Design Challenges of Intra- and Inter- Chiplet Interconnection

Special Session Paper

Chixiao Chen, *Fudan University*, Jieming Yin, *Lehigh University*, Yarui Peng, *University of Arkansas*, Maurizio Palesi, *University of Catania*, Wenxu Cao and Letian Huang, *University of Electronic Science and Technology of China*, Amit Kumar Singh, *Univeristy of Essex*, Haocong Zhi, *South China University of Technology*, and Xiaohang Wang, *Zhejiang University*

***Abstract*—In a chiplet-based many-core system, intra- and inter- chiplet interconnection is key to system performance and power consumption. There are a few challenges in intra- and inter- chiplet interconnection network: 1) Fast and accurate simulation is necessary to analyze the performance metrics. 2) Efficient network architecture for inter- and intra- chiplet is necessary, including topology, PHY design and deadlock free routing algorithms, *etc*. 3) Deep learning based AI systems are demanding more computation power, which calls for the need of efficient and low power chiplet-based systems. This paper proposes network designs to address these challenges and provides future research directions.**

***Index Terms*—Chiplet, inter- and intra-chiplet interconnection.**

I. Introduction

TO address the so called area wall problem, multi-chiplet based systems become a promising design paradigm in the post-Moore era. Companies like Intel, AMD, Apple, *etc*. design or fabricate state-of-the-art CPU or GPU chips using the chiplet integration technology. Inter- and intra- chiplet interconnection network is key to chiplet-based many-core systems. There are a few design challenges in optimizing inter- and intra- chiplet interconnection networks as follows.

- Chiplet-based many-core systems might integrate a large number of chiplets or cores. For example, the Celebras system-on-wafer chip system has 200,000 AI cores. Simulators are needed to accurately simulate large-scale chiplet-based many-core system with fast speed and high accuracy.
- Inter-chiplet interconnections have lower bandwidth and much higher latency compared to their intra-chiplet counterparts, due to pin limit, additional processing overhead of PHY, and longer wires in interposer/RDL. Therefore, inter- and intra- chiplet interconnection networks should be carefully designed to provide high efficient inter-chiplet communication.
- Chiplet-based many-core systems are designed to meet the ever-growing computation demand from various applications, like AI and high performance computing, *etc*.

To address the above challenges, in this paper, the chiplet-based many-core system simulator design is detailed in Section II, followed by inter- and intra-chiplet PHY, network, and routing algorithm designs in Sections III to V. Cross-layer design is introduced in Section VI and a chiplet-based DNN accelerator design is detailed in Section VII.

II. Simulation Methodology for Chiplet-based Many-core System

Simulators, especially those cycle accurate ones, are needed for early stage design space exploration for chiplet-based systems. However, current multi-core simulators can not be used directly for multi-chiplet system simulation due to lack of accurate interconnection modelling for inter-chiplet communication and incapability of large-scale parallel simulation. Therefore, we propose a methodology of simulating multi-chiplet systems by integrating and modifying open-source simulators. This methodology supports parallel simulation for large-scale systems with accurate modeling of inter- and intra-chiplet interconnection, and has both distributed and shared memory model for multi-chiplet systems [8] (available for free download in https://github.com/FCAS-SCUT/).

The multi-chiplet simulation system consists of single-chiplet simulators and an inter-simulator-process communication and synchronization protocol. The existing simulators (*e.g.*, gem5, sniper, *etc.*) simulate individual chiplets and run in parallel, acting as the single-chiplet simulators of the simulation system. The inter-simulator-process communication and synchronization protocol is proposed to simulate the inter-chiplet communication. The multi-chiplet system has distributed, shared or hybrid (*e.g.* globally distributed but a few chiplets share memory address) memory models.

Following the layers in the multi-chiplet system design as in Fig. 1, the proposed simulator framework is comprised by the following layers. In the circuit and physical layer, the model of

Xiaohang Wang is the corresponding author.

This work was supported in part by the National Natural Science Foundation of China under Grant 61971200, in part by Zhejiang Lab under Grants 2021LE0AB01 and 2021PC0AC01, in part by the Open Research Grant of State Key Laboratory of Computer Architecture Institute of Computing Technology, Chinese Academy of Sciences under Grant CARCH201916, in part by Major Scientific Research Project of Zhejiang Lab under Grant 2021LE0AC01, in part by the Key Technologies R&D Program of Jiangsu (Prospective and Key Technologies for Industry) under Grant BE2021003, and in part by the National Key Research and Development Program of China under Grant 2019QY0705.

