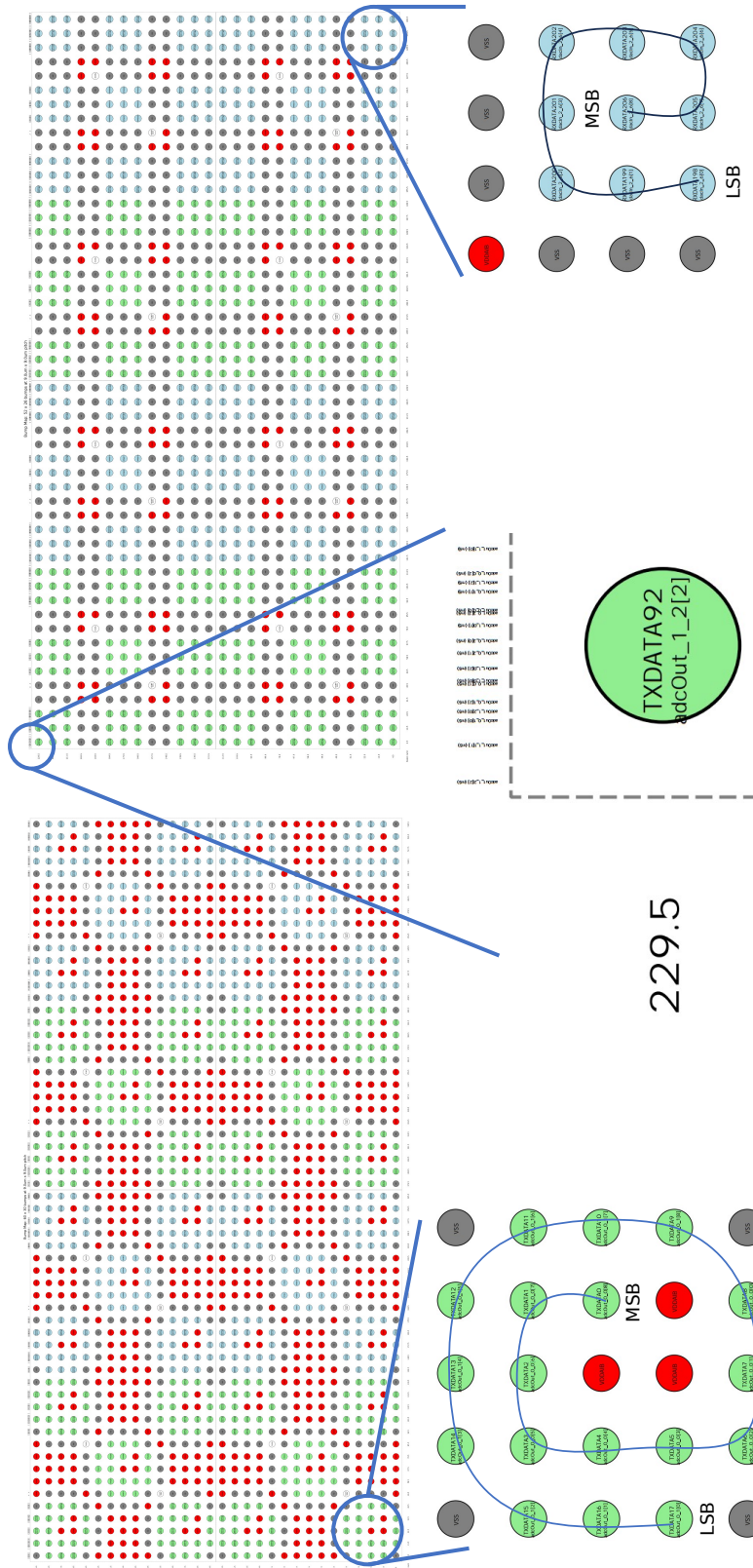
Figure 5.11: Doodle-generated visualizations for coding (left) and shifting (right) redundancy.

dashed outline and no fill. Helpful text, such as a title and rulers for each bump array row
and column are overlaid around the bump map. Finally, pins are drawn as squares with size
derived from the pin layer's pitch with the pin name printed beside it.

Fig. 5.11 shows an example of the generated visualization for both redundancy schemes,
with inset details of data statistic-aware signal-to-bump assignment described in Sec. 5.4,
pin placement, and rulers shown. This visualization is the best demonstration of how this
generator replaces the status quo of spreadsheet-based bump mapping and floorplanning.

5.4.4.3 Timing: Clocks (Transmit Modules)

One of the challenges of meeting the D_{tx} specification with digital tools is the need to
concurrently optimize the forwarded clock path with the data path.

In transmit modules, the single incoming clock must be split into the clocks going
to the IO cells and the forwarded clock. Logically, each redundancy module outputs a
`clksToTx` output to create the IO cell clock from. In SDC, this looks like the following
(`set_clock_uncertainty` is omitted):

```
1  create_clock [get_ports clocks_TXCKP0] -name TXCKP0 -period $tPeriod
2  create_generated_clock -name io_TXCKP0 -source [get_ports clocks_TXCKP0]
   ↪  -divide_by 1 [get_pins */clksToTx_0]
3  create_generated_clock -name out_TXCKP0 -source [get_ports
   ↪  clocks_TXCKP0] -divide_by 1 [get_ports TXCKP0]
4  set_clock_groups -group {TXCKP0 io_TXCKP0 out_TXCKP0}
```

Listing 5.1: Transmit clock SDC example.

While the clock is all *logically* the same net (and therefore may be optimized accordingly
by synthesis), generating multiple clocks is required to a) coerce P&R to perform the correct
optimization order, and b) make it more seamless to switch between redundancy schemes. In
P&R, the IO cells' clock needs to be optimized using Clock Tree Synthesis (CTS), while the
forwarded clock must be treated as a datapath instead. If the forwarded clock is erroneously
fixed after CTS, P&R will attempt to meet timing by inserting buffers in the data path (i.e.,
before or after the flip-flop in Tx IO cells), which is undesired because it increases the corner-
to-corner delay spread and adds unnecessary power. Therefore, a combination of generated
clocks, `set_data_check`, and tool configuration selecting clock-as-datapath optimization is
required (described in section 5.5.1).

For coding redundancy, an extra `out_TXCKR0` is created on the redundant clock bump. For
shifting redundancy, the clock muxing is marginally more complex and must be managed via
separate SDC constraints depending on a case analysis on the `shift` pins of the redundancy
muxes. For example, the shifted case analysis for module #2's clock domain (note the source
of `out_TXCKP2`) has clock constraints as follows:

```
1  set_case_analysis 1 [get_pins shifting/*Muxes*/shift]
2  create_clock [get_ports clocks_TXCKP2] -name TXCKP2 -period $tPeriod
3  create_generated_clock -name io_TXCKP2 -source [get_ports clocks_TXCKP2]
   ↪  -divide_by 1 [get_pins */clksToTx_2]
4  create_generated_clock -name out_TXCKP2 -source [get_ports
   ↪  clocks_TXCKP0] -divide_by 1 [get_ports TXCKP2]
5  set_clock_groups -group {TXCKP2 io_TXCKP2 out_TXCKP2}
```

Listing 5.2: Transmit clock with shift redundancy SDC example.

5.4.4.4   Timing: Clocks (Receive Modules)

First, the main clock is created at the receive bump, followed by generated clocks for the data IO cells and core-side output clock with an inversion applied. In SDC, this looks like the following (set_clock_uncertainty is omitted):

```
1  create_clock [get_ports RXCKP0] -name RXCKP0 -period $tPeriod
2  create_generated_clock -name io_RXCKP0 -source [get_ports RXCKP0]
   ↪  -divide_by 1 -invert [get_pins */clksToRx_0]
3  create_generated_clock -name out_RXCKP0 -source [get_ports RXCKP0]
   ↪  -divide_by 1 -invert [get_ports clocks_RXCKP0]
4  set_clock_groups -group {RXCKP0 io_RXCKP0 out_RXCKP0}
```

Listing 5.3: Receive clock SDC example.

For coding redundancy, an clock is created for the redundant clock bump, named RXCKR0, for example. Since these clocks are mutually exclusive, a set_false_path is also emitted. A similar strategy of emitting separate SDC constraints for the default and shifted case is used for shifting redundancy.

5.4.4.5   Timing: Delays (Transmit Modules)

Because this circuit architeture has only one sequential element between input and output (Fig. 5.3), CTS cannot borrow time between sequential elements to meet input delay on one side and output delay on the other. Therefore, specification of input/output delays and clock tree insertion delays (ID) must be derived from the floorplan.

The clock tree ID should be wholly constrained by the core-side input delay and minimized (i.e., no excess buffers in the clock tree distribution). To do this, the clock tree ID must be estimated pre-synthesis using a ps/mm model, with input delays adjusted accordingly. It would be an under-constrained problem if, instead of using this distance-based delay estimation, the tool was allowed to adjust the input delay constraint using the calculated clock tree network delay instead. The ID of a given module is approximated by the module's

position in the array relative to the core-facing pin edge multiplied by the average routing
delay in the module, which is calculated as the average module width and height multiplied
by the ps/mm value. Using the average is sufficient because during bump map creation, the
dimensions of each module is made to be roughly square.

For clocks, the following is an example of calculated delays for pins along the North edge
and modules at coordinates (0, 0) and (0, 1):

$$D_{mod} = \frac{rows \times pitch_V + cols \times pitch_H}{2} \times \frac{psmm}{1e6}$$

$$D_{(0,0)} = 2.5 \times D_{mod}$$

$$D_{(0,1)} = 1.5 \times D_{mod}$$

Traversal across a module incurs a delay of D_{mod}, and the clock tree ID within each
module is also D_{mod} due to the approximate routing of the clock tree, i.e., a half-perimeter
wire length (HPWL) calculation from the clock tree root's location in the center of the
module to the farthest bump in the module. Finally, traversing from the edge of the module
to the center accounts for the additional factor of 0.5. A visual example of this calculation
is shown in the left side of Figure 5.12.
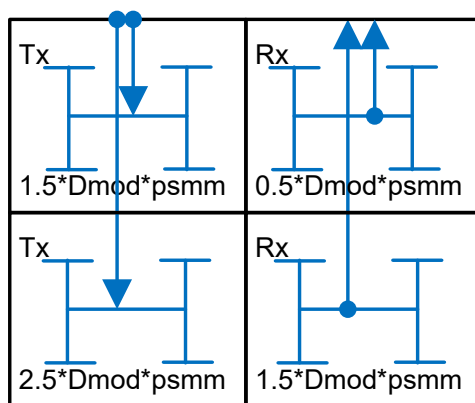
Figure 5.12: HPWL calculation for clock tree insertion and core-facing input/output delays
(pins on North edge).

Input delays for transmit data are calculated exactly the same way, except the $1 \times D_{mod}$
factor due to the clock tree's HPWL is not present. The input delay min/max budget is also
dependent on the type of redundancy used. With redundancy, because there is logic before
the retimer, the budget is set to within the second quartile of the clock period. Without
redundancy, the budget is set to within the third quartile. This sufficiently minimizes the
clock tree ID while ensuring that hold buffers are not inserted in the data path. Finally,
this clock tree ID and input delay is written out for each pin using the SDC commands
`set_clock_latency` and `set_input_delay`.

