Holistic Chiplet–Package Co-Optimization for Agile Custom 2.5-D Design

MD Arafat Kabir, *Graduate Student Member, IEEE*, and Yarui Peng, *Member, IEEE*

***Abstract*—With the increasing popularity and applications of 2.5-D integration, both chip and packaging industries are making significant progress in this direction. In advanced high-density 2.5-D packages, package redistribution layers become similar to chiplet back-end-of-line routing layers, and the gap between them scales down with pin density improvement. Chiplet–package interactions become significant and severely affect system performance and reliability. Moreover, 2.5-D integration offers opportunities to apply novel design techniques. The traditional die-by-die design approach neither carefully considers these interactions nor fully exploits the cross-boundary design opportunities. In this article, we present a holistic chiplet–package co-optimization flow for high-density 2.5-D packaging technologies with little performance overhead and zero pipeline-depth increase. Our holistic extraction can capture all parasitics from chiplets and the package and improve system performance through iterative optimizations. Both drop-in and pay-as-you-use design methodologies are implemented for agile development and quick turn-around time. To prove the effectiveness of our flow, we implement several design cases of a microcontroller system in TSMC 65-nm technology. Our design methodologies can reduce the performance gap by 85% with respect to the 2-D reference design after holistic optimizations. We demonstrate design flexibility and development cost-saving by presenting several flavors of a three chiplets system. To validate our flow in silicon, we tape-out a chip in TSMC 65-nm technology with measured data and validated functionality.**

***Index Terms*—2.5-D design, agile design, chiplet–package co-optimization, holistic flow, redistribution layer (RDL) planning.**

I. Introduction

THE demands for increased functionality and performance for applications, such as 5G, artificial intelligence, and high-performance computing, are pushing modern chips to the reticle limit. The industry responses with a modular design approach, in which a large system is broken down into smaller functional blocks and then assembled as a complete system. Traditionally, a printed circuit board (PCB) is used as the integration platform. Though PCBs are mature and cost-effective, because of long and wide traces, they suffer from high inductance and capacitance, limited bandwidth, and severe power loss. To overcome these drawbacks, the industry introduces the System-in-Package (SiP) design approach, such as 2.5-D and 3-D packagings, which leverages the fan-out wafer-level-packaging (FOWLP) technology. With transistor scaling saturated, these SiP designs are becoming popular in high-density applications, such as mobile phones and tablets [2]. Moreover, 2.5-D packaging enables heterogeneous integration [3], [4] and high-bandwidth interdie communication [5]. It also offers promising hardware security applications [6], [7]. The increasing interests are driving the industry to develop compact and high-performance FOWLP solutions [8]. As shown in Fig. 1(a), in system integration schemes through PCBs, the packages become sufficiently large compared to chiplets. The FOWLP integration solutions, as depicted in Fig. 1(c), have chip-scale packages with very fine pitch and shorter interconnects, making them promising candidates for high-performance system design. In the last few years, the industry has developed so many FOWLP technologies [9]–[11]. In every iteration of these technologies, the package wires become thinner and denser, bringing chips and packages very close with a reduced pad size.

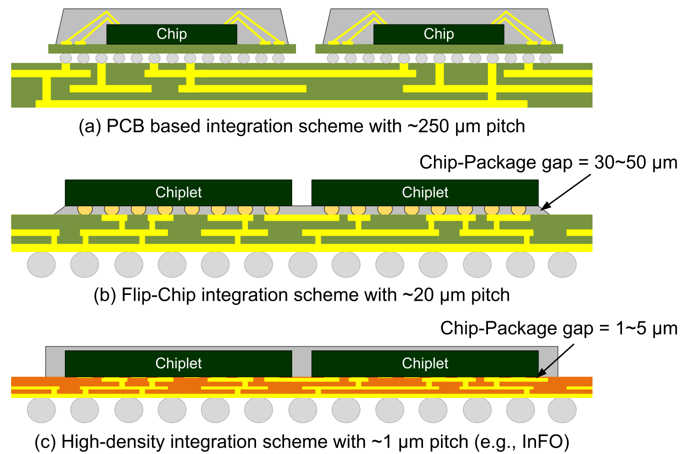


Fig. 1. 2.5-D integration schemes. (a) PCB-based system. (b) Flip-chip with an organic interposer. (c) High-density integration scheme such as wafer-level-packaging.

In the traditional flow, 2.5-D systems are designed in a die-by-die (DbD) approach where each chiplet is designed independently as a single unit, and then, all chiplets are mounted on the package as a complete system. The analysis

Manuscript received December 14, 2020; revised February 25, 2021; accepted February 28, 2021. Date of publication March 30, 2021; date of current version May 17, 2021. This work was supported by the National Science Foundation under Grant 1755981. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Recommended for publication by Associate Editor W. T. Beyene upon evaluation of reviewers' comments. *(Corresponding author: MD Arafat Kabir.)*

The authors are with the Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR 72701 USA (e-mail: makabir@uark.edu; yrpeng@uark.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCPMT.2021.3069724.