Manuscript received 1 Jul, 2022; revised 1 Aug, 2022; accepted 24 Aug, 2022. This article was presented at the NoCS 2022 and appears as part of the Design & Test special issue.

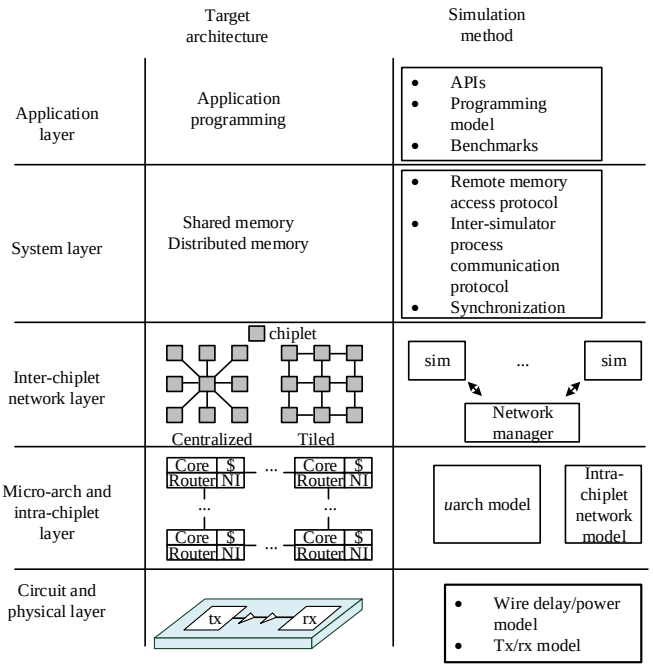


Fig. 1. The overview of the simulation framework.

latency and power are from [2]. In the micro-architectural and intra-chiplet layer, open-source simulators are used to simulate the pipelines of the routers or the cores. Each individual chiplet is simulated by an existing open-source simulator. In the inter-chiplet network layer, a centralized network manager can configure different inter-chiplet network topologies according to configuration files.

In the system layer, both distributed and shared memory models are simulated with the timing model and functional model files. The functional model files carry data packets and the timing model files accumulate latency of packets. The memory addresses are either private or shared among the chiplets, which are distinguished by address tables. In the application layer, APIs (Application Programming Interface) are provided for the programmer (benchmark developer) for remote communication. Timing and functional model files are generated by `m5opt` in full system (FS) mode simulators like gem5 or by system call handlers in syscall emulation (SE) mode simulators.

A. Path Forward

With chiplet integration technology, more cores/memory units can be integrated. For the system-on-wafer chip, there can be millions of cores/memory units. Designing a fast and accurate simulator for million-scale system becomes a must.

III. Digital Die-to-Die Physical Layer Design

As 2.5D chiplet technology develops, the inter-Chiplet data communication was getting more concern. Traditional Ser-Des high-speed links, which are normally adopted for inter-chip data transmission through PCB wirelines, can achieve up to 112 Gbps [4] with only two differential pairs. However, they consume huge costs of power, area, and delay thanks to the complex signal processing blocks, not necessary for chiplet scenarios. Moreover, such high-speed links' physical layer (PHY) contains analog equalizers, comparators, and even Giga-hertz-sampling-rate analog-to-digital converters (ADCs), making it difficult to port between different fabrication technology. Tedious analog redesign efforts are also required. In this section, we present an all digital PHY design method for die-to-die communication in chiplet technology. Compared with traditional Ser-Des, it features simple circuit topology, low power consumption, and good portability.

A. All Digital Die-to-Die PHY Implementation

Fig. 2 shows the overall system of a digital PHY, including a pair of a transmitter (TX) and a receiver (RX). TX converts the parallel data flow from the processor side core into a quadruple data rate serial data stream with a dedicated designed parallel-to-serial module. Rather than PLL or multi-phase DLL, the PHY's clock is generated by a frequency doubler using digital-controlled delay-lines (DCDL). As a result, the data rate will be four times of input data signal due to doubling clock and DDR.