Output delay (D_{tx}) is, as described above, done by forcing the forwarded clock to be adjusted instead. This is accomplished using individual `set_data_check`s from the forwarding clock output (e.g., `[get_clocks out_TXCKP0]`) to each data bit in the module, instead of `set_output_delay`. This was found to be more reliable in coercing clock tree synthesis to adjust the clock as though it was a data path. For coding redundancy, an additional set of `set_data_check`s are added to ensure that D_{tx} is met regardless of which clock is forwarded. Since `set_data_check` is a same-cycle constraint and the receive side is clocked on the falling edge of the forwarded clock and output clock duty cycle cannot be easily constrained using fully-digital design, extra calculation is required for the constraint value. The delay is calculated from the specification curve (Eqns. 5.1) by plugging in the nominal supply voltage $\pm$ 10% (for setup/hold corner analysis) and the clock period. Lane-to-lane skew is also accounted for (but not constrained — see below) in the constraint, by adding the skew budget above max delay value. In summary the values for `set_data_check` commands are calculated as follows, where t_j is the clock uncertainty that must be de-embedded from the constraint to avoid over-pessimism and t_{skew} is the lane-to-lane skew budget:

$$D_{tx,max(\mathrm{SDC})} = -D_{tx,max}(0.9 \cdot V_{cc,nom}, t_{period}) + t_{skew} - t_j - 0.5 \cdot t_{period}$$
$$D_{tx,min(\mathrm{SDC})} = D_{tx,min}(1.1 \cdot V_{cc,nom}, t_{period}) - t_j + 1.5 \cdot t_{period}$$

5.4.4.6 Timing: Delays (Receive Modules)

Clock latency estimation is similar to the transmit side, except that the ID for the `io_RXCKP#` clocks are always equal to 1 module delay and the output clock delay is reduced by 1 module delay.

Input delay is also calculated using the specification timing parameters. D_{rx}, supply noise, pulse width jitter, channel-induced eye closure, and data/clock differential jitter parameters must be summed together to give the minimum and maximum possible arrival times of the input data relative to the received clock. Contributions of the jitter parameters are split half/half between the min/max delay, which is likely mildly skewed, however, their contributions are relatively small. Including clock uncertainty in the timing check is sufficient margin on top of this error. The related clock must be the inverted clock as created at the receive bump. Coding redundancy has a duplicate set of input delay constraints against the redundant clock bump. The input delay is calculated as follows (see [108] for more details on the other timing variables):

$$D_{rx,max(\mathrm{SDC})} = D_{rx,max}(0.9 \cdot V_{cc,nom}, t_{period}) \frac{1 + \alpha_{trx} n_{vcc}}{2} + \frac{j_{pw}}{2} + \frac{Ch}{2} + \frac{t_j}{\sqrt{2}}$$
$$D_{rx,min(\mathrm{SDC})} = D_{rx,min}(1.1 \cdot V_{cc,nom}, t_{period}) \frac{1 - \alpha_{trx} n_{vcc}}{2} - \frac{j_{pw}}{2} - \frac{Ch}{2} - \frac{t_j}{\sqrt{2}}$$

Output delay is far simpler, as it can be constrained relative to the pass-through receive clock, whose delay does not need to be as tightly constrained as on the transmit side. To

remove redundancy scheme dependence, the output delay is constrained to the first half of
the clock period.

5.4.4.7 Timing: Lane-to-Lane Skew

Constraining lane-to-lane skew is critical for preserving timing margin across the entire
module. This is done by pairwise `set_data_check`s between lanes in each clock domain.
While not specified by the standard, the output pins for receive data are also be constrained
to have the same lane-to-lane skew to facilitate integration. Clock uncertainty is once again
de-embedded from the constraint so that only the skew specification is considered. In SDC,
if the variable `sigs` is set to the list of Tx bumps in the module or core-facing Rx pins:

```
1  foreach s1 $sigs {
2      foreach s2 $sigs {
3          if {$s1 ne $s2} {
4              set_data_check [expr -$tskew - $tj] -from [get_ports $s1]
   ↪   -to [get_ports $s2]
5          }
6      }
7  }
```

Listing 5.4: Constraining lane-to-lane skew.

5.4.4.8 Timing: Other

Finally, a few other timing constraints are required. Most importantly, the bump load ca-
pacitance must be estimated as a function of the bump technology. For the purposes of
initial exploration, the bump load for 9μm face-to-face hybrid bonding is estimated to be
40fF for the entire channel, and resistance is negligible. The load should be extrapolated
downwards for smaller pitches, though at this time, a curve for this has not yet been devel-
oped, especially for TSVs because the information is proprietary. For all other outputs, the
load is estimated at 5fF and capped at 10fF.

Next, transition times are constrained to provide a good tradeoff between jitter accumu-
lation and power consumption. Internally, and for core-side inputs, transition time is set to
a maximum of 50ps. At the bumps, it is constrained between $t_{period}/10$ and $t_{period}/6$.

Lastly, duty cycle distortion and data rise/fall differential delay can be constrained using
`set_min_pulse_width` as $0.48 \times t_{period}$ for clocks and $t_{period} + t_{jpw}/2$ for data, respectively.

A summary of all of the SDC constraints visualized against the signaling waveform is
given in Figure 5.13.

Clock/Data   **Waveform**   **SDC command(s)**

Legend:
→ Tech LIB arc
→ SDC constraint
Constraint useful
for redundancy

clocks_TXCKP# (input)

```
create_clock -name TXCKP# -period $Tcyc [get_ports clocks_TXCKP#]
set_clock_groups -group {TXCKP# io_TXCKP# out_TXCKP#} -group ...
```

TXCKP# (after clock tree)

```
create_generated_clock -name io_TXCKP# -source clocks_TXCKP# -divide_by 1 [get_pins */clksToTx_#]
set_clock_latency -min/max Dmod [get_clocks io_TXCKP#]
set_clock_skew Tskew/2 [get_clocks io_TXCKP#]
```

Tx data bit (input)

```
set_input_delay -clock TXCKP# -min/max Dmod [get_ports <data>]
Tx:
create_generated_clock -name out_TXCKP# -source clocks_TXCKP# -divide_by 1 [get_ports TXCKP#]
set_clock_latency -min/max Dmod [get_clocks out_TXCKP#]
set_min_pulse_width [expr 0.48*Tcyc] [get_ports TXCKP*]
set_min_transition [expr Tcyc/10] [get_ports TXCKP*]
set_max_transition [expr Tcyc/6] [get_ports TXCKP*]
set_load Cbump [get_ports TXCKP*]
```

TXCKP# (at bump)
RXCKP# (at bump)

```
Rx:
set_input_transition -min [expr Tcyc/10] [get_ports RXCKP*]
set_input_transition -max [expr Tcyc/6] [get_ports RXCKP*]
create_clock -name invert_RXCKP# -period $Tcyc [get_ports RXCKP#]
Tx:
set_data_check [expr Dtxmax – Tcyc/2] -fall_from [get_ports TXCKP#] -to [get_ports TXDATA#] -setup
set_data_check [expr Dtxmin + 1.5*Tcyc] -fall_from [get_ports TXCKP#] -to [get_ports TXDATA#] -hold
set_data_check Tskew -from [get_ports TXDATA#] -to [get_ports <other TXDATA in module>]
set_min_pulse_width [expr Tcyc + Tjpw/2] [get_ports TXDATA*]
set_min_transition [expr Tcyc/10] [get_ports TXDATA*]
set_max_transition [expr Tcyc/6] [get_ports TXDATA*]
set_load $Cbump [get_ports TXDATA*]
```

Data bit (at bump)

```
Rx:
set_input_transition -min [expr Tcyc/10] [get_ports RXDATA*]
set_input_transition -max [expr Tcyc/6] [get_ports RXDATA*]
set_input_delay -clock RXCKP# -min [expr Dtxmin – Tjpw/2 – Tch/2 – Tj/2] [get_ports RXDATA#]
set_input_delay -clock RXCKP# -max [expr Dtxmax + Tjpw/2 + Tch/2 + Tj/2] [get_ports RXDATA#]
set_false_path -from [get_clocks RXCKP#] -to [get_clocks RXCKR#]
set_false_path -from [get_clocks RXCKR#] -to [get_clocks RXCKP#]
```

RXCKP# (after
inversion)

```
create_generated_clock -name RXCKP# -invert -source [get_ports RXCKP#] [get_pins <iocell_RXCKP#/o>]
set_clock_groups -asynchronous -group RXCKP# -group ...
create_generated_clock -name io_RXCKP# -source [get_pins <iocell_RXCKP#/o>] -divide_by 1 [get_pins */clksToRx_#]
create_generated_clock -name out_RXCKP# -source [get_pins <iocell_RXCKP#/o>] -divide_by 1 [get_pins */clksToRx_#]
set_clock_latency -min/max Dmod [get_clocks io_RXCKP#]
```

RXCKP# (after clock tree)
and clocks_RXCKP# (output)

```
set_clock_latency -min/max Dmod [get_clocks out_RXCKP#]
set_clock_skew Tskew/2 [get_clocks io_RXCKP#]
set_min_pulse_width [expr 0.48*Tcyc] [get_ports out_RXCKP#]
set_max_capacitance Cint [get_ports clocks_RXCKP*]
```

Rx data bit (output)

```
set_output_delay Tcyc/2 -clock [get_clocks out_RXCKP#] -max [get_ports <data>]
set_output_delay 0 -clock [get_clocks out_RXCKP#] -min [get_ports <data>]
set_max_capacitance Cint [get_ports <data>]
set_data_check Tskew -from [get_ports <data>] -to [get_ports <all other data>]
```

(Global/internal signals)

Latency is
unconstrained

```
create_clock -name clock -period Tslow [get_ports clock]
set_clock_uncertainty [expr Tcyc/10] [all_clocks]
set_propagated_clocks [all_clocks]
set_false_path -from <config bits>
set_max_transition 0.05 [current_design]
set_load 0.005 [all_outputs]
set_max_dynamic_power [expr 0.05pJ/b * bits / Tcyc(in ns)]mW
```

Figure 5.13: Visualized SDC constraints.

5.4.4.9 File Emission

During compilation, the bump map and list of IO cell instances is passed to various collateral emission functions. Each IO cell instance has a `forBump` field that attaches it to a specific bump object in the bump map. As the bump map and IO cells are traversed, the fields contained within each bump instance and the IO cell instance name are used to generate all of the collateral as previously described. Finally, the collateral files are written out using the `ElaborationArtefacts` API of the `rocket-chip` project. In this way, the generator does not require any custom FIRRTL/CIRCT passes. Although Chisel Aspects [79] were considered, `InjectingAspect` was removed since Chisel 3.6 and FIRRTL → CIRCT migration, so it is no longer possible to inject information to Chisel constructs from an entirely separate framework, as demonstrated by the Chisel Floorplanning framework [129]. Since only the `instanceName` of IO cells are required, and all other information is handled using internal data structures, and the logic is derived from those data structures, use of the remaining

`InspectingAspect` would be overkill for this use case.

5.5   Evaluation

Implementation experiments were performed using the Hammer PD flow generator [25], targeting Intel 16 [44] and Cadence Genus/Innovus 23.1. The generator was executed within Chipyard v1.10.0 [23] atop Chisel 3.5.1. Although the specific process design kit used does not have hybrid bond bump cells or TSVs, PD constraints (KOZs, loads) are sufficient to emulate their effects.

5.5.1   Physical Design

Because the timing and power targets for a 3D D2D PHY are extrapolated from manual circuits and layout, P&R tools require extensive tweaking to meet these targets. Some generalized optimizations yielding positive results include:

- Clock as datapath optimization on forwarded clocks and enabling/increasing the priority of data-to-data checks

- Strong placement guides for IO cell logic

- Adjusting relative timing, power, and design rule violation (DRV) optimization efforts

- Connectivity and SDC-driven skew grouping

- Aggressive useful skew optimization

- Enabling signoff timing optimization with on-chip variation (OCV), waveform-based models, and path propagation in timing calculation

- Non-default routing (NDR) rules, especially for clocks

- Selectively disabling hold fixing on certain path groups

Certain settings were observed to be not useful or even adversely affected quality of results:

- Physical synthesis and effort spent on post-synthesis timing closure–due to the dependence on clock latency. Timing optimization mostly occurs in P&R instead.

- Constraining overall datapath latency, because it effectively demotes the priority of D_{tx} and D_{rx}-derived constraints.

- Allowing clock tree rebuffering during timing optimization can add undesired clock tree skew and power.

- Forcing polarity in the datapath, e.g., inverting flip-flop followed by inverter to reduce stages often induces extra buffering to meet other constraints like transition time. Using custom IO cells is recommended instead.

