Challenges and Opportunities to Enable Large-Scale Computing via Heterogeneous Chiplets

Invited Paper

Zhuoping Yang*, Shixin Ji*, Xingzhen Chen*, Jinming Zhuang*, Weifeng Zhang†, Dharmesh Jani‡, Peipei Zhou*
* University of Pittsburgh, † Lightelligence, ‡ Meta
Email: {zhuoping.yang, peipei.zhou}@pitt.edu*,
weifeng.zhang@lightelligence.ai† , janidb@meta.com‡

***Abstract*—Fast-evolving artificial intelligence (AI) algorithms such as large language models have been driving the ever-increasing computing demands in today's data centers. Heterogeneous computing with domain-specific architectures (DSAs) brings many opportunities when scaling up and scaling out the computing system. In particular, heterogeneous chiplet architecture is favored to keep scaling up and scaling out the system as well as to reduce the design complexity and the cost stemming from the traditional monolithic chip design. However, how to interconnect computing resources and orchestrate heterogeneous chiplets is the key to success. In this paper, we first discuss the diversity and evolving demands of different AI workloads. We discuss how chiplet brings better cost efficiency and shorter time to market. Then we discuss the challenges in establishing chiplet interface standards, packaging, and security issues. We further discuss the software programming challenges in chiplet systems.**

***Index Terms*—Chiplet, interconnect, advanced packaging, security, programming abstraction, heterogeneous computing, large language model (LLM), generative AI**

I. Introduction

Artificial intelligence (AI) and deep learning (DL) have provided an effective way to address complicated tasks in applications including computer vision, natural language processing, etc. Many hardware accelerators including GPUs, dedicated DL application-specific integrated circuits (ASICs), and FPGAs, are proposed to increase both throughput and energy efficiency. GPUs achieve high throughput by massive parallelism, and they are widely used in deep neural network (DNN) training as they can hugely exploit the parallelism in large batches of training data. From the deep learning inference aspect, for example in real-time application scenarios where low latency is required, ASICs are designed in the seek for better customization. However, designing ASICs is not trivial. Companies would spend three or four years developing their first ASIC from scratch, and two or three years releasing a subsequent ASIC [1], not to mention the huge costs in dollars (millions to billions). FPGAs offer greater programmability compared to ASICs, enabling rapid and economical product updates, albeit with some trade-offs in performance. In fact, to achieve higher energy efficiency for DL tasks while providing flexibility, there is a trend that CPUs, FPGAs, GPUs, and accelerators (Accs) are integrated into the same system-on-chip (SoC). Intel FPGAs Stratix 10 NX FPGA [2] and AMD Versal ACAP architecture [3]–[6] integrate AI tensor cores into the FPGA fabric (FPGA+Acc). Intel CPUs integrate vector units to support SSE or AVX instructions. The latest AMD Ryzen™ AI CPUs also integrate dedicated AI engines in the CPUs (CPU+Acc) [7]. Nvidia Jetson Orin GPUs have integrated tensor cores, multimedia cores, and NVDLA with the CUDA cores (GPU+CPU+Accs).

In the abovementioned heterogeneous systems, integration can be achieved with intellectual properties (IPs) designed by the same company or through licensed IPs from other companies. The emerging chiplet technologies are enabling novel heterogeneous integration across different IP vendors. Traditional chip is implemented on a monolithic silicon die but the die size is approaching the lithographic reticle limits due to growing complicated functionality and slow down of process technology. Chiplets are smaller chips disaggregated from an SoC and optimized for in-package communication [8]. In the vision of the Open Compute Project subgroup Open Domain Specific Architecture (ODSA) [9], chiplets can be easily reused and integrated on an interposer even if chiplets are manufactured by different vendors. However, there remain enormous research questions to be explored. In this paper, we focus on how chiplets can meet new demands in generative AI workload, and discuss the challenges in both hardware and software with potential solutions.