Multiple tri-state gates from the standard cell library are used for TX drivers. To configure various driving strengths, each TX driver contains 16 parallel tri-state gates. To verify the effectiveness, we extract *s*-parameters of three different channels. As the results show, 14 tri-state gates are required to drive a 4.70-mm channel if the eye-diagram width up to 0.5UI (unit interval), while only 7 and 8 tristate gates are needed to achieve similar performance on the 1.33-mm and 2.34-mm channels.

The PHY's RX features a termination-resistor-less design. Thanks to the low loss channel characteristic, we eliminate the termination resistor and use a standard inverter cell as the front-end comparator in RX.

All components of the proposed PHY are from standard cell library, indicating that it can be implemented by the standard digital placement and routing flow. In practice, the commercial EDA tools can accelerate the development process of this PHY and can be easily ported among different technology. Simulation results shows that the entire PHY consumes 13.03mW under 6.4Gbps data rate, achieving the power efficiency of 0.41pJ/bit.

B. Path Forward

Though many designs inherit traditional high-speed analog Ser-Des paradigms, we still have a good vision for the future. Physical interconnect standard is a key knob enabling versatile multi-chiplet systems. Given that dies designed by different vendors are combined into a *integrated-chip-system*, all interfaces have to obey the same rule. Under the trend, recent standards such as BoW and UCIe are attracting more attention.

IV. In-Package Network Design

When designing chiplet-based systems, ensuring routing correctness can be challenging. Specifically, integrating individually designed chiplets into the same package might

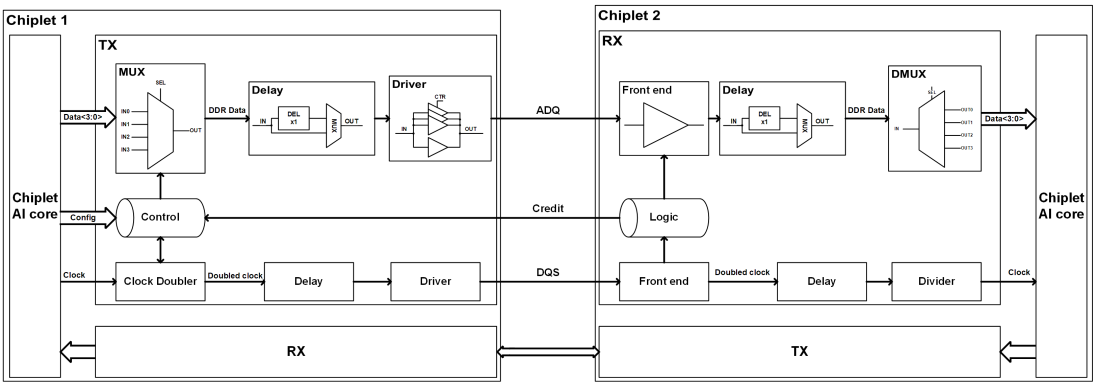


Fig. 2. The overall Die-to-Die PHY architecture.

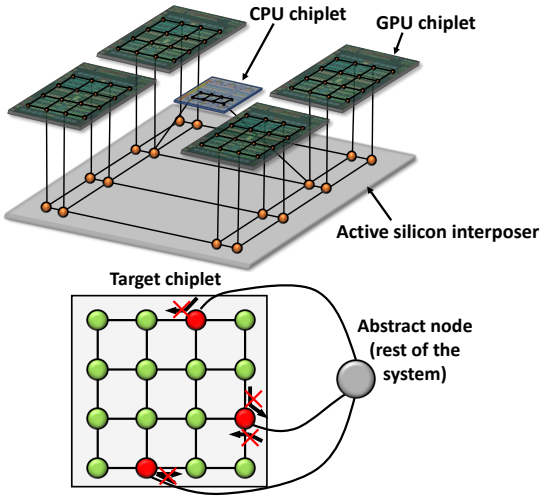


Fig. 3. Baseline chiplet-based system (top) and the proposed MTR methodology (bottom).

cause the final system to be deadlocked, even if each chiplet is deadlock-free. In this section, we present Modular Turn Restriction (MTR), a composable routing methodology that enables modular design and integration of heterogeneous systems. Our methodology imposes turn restrictions applied only to traffic as it flows into or out of the chiplets from the interposer. Using MTR, each individual chiplet as well as the interposer is free to implement its own NoC topology and local routing algorithm.