5.5.2 Analysis

After STA, post-processing timing reports is required to verify compliance with timing specifications and debug PD issues such as undesired buffer insertion. An Innovus script is included in the generator repository that reports the following into CSV files:

- Clocks: rise/fall latency, DCD, insertion delay, skew

- Transmit data: min/max D_{tx} and slack, skew at flip-flop and output, input slack, data latency

- Receive data: slack and sampling aperture margin, output slack, data latency

- For all arcs: cumulative manhattan length, wire delay percentage, path cells

- Static power for random data patterns

- Logic cell density and routing utilization

5.6 Results

Results presented herein are a snapshot of the redundancy design space exploration process enabled by this generator. The following design parameters are chosen: 1) 9μm, the highest hybrid bonding pitch [108], to simulate the worst case timing and power scenario, 2) 4 Tx/4 Rx module array, to exercise all logic cases in module-shift redundancy, 3) 4:1 (signal:coded bits) coding redundancy ratio, to provide worst-case area overhead, and 4) 72 bits per module, to provide enough bits in the same clock domain for most data transfer use cases. Figures 5.14a and 5.14b show the generated layouts for coding and shifting redundancy, respectively.

Table 5.2 lists particular metrics of note as extracted from a typical P&R run. Note that routing utilization in the vertical direction is parallel to pin-to-IO cell routing, KOZ blockages are omitted from density and utilization calculations, and static power consumption is reported assuming random data statistics. The results show that despite its larger area and better signal:P/G ratio (indicative of low packing density), 4:1 coding redundancy offers similar power consumption as shifting redundancy for this configuration. However, larger signal:coded bit ratios will add more coding logic and shrinking the bond pitch will decrease the routing-dominated power of shifting redundancy. The redundancy overhead is also reflected in cell density and routing utilization. Shifting redundancy has higher routing utilization (visible in Fig. 5.14b) even after accounting for PHY dimensions, suggesting that

(a) Coding redundancy

(b) Shifting redundancy
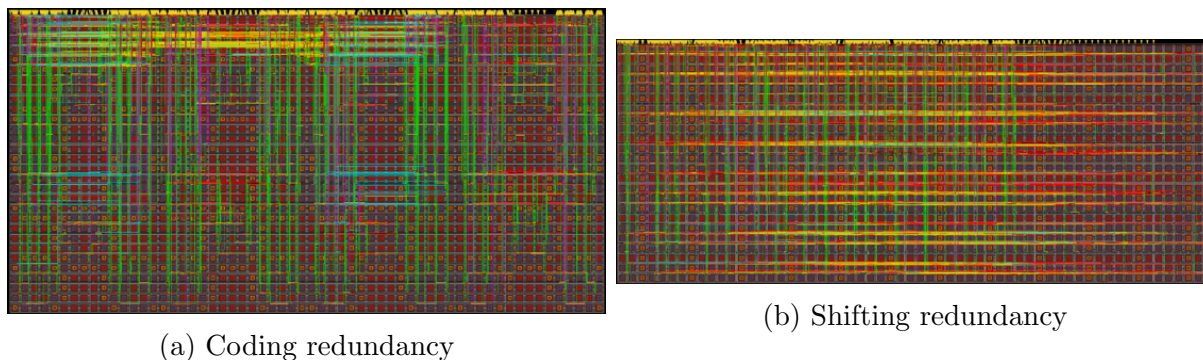
Figure 5.14: Innovus generated layouts.

Table 5.2: Metrics for coding and shifting redundancy.

Metric	Coding	Shifting (no shift)	Shifting (all shift)
		Redundancy Configuration	
Dimensions (rows × cols)	30 × 60	24 × 60	
Signal to P/G bump ratio	0.35	1.33	
Logic cell density (%)	6.1	6.8	
Routing util. (vert., %)	3.2	2.5	
Routing util. (horiz., %)	1.3	2.8	
Power ($\alpha = 0.5$, fJ/b)	24.7	17.4	24.1
Tx clock DCD (%)	-5.5 - 3.1	-2 - 4.9	0.9 - 12.3

it will run out of routing resources first as bond pitch scales down. Thus, at very fine pitches, coding redundancy may be necessary despite the power impact.

Timing-wise, the forwarded clock DCD constraint was never met, especially for shifting redundancy, likely due to the low priority of maintaining clock duty cycle in flip-flop based logic. The DCD is plotted for each clock domain (module) for coding (Fig. 5.15a) and shifting (Fig. 5.15b) redundancies, revealing no systematic reasons for why the DCD is worse for some domains or is positive or negative, except that shifting worsens it likely due to the higher number of cells in the clock path. To account for this DCD, the D_{tx} constraint is imposed relative to the falling edge of the forwarded clock to account for DCD to meet D_{rx} on the other die. However, meeting D_{tx} was still difficult, especially for shifting redundancy, as shown when comparing Fig. 5.16a to Figs. 5.16b and 5.16c. To interpret these plots, note that all points should be within the span of the shaded regions; however, there are clearly large variations depending on the shifting case analysis. This suggests that additional ps/mm model adjustments, transmit input delay tuning, and/or additional settings on the forwarded clock timing optimization are needed to satisfy both case analyses. However, in

general, the delay spread between the minimum D_{tx} in the hold corner and the maximum D_{tx} in the setup corner is still within specification when also accounting for lane-to-lane skew. Excessive DCD may have also contributed to difficulty in meeting D_{tx} for certain modules. Run-to-run results were also inconsistent, reflecting the stochastic nature of P&R algorithms. Thankfully, transmit input, receive input/output, lane-to-lane skew, and DRV constraints were generally easy to meet, and could be additional knobs for implementers to meet remaining violating arcs.

(a) DCD in coding redundancy

(b) D_{tx} in shifting redundancy

Figure 5.15: Forwarded clock duty cycle distortion.

5.7 Future Work

This generator is a proof of concept that calls for multiple additional PD optimizations and analyses to be explored and executed. This project would benefit from implementation and more detailed analysis of:

- **NDRs**: Industry-standard flows use NDRs for their clock distribution, which could make a large difference in meeting clock and delay constraints.

- **Additional corners**: The PD flow described in Sec. 5.5 was not set up for the permutations of modes (case analyses) and PVT corners like those in industry-standard flows. In theory, the UCIe-3D™ timing curve should make it just as easy to close timing at even lower-voltage corners than the ones used herein; however, it remains to be seen how technology- and PD tool-agnostic that curve really is.

(a) Coding redundancy

(b) Shifting redundancy (no modules shifted case)
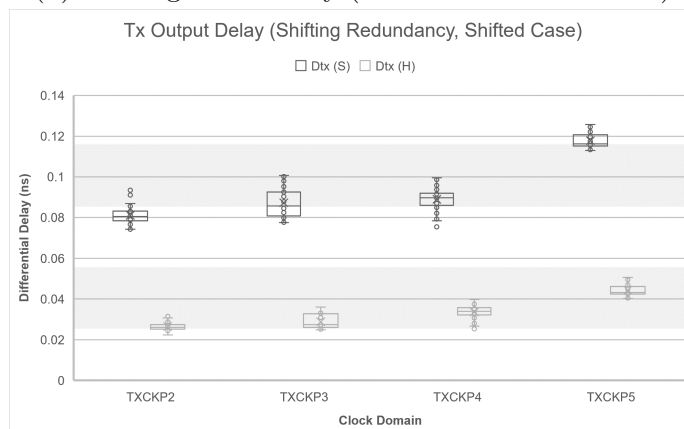
(c) Shifting redundancy (all modules shifted case)

Figure 5.16: D_{tx} measurements. $D_{tx,min} + t_{skew}$ indicated by lower shaded region, $D_{tx,max} + t_{skew}$ indicated by upper shaded region.

- **Power**: Extracted simulations using real bump and TSV models stimulated by generated data patterns for each data statistic would provide a more accurate power estimate. It would help prove whether and where the signal mapping techniques (spiral, sawtooth) and bus inversion are effective in reducing power.

- **SI/PI**: Similar to above, extracted simulations would help validate whether digital construction is sufficient in meeting the random and supply-induced jitter specifications. If not, additional tuning or configurability on the transition time constraints would be required, combined with additional NDR and threshold voltage library selection tuning in P&R.

- **Porosity**: The density and utilization numbers in Table 5.2 are clearly quite low even at 9μm pitch, meaning the placement and routing resources are underutilized. In a more deeply-scaled technology, this underutilization would be even more pronounced. In the current implementation methodology, the entire PHY area is assumed to be blocked off from the rest of the chip, which is a significant routing resource impact. Additional constraints, like channel-level placement and routing blockages, should be generated so that space is leftover for the rest of the chiplet to place and route in, potentially reducing the floorplan blockage of these 3D D2D PHYs, like in the right side of Fig. 4.8. PD tools do not generally support this use case, which requires new constraints such as partitioned SI/PI budgets across the levels of hierarchy and extra work to generate the correct abstract views.

The next step will be to test this generator with a wider range of technology nodes to see how well this approach scales, and also tapeout multiple configurations on a test chip to validate it. To make that happen, some more logic and functionality needs to be implemented and simulated:

- **Architecural Partitioning**: As described in Section 4, pitches above 9μm use μbumps, which often use hexagonal arrangements. This would require a different bump map data structure and other circuit architectural changes. Custom I/O cells are required at higher pitches due to higher output loading and CDM ESD.

- **BIST logic**: An at-rate bit-error rate tester (BERT) needs to be implemented in order to configure the redundancy logic. There will need to be an option of including this inside or outside the main PHY. Since it will be shared amongst modules, it will be challenging to distribute the test patterns across the PHY to work with all the different clock domains, while introducing with minimal overhead. Area and power will also need to be re-analyzed.

- **Coding Redundancy**: In the same vein, the fixed encoder and configurable decoder was not analyzed in this work, so the extra path delay hypothesis of the configurable decoder must be determined. If timing can be closed on the receive side, the removal of sideband signaling would make the BIST logic significantly simpler. In terms of

physical mapping, coding clusters need not be circles; hexagons would result in better packing density but would require more advanced algorithms for hexagonal arrangements.

- **Additional register control protocols**: The ability to add custom protocols (there are many in use) without requiring or triggering Diplomacy [130] would help make generator adoption much more seamless.

- **Verification Collateral**: The generated PHY should be verified against the spec using timing-annotated and post-extraction mixed-signal simulation. Simulation decks containing specific measure statements, for example, would be a feasible generator output.

- **Model Collateral**: At this time, standardization is ongoing of a chiplet model file that describe a PHY instance's characteristics to chiplet system designers. Additional scripts pre- and post-PD flows are needed to assemble this model.

- **Chisel Aspects** [79]: The Chisel Floorplanning framework [129] utilized Chisel Aspects to perform floorplanning in a manner that kept the code generating the logic and floorplan constraints separate. Due to the importance of the bump map data structure and the logic simplicity in this generator, significant re-architecting would be required to adhere to the Aspects design paradigm.

Finally, this generator will be released as open-source software with the goal of integrating it into Chipyard. As for industry adoption, some barriers remain, such as the low adoption of Chisel. However, industry already uses internal scripts that generate Verilog already, and so the techniques used in this generator could be adapted into those tools.

5.8 Conclusion

In this work, a Chisel generator is proposed that demonstrates the ability to generate standards-compliant 3D D2D PHYs in one shot from a set of parameters, a data bundle, and highly-constrained physical design. This generator also serves as an tool for evaluating features such as redundancy and the effects of technology scaling, providing a model of how interconnect standards can evolve to meet the needs of future 3D integrated systems. Finally, the results validate the proposed UCIe-3D™ architecture and performance targets.

Chapter 6

Discussion and Conclusion

In this dissertation, a multi-part exploration of integration paradigms for future sub-THz massive MIMO systems has been presented. Through the process of designing and implementing a pair of baseband ASICs for scalable linear massive MIMO arrays, key insights into the challenges of integrating two-dimensionally scalable arrays at sub-THz frequencies have been revealed, resulting in a drive to study and implement standardized 3D die-to-die interconnects in order to intercept industry roadmaps ranging from 6G to process and packaging technologies.