II. Infrastructure Challenges from Diverse and Evolving AI Workloads

In the acceleration of AI tasks, both communication and computation play crucial roles. One way to improve the computation utilization is to overlap the time of communication with the computation. If the communication time is less than or equal to the computation, no hardware resources are idling. However, the communication and computation demands vary in different applications, e.g., GPT [11] and BERT [12]. We use arithmetic intensity [13] to characterize the algorithms' data reuse ratio and computation-to-communication (CTC) ratio [14] to characterize the execution time between computation and communication.

$$\begin{gathered} Arithmetic\ Intensity = \frac{\#Operations}{\#DataMovement} \\ CTC = \frac{Time_{Comp.}}{Time_{Comm.}} = \frac{\#Operations/HW_{ops}}{\#DataMovement/HW_{bw}} \end{gathered} \tag{1}$$

GPT and BERT are both Transformer model variances. As shown in Figure 1 BERT repeats encoder blocks while GPT is built using decoder blocks. In generation tasks, the input sequence is first transformed into tokens. We can mask some tokens and pass the masked tokens to BERT and BERT predicts all masked tokens in only one inference. Different from

TABLE I: Comparisons between chiplet and PCB, monolithic ASIC [10].

Integration Technology	Design Cycle	Cost/$	Integration	Energy Efficiency	Performance
Monolithic ASIC	>1 year	>1,000,000	+++	+	+++
Chiplet	months	1,000-1,000,000	++	++	++
PCB	weeks	100-10,000	+	+++	+

BERT generation, GPT only generates one next token based on previous tokens and the generation process can be further split into two sub-tasks, prompt processing and token generation. In prompt processing, GPT processes all tokens, predicts the next one, and stores intermediate data to avoid recomputation in the token generation processes. This technique is referred to as KV-cache. Then, in the token generation, GPT only processes the last token and generates the next token using KV-cache. Therefore, GPT has a very low arithmetic intensity in the token generation process whereas BERT has a higher arithmetic intensity. For example, when the sequence length is 512, the arithmetic intensity of GPT-2 and BERT-Large are 2 and 266 respectively [13]. Similarly, the variances in arithmetic intensity also exist in different operations within the same model. Layers including convolution and matrix-multiply are more computationally intensive while activations such as Softmax are more demanding in data movement. The variances of arithmetic intensity in different workloads entail different demands in bandwidth and computation resources when designing hardware accelerators.

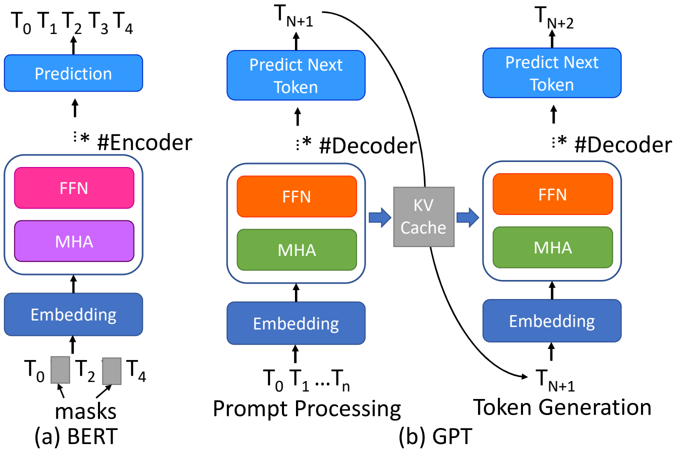

Fig. 1: Different processes in BERT and GPT.

Additionally, the differences between the growth of peak hardware throughput and bandwidth are growing, which means the design complexity also increases as more data reuse and higher arithmetic intensity are needed. There is a trend showing that every two years, the peak hardware throughput increases 3x but the bandwidth only increases 1.6x [15]. Another trend is that the gap between the hardware memory size and AI model size surges. In recent two years, the AI model has increased 410x while hardware memory only increases 2x [15]. These emerging demands are pushing innovations in designing scalable hardware acceleration systems.

III. Chiplets: A Solution for Rapid Heterogeneous System Development

To catch up with the increasing demands, more circuits are integrated into chips. However, simply scaling the monolithic ASIC accelerators is hardly practical. The chiplet technique is becoming a promising solution to improve performance and energy efficiency and decrease the cost and time to market (design cycle column in Table I), which also suits the different requirements of customers.

There are two limitations in scaling up monolithic ASIC chips: the chip area and the yield rate. The die size of CPUs and GPUs grew 14% and 8% respectively every year [16] from 2006 to 2020, which is quickly approaching the lithographic reticle limit. Furthermore, the yield will decrease with the increase in die sizes, which causes more waste, leading to higher costs. One way to keep scaling while minimizing the cost is to fabricate small chips and then integrate them. Traditionally, multiple chips can be connected via printed circuit boards (PCBs). However, PCB systems are difficult to provide high-density inter-chip connections, which is required in many applications. The PCB systems cannot pack more than 400 connections in 1 cm^2 because the PCBs are prone to warpage and the distance between solder bumps (used to connect the PCBs and chips) cannot be less than 0.5 mm [17]. Besides, due to the wire length and parasitic parameters the PCB systems need more power in interconnection and the signal frequency is low [8]. The chiplet technique is proposed to keep the performance scaling and maintain a reasonable cost. This is done by partitioning a monolithic chip into several smaller chiplets and reassembling them into a system-in-package (SiP). This method enables further scaling in the silicon area and decreases the cost. Table I compares chiplet technology, monolithic ASIC, and PCB in various metrics [10].