Digital Object Identifier 10.1109/TCPMT.2021.3069724

2156-3950 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

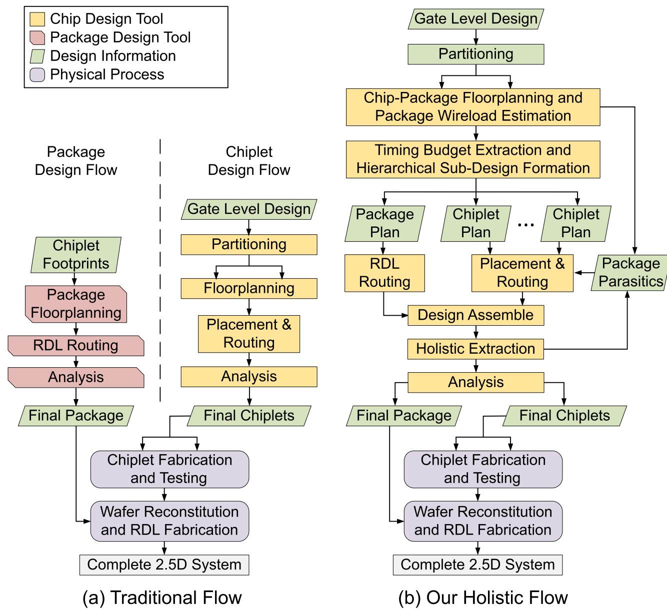


Fig. 2. (a) Traditional die-by-die design flow of a 2.5-D system versus (b) our proposed holistic (Holi) iterative optimization flow.

and optimization of chiplets and the package are also conducted separately without consideration of the interactions between them [12], [13]. Fig. 2(a) illustrates this traditional flow in which chiplets and the package never interact with each other until they are fabricated and assembled. During package design, a chiplet is approximated as a ground mesh or plane. In this approach, it is possible to achieve the shortest possible turn-around time using off-the-shelf chiplets in a plug-and-play manner [12]. This flow is sufficient when the gap between chiplets and the package is large enough to make the interactions between them minimal. As shown in Fig. 1(b), in flip-chip WLP, this gap is around 30–50 μm. In such integration technologies, the traditional flow can be used without any critical problem. However, due to the industry's aggressive development and the adoption of bumpless contact pads [4], this gap is decreasing rapidly [14]–[17]. Within a few years, this gap is reduced from tens of μm to 1.5 μm [16]. With this scaling trend, the chiplet–package gap will reduce to the submicrometer level, making the interactions between chiplets and the package more significant and critical to system reliability. To handle such high-density integration schemes, a cross-boundary design flow is required, which can capture these interactions during the design and optimization steps of both chiplet and package.

In the die-by-die approach, the complete system is not considered as a whole. Therefore, it is not possible to obtain a globally optimized system though individual chiplets may perform well. Because of the pin-dominated nature of package routing, it can get overly complicated, introducing unnecessary package overhead due to detours if chiplet pins are not planned properly. As all the chiplets work together as a single system, timing optimizations need to be performed at the system level. While planning the package, it may be necessary to rearrange the chiplets pin configurations to obtain a compact package routing to minimize package wire delays. The postdesign analysis tools need to consider chiplets and package interactions to avoid signal reliability issues, potentially causing system failure.

Apart from heterogeneous integration, 2.5-D integration technology enables the chiplet design approach. A large ASIC chip can be partitioned into smaller chiplets in order to increase yield through the use of the known-good-dies (KGD) [18]. In such systems, to ensure reliable interchiplet communication, an additional stage in the pipeline, such as SerDes [3], [15], would be necessary to hide IO overhead. This would require changes at the architecture level. Changes in architecture require sufficient engineering efforts and are not so quick and flexible. Though this is not an issue for large design houses, small ASIC design companies may not have enough resources and time for such architecture exploration. In that case, a large engineering design margin needs to be left such that the IOs from different chiplets can communicate with each other within the design tolerance. Novel IO designs [19], [20] have been proposed for 2.5-D systems, which will significantly reduce the IO overhead and power consumption. However, as these cells are not designed for driving long redistribution layer (RDL) wires with many technology variations, parasitics and static timing analysis (STA) must be performed very carefully to avoid potential violations to the overall system performance and signal integrity issues.