6.1   System Integration Concept

Given the domain-specific roadmaps in Table 2.2, the work presented herein on standardized 3D D2D interconnects, and closely related work on piezoelectric resonator-based fine-grained power conversion (partially published in [64]), the following figures show a few physical mock-ups for how a 3D-integrated, scalable sub-THz massive MIMO system would look like. In Figs. 6.1a and 6.1b, a 3D mockup is presented with a digital chiplet stacked on top of a pair of mixed-signal/RF chiplets, with a special power delivery structure and package-embedded power management IC (PMIC) placed in the middle. On-package antennas are placed on the opposite face of the package, necessitating a cavity in the carrier PCB. This cavity is made possible with in-package power conversion, which dramatically reduces the required current at the PCB/package interface. Array scalability is achieved by tiling the chiplet stack horizontally, connected by 2.5D D2D interconnects such as UCIe™. Finally, data processed through the chiplet daisy chains are aggregated in a host FPGA for managing the fronthaul/backhaul interface.

However, this configuration is not the only feasible option given current technology developments and continued pressure from $\lambda/2$ scaling. In Fig. 6.2, a side view of the concept above (Concept 1) is presented along with two additional stacking configurations. In Concept 2, the RF chiplet is divorced from the mixed-signal chiplet, in a case driven by $\lambda/2$ area scaling and improved performance of on-chip antennas compared to in-package antennas. In

(a) Top view

(b) Bottom view
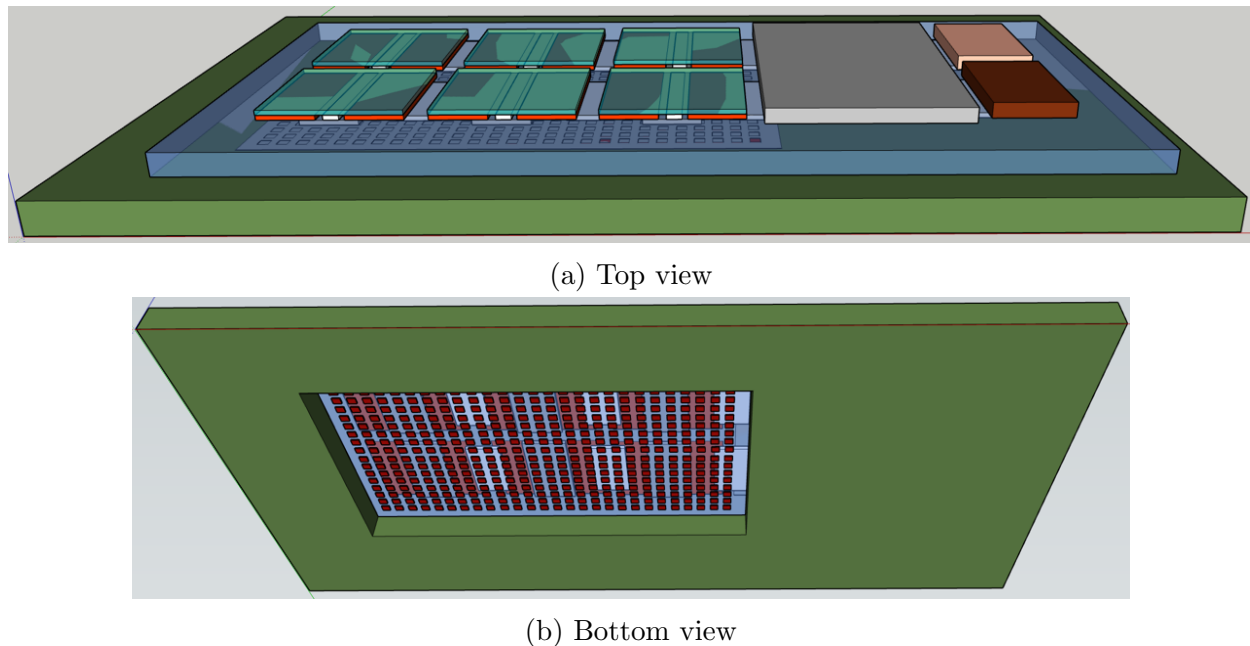
Figure 6.1: 3D mockup of 2-high chiplet stack and on-package antennas

Concept 3, the stacking configuration is inverted, and the PMIC is moved to the bottom side of the package for improved package substrate manufacturability. However, both of these additional concepts require more advanced thermal dissipation technologies. Concept 2 has a long path from the RFIC to any heat sink mounted on the top side of the package, whereas Concept 3 has structures on both faces of the package preventing heat sink attachment. This and remaining challenges are discussed further below in Sec. 6.3.

6.2 Summary of Contributions

This work includes the following contributions:

- The design of an ASIC for baseband processing for 16 antennas and 16 simultaneous users in the TSMC 16FFC process technology (Section 2.2).

- The design, implementation, and testing of an ASIC for baseband processing for 8 antennas and 8 simultaneous users in the Intel 22FFL process technology (Section 2.3).

- A review of the scaling an integration challenges of sub-THz two-dimensional massive MIMO arrays, making the case for 3D integration of domain-specialized chiplets (Section 2.4).
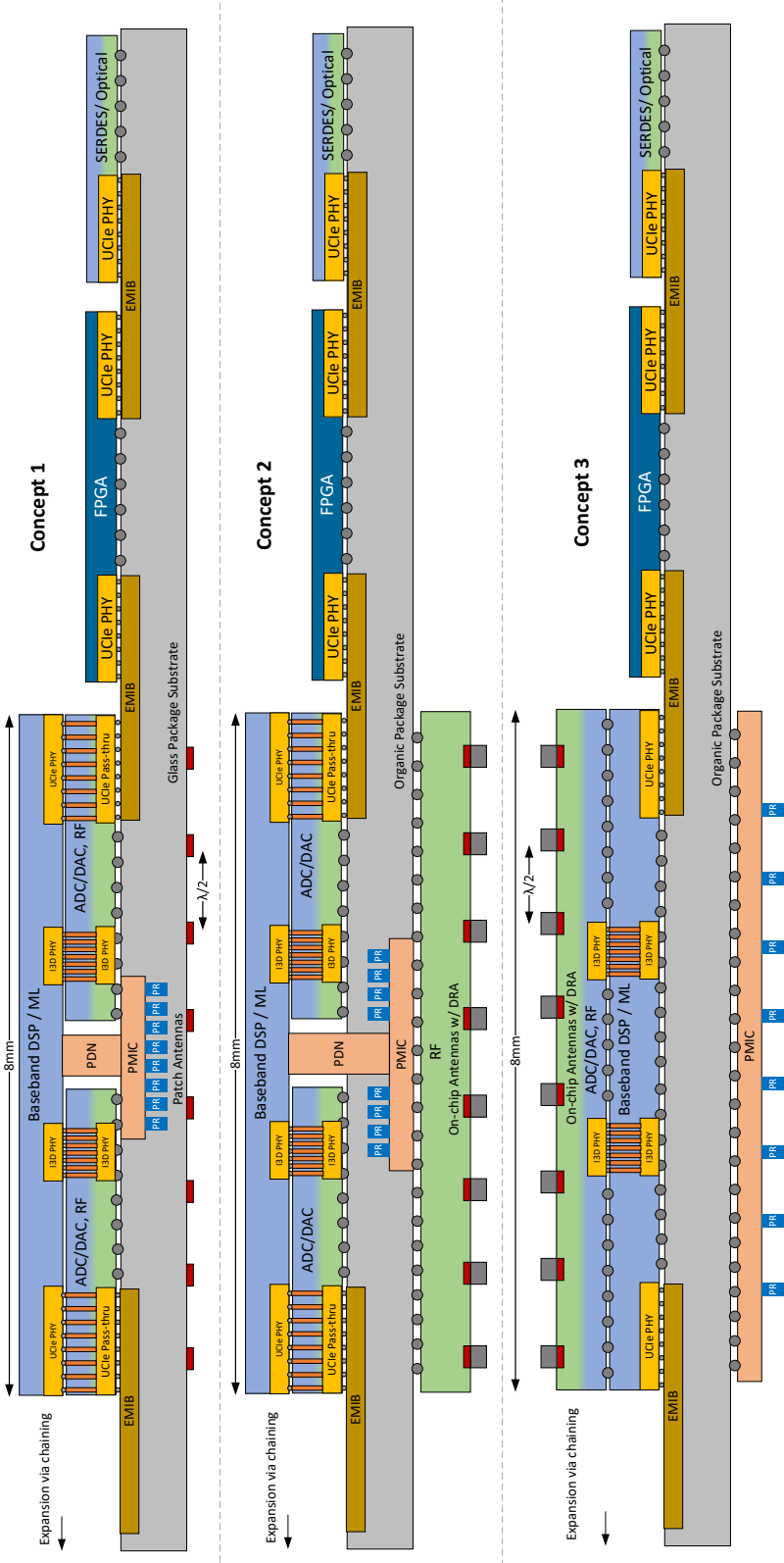
Figure 6.2: Three potential concepts for the final integrated system, demonstrating the diverse range of possibilities for antenna integration, 3D interconnect, power management, and more.

- Improvements to a physical design flow framework, resulting in significantly improved reuse between projects implemented in diverse process technologies, such as between the two Spine ASICs, and workforce development via its integration into multiple courses' teaching material (Chapter 3).

- A comprehensive study of technology constraints for standardizing 3D die-to-die interconnects, resulting in a framework for how the interconnects should be specified given routing, transistor, ESD, and other scaling roadmaps (Chapter 4).

- Informed by the above, a Chisel generator for exploring and implementing standardized 3D die-to-die interconnects, demonstrating both a data structures-driven design methodology and validation of UCIe-3D™ architecture in the Intel 16 process technology (Chapter 5).

6.3   Future Work

6.3.1   System Integration

Referring back to Table 2.2 studying the domain-specific roadmaps in light of the $\lambda/2$ dilemma, it is evident that specific research is not yet on track to meet the requirements of these future systems. Specifically, the following future work is needed:

- **RF front-ends**: While this area is most mature in the context of pushing towards higher frequencies, the plateauing of performance at more mature process nodes is widening the gap between technologies optimal for RF and digital. Heterogeneous integration technologies (bonding, 3D interconnect) must account for this disparity. The work in Chapter 5 is a step in this direction, as it aims to widen compatible process technologies for 3D interconnects, but more work is needed to prove that this is true.

- **Mixed-Signal**: Very high-speed, moderate-resolution ADCs and DACs are rapidly maturing for uses in 224+Gb/s wireline links; however, more attention needs to be paid to reducing circuit area so that a set of two ADCs and two DACs can fit within the area of a single antenna along with with the required 3D interconnect. Not mentioned in Table 2.2 but in this domain is clock distribution–a significant challenge for scalable HI systems. Wave-like distribution methods such as those in [57] are most practical because the modules in the array do not need to be phase synchronous with each other. However, noise and jitter can accumulate, leading to a need for continued research into topologies such as distributed PLLs and/or resonant clocking.

- **Baseband digital**: The ongoing research in this area is promising, but research should focus on more low-latency distributed algorithms for array scaling. At the moment, the algorithms in [60] and [61] are hampered by computational complexity (e.g. matrix

inversion) or latency from iterative or chaining operations, thereby limiting array size for a given OFDM symbol length. AI/ML-based approaches, while undergoing intense research, do not yet address the scaling aspect either.

- **Antenna integration**: Research into antenna-in-package (AiP) [46] and antenna-on-chip (AoC) [131] topologies is highly active, with tradeoffs between performance and cost. Improvements in AoC radiation efficiency and fractional bandwidth are approaching that of AiP; however, some are only possible by overlaying materials such as dielectric resonators and therefore require the RFIC to be exposed to air on the top layer. In the context of the 3D-integrated system, this may not be feasible once thermal and mechanical constraints are considered. Therefore, continued research into substrate materials is needed, with LTCC, glass, and quartz currently all showing promise [46], [47].