The chiplet technology has been adopted and shown great improvement in real-world products. For example, the first-generation AMD EPYC CPU processor [18] is based on chiplet architecture, which consists of four chiplets in the 14nm process node. Compared with the monolithic design, the chiplet-based architecture integrates more silicon area in total, which exceeds the reticle limits. Besides, the cost of the chiplet-based architecture is 41% cheaper than a monolithic design [16].

Additionally, chiplet technology allows the integration of chiplets in different process nodes, which provides more choices for different IPs. For example, unlike digital circuits, analog circuits or mixed-signal IPs do not benefit greatly from the advanced technology node, and it is also difficult and time-consuming to apply new technology to them [19]. Therefore, it is necessary to manufacture these IPs in a more mature technology node. Chiplet allows us to integrate these analog or mixed-signal chiplets together with digital circuit chiplets that are fabricated in more advanced technology. Besides, designers can select various chiplets to fulfill specific requirements. For example, GPT-based applications have high demands in memory bandwidth [20]. Therefore, to decrease inference latency, chiplets with more memory channels or higher I/O bandwidth are preferred to be integrated into the SiP.

IV. Hardware Design Challenges

A. Chiplet Interfaces

To achieve heterogeneous integration, further efforts are required for chiplet interconnection design, which includes interconnection protocols standardization, routing algorithms in SiP, and system simulation supports for chiplet system design.

1) Chiplet-based protocols and interfaces: To enable communication between chiplets, chiplets need to follow the same transmission protocol. However, different vendors adopt different protocols, which hinders the heterogeneous integration. Besides, it is difficult to develop a unified protocol that suits various applications as different applications may have different requirements.

Serial interface and parallel interface are two categories of inter chiplet communication interfaces. The serial interface only requires a pair of differential connections to facilitate data transmission in the physical layer, while the parallel interface uses multiple connections [31], [32]. For example, ultra-short-reach (USR) [21], [33] is a serial interface [32] designed for die-to-die interconnect, which aims to ultra-short electrical interconnect and has a low power consumption ($< 0.6pJ/b$) [21]. Compared with serial interface, parallel interface typically has hundreds of connections and thus can achieve the same bandwidth as the serial interface with a much lower connection rate [32]. At present, there are many parallel interface standards, such as Advanced Interface Bus (AIB) [22], Bunch of Wires (BoW) [24], high bandwidth memory (HBM) interconnect [26], Low-voltage-In-Package-INter-CONnect (LINPINCON) [27], Universal Chiplet Interconnect Express (UCIe) [34], Advanced Cost-driven Chiplet (ACC) [30]. Table II shows the differences between the existing chiplet interfaces. These interfaces provide different bandwidth, latency, etc, and have different demands on the package. It is difficult to develop a unified interface that is the best solution for all applications. Heterogeneous integration has another opportunity when integrating off-the-shelf chiplets equipped with different chiplet interfaces. For example, designing a hub chiplet that communicates to two different off-the-shelf chiplets that are equipped with two different die-to-die (D2D) IPs. However, designers have to pay two times D2D IP non-recurring engineering (NRE) cost for the hub chiplet, which brings extra cost and design complexity.

2) Passive and active interposer: Silicon interposer is one of the chiplet packaging technology. The passive interposers only have wires while the active interposes have active components and allow offloading digital logic circuits to the interposer. The active interposer has many advantages. For example, the routers do not have to be placed on the chiplets, which decreases the silicon area of the chiplets. Repeaters can be integrated for long signals to improve the frequency. The active interposer also provides a way to distribute low-jitter and low-skew clocks for all routers [35]. However, there are also design challenges related to the active interposer. First, the network topology may have an impact on the performance due to data contention. [36] proposes a set of chiplet interconnect topologies to balance the data traffic, avoid hot spots, and provide better performance. Second, the multi-chiplet systems are prone to have deadlock issues, even if each chiplet has been verified for the functionality separately [37], [38]. The deadlock typically results from the circular data dependencies among different chiplets.