A. Routing Design Challenge for Chiplet-based Systems

In multi-chip SoCs, chiplets can be independently designed by different vendors. As chiplets may be deployed in multiple products, including future products not even defined at chiplet-design time, their global SoC routing information may not be available. Figure 3 (top) shows a multi-chiplet system, consisting of four GPU chiplets and a CPU chiplet. Each of the GPU and CPU chiplet contains a local NoC. These five chiplets are stacked on an active interposer that implements its own NoC to interconnect the chiplets and other common system functionality. Designing the in-package network for such a system is challenging, because while each individual chiplet's and interposer's NoC may be deadlock-free, they can still be connected together in a manner that introduces deadlocks in the final SoC (channel dependence loops that involve multiple chiplets can be formed easily). Most existing deadlock-free routing algorithms assume that complete system-level information is available, which does not necessarily hold in chiplet-based systems. Therefore, these approaches are not amenable to routing for modular, independently-designed chiplets that may be reused in multiple SoC designs.

B. Modular Turn Restriction Methodology

MTR [7] leverages a simple-yet-powerful insight: from an individual chiplet's perspective, the rest of the system can be abstracted away into a single node. Turn restrictions are carefully applied to only the boundary routers that connect the chiplet to the abstract node, leading to tractable analysis and optimization at the granularity of individual chiplets. MTR consists of three important steps as follows.

Step 1: Select boundary routers for the target chiplet. A boundary router connects the chiplet to the interposer. Chiplet designers need to decide the number of boundary routers and their placement. The number of boundary routers determines the throughput a chiplet can sustain for sending/receiving off-chiplet traffic. Given an internal chiplet-level routing algorithm, the placement of boundary routers affects their inbound (from interposer to the chiplet) and outbound (from chiplet to interposer) reachability and the on-chip traffic distribution.

Step 2: Apply turn restrictions on boundary routers. Once the boundary routers are determined, we can abstract away rest of the system into a single node, as shown in Figure 3 (bottom). We use turn restrictions to break cycles containing the abstract node and a pair of boundary routers. The abstract node represents the rest of the system that designers of individual chiplets do not need to have knowledge of, hence turn restrictions do not apply to the abstract node. When choosing prohibited turns for boundary routers, connectivity must be preserved, so turn restrictions that cause a disconnected NoC are prohibited.

Step 3: Configure the interposer NoC. Packets are routed from one boundary router to another through the interposer. The system integrator need to program the interposer's routing tables properly by taking into account the turn restrictions of all chiplets. To do that, certain chiplet-level information must be provided to the interposer. First, the system integrator need to know the on-chip nodes (endpoints) that are reachable from

each individual boundary router given the turn restrictions. Second, we optionally use the topological distances between each boundary router and its reachable on-chip nodes to optimize routing distances and load balancing.

Following the above steps, chiplet designers have the freedom to optimize their local NoC topology and routing algorithm, while the resulting system is guaranteed to be deadlock free. In terms of microarchitectural design, each chiplet needs to implement two different routing tables. The first handles intra-chiplet traffic that never goes to the interposer. The second routing table directs outbound traffic to the appropriate boundary router.

C. Path Forward

Future chiplet-based systems can have a mix of 2.5D and 3D integration (some chiplets are integrated in a 2.5D manner while some are 3D stacked). Finding an optimal placement and designing/optimizing in-package network topologies can be an important step during system integration.

V. Deadlock-free Design: Model and Algorithms

In this section, we propose to use the tree model to run the turn restriction algorithm (TRA) in the aforementioned modular turn restriction (MTR) methodology, and propose an improved method Presort-TRA to accelerate TRA. The Presort-TRA is proved to reduce the number of iterations of TRA by up to 50%.

A. The Tree Model for TRA

Suppose the NoC on the target chiplet generates N different candidate boundary turns with restrictions. TRA can be depicted as a tree called recursive combinatorial tree (RCT), composed by all candidate boundary turns labeled 1 to N in random order. Fig. 4(a) shows an example of generating the boundary turns given an inter-chiplet network, where the target chiplet consists of 3 boundary routers labeled $R1$ to $R3$, corresponding to 6 boundary turns labeled as ① to ⑥. The corresponding RCT of the system is shown in Fig. 4(b).