- **Power conversion**: In the 3D-integrated system presented above, power delivery must be highly granular, with multiple domains occupying the same $\lambda/2^2$ area. The control scheme proposed in [64] was born from an investigation into whether piezoelectric resonator-based power conversion could be promising for this degree of miniaturization, where the control circuits and capacitors can be integrated onto cutting-edge process nodes. However, it is not yet clear whether PRs can be manufactured and bonded at the wafer-scale, as significant issues surrounding fab contamination and yield are yet to be solved. Therefore, more work is needed in more traditional power conversion topologies such as those in [63], which are thankfully being driven by hyperscale processors.

- **Thermal dissipation**: Stacking multiple domains within an area of less than a square millimeter is a significant challenge for thermal dissipation, which is projected to be approximately 200W/cm^2 in this system (budgeting 100mW for RF, <1W for mixed-signal, <1W for baseband per mm^2). Beyond the current paradigm of utilizing thermal bumps, vias, and TSVs for vertical heat spreading, more research using micro heat pipes/vapor chambers [132] and microfluidics [104] is needed to achieve the required thermal dissipation.

Finally, a key goal of 6G is to enable joint communication and sensing, where the same array can be used for communication, radar, and control [6]. Apart from algorithmic differences between the two applications, it remains to be studied what additional architectural opportunities and challenges this presents.

6.3.2 Design Methodologies

The Hammer framework from Chapter 3 and the 3D interconnect generator from Chapter 5 are still in their infancy relative to what is needed for broader adoption. The following future work is needed to make them more impactful:

- Hammer suffers from usability problems. Purely command-line interfaces and manual post-processing is not conducive to scaling design up to larger chips and teams, where project management from design space exploration all the way to signoff is just as important as Hammer's tenets of reuse and abstraction. To this end, there is ongoing work on overlaying Hammer upon a directed acyclic graph-based (DAG) workflow management tool like Airflow [133] to tackle these issues. Dynamic creation of hierarchical DAGs effectively encodes what the hooks and flow control features of Hammer achieves, without having to manage multiple different ways of specifying them. This infrastructure will also make hierarchical design easier, as the DAG can support both bottom-up (currently available) and top-down (in development) methodologies.

- Though many additional tools have been integrated into Hammer, many significant ones are still needed. For one, support for the Synopsys suite of tools is particularly lacking, as some of these tools have better performance than their Cadence counterparts. Higher-level tools such as high-level synthesis and AI-assisted mixed placers should also be actively benchmarked and integrated to improve design productivity. Additional signoff tools like power intent verification, thermal analysis, and more as well as a checklisting tool for ensuring that all steps have been executed prior to tapeout are needed to produce better-quality designs.

- The 3D interconnect generator is currently tailored to the UCIe-3D™ architecture and Intel 16 process technology. Learnings from experiments with additional process technologies and circuit architectures (e.g., for clocking) should be incorporated to make it more general. Some of the algorithms contained within require further optimizations, and machine learning techniques could potentially be applied as the constraint space becomes more complex.

Bibliography

[1] K. Okada, *Beyond 5g/6g mmwave and terahertz communications technologies*, 2022. [Online]. Available: `https://dl.cdn-anritsu.com/en-en/about-anritsu/r-d/technical/e-30/30-00.pdf`.

[2] "6g the next hyper-connected experience for all," Samsung Research, Tech. Rep., Dec. 2020. [Online]. Available: `https://cdn.codeground.org/nsr/downloads/researchareas/20201201%5C_6G%5C_Vision%5C_web.pdf`.

[3] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: `10.1002/j.1538-7305.1948.tb01338.x`.

[4] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962. DOI: `10.1109/tit.1962.1057683`.

[5] R. W. Chang, "Synthesis of band-limited orthogonal signals for multichannel data transmission," *Bell system technical journal*, vol. 45, no. 10, pp. 1775–1796, 1966. DOI: `10.1002/j.1538-7305.1966.tb02435.x`.

[6] W. Jiang, Q. Zhou, J. He, *et al.*, "Terahertz communications and sensing for 6g and beyond: A comprehensive review," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024. DOI: `10.1109/COMST.2024.3385908`.

[7] *Data centres and data transmission networks*, 2023. [Online]. Available: `https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks`.

[8] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE transactions on wireless communications*, vol. 9, no. 11, pp. 3590–3600, 2010. DOI: `10.1109/twc.2010.092810.091092`.

[9] B. Sadhu, X. Gu, and A. Valdes-Garcia, "The more (antennas), the merrier: A survey of silicon-based mm-wave phased arrays using multi-ic scaling," *IEEE Microwave Magazine*, vol. 20, no. 12, pp. 32–50, 2019. DOI: `10.1109/mmm.2019.2941632`.

[10] J. Howell, "Microstrip antennas," *IEEE Transactions on Antennas and Propagation*, vol. 23, no. 1, pp. 90–93, 1975. DOI: `10.1109/TAP.1975.1141009`.

[11] A. Puglielli, A. Townley, G. LaCaille, *et al.*, "Design of energy-and cost-efficient massive mimo arrays," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 586–606, 2015. DOI: `10.1109/jproc.2015.2492539`.

[12] A. Puglielli, "System architecture and signal processing techniques for massive multiuser antenna arrays," Ph.D. dissertation, University of California Berkeley, 2017.

[13] J. Dunn, B. Nikolic, A. Niknejad, and E. Alon, "An open, scalable massive mimo testbed operating at e-band frequencies," *Master's thesis, EECS Department, University of California, Berkeley*, 2020.

[14] P. Harris, W. B. Hasan, S. Malkowsky, *et al.*, "Serving 22 users in real-time with a 128-antenna massive mimo testbed," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016, pp. 266–272. DOI: `10.1109/SiPS.2016.54`.

[15] C. W. Shepard, R. Doost-Mohammady, R. E. Guerra, and L. Zhong, "Argosv3: An efficient many-antenna platform," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 501–503. DOI: `10.1145/3117811.3119863`.

[16] L. Kuai, J. Chen, Z. H. Jiang, *et al.*, "A n260 band 64 channel millimeter wave full-digital multi-beam array for 5g massive mimo applications," *IEEE Access*, vol. 8, pp. 47640–47653, 2020. DOI: `10.1109/access.2020.2978070`.

[17] M. Chung, L. Liu, A. Johansson, *et al.*, "Lumami28: Real-time millimeter-wave multiuser mimo systems with antenna selection," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 7944–7960, 2023. DOI: `10.1109/twc.2023.3257195`.

[18] R. Lu, C. Weston, D. Weyer, F. Buhler, D. Lambalot, and M. P. Flynn, "A 16-element fully integrated 28-ghz digital rx beamforming receiver," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 5, pp. 1374–1386, 2021. DOI: `10.1109/JSSC.2021.3067504`.

[19] M. Fowler, J. Highsmith, *et al.*, "The agile manifesto," *Software development*, vol. 9, no. 8, pp. 28–35, 2001.

[20] J. Bachrach, H. Vo, B. Richards, *et al.*, "Chisel: Constructing hardware in a scala embedded language," in *DAC Design automation conference 2012*, IEEE, 2012, pp. 1212–1221. DOI: `10.1145/2228360.2228584`.

[21] K. Asanovic, R. Avizienis, J. Bachrach, *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, pp. 6–2, 2016.

[22] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The risc-v instruction set manual, volume i: User-level isa, version 2.0," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54*, p. 4, 2014.

[23] A. Amid, D. Biancolin, A. Gonzalez, *et al.*, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020. DOI: `10.1109/mm.2020.2996616`.

[24] S. Karandikar, H. Mao, D. Kim, *et al.*, "Firesim: Fpga-accelerated cycle-exact scale-out system simulation in the public cloud," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2018, pp. 29–42. DOI: `10.1109/isca.2018.00014`.

[25] H. Liew, D. Grubb, J. Wright, *et al.*, "Hammer: A modular and reusable physical design flow tool," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1335–1338. DOI: `10.1145/3489517.3530672`.

[26] E. Chang, J. Han, W. Bae, *et al.*, "Bag2: A process-portable framework for generator-based ams circuit design," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2018, pp. 1–8. DOI: `10.1109/cicc.2018.8357061`.

[27] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998. DOI: `10.1109/jproc.1998.658762`.

[28] X.-W. Lin, V. Moroz, X. Xu, *et al.*, "Heterogeneous integration enabled by the state-of-the-art 3dic and cmos technologies: Design, cost, and modeling," in *2021 IEEE International Electron Devices Meeting (IEDM)*, 2021, pp. 3.4.1–3.4.4. DOI: `10.1109/IEDM19574.2021.9720707`.

[29] K. Namura, J. M. Kühn, T. Adachi, *et al.*, "Mn-core-a highly efficient and scalable approach to deep learning," in *2021 Symposium on VLSI Circuits*, IEEE, 2021, pp. 1–2. DOI: `10.23919/vlsicircuits52068.2021.9492395`.

[30] U. Rathore, S. S. Nagi, S. Iyer, and D. Marković, "A 16nm 785gmacs/j 784-core digital signal processor array with a multilayer switch box interconnect, assembled as a 2× 2 dielet with 10μm-pitch inter-dielet i/o for runtime multi-program reconfiguration," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 52–54. DOI: `10.1109/isscc42614.2022.9731582`.

[31] N. Nassif, A. O. Munch, C. L. Molnar, *et al.*, "Sapphire rapids: The next-generation intel xeon scalable processor," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 44–46. DOI: `10.1109/isscc42614.2022.9731107`.

[32] O. Geva, C. Berry, R. Sonnelitter, *et al.*, "Ibm telum: A 16-core 5+ ghz dcm," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 46–48. DOI: `10.1109/isscc42614.2022.9731541`.

[33] T. Burd, W. Li, J. Pistole, *et al.*, "Zen3: The amd 2nd-generation 7nm x86-64 microprocessor core," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 1–3. DOI: `10.1109/isscc42614.2022.9731678`.

[34] W. Gomes, S. Khushu, D. B. Ingerly, *et al.*, "8.1 lakefield and mobility compute: A 3d stacked 10nm and 22ffl hybrid processor system in 12× 12mm 2, 1mm package-on-package," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2020, pp. 144–146. DOI: `10.1109/isscc19947.2020.9062957`.

[35] W. Gomes, A. Koker, P. Stover, *et al.*, "Ponte vecchio: A multi-tile 3d stacked processor for exascale computing," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 42–44. DOI: 10.1109/isscc42614.2022. 9731673.

[36] F. Sheikh, R. Nagisetty, T. Karnik, and D. Kehlet, "2.5 d and 3d heterogeneous integration: Emerging applications," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 4, pp. 77–87, 2021. DOI: 10.1109/mssc.2021.3111386.

[37] E. Naviasky, L. Iotti, G. LaCaille, B. Nikolić, E. Alon, and A. Niknejad, "14.1 a 71-to-86ghz packaged 16-element by 16-beam multi-user beamforming integrated receiver in 28nm cmos," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 64, 2021, pp. 218–220. DOI: 10.1109/isscc42613.2021.9365999.

[38] J. Wright, "Design of a lightweight serial link generator for test chips," *Master's thesis, EECS Department, University of California, Berkeley*, 2017.

[39] A. Wang, P. Rigge, A. Izraelevitz, C. Markley, J. Bachrach, and B. Nikolić, "Aced: A hardware library for generating dsp systems," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6. DOI: 10.1109/dac.2018.8465790.

[40] Y. Dai, G. LaCaille, H. Liew, J. Dunn, and B. Nikolić, "A scalable massive mimo uplink baseband processing generator," in *ICC 2021-IEEE International Conference on Communications*, IEEE, 2021, pp. 1–6. DOI: 10.1109/icc42927.2021.9500566.

[41] *Amba 4 axi4-stream protocol*, Revision 1.0, ARM, Mar. 2010.

[42] W. Terpstra, "Tilelink: A free and open source, high performance scalable cache coherent fabric designed for risc-v," in *Proc. 7th RISC-V Workshop*, 2017.

[43] J. Zhao, A. Agrawal, B. Nikolic, and K. Asanović, "Constellation: An open-source soc-capable noc generator," in *2022 15th IEEE/ACM International Workshop on Network on Chip Architectures (NoCArc)*, IEEE, 2022, pp. 1–7. DOI: 10.1109/nocarc57472. 2022.9911299.