3) Pre-silicon hardware simulator for chiplet-based architecture: The scale of chiplet-based architecture is much larger than the scale of the monolithic chip. Therefore, we need a more powerful, functional, and multi-chiplet scenario-oriented simulator in the pre-silicon phase.

Though we have several multi-core simulators, we can't use these multi-core simulators for the design of multi-chiplet system [39]. In the design of the chiplet-based system, we also need to accurately model the routing layer between several chiplets, which is also not considered in the multi-core simulation. There are some existing works on multi-chiplet simulations [36], [40], [41]. As the scale of the chiplet-based systems keeps increasing we call for more efficient and comprehensive simulators to provide robust support for chiplet-based design.

B. Package Related Issues

Testing. The chiplet should not be a black box to the system designer. The marketplace also needs standards to describe the chiplets in terms of testing, thermal, I/O, etc [42]. Chiplets potentially enable a marketplace where product developers can buy modules for multiple vendors and assemble them at a much lower cost than designing a chip from scratch. However, product developers still have to design their own interposer and package. A report shows that packaging costs can be comparable with designing a chip due to the low yield of the complex packaging processing and the bounding defects [43]. In current packaging technologies, we cannot detach a bonded die from the substrate without damaging the die [8]. Therefore, comprehensive tests before the chiplets are assembled are desired. These tests on individual chiplets can be provided by the product developers to the packaging vendors and should cover as many user cases as possible to increase the test coverage. If the testing is performed after the packaging process, and the exact chips with errors are identified, we can not do repackaging as the repackaging process will damage all the good chiplets as well as the package. However, this is still an open area for academia, as well as industry [42], [44], [45], and more works to develop the

TABLE II: Comparisons between different chiplet interfaces (data accessed in 2023/11).

Protocol	Institution	Typical Energy Efficiency (pJ/bit)	Maximum Speed (Gbps/wire)	Fault Tolerance Mechanism
USR [21]	\	<0.6 [21]	>20 [21]	N/A
AIB [22], [23]	Intel	0.85 (Gen1)	2 (Gen1) [22], 6.4 (Gen2) [23]	N/A
BoW [24], [25]	ODSA	<0.25-1.0 [25]	32 [24]	N/A
HBM [26]	JESDC	\	6.4	ECC
LINPINCON [27]	TSMC	0.424	2.8	N/A
UCIe [28]	UCIe Union	0.25-1.25 [28]	32 GT/s [28]	CRC + Retransmission
AAC [29], [30]	China Chiplet Industry Alliance	2.5 [29]	128 [30]	CRC + BER + Retransmission

standards which include reducing the required number of pins for testing and standardizing the test interfaces are desired [8].

Thermal. Recent research has shown that the package design cannot guarantee full functionality without considering the chiplets' thermal characteristics [46]–[49]. Dark silicon describes a phenomenon that in a many-core system, not all cores can be active at the same time due to certain thermal constraints. This problem is alleviated in the chiplet-based system. However, we still have to address the thermal issues since chiplets are placed close to each other to minimize the wire length and package size. The thermal issues can be addressed using both online management and offline design optimization. In run-time management, the computational-intensive tasks can be scheduled to non-adjacent chiplets. Besides, the thermal issue can be addressed using dynamic voltage and frequency scaling (DVFS) or run-time scheduling after the chiplet systems are manufactured [50], but these methods sacrifice performance. In offline design optimization, thermal-aware chiplet placement approaches avoid hot spots by increasing distance or inserting low-power chiplets between high-power chiplets [46], [49]. [46] shows it can achieve 20 % peak temperature decrease without sacrificing performance or using expensive cooling technology.

Package and chiplet co-design. The achievable system performance is determined by both the package design and the chiplet design. [47], [48] propose a holistic package design flow that takes chiplets' netlist into consideration. However, this analysis would be hard to perform in the chiplet marketplace since the vendors may be reluctant to share their intellectual properties. Therefore, the vendors and customers need to find standard models to describe the electrical properties of chiplets and enable fast simulation without revealing any detailed circuit designs as discussed in standards [42].