In the RCT, a node with label k is the boundary turn has $N-k$ child nodes labeled from $k+1$ to N. When executing TRA, a depth first search (DFS) algorithm is applied on the tree as shown in Fig. 4(b). The sibling nodes are visited in a random order in TRA. Fig. 4(b) shows an example of TRA. Once a node in a higher level is visited, *e.g.* from level 1 to level 2, the boundary turn with the current node's label is restricted. When the searched returns from that higher level, the restricted node is released. Therefore, once a new node in the RCT is visited, a new turn restriction pattern is evaluated. Thus, each node except the root node in the RCT corresponds to a distinct combination of restricted boundary turns, which is represented by the node itself along with all of its non-root parent nodes. For example, in Fig. 4(b), node m and its parent nodes n and j have the turn restriction combination of $\{2, 3, 6\}$, and node k and its parent node j have the combination of $\{2, 5\}$. In addition, MTR requires limited the number of restricted boundary turns. In Fig. 4, the maximum number of the restrictions is set to be 3. Thus, there are 3 non-zero levels in the RCT. The objective function is defined as $\Phi = \frac{AverageDistance}{AverageReachability}$ [7] in the search algorithm. TRA searches through all boundary turns to minimize Φ.

B. Presort-TRA

The efficiency of TRA can be improved by choosing the orders to label the boundary turns and to visit the sibling nodes of RCT. Therefore, the Presort-TRA algorithm is proposed to accelerate TRA by selecting the labeling order of boundary turns and the searching order of the sibling nodes. The Presort-TRA has 2 steps:

1) Presorting. All of the N candidate boundary turns are labeled from 1 to N in a descending order by their Φ values.
2) Searching. The RCT of each presorted boundary turns is formed. The DFS algorithm is performed on the RCT to search the optimal combination of restricted boundary turns with minimal Φ. Presort-TRA visits the sibling nodes in the RCT by following the descending order of their labels.

An example of how Presort-TRA works is shown in Fig. 4(b).

VI. Cross-Boundary Chiplet-Package Co-Design

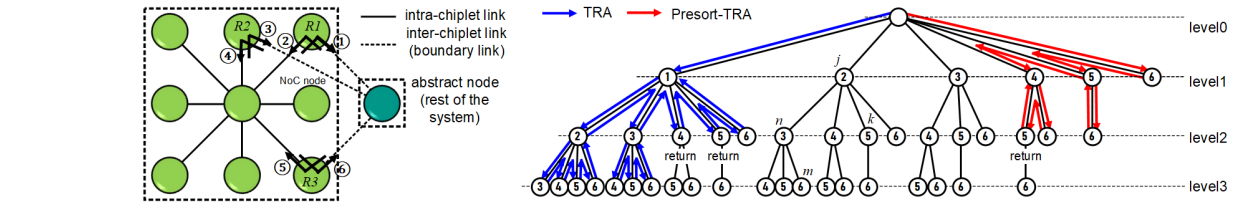
A. Co-Design Methodologies and Benchmark Design

2.5D chiplet design is becoming increasingly popular as a low-cost scalable solution to further push computational performance beyond traditional More-Moore scaling. The traditional die-by-die design flow separates engineers and CAD tools into two distinct domains: VLSI and packaging. This decoupled strategy is effective for the industry to implement in their workflow by allowing design engineers to focus on a smaller knowledge domain while isolating design efforts and responsibilities. Especially for advanced 2.5D/3D packaging, it prevents the chiplets from reaching its full potential. A conservative interface will ensure compatibility but also inevitably result in large design tolerances and reduce performance to achieve a broader reception.

One obvious solution is extending the 2D design flow into a holistic approach by including every component into the design scope. The holistic system functions like a top-level giant chip design while each individual chiplets are like macros inside. It remains very compatible with the traditional physical design flow. However, this inevitability introduces other practical concerns: IP protection, responsibility for integration, and fragmentation of heterogeneous integration.