[44] B. Sell, B. Bigwood, S. Cha, *et al.*, "22ffl: A high performance and ultra low power finfet technology for mobile and rf applications," in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2017, pp. 29–4. DOI: 10.1109/iedm.2017.8268475.

[45] A. Izraelevitz, J. Koenig, P. Li, *et al.*, "Reusability is firrtl ground: Hardware construction languages, compiler frameworks, and transformations," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2017, pp. 209–216. DOI: 10.1109/iccad.2017.8203780.

[46] X. Gu, D. Liu, and B. Sadhu, "Packaging and antenna integration for silicon-based millimeter-wave phased arrays: 5g and beyond," *IEEE Journal of Microwaves*, vol. 1, no. 1, pp. 123–134, 2021. DOI: 10.1109/JMW.2020.3032891.

[47] T. Chaloun, L. Boccia, E. Arnieri, *et al.*, "Electronically steerable antennas for future heterogeneous communication networks: Review and perspectives," *IEEE Journal of Microwaves*, vol. 2, no. 4, pp. 545–581, 2022. DOI: 10.1109/JMW.2022.3202626.

[48] Z. Guo, A. Mostafa, A. Elshazly, *et al.*, "A 112.5 gb/s adc-dsp-based pam-4 long-reach transceiver with >50db channel loss in 5nm finfet," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 116–118. DOI: 10.1109/isscc42614.2022.9731650.

[49] C. F. Poon, W. Zhang, J. Cho, *et al.*, "A 1.24-pj/b 112-gb/s (870 gb/s/mm) transceiver for in-package links in 7-nm finfet," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 4, pp. 1199–1210, 2022. DOI: 10.1109/jssc.2022.3141802.

[50] S. Li, M.-S. Lin, W.-C. Chen, and C.-C. Tsai, "Interconnect in the era of 3dic," in *2022 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2022, pp. 1–5. DOI: 10.1109/cicc53496.2022.9772820.

[51] S. Wang, T. Liang, A. Alhamed, and G. M. Rebeiz, "A 23–46-ghz fully planar 8×8 multistandard 5g phased array with ofdm 400-mhz 64-qam waveforms at 40–44-dbm eirp," *IEEE Transactions on Microwave Theory and Techniques*, 2024. DOI: 10.1109/tmtt.2024.3397950.

[52] E. Ghaderi and S. Gupta, "A four-element 500-mhz 40-mw 6-bit adc-enabled time-domain spatial signal processor," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1784–1794, 2020. DOI: 10.1109/jssc.2020.3040702.

[53] C.-C. Lin, C. Puglisi, V. Boljanovic, *et al.*, "Multi-mode spatial signal processor with rainbow-like fast beam training and wideband communications using true-time-delay arrays," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 11, pp. 3348–3360, 2022. DOI: 10.1109/jssc.2022.3178798.

[54] S. Lee, S. Hara, T. Yoshida, *et al.*, "An 80-gb/s 300-ghz-band single-chip cmos transceiver," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 12, pp. 3577–3588, 2019. DOI: 10.1109/JSSC.2019.2944855.

[55] Z. Hu, M. Kaynak, and R. Han, "High-power radiation at 1 thz in silicon: A fully scalable array using a multi-functional radiating mesh structure," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 5, pp. 1313–1327, 2018. DOI: 10.1109/jssc.2017.2786682.

[56] X. Chen, X. Yi, M. I. W. Khan, *et al.*, "A 140-ghz fmcw tx/rx-antenna-sharing transceiver with low-inherent-loss duplexing and adaptive self-interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 12, pp. 3631–3645, 2022. DOI: 10.1109/jssc.2022.3202814.

[57] J. Zhang, B. Dai, X. Meng, *et al.*, "24.2 a scalable 134-to-141ghz 16-element cmos 2d $\lambda/2$-spaced phased array," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 67, 2024, pp. 414–416. DOI: 10.1109/isscc49657.2024.10454376.

[58] H. Wang, P. M. Asbeck, and C. Fager, "Millimeter-wave power amplifier integrated circuits for high dynamic range signals," *IEEE Journal of Microwaves*, vol. 1, no. 1, pp. 299–316, 2021. DOI: 10.1109/jmw.2020.3035897.

[59] B. Murmann, "Short course: High speed/high performance data converters: Metrics, architectures, and emerging topics," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 567–568. DOI: 10.1109/ISSCC42614.2022.9731697.

[60] K. Li, J. McNaney, C. Tarver, *et al.*, "Design trade-offs for decentralized baseband processing in massive mu-mimo systems," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2019, pp. 906–912. DOI: 10.1109/ieeeconf44664.2019.9048727.

[61] L. Van der Perre, L. Liu, and E. G. Larsson, "Efficient dsp and circuit architectures for massive mimo: State of the art and future directions," *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4717–4736, 2018. DOI: 10.1109/TSP.2018.2858190.

[62] M. A. Albreem, A. H. Alhabbash, S. Shahabuddin, and M. Juntti, "Deep learning for massive mimo uplink detectors," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 741–766, 2021. DOI: 10.1109/comst.2021.3135542.

[63] R. Jain, S. Xu, R. Kaushal, *et al.*, "28.6 an 87% efficient 2v-input, 200a voltage regulator chiplet enabling vertical power delivery in multi-kw systems-on-package," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 67, 2024, pp. 466–468. DOI: 10.1109/isscc49657.2024.10454349.

[64] M. Touhami, H. Liew, and J. Boles, "Phase-shift voltage regulation of dc-dc converters based on piezoelectric resonators," in *2024 IEEE Workshop on Control and Modeling for Power Electronics (COMPEL)*, IEEE, 2024, forthcoming.

[65] I. Abdo, C. Da Gomez, C. Wang, *et al.*, "A bi-directional 300-ghz-band phased-array transceiver in 65-nm cmos with outphasing transmitting mode and lo emission cancellation," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 8, pp. 2292–2308, 2022. DOI: 10.1109/jssc.2022.3179166.

[66] Y.-J. Mii, "Semiconductor innovations, from device to system," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, IEEE, 2022, pp. 276–281. DOI: 10.1109/vlsitechnologyandcir46769.2022.9830423.

[67] E. Wang, C. Schmidt, A. Izraelevitz, *et al.*, "A methodology for reusable physical design," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, IEEE, 2020, pp. 243–249. DOI: 10.1109/isqed48828.2020.9136999.

[68] T. Ajayi, V. A. Chhabria, M. Fogaça, *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–4. DOI: 10.1145/3316781.3326334.

[69] L. T. Clark, V. Vashishtha, L. Shifren, *et al.*, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016. DOI: `10.1016/j.mejo.2016.04.006`.

[70] R. Goldman, K. Bartleson, T. Wood, V. Melikyan, and E. Babayan, "Synopsys' educational generic memory compiler," in *10th European workshop on microelectronics education (EWME)*, IEEE, 2014, pp. 89–92. DOI: `10.1109/ewme.2014.6877402`.

[71] A. Shafaei, Y. Wang, X. Lin, and M. Pedram, "Fincacti: Architectural analysis and modeling of caches with deeply-scaled finfet devices," in *2014 IEEE Computer Society Annual Symposium on VLSI*, 2014, pp. 290–295. DOI: `10.1109/ISVLSI.2014.94`.

[72] C. Schmidt, J. Wright, Z. Wang, *et al.*, "4.3 an eight-core 1.44 ghz risc-v vector machine in 16nm finfet," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 64, 2021, pp. 58–60.

[73] F. Guo, N. Krzysztofowicz, A. Moreno, *et al.*, "A heterogeneous soc for bluetooth le in 28nm," in *2023 IEEE Hot Chips 35 Symposium (HCS)*, 2023, pp. 1–11. DOI: `10.1109/HCS59251.2023.10254696`.

[74] S. Datta, B. Richards, H. Liew, Y. Kim, D. Sun, and J. M. Rabaey, "Hdbinarycore: A 28nm 2048-bit hyper-dimensional biosignal classifier achieving 25 nj/prediction for emg hand-gesture recognition," in *ESSCIRC 2023- IEEE 49th European Solid State Circuits Conference (ESSCIRC)*, 2023, pp. 229–232. DOI: `10.1109/ESSCIRC59616.2023.10268684`.

[75] R. T. Edwards, M. Shalan, and M. Kassem, "Real silicon using open-source eda," *IEEE Design & Test*, vol. 38, no. 2, pp. 38–44, 2021. DOI: `10.1109/mdat.2021.3050000`.

[76] M. Shalan and T. Edwards, "Building openlane: A 130nm openroad-based tapeout-proven flow," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–6. DOI: `10.1145/3400302.3415735`.

[77] A. Olofsson, W. Ransohoff, and N. Moroze, "A distributed approach to silicon compilation," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1343–1346. DOI: `10.1145/3489517.3530673`.

[78] A. Carsello, J. Thomas, A. Nayak, *et al.*, "Mflowgen: A modular flow generator and ecosystem for community-driven physical design," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1339–1342. DOI: `10.1145/3489517.3530633`.

[79] A. Izraelevitz, *Unlocking Design Reuse with Hardware Compiler Frameworks*. University of California, Berkeley, 2019.

[80] *Advanced interface bus for 3d (aib-3d) specification*, Revision 0.5, Chips Alliance, Sep. 2021.

[81] J. Lee, K. Cho, C. K. Lee, *et al.*, "13.4 a 48gb 16-high 1280gb/s hbm3e dram with all-around power tsv and a 6-phase rdqs scheme for tsv area optimization," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 67, 2024, pp. 238–240. DOI: 10.1109/isscc49657.2024.10454440.

[82] H. Tsugawa, H. Takahashi, R. Nakamura, *et al.*, "Pixel/dram/logic 3-layer stacked cmos image sensor technology," in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2017, pp. 3–2. DOI: 10.1109/iedm.2017.8268317.

[83] J. Wuu, R. Agarwal, M. Ciraula, *et al.*, "3d v-cache: The implementation of a hybrid-bonded 64mb stacked cache for a 7nm x86-64 cpu," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 65, 2022, pp. 428–429. DOI: 10.1109/isscc42614.2022.9731565.

[84] F. Sheikh and P. Fischer, "New directions in heterogeneous integration: Nano-micro- and macro-3d ics," in *Proc. DARPA ERI Summit*, 2020.

[85] D. D. Sharma, G. Pasdast, Z. Qian, and K. Aygun, "Universal chiplet interconnect express (ucie): An open industry standard for innovations with chiplets at package level," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 9, pp. 1423–1431, 2022. DOI: 10.1109/tcpmt.2022.3207195.

[86] *Advanced interface bus (aib) specification*, Revision 2.0.2, Chips Alliance, Jun. 2021.

[87] G. Sisto, R. Chen, R. Chou, *et al.*, "Design and sign-off methodologies for wafer-to-wafer bonded 3d-ics at advanced nodes," in *2021 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, IEEE, 2021, pp. 17–23. DOI: 10.1109/slip52707.2021.00011.

[88] D. Ingerly, S. Amin, L. Aryasomayajula, *et al.*, "Foveros: 3d integration and the use of face-to-face chip stacking for logic devices," in *2019 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2019, pp. 19–6. DOI: 10.1109/iedm19573.2019.8993637.

[89] A. Elsherbini, S. Liff, J. Swan, K. Jun, S. Tiagaraj, and G. Pasdast, "Hybrid bonding interconnect for advanced heterogeneously integrated processors," in *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*, IEEE, 2021, pp. 1014–1019. DOI: 10.1109/ectc32696.2021.00166.

[90] Y. Kagawa, T. Kamibayashi, Y. Yamano, *et al.*, "Development of face-to-face and face-to-back ultra-fine pitch cu-cu hybrid bonding," in *2022 IEEE 72nd Electronic Components and Technology Conference (ECTC)*, IEEE, 2022, pp. 306–311. DOI: 10.1109/ectc51906.2022.00057.