C. Security Issues: Threat Models and Protections

The security of chiplet systems also poses significant challenges for both chiplet vendors and designers. Here, we focus on the security issues from the product designers' perspective. Combining different chiplets from various vendors, chiplet systems are more vulnerable to hardware security threats. On the one hand, chiplet systems expose more vulnerabilities for attacks [51]. The interaction among chiplets may lead to more system side-effects. the design flaws or malicious circuits not only appear in the chiplet but also can be inserted in the integration stage or even in an active interposer [52]. On the other hand, the complicated computing architecture and mix-trust environment will bring more difficulties in developing and deploying the protection methods on the chiplet systems.

1) Potential threats: From the product designers' perspective, side-channel attacks, fault-injection attacks, and hardware trojans are common threats, and these threats can also happen in chiplet systems.

Side-channel attack is the attack that utilizes the unintentional leakage information of the computing systems, which includes memory, timing, power, electromagnetic, etc. For example, Zombie Load [53] successfully extracts sealing keys from the Intel SGX [54] enclave by using leaking memory data of hyperthreading. Recently, some attacks have also been proposed on accelerator systems like GPU systems [55] or in-memory computing (IMC) systems [56]. In chiplet systems, the interconnect channels may become new victims of side-channel attacks. For example, the contention in the interconnect could cause key leakage in cryptographic systems [57].

Fault-injection attacks use malicious circuits to trigger faults in aspects like voltage [58], temperature [59], electromagnetic [60], and so on. These threats could also exist in the chiplet systems, which enables attackers to manipulate the behavior of the chiplet systems in multiple ways.

Hardware trojan is malicious modification or inclusion of additional malicious circuits. The hardware trojan could be inserted into the system in each stage of the supply chain including design, implementation, and integration [52]. Chiplet systems can be prone to hardware trojans since there are more parts and entities, e.g., untrusted vendors in the supply chain.

2) Potential protection methods: The novel and complicated architecture of chiplet systems also brings challenges in adopting protection methods. We summarize existing efforts that could be adopted in chiplet systems below.

Trusted execution environments (TEEs) provide an isolated space, namely an enclave, in which programs can be safely executed even in an untrusted CPU. Initially, TEEs like Intel SGX [54] only protect the basic CPU architecture, while recently more methods have been proposed for heterogeneous architecture with accelerators like GPU [61] and FPGA [62]. The rise of chiplet systems calls for more advanced methods: a TEE may need to provide a safe environment across multiple chiplets with different types. We need a generalized TEE design framework that can be applied to different chiplets, and different D2D interfaces, to minimize the design efforts.

The novel architecture of chiplet systems also brings new methods and opportunities for protection. [63] proposed a new concept of "Root of Trust", which guarantees security by utilizing an active interposer. [64] improves the obfuscation method, which adopts a hybrid split manufacturing methodology and uses different vendors to manufacture the obfuscated circuits. Therefore, in chiplet systems, the chip design can be further spilled and obfuscated, thus providing more security. [52] proposes to integrate additional trusted chiplets such as chiplet-based hardware security modules (CHSM) and chiplet-based security IPs (CSIP). The CHSM is an FPGA chiplet from trusted vendors and features many sensors to detect probing or fault injection attacks. The reconfigurability of CHSM allows remote security policy updates in response to zero-day attacks. CSIP can prevent attackers that have physical access to the system by establishing encrypted communication between chiplets. To achieve security, CSIP must be provided by trusted vendors and has cryptographic modules, physically unclonable functions (PUFs), etc.

V. Software Design Challenges

Programming software on chiplet systems is challenging. In existing host-device systems, CPUs can offload tasks to accelerators such as FPGAs and GPUs and runtime is needed to manage the data movement and execution. To enable the host orchestration, vendor-dependent runtimes are required in the host machine. In a chiplet system, the application will probably be executed based on a number of different runtimes. If a runtime does not cooperate with others and assumes full ownership of the host system, conflicts may occur which might lead to erroneous behaviors. Besides, different accelerators usually need different development environments and design kits. This means the developers have to write code that runs on each chiplet separately. This not only introduces extra design efforts but also reduces the portability of the application.

A unified programming infrastructure seems a promising solution. Though many unified programming methods have been proposed, it is still an open question that which infrastructure fits the chiplet systems best. Here we summarize some existing explorations.

In 2014, SYCL [65] was proposed to achieve heterogeneous device programming for applications. As an open industry standard, SYCL utilizes the C++ programming model. Several implementations of SYCL have been introduced, including Intel oneAPI [66]. oneAPI is designed as a unified development environment for different kinds of accelerators from different vendors. Based on the Level Zero API, which acts as the lowest-level interface, oneAPI provides a whole software stack including system software, developer interface, etc.