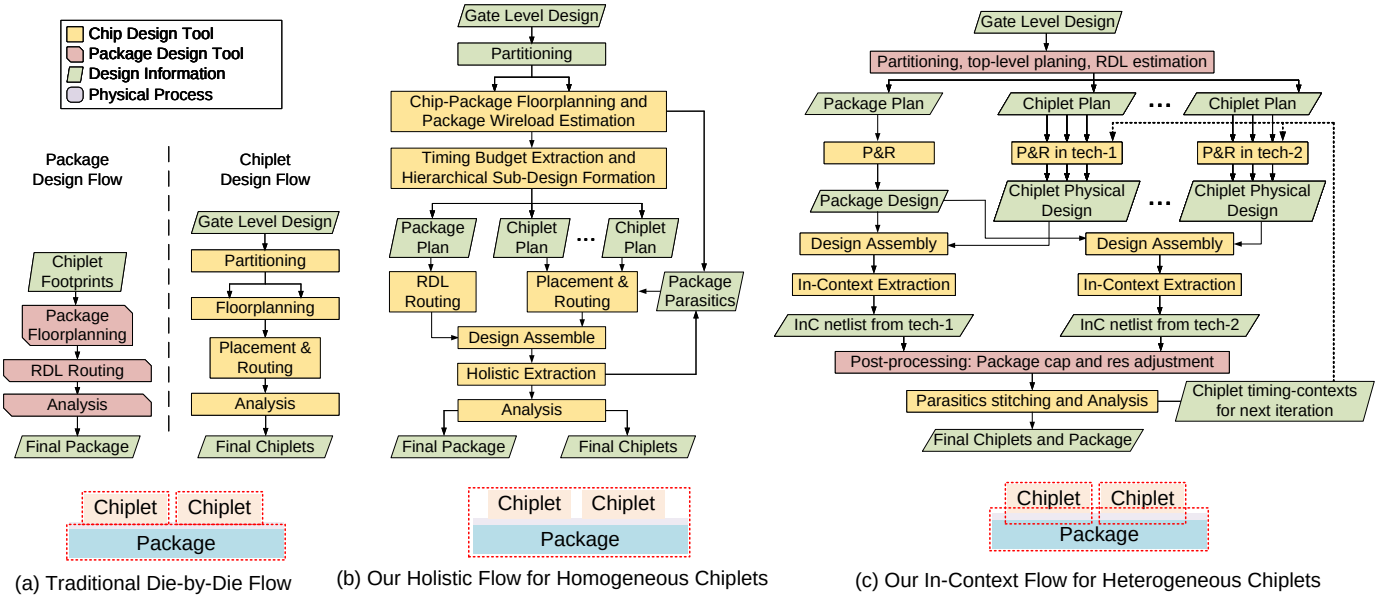
To break the design boundary without imposing the need for detailed layout information from each chiplet, we designed a novel in-context design flow. Only a few top metal layers from each chiplet is exposed to the top level as the "interface layers." Similar to the Object-Oriented-Programming, each chiplet only need to share their public abstract view while holding the IP-sensitive private detailed implementations. This approach does not require complete design files from every component, while it can still capture most chiplet-package

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2022.3203005



(a) The inter-chiplet network (b) The RCT and the searching procedure

Fig. 4. An example of the tree model and the inter-chiplet network. The boundary turns in the target chiplet are used to generate the RCT, and a search algorithm is applied to search the optimal combination of restricted boundary turns.



(a) Traditional Die-by-Die Flow (b) Our Holistic Flow for Homogeneous Chiplets (c) Our In-Context Flow for Heterogeneous Chiplets

Fig. 5. Comparison of three extraction flows.

coupling for parasitic extraction, noise, timing, and power analysis. Revealing the non-critical properties, our in-context extraction remains heterogeneous-friendly and ensures IP protection. All three methods are compared in Fig. 5.

To demonstrate our 2.5D design methodologies, we design a micro-controller system based on ARM Cortex-M0 with seven metal layers for chiplet routing and three RDL layers. We then compare different partition methodologies and choose to utilize the knowledge of the system architecture to come up with an architecture-aware partition.

With our holistic flow [2], both package and chiplets are assembled into the same VLSI design environment. Therefore, we can extract the distributed parasitic netlist of the entire chip-package system and perform timing and power analysis. Then we compare the results with the monolithic 2D implementation. Using the traditional die-by-die flow, chiplets and packages are separately optimized. As a result, the highest system frequency drops to 245 MHz for the unoptimized 2.5D system, which is much worse compared to the 2D monolithic implementation (333 MHz). However, with an iterative timing optimization using the holistic extraction, the timing degradation is almost eliminated, and the system performance is comparable to a single chiplet (300MHz). Effective for homogeneous designs, the holistic extraction is still computationally expensive to process the entire 2.5D system layout.