[91] L. Arnaud, S. Moreau, A. Jouve, *et al.*, "Fine pitch 3d interconnections with hybrid bonding technology: From process robustness to reliability," in *2018 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2018, pp. 4D–4. DOI: 10.1109/irps.2018.8353591.

[92] A. Elsherbini, K. Jun, R. Vreeland, *et al.*, "Enabling hybrid bonding on intel process," in *2021 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2021, pp. 34–3. DOI: `10.1109/iedm19574.2021.9720586`.

[93] S. Moreau, J. Jourdon, S. Lhostis, *et al.*, "Hybrid bonding-based interconnects: A status on the last robustness and reliability achievements," *ECS Journal of Solid State Science and Technology*, vol. 11, no. 2, p. 024 001, 2022. DOI: `10.1149/ma2021-0214662mtgabs`.

[94] L. Liebmann, J. Smith, D. Chanemougame, and P. Gutwin, "Cfet design options, challenges, and opportunities for 3d integration," in *2021 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2021, pp. 3–1. DOI: `10.1109/iedm19574.2021.9720577`.

[95] A. Jourdain, M. Stucchi, G. Van der Plas, G. Beyer, and E. Beyne, "Buried power rails and nano-scale tsv: Technology boosters for backside power delivery network and 3d heterogeneous integration," in *2022 IEEE 72nd Electronic Components and Technology Conference (ECTC)*, IEEE, 2022, pp. 1531–1538. DOI: `10.1109/ectc51906.2022.00244`.

[96] T. Lu, C. Serafy, Z. Yang, S. K. Samal, S. K. Lim, and A. Srivastava, "Tsv-based 3-d ics: Design methods and tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1593–1619, 2017. DOI: `10.1109/tcad.2017.2666604`.

[97] R. Farjadrad, M. Kuemerle, and B. Vinnakota, "A bunch-of-wires (bow) interface for interchiplet communication," *IEEE Micro*, vol. 40, no. 1, pp. 15–24, 2019. DOI: `10.1109/mm.2019.2950352`.

[98] B. Sell, S. An, J. Armstrong, *et al.*, "Intel 4 cmos technology featuring advanced finfet transistors optimized for high density and high-performance computing," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2022, pp. 282–283. DOI: `10.1109/VLSITechnologyandCir46769.2022.9830194`.

[99] C.-Y. Lin, R.-H. Liu, and M.-D. Ker, "Esd self-protection design on 2.4-ghz t/r switch for rf application in cmos process," in *2016 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2016, EL–1. DOI: `10.1109/irps.2016.7574602`.

[100] S. Mitra, R. Gauthier, J. Li, *et al.*, "Esd protection using grounded gate, gate non-silicided (gg-gns) esd nfets in 45nm soi technology," in *EOS/ESD 2008-2008 30th Electrical Overstress/Electrostatic Discharge Symposium*, IEEE, 2008, pp. 312–316.

[101] A. Wang, *Practical ESD Protection Design*. John Wiley & Sons, 2021. DOI: `10.1002/9781119850434`.

[102] "White paper 2: A case for lowering component-level cdm esd specifications and requirements part ii: Die-to-die interfaces," Industry Council on ESD Target Levels, Tech. Rep., Aug. 2023.

[103] A. Kosuge and T. Kuroda, "Proximity wireless communication technologies: An overview and design guidelines," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 11, pp. 4317–4330, 2022. DOI: 10.1109/tcsi.2022.3210200.

[104] R. Van Erp, R. Soleimanzadeh, L. Nela, G. Kampitsis, and E. Matioli, "Co-designing electronics with microfluidics for more sustainable cooling," *Nature*, vol. 585, no. 7824, pp. 211–216, 2020. DOI: 10.1038/s41586-020-2666-1.

[105] V. Kolesov, V. Rajan, and R. Nagisetty, "3dic design challenges, early solutions and future recommendations," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2021, pp. 1–4. DOI: 10.1109/cicc51472.2021.9431513.

[106] R. Munoz, "Scaling the chiplet adoption wall," *MEPTEC Rep. Fall*, pp. 14–18, 2021.

[107] H. Liew, F. Sheikh, D. Kehlet, and B. Nikolić, "Silicon process technology constraints for standardized vertical die-to-die interconnects," in *2023 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2023, pp. 1–6. DOI: 10.1109/cicc57935.2023.10121246.

[108] *Universal chiplet interconnect express (ucie) specification*, Revision 2.0, Universal Chiplet Interconnect Express Consortium, Aug. 2024.

[109] L. Bamberg, J. M. Joseph, A. García-Ortiz, and T. Pionteck, *3D Interconnect Architectures for Heterogeneous Technologies: Modeling and Optimization.* Springer, 2022. DOI: 10.1007/978-3-030-98229-4.

[110] T. F. Wu, H. Liu, H. E. Sumbul, *et al.*, "11.2 a 3d integrated prototype system-on-chip for augmented reality applications using face-to-face wafer bonded 7nm logic at $<2\mu$m pitch with up to 40% energy reduction at iso-area footprint," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 67, 2024, pp. 210–212. DOI: 10.1109/isscc49657.2024.10454529.

[111] I. Koren and Z. Koren, "Defect tolerance in vlsi circuits: Techniques and yield analysis," *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1819–1838, 1998. DOI: 10.1109/5.705525.

[112] L. Jiang, Q. Xu, and B. Eklow, "On effective through-silicon via repair for 3-d-stacked ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 559–571, 2013. DOI: 10.1109/tcad.2012.2228742.

[113] Q. Xu, H. Geng, T. Ni, *et al.*, "Fortune: A new fault-tolerance tsv configuration in router-based redundancy structure," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 10, pp. 3182–3187, 2021. DOI: 10.1109/tcad.2021.3133484.

[114] W.-H. Lo, K. Chi, and T. Hwang, "Architecture of ring-based redundant tsv for clustered faults," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 12, pp. 3437–3449, 2016. DOI: 10.1109/tvlsi.2016.2558514.

[115] Q. Wang, Z. Liu, J. Jiang, N. Jing, and W. Sheng, "A new cellular-based redundant tsv structure for clustered faults," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 458–467, 2018. DOI: `10.1109/tvlsi.2018.2876906`.

[116] Q. Xu, W. Sun, S. Chen, Y. Kang, and X. Wen, "Cellular structure-based fault-tolerance tsv configuration in 3d-ic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 5, pp. 1196–1208, 2021. DOI: `10.1109/tcad.2021.3084920`.

[117] I. Lee, M. Cheong, and S. Kang, "Highly reliable redundant tsv architecture for clustered faults," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 237–247, 2018. DOI: `10.1109/tr.2018.2864591`.

[118] M. Cheong, I. Lee, and S. Kang, "A 3-d rotation-based through-silicon via redundancy architecture for clustering faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 9, pp. 1925–1934, 2019. DOI: `10.1109/tcad.2019.2927485`.

[119] T. Ni, Y. Yao, H. Chang, *et al.*, "Lchr-tsv: Novel low cost and highly repairable honeycomb-based tsv redundancy architecture for clustered faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2938–2951, 2019. DOI: `10.1109/tcad.2019.2946243`.

[120] Y. Lee, D. Han, S. Lee, and S. Kang, "A circular-based tsv repair architecture," in *2021 18th International SoC Design Conference (ISOCC)*, IEEE, 2021, pp. 1–2. DOI: `10.1109/isocc53507.2021.9613904`.

[121] S. Park, M. Cheong, D. Han, and S. Kang, "Herringbone-based tsv architecture for clustered fault repair and aging recovery," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 1142–1153, 2021. DOI: `10.1109/tcad.2021.3076141`.

[122] H. Lee, S. H. Shin, Y. Yoo, and S. Kang, "Trust: Through-silicon via repair using switch matrix topology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022. DOI: `10.1109/tcad.2022.3220711`.

[123] L. Chen, R. Kyng, Y. P. Liu, R. Peng, M. P. Gutenberg, and S. Sachdeva, "Maximum flow and minimum-cost flow in almost-linear time," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2022, pp. 612–623. DOI: `10.1109/focs54457.2022.00064`.

[124] D. K. Maity, S. K. Roy, and C. Giri, "Tsv-cluster defect tolerance using tree-based redundancy for yield improvement of 3-d ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 8, pp. 1500–1510, 2020. DOI: `10.1109/tcad.2020.3021341`.

[125] R. P. Reddy, A. Acharyya, and S. Khursheed, "A cost-effective fault tolerance technique for functional tsv in 3-d ics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, pp. 2071–2080, 2017. DOI: 10.1109/tvlsi.2017.2681703.

[126] E. W. Weisstein, *Square Packing – from Wolfram MathWorld — mathworld.wolfram.com*, [Accessed 2023], 2005. [Online]. Available: https://mathworld.wolfram.com/SquarePacking.html.

[127] E. W. Weisstein, *Circle Packing – from Wolfram MathWorld — mathworld.wolfram.com*, [Accessed 2023], 2002. [Online]. Available: https://mathworld.wolfram.com/CirclePacking.html.

[128] D. Gurnell, "Creative scala," 2020.

[129] J. C. Wright, "Physically aware design of generated systems-on-chip," Ph.D. dissertation, University of California, Berkeley, 2021.

[130] H. Cook, W. Terpstra, and Y. Lee, "Diplomatic design patterns: A tilelink case study," in *1st Workshop on Computer Architecture Research with RISC-V*, 2017.

[131] T. Maiwald, T. Li, G.-R. Hotopan, *et al.*, "A review of integrated systems and components for 6g wireless communication in the d-band," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 220–256, 2023. DOI: 10.1109/JPROC.2023.3240127.

[132] S. Lu and K. Vafai, "Thermal performance optimization of the three-dimensional integrated circuits employing the integrated chip-size double-layer or multi-layer microchannels," *ASME Journal of Heat and Mass Transfer*, vol. 145, no. 3, p. 032 501, 2023. DOI: 10.1115/1.4055245.

[133] B. P. Harenslak and J. de Ruiter, *Data pipelines with apache airflow*. Simon and Schuster, 2021.

Appendix A

Hammer

A.1   Path Resolution in TechJSON

The "path" for any library listing can be one of five types:

1. Absolute path: the path starts with "/" and refers to an absolute path on the filesystem
   `/path/to/a/lib/file.lib` → `/path/to/a/lib/file.lib`

2. Tech plugin relative path: the path has no "/"s and refers to a file directly inside the tech plugin folder (no subdirectories allowed, else it conflicts with 3-5 below!)
   `techlib.lib` → `<tech plugin package>/techlib.lib`

3. Tech cache relative path: the path starts with an identifier which is "cache" (this is used in the SKY130 plugin)
   `cache/primitives.v` → `tech-<tech name>-cache/primitives.v`

4. Install relative path: the path starts with an install/tarball identifier (installs.id, tarballs.root.id) and refers to a file relative to that identifier's path
   If `pdkroot` is `/nfs/ecad/tsmc100/stdcells`, then
   `pdkroot/dac/dac.lib` → `/nfs/ecad/tsmc100/stdcells/dac/dac.lib`

5. Library `extra_prefix` path: the path starts with an identifier present in the provided library's `extra_prefixes` field
   `lib1/cap150f.lib` → `/design_files/caps/cap150f.lib`

A.2   Sky130 Multi-Library Conversion

```
 1  {
 2    "name": "Skywater 130nm Library",
 3    "grid_unit": "0.001",
 4    "installs": [
 5      {
 6        "id": "$SKY130_NDA",
 7        "path": "technology.sky130.sky130_nda"
 8      },
 9      {
10        "id": "$SKY130A",
11        "path": "technology.sky130.sky130A"
12      },
13      {
14        "id": "$SKY130_CDS",
15        "path": "technology.sky130.sky130_cds"
16      }
17    ],
18    "libraries": [
19      {
20        "lef_file": "cache/sky130_fd_sc_hd__nom.tlef",
21        "verilog_sim": "cache/primitives.v",
22        "provides": [
23          {
24            "lib_type": "technology"
25          }
26        ]
27      }
28    ]
29  }
```