Besides SYCL, MLIR [67] is also a desired choice. As a compiler infrastructure, MLIR is adopted in different projects like TensorFlow Graphs and Fortran IR, as well as various domain-specific compilers. Some implementations of MLIR on heterogeneous hardware systems are also been made recently. ScaleHLS [68] uses MLIR in the FPGA high-level synthesis (HLS) to suit the intrinsic hierarchies of HLS design and enables larger design space and more optimization chances. HeteroCL [69] proposed a programming infrastructure, decoupling algorithm specifications with hardware customization, which gives developers an efficient way to explore larger design space and higher performance.

To fully unleash the performance of chiplet systems, there are many more aspects including higher-level software tools, including how to map and partition the workload onto heterogeneous chiplets, how to do mapping and architecture co-design for chiplet systems, etc. H2H [70] proposes a communication-aware mapping algorithm to map heterogeneous models to heterogeneous systems. CHARM [4] proposes a software mapping framework to map heterogeneous kernels within end-to-end deep learning applications and heterogeneous components within an SoC. Such discussions should be considered in chiplet system scenarios and reevaluated.

On top of that, how to perform efficient design space explorations (DSE) in chiplet systems is also much needed. For heterogeneous systems, the techniques needed for fine-grained control like pipeline or parallelism require a deep understanding of the architecture from the software developers. There is a need for tools that can: (1) describe different optimization techniques on different chiplet architectures; (2) automatically find the optimal configurations on these heterogeneous chiplet systems. For example, for FPGA accelerators, AutoDSE [71] develop a DSE framework, aiming to solve the optimization bottleneck, usually application-specific, to enable common software developers to produce high-quality FPGA programs. Similar DSE tools are needed in more complicated systems, e.g., the chiplet system.

VI. Conclusion

With the boom in AI models and fast-increasing demands on performance and energy efficiency, heterogeneous integration based on chiplet becomes a promising way to keep scaling while minimizing the total cost. This paper summarizes the current challenges and calls for innovations and collaborative efforts to address these challenges.

Acknowledgement

We acknowledge the support from the University of Pittsburgh New Faculty Start-up Grant and National Science Foundation awards #2213701, #2217003, #2324864.

References

[1] A. Reuther *et al.*, "Ai and ml accelerator survey and trends." Institute of Electrical and Electronics Engineers Inc., 2022.

[2] Langhammer *et al.*, "Stratix 10 nx architecture," *ACM TRETS*, vol. 15, no. 4, pp. 1–32, 2022.

[3] B. Gaide *et al.*, "Xilinx adaptive compute acceleration platform: Versaltm architecture," in *ACM/SIGDA FPGA*, 2019.

[4] J. Zhuang *et al.*, "CHARM: Composing Heterogeneous AcceleRators for Matrix Multiply on Versal ACAP Architecture," in *ACM/SIGDA FPGA*, 2023.

[5] Z. Yang *et al.*, "AIM: Accelerating Arbitrary-precision Integer Multiplication on Heterogeneous Reconfigurable Computing Platform Versal ACAP," in *ICCAD*, 2023.

[6] J. Zhuang *et al.*, "High Performance, Low Power Matrix Multiply Design on ACAP: from Architecture, Design Challenges and DSE Perspectives," in *2023 60th ACM/IEEE Design Automation Conference DAC*, 2023.

[7] AMD, "Amd ryzen™ ai software platform," https://www.amd.com/en/developer/resources/ryzen-ai-software-platform.html.

[8] M. Hutner *et al.*, "Special session: Test challenges in a chiplet marketplace," in *38th VTS*. IEEE, 2020.

[9] "Open domain-specific architecture ¿¿ open compute project," https://www.opencompute.org/projects/open-domain-specific-architecture.

[10] T. Li *et al.*, "Chiplet heterogeneous integration technology—status and challenges," *Electronics*, vol. 9, no. 4, p. 670, 2020.

[11] A. Radford *et al.*, "Improving language understanding by generative pre-training," 2018.

[12] J. Devlin *et al.*, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[13] "Full stack optimization of transformer inference: a survey," 2 2023. [Online]. Available: http://arxiv.org/abs/2302.14017

[14] C. Zhang *et al.*, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *ACM/SIGDA FPGA*. ACM, 2015.

[15] A. Gholami *et al.*, "Ai and memory wall," *RiseLab Medium Post*, 2021.