Designed for heterogeneous integration, our in-context flow can be used to accelerate the extraction process [3]. It only includes essential interface layers from both the package and chiplet during extraction and then emerges the parasitic database with post-processing. Also, multiple dies are extracted separately to allow extracting heterogeneous chiplets in parallel. We compare the extraction accuracy of holistic extraction to in-context extraction using our 2.5D design. Our in-context extraction achieves less than 1% error compared to holistic design. This allows the whole heterogeneous systems to achieve the same 300MHz max frequency. Our in-context extraction remains heterogeneous-friendly and new rule decks can be calibrated incrementally by reusing existing rule decks. This approach does not require complete design files from every component, while it can still capture most chiplet-package coupling for parasitic extraction, noise, timing, and power analysis.

B. Paths Forward

Our heterogeneous 2.5D design flow and CAD tool PowerSynth [1] will further enable integrating both Si chips with SiC power electronics devices while ensuring performance, reliability, and low cost.

VII. Multi-Objective Hardware-Mapping Co-Optimisation for Chiplet-based DNN Accelerators

The quest towards computation efficiency together with the ever-increasing computation demand from emerging workloads is leading to the adoption of a scalable design paradigm

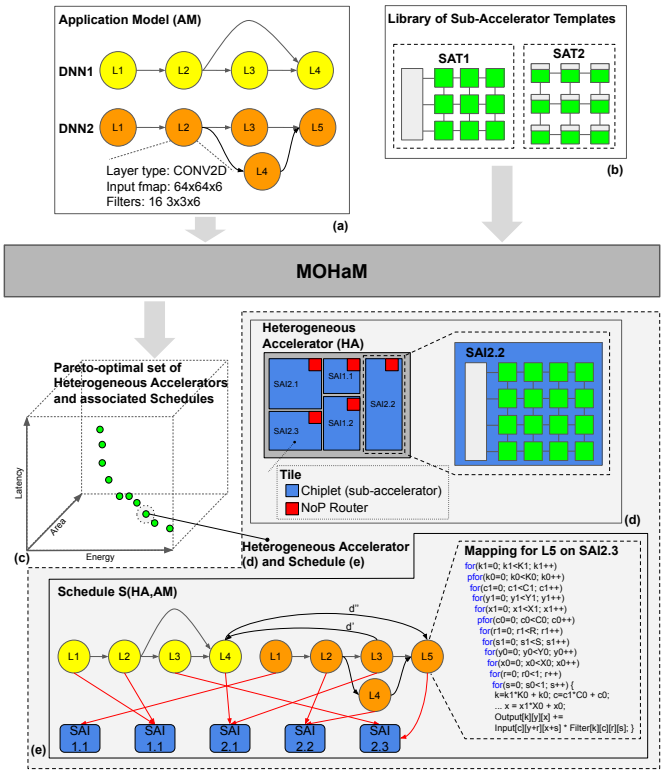


Fig. 6. The overall flow of MOHaM.

which combines multiple sub-accelerators (SAs) to build a large accelerator system. Such SAs can come in the form of chiplets that are connected by means of a Network-on-Package (NoP). In this context, hardware configuration (*i.e.*, the number, placement, interconnection of the chiplets and their configuration, *i.e.*, number of processing elements, buffer sizes, *etc*.) and mapping strategy (*i.e.*, how the workload is spatially and temporally scheduled) are the top two most important factors determining the overall accelerator performance.

This section introduces a Multi-Objective Hardware-Mapping co-optimisation framework (MOHaM) for MCM-based multi-tenant DNN accelerators. It is a first attempt on simultaneous exploration of hardware configuration and mapping strategy for multi-tenancy aimed at deriving Pareto-optimal system instances that optimize towards multiple conflicting design opbectives.

A. MOHaM Overview

The inputs and outputs of MOHaM are reported in Fig. 6. It takes in input the *application model* and a library of *parameterized sub-accelerators templates* and provides in output the Pareto-optimal set of *heterogeneous accelerators* with the corresponding *optimal schedules* that minimize energy, latency, and area.

An application model (AM) is a set of DNN models that generate the workload [Fig. 6(a)]. The DNN models in the AM are assumed to be independent each other and thus can be executed in parallel. A parameterized sub-accelerator template (SAT) is a reconfigurable accelerator supporting different mappings by means of reconfiguration and parameterized in terms of number of PEs and buffer sizes [Fig. 6(b)]. When each of the free parameters of a SAT is set, we obtain a SAT instance (SAI).