Listing A.1: Snippet of hard-coded selection of technology LEF in `sky130.tech.json`.
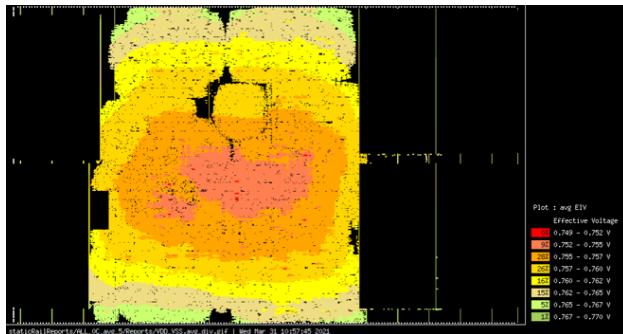
```python
1  def gen_config(self) -> None:
2      """Generate the tech config, based on the library type selected"""
3      slib = self.get_setting("technology.sky130.stdcell_library")
4      SKY130A = self.get_setting("technology.sky130.sky130A")
5      SKY130_CDS = self.get_setting("technology.sky130.sky130_cds")
6      SKY130_CDS_LIB = self.get_setting("technology.sky130.sky130_scl")
7
8      # Select tech LEF and behavioral Verilog
9      if slib == "sky130_fd_sc_hd":
10         libs += [
11             Library(lef_file="$SKY130A/sky130_fd_sc_hd__nom.tlef",
12                     verilog_sim="cache/primitives.v",
13                     provides=[Provide(lib_type="technology")]),
14         ]
15     elif slib == "sky130_scl":
16         libs += [
17             Library(lef_file="$SKY130_SCL/lef/sky130_scl_9T.tlef",
18                     verilog_sim="$SKY130_SCL/verilog/sky130_scl_9T.v",
19                     provides=[Provide(lib_type="technology")]),
20         ]
21     else:
22         raise ValueError(
23             f"Incorrect standard cell library selection: {slib}")
```
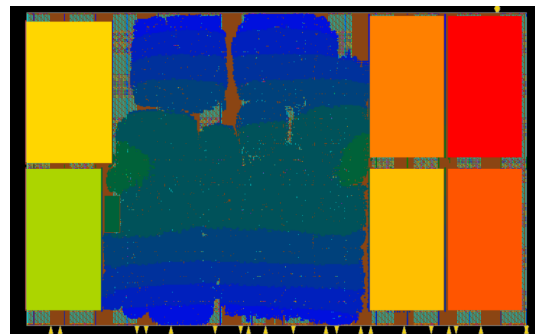
Listing A.2: Snippet of code-based selection of technology LEF in `Sky130Tech` class.

A.3 Tutorials and Labs

- Initial demo video: `https://youtu.be/TiXeocDWdFA`

- Latest tutorial documentation:
  `https://chipyard.readthedocs.io/en/stable/Chipyard-ASAP7-Tutorial.html`

- Latest tutorial presentation slides and recording:
  `https://fires.im/asplos-2023-tutorial/`

(a) Voltus rail analysis

(b) Innovus early rail analysis

Figure A.1: Rail analysis results on the ASAP7 tutorial configuration.

Appendix B

Chisel Generator

B.1   Parameter Tables

Table B.1: Global Technology Parameters

Parameter	R/O	Description
rate	R	The data rate of the interconnect, in GT/s
timingFile	R	The path to the timing header file containing timing constants/curves
pitch	R	The minimum bond pitch, in microns
pitchOvrdH	O	Override of pitch in the horizontal dimension
pitchOvrdV	O	Override of pitch in the vertical dimension
maxNode	R	The largest/oldest technology node (in nm) expected to be used in the 3D stack
ubOrHb	R	The type of bump, either micro-bump or hybrid bond
maxParticleSize	R	The maximum size of a particle to be repaired by coding redundancy, measured in span (diameter) of bond pitches
patternOvrd	O	Overrides the bump pattern of the technology
sigsPerPGOvrdH	O	Overrides the number of signal bumps per power/ground bump in the horizontal dimension
sigsPerPGOvrdV	O	Overrides the number of signal bumps per power/ground bump in the vertical dimension
modSize	R	The maximum number of signals (Tx/Rx, each), per module

Table B.2: Global Design Parameters

Parameter	R/O	Description
redundArch	R	Selects the redundancy architecture
redundRatio	R	Denotes the maximum number of signal bumps per redundant bump
hasDBI	R	Denotes whether data bus inversion logic is included with coding redundancy
deskewArch	R	Selects the clock de-skew architecture
pinSide	R	Denotes which edge of the PHY the core-facing pins are located, before any mirroring/rotation
dataBundle	R	A list of data names and widths in the leader chiplet
dataStatistic	R	The most common data switching statistic of the buses in the dataBundle

Table B.3: Instance Technology Parameters

Parameter	R/O	Description
node	R	The technology node (in nm) of the instance
layerPitch	R	A map of layer name -¿ track pitch, in microns, for layers used for pins and routing
viaKOZRatio	R	The ratio of via stack keep-out zone diameter to bond pitch
bprKOZRatio	O	The ratio of backside power via keep-out zone diameter to bond pitch
tsvKOZRatio	O	The ratio of TSV keep-out zone diameter to bond pitch
powerF	R	Denotes if power/ground is supplied to/from the face of the chiplet
powerB	R	Denotes if power/ground is supplied to/from the back of the chiplet
vNom	R	The nominal voltage of the chiplet, in volts
psmm	R	Typical on-chip routing delay (picoseconds per millimeter model)

Table B.4: Instance Design Parameters

Parameter	R/O	Description
isLeader	R	Denotes if the chiplet is the leader in the stack
faceToStack	R	Denotes if the chiplet's face is facing the rest of the stack
mirrorAxis	R	When chiplet is stacked, about which axis does mirroring occur
pinSide	R	Denotes which edge of the PHY the core-facing pins are located, before any mirroring/rotation, overriding the global parameter
bumpOffset	R	The offset of the bump array from the pin edge, in microns
baseAddress	R	The address offset for the status/control registers (SCRs)
testProtocol	R	Denotes which test protocol to implement
bumpCellName	O	The name of the bump cell to use, may be used by collateral generator
ioCellName	O	The name of the IO buffer cell to use, may be used by collateral generator

B.2   Containers

The following `I3DBump` and `I3DCore` containers are tremendously useful for pattern matching and filtering when processing the bump map data structure during compilation.

Table B.5: `I3DBump` Container Members

Member	R/O	Description
bumpName	R	The port name at the bump
relatedClk	O	The clock domain in which the bump resides
coreSig	O	The core signal container instance
location	R	The bump position (x, y), in microns
modCoord	R	The coordinate of the module this bump is contained in
sdcConstraints	R	A function that returns SDC constraints for this bump class

Table B.6: `I3DCore` Container Members

Member	R/O	Description
name	R	The base signal (port) name, as derived from the flattened data bundle
bitIdx	R	The bit index in the bus, as derived from the flattened data bundle
ioType	R	Encodes the direction of the data, as derived from the flattened data bundle
relatedClk	O	The clock domain the signal is contained within, used for input/output delay constraints
pinLocation	R	The calculated pin location
pinLayer	R	The calculated pin layer
fullName	R	A function that appends the bit index to the name
muxedClk	R	For shifting redundancy, this function returns the muxed clock domain

B.3   Logic Generation Example Listings

```scala
class ModuleBundle(
  val modCoord: I3DCoordinates[Int], coreFacing: Boolean)
  (implicit p: I3DParams) extends Record with AutoCloneType {
  // First, extract the bumps corresponding to this module index
  val modSigs: Seq[I3DBump] = p.flatBumpMap.filter(b => b match {
    case _:Pwr | _:Gnd => false
    case _ => b.modCoord.get == modCoord
  })
  // Filter out redundant bumps in coding redundancy if core-facing
  .filterNot(_.coreSig.isEmpty && coreFacing && !modCoord.isRedundant)
  // elements map is the bump/core signal name -> (cloned) ioType
  // If core-facing, get the coreSig name, else get the bumpName
  val elements: SeqMap[String, Data] = SeqMap(modSigs.map(b => {
    if (modCoord.isRedundant ||  // redundant module, get from bumpName
      (!coreFacing && b.coreSig.isEmpty)) { // redundant coding bump
      val dtype = if (b.bumpName.contains("CK")) Clock() else UInt(1.W)
      b.bumpName -> (
        if (coreFacing ^ b.bumpName.contains("TX")) Output(dtype)
        else Input(dtype))
    } else if (coreFacing) {  // get from coreSig name and flip
      b.coreSig.get.fullName -> Flipped(b.coreSig.get.cloneIoType)
    } else   // bump facing, get from bumpName
      b.bumpName -> b.coreSig.get.cloneIoType
  }):_*)  // magic to convert Seq[Map] to SeqMap
  def apply(elt: String): Data = elements(elt)
}
```

Listing B.1: `ModuleBundle` Generation

```scala
1   // Hook up to the data bundle at the top level
2   // p contains the implicit global parameters
3   def connectDataBundle(dataBundle: Bundle): Unit = {
4     // Connection inwards is straightforward
5     // Convert to Seq of bools and connect each core bit to each data bit
6     val dbIn = dataBundle.getElements.withFilter(
7       DataMirror.directionOf(_) == ActualDirection.Input
8     ).flatMap(_.asUInt.asBools)
9     val eltIn = p.flatTxOrder.flatMap(elements.get(_))
10    (eltIn zip dbIn).foreach{ case (c, p) => c := p }
11    // Connection outwards is more complicated
12    // Need to iterate through each element in the data bundle
13    // Then concatenate the correct number of core bits together to
    ↪  assign
14    val dbOut = dataBundle.getElements.filter(
15      DataMirror.directionOf(_) == ActualDirection.Output)
16    val eltOut = p.flatRxOrder.flatMap(elements.get(_))
17    var i = 0
18    dbOut.foreach{ c =>
19      val w = c.getWidth
20      c := VecInit(eltOut.slice(i, i+w)).asTypeOf(c)
21      i += w
22    }
23  }
```

Listing B.2: `CoreBundle` to `dataBundle` Connection

```scala
class Encoder(val modIdx: I3DCoordinates[Int])
  (implicit p: I3DParams) extends RawModule {
  val core = IO(new ModuleBundle(modIdx, coreFacing = true))
  val bumps = IO(new ModuleBundle(modIdx, coreFacing = false))
  val faulty = IO(Input(Vec(p.sigsPerCluster,
                  UInt(log2Ceil(p.numClusters - 1).W))))
  // Process bits into coding groups
  // Last cluster may have fewer bits (pad it)
  val cBits = core.getElements
    .filterNot(DataMirror.checkTypeEquivalence(_, Clock()))  // no clks
    .map(d => Some(d.asUInt))  // convert to Option
    .padTo(p.numClusters * p.sigsPerCluster, None)  // last cluster
    .grouped(p.sigsPerCluster).toSeq  // group by cluster
    .transpose  // to get bits for each encoder
  val bsBits = bumps.getElements
    .filterNot(DataMirror.checkTypeEquivalence(_, Clock()))  // no clks
    .dropRight(p.sigsPerCluster)  // drop redundant bits
    .map(d => Some(d.asUInt))
    .padTo(p.numClusters * p.sigsPerCluster, None)
    .grouped(p.sigsPerCluster).toSeq
    .transpose
  val brBits = bumps.getElements
    .filterNot(DataMirror.checkTypeEquivalence(_, Clock()))  // no clks
    .takeRight(p.sigsPerCluster)
  // Signal logic
  (cBits zip bsBits).zipWithIndex.foreach { case ((cGrp, bGrp), i) =>
    // Invert is a n-input mux controlled by faulty
    val inv = MuxLookup(faulty(i), 0.U,
      (0 until p.numClusters - 1).map(j =>
        j.U -> cGrp(j).getOrElse(0.U(1.W))
      ))
    // XOR bits with inv
    (bGrp zip cGrp).foreach { case (b, c) =>
      if (b.isDefined) b.get := c.get ^ inv }
    // Redundant bit is invert
    brBits(i) := inv
  }
}
```

Listing B.3: Subset of `Encoder` Generator.