[16] G. H. Loh *et al.*, "Understanding chiplets today to anticipate future integration opportunities and limits," in *DATE*, 2021.

[17] P. Gupta *et al.*, "Goodbye, motherboard. bare chiplets bonded to silicon will make computers smaller and more powerful: Hello, silicon-interconnect fabric," *IEEE Spectrum*, vol. 56, no. 10, pp. 28–33, 2019.

[18] S. Naffziger *et al.*, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families: Industrial product," in *ACM/IEEE 48th ISCA*. IEEE, 2021.

[19] R. Munoz, "Furthering moore's law integration benefits in the chiplet era," *IEEE Design & Test*, pp. 1–1, 2023.

[20] H. Peng *et al.*, "Chiplet cloud: Building ai supercomputers for serving large generative language models," *arXiv:2307.02666*, 2023.

[21] A. C. Carusone *et al.*, "Ultra-short-reach interconnects for package-level integration," in *IEEE OI*, 2016.

[22] Intel. (2022) Advanced Interface Bus (AIB) Specification, Revision 2.0.3. https://github.com/chipsalliance/AIB-specification/blob/master/AIB_Specification%202_0.pdf. Update Date:2022/06/17.

[23] David Kehlet, "Accelerating Innovation Through A Standard Chiplet Interface: The Advanced Interface Bus (AIB)," https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/accelerating-innovation-through-aib-whitepaper.pdf, 2022, update Date:2022/06/17.

[24] R. Farjadrad *et al.*, "A bunch-of-wires (bow) interface for interchiplet communication," *IEEE Micro*, vol. 40, no. 1, pp. 15–24, 2020.

[25] "Bunch of Wires (BoW) PHY Specification, DRAFT Version 2.0," https://www.opencompute.org/documents/bow-specification-v2-0d-1-pdf, update Date:2023/03.

[26] JEDEC, "High Bandwidth Memory DRAM (HBM3)," January,2022. [Online]. Available: https://www.jedec.org/document_search?search_api_views_fulltext=JESD238

[27] M.-S. Lin *et al.*, "A 16nm 256-bit wide 89.6gbyte/s total bandwidth in-package interconnect with 0.3v swing and 0.062pj/bit power in info package," in *IEEE HCS*, 2016.

[28] "Universal Chiplet Interconnect Express (UCIe) Specification, Revision 1.1, Version 1.0," https://www.uciexpress.org/specifications, update Date: 2023/11.

[29] "Advanced Cost-driven Chiplet Interface (ACC 1.0)," http://www.iiisct.com/smart/upload/CMS1/202303/ACC1.0.pdf, update Date: 2023/11.

[30] K. Ma, "Introducing acc 1.0: Advanced cost-driven chiplet interface standard," in *The 3rd HiPChips Conference at ISCA*, 2023. [Online]. Available: https://hipchips.github.io/isca2023/

[31] S. Ardalan *et al.*, "An open inter-chiplet communication link: Bunch of wires (bow)," *IEEE Micro*, vol. 41, no. 1, pp. 54–60, 2020.

[32] X. Ma *et al.*, "Survey on chiplets: interface, interconnect and integration methodology," *CCF THPC*, 2022.

[33] B. Dehlaghi *et al.*, "Ultra-short-reach interconnects for die-to-die links: Global bandwidth demands in microcosm," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 2, pp. 42–53, 2019.

[34] D. Das Sharma *et al.*, "Universal chiplet interconnect express (ucie): An open industry standard for innovations with chiplets at package level," *IEEE CPMT*, 2022.

[35] D. Stow *et al.*, "Investigation of cost-optimal network-on-chip for passive and active interposer systems," in *2019 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*. IEEE.

[36] S. Bharadwaj *et al.*, "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *DAC*, 2020.

[37] E. Taheri *et al.*, "Deft: A deadlock-free and fault-tolerant routing algorithm for 2.5 d chiplet networks," in *DATE*. IEEE, 2022.

[38] J. Yin *et al.*, "Modular routing design for chiplet-based systems," in *45th ISCA*. IEEE, 2018.

[39] C. Chen *et al.*, "Design challenges of intrachiplet and interchiplet interconnection," *IEEE Design & Test*, 2022.

[40] H. Zhi *et al.*, "A methodology for simulating multi-chiplet systems using open-source simulators," in *NANOCOM*, 2021.