Each point of the Pareto-optimal set provided by MoHaM represents a heterogeneous accelerator (HA) and its specific schedule [Fig. 6(c)]. A HA is specified by the set of its SAIs, the NoP that allows chiplets to communicate each other and with the external DRAM through the set of available memory interfaces (MIs), and a placement function that, for each SAI and MI, returns the tile where they are placed on. For instance, Fig. 6(d) shows a HA formed by five sub-accelerators chiplets interconnected by a NoP. The sub-accelerators are instances of two parameterized sub-accelerator templates, namely, SAT1 and SAT2 shown in Fig. 6(b). Fig. 6(e) shows an example schedule. Black edges denote the layers dependencies whereas red edges denote the mappinglayer M into the sub-accelerators. Here, both L3 of DNN2 and L4 of DNN1 are mapped on the same SAI2.1 (i.e., instance 1 of SA2). Dependency d' defines their execution order, that is, L3 has to be executed before L4. Similarly, d'' defines the execution order between L4 of DNN1 and L5 of DNN2: that is, L4 has to be executed before L5.

B. MOHaM Optimization Engine

MOHaM optimization engine adopts a two-step approach. In both steps of the search the Timeloop/Accelergy [5] framework is used as the cost model.

The first step is the mapping of each layer of the AM onto each SA template in the library. This step is built by leveraging MEDEA [6] which allows the search for a Pareto set of mappings of a layer on a specific architecture, using a genetic algorithm approach augmented with custom genetic operators.

In the second step, the Pareto mappings found for each layer are considered for the global scheduling search. The global scheduler is based on the NSGA-II multi-objective genetic algorithm. The selection and survival phases are those of the original algorithm. However, several custom genetic operators have been implemented to increase sampling efficiency, thus finding better individuals in less time, but also because only a small part of the genomes are valid. Searching with default random mutation and crossover operators is therefore not feasible. The result of a global scheduler run is Pareto-optimal set of accelerators composed of heterogeneous sub-accelerators and, for each of them, the optimal schedule in such a way to minimize energy, makespan, and area.

C. Path Forward

Future research in this area should be devoted to the exploration of design space taking into account the architectural parameters of the communication sub-system and the different silicon interposer technologies.

VIII. Conclusion

In this paper, challenges in designing inter- and intra-chiplet interconnection systems in chiplet-based systems were discussed. We expect the future lies in joint consideration of all possible aspects, *i.e.*, cross-level optimization and design.

REFERENCES

[1] I. Al Razi, Q. Le, T. Evans, S. Mukherjee, H. A. Mantooth, and Y. Peng, "PowerSynth design automation flow for hierarchical and heterogeneous 2.5D multi-chip power modules," *IEEE Trans. Power Electronics*, vol. 36, no. 8, pp. 8919–8933, 2021.

[2] M. Kabir and Y. Peng, "Chiplet-package co-design for 2.5D systems using standard ASIC CAD tools," in *ASPDAC*, 2020.

[3] M. Kabir, D. Petranovic, and Y. Peng, "Coupling extraction and optimization for heterogeneous 2.5D chiplet-package co-design," in *ICCAD*, 2020.

[4] J. Kim, A. Balankutty, R. Dokania, A. Elshazly, H. S. Kim, S. Kundu, S. Weaver, K. Yu, and F. O'Mahony, "A 112Gb/s PAM-4 transmitter with 3-Tap FFE in 10nm CMOS," in *ISSCC*, 2018.

[5] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: a systematic approach to DNN accelerator evaluation," in *ISPASS*.

[6] E. Russo, M. Palesi, S. Monteleone, D. Patti, G. Ascia, and V. Catania, "MEDEA: a multi-objective evolutionary approach to DNN hardware mapping," in *DATE*, 2022.

[7] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. S. B. Altaf, N. E. Jerger, and G. H. Loh, "Modular routing design for chiplet-based systems," in *ISCA*, 2018.

[8] H. Zhi, X. Xu, W. Han, Z. Gao, X. Wang, M. Palesi, A. K. Singh, and L. Huang, "A methodology for simulating multi-chiplet systems using open-source simulators," in *NANOCOMM*, 2021.