[41] G. Krishnan *et al.*, "Siam: Chiplet-based scalable in-memory acceleration with mesh for deep neural networks," *ACM TECS*, 2021.

[42] A. Mastroianni *et al.*, "Proposed standardization of heterogenous integrated chiplet models." IEEE, 2021.

[43] Y. Feng *et al.*, "Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration," in *DAC*, 2022.

[44] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," *IEEE Std 1687-2014*, pp. 1–283, 2014.

[45] "IEEE Standard for Test Access Architecture for Three-Dimensional Stacked Integrated Circuits," *IEEE Std 1838-2019*, 2020.

[46] Y. Ma *et al.*, "Tap-2.5d: A thermally-aware chiplet placement methodology for 2.5d systems," *DATE*, 2021.

[47] M. D. Kabir *et al.*, "Holistic 2.5d chiplet design flow: A 65nm shared-block microcontroller case study," *IEEE SOCC 2020*.

[48] A. Kabir *et al.*, "Coupling extraction and optimization for heterogeneous 2.5d chiplet-package co-design," 2020.

[49] F. Eris *et al.*, "Leveraging thermally-aware chiplet organization in 2.5 d systems to reclaim dark silicon," in *DATE*. IEEE, 2018.

[50] X. Li *et al.*, "Power management for chiplet-based multicore systems using deep reinforcement learning." IEEE Computer Society, 2022.

[51] W. Hu *et al.*, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *TCAD*, 2020.

[52] M. S. U. I. Sami *et al.*, "Enabling security of heterogeneous integration: From supply chain to in-field operations," *IEEE Design & Test*, 2023.

[53] M. Schwarz *et al.*, "Zombieload: Cross-privilege-boundary data sampling," in *CCS*, 2019.

[54] F. McKeen *et al.*, "Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave," in *HASP*, 2016.

[55] C. Luo *et al.*, "Side-channel timing attack of rsa on a gpu," *TACO*, vol. 16, no. 3, pp. 1–18, 2019.

[56] Z. Wang *et al.*, "Side-channel attack analysis on in-memory computing architectures," *IEEE TETC*, 2023.

[57] M. Dai *et al.*, "Don't mesh around:{Side-Channel} attacks and mitigations on mesh interconnects," in *USENIX Security*, 2022, pp. 2857–2874.

[58] J. Krautter *et al.*, "Fpgahammer: Remote voltage fault attacks on shared fpgas, suitable for dfa on aes," *IACR CHES*, pp. 44–68, 2018.

[59] M. M. Alam *et al.*, "Ram-jam: Remote temperature and voltage fault attack on fpgas using memory collisions," in *2019 FDTC*. IEEE, 2019.

[60] M. A. Elmohr *et al.*, "Em fault injection on arm and risc-v," in *2020 ISQED*. IEEE, 2020, pp. 206–212.

[61] S. Volos *et al.*, "Graviton: Trusted execution environments on {GPUs}," in *OSDI*, 2018, pp. 681–696.

[62] M. Zhao *et al.*, "Shef: Shielded enclaves for cloud fpgas," in *ASPLOS*, 2022, pp. 1070–1085.

[63] M. Nabeel *et al.*, "2.5 d root of trust: Secure system-level integration of untrusted chiplets," *IEEE TC*, vol. 69, no. 11, pp. 1611–1625, 2020.

[64] Y. Safari *et al.*, "Hybrid obfuscation of chiplet-based systems," in *DAC*. IEEE, 2023, pp. 1–6.

[65] "SYCL Overview," https://www.khronos.org/sycl/.

[66] R. W. Wisniewski *et al.*, "A holistic systems approach to leveraging heterogeneity," in *2021 PEHC*. IEEE, 2021, pp. 27–33.

[67] C. Lattner *et al.*, "MLIR: Scaling compiler infrastructure for domain specific computation," in *2021 IEEE/ACM CGO*. IEEE, 2021.

[68] H. Ye *et al.*, "Scalehls: A new scalable high-level synthesis framework on multi-level intermediate representation," *HPCA*, 2022.

[69] Y.-H. Lai *et al.*, "Heterocl: A multi-paradigm programming infrastructure for software-defined reconfigurable computing," in *FPGA*, 2019.

[70] X. Zhang *et al.*, "H2h: Heterogeneous model to heterogeneous system mapping with computation and communication awareness," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 601–606.

[71] A. Sohrabizadeh *et al.*, "Autodse: Enabling software programmers to design efficient fpga accelerators," *TODAES*, 2022.