In this article, we present our holistic 2.5-D chiplet–package codesign and optimization flow that employs a cross-boundary strategy to design chiplets and the package together. Fig. 2(b) shows the overall steps of our flow. In this flow, chiplets and the package are assembled in a common design environment during planning and analysis steps for holistic consideration. This shared layout database allows exchanging necessary cross-boundary design information to capture coupling and mutual interactions, which is essential to achieve high analysis accuracy, co-optimization of the chiplets, and reliable system design.

Through the study of several 2.5-D design cases of an ARM Cortex-M0-based microcontroller system, we illustrate the effectiveness and flexibility of our flow. To verify our flow in silicon, we taped out and studied a chip that is designed using our flow in TSMC 65-nm technology. Through the work presented in this article, we claim the following contributions: 1) ASIC-CAD-compatible holistic flow that can design, optimize, and analyze 2.5-D systems with high-density FOWLP technologies; 2) a study of the necessity and effectiveness of holistic extraction and STA on 2.5-D systems designed in commercial technologies; 3) illustration of design flexibility and speed offered by our holistic flow with both drop-in and pay-as-you-use design strategies; and 4) silicon validation of our flow with a 2-D/2.5-D tape-out design in TSMC 65-nm technology.

Our flow is most useful when a 2.5-D system needs to be designed from its register-transfer level (RTL) description targeting the best performance achievable and take advantage of the modularity, flexibility, customization, and yield benefits offered by the 2.5-D integration technology. To the best of our knowledge, there exists no previous work that implements silicon-verified holistic flow to design 2.5-D systems and

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

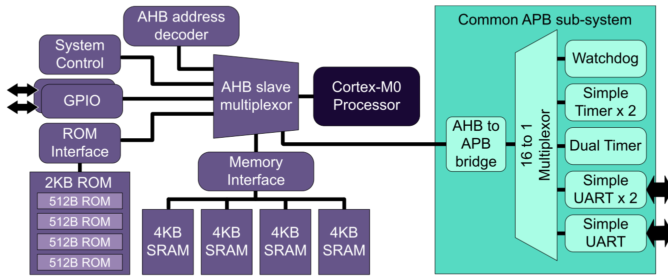


Fig. 3. System architecture of the ARM Cortex-M0-based microcontroller.

illustrates its effectiveness, flexibility, and speed through the study of design cases in commercial technologies. Our holistic flow for 2.5-D systems is available at [1].

II. Design Settings and CAD Flow

A. Architecture and Chiplet Partitions

Fig. 3 shows the system architecture of the microcontroller system we use to illustrate our flow. It has an ARM Cortex-M0 processor core connected to the rest of the system through AMBA high-performance bus (AHB). The AHB bus is connected to the system controller, two GPIO modules, an ROM table, the memory interface, and an advanced peripheral bus (APB) subsystem. The APB subsystem consists of a watchdog timer, two simple timers, a dual-timer, UART modules, and so on. The system has a total of 16-KB memory divided into four 4-KB banks. The memory interface is designed in a way that each bank occupies a contiguous address range. As a result, the system can operate even if some of the memory banks addressed by the upper address range are not present. RAM and ROM macros are compiled using ARM memory compilers.

To implement the microcontroller as a 2.5-D system, we partition it into two chiplets. We studied several partitioning algorithms and design schemes to understand the impact of package wires during the partition stage [21]. We compared area-balanced partitions using hMetis [22] and FLARE [23] algorithms, logic-versus-memory scheme, and architecture-aware scheme. In the balanced-area and logic-versus-memory partitions, the chiplet areas are not sufficient to accommodate all the pins. In the architecture-aware partition scheme, we use our knowledge of architecture to come up with a partition in which the chiplets can accommodate all of their pins with reasonable pin-pitch. We use the architecture-aware partition scheme for our experimental studies presented in the latter part of this article. This scheme helps us illustrate the drop-in design approach, which allows several flavors of a 2.5-D system with zero design cost. In this partition scheme, we gather all core logic and 8 KB of memory residing in the lower 8k address range into a Core chiplet. In the other Mem chiplet, we only keep the rest of 8 KB of memory with a few control logic. As a result, the Core chiplet can operate as a standalone system with or without the Mem chiplet.

B. Technology Settings

We use the TSMC 65-nm technology to implement a 2-D design and 2.5-D chiplets. In our holistic flow, we need a unified environment where both chiplet and package designs can be imported together for holistic planning and extraction. Moreover, there is no publicly available PDK to design 2.5-D packages for academic study. Therefore, we modified the PDK to create a unified chiplet–package codesign environment with all chiplet and package layers together.

Table I shows the settings of our modified PDK. We use the lower seven metal layers (M1-M7) with their original settings for chiplet internal routing. The parameters of M8, M9, M10, and the relevant dielectric layers are modified to mimic the attributes of TSMCs InFO package routing layers. Though the most advanced InFO flavors can handle 0.8-μm/0.8-μm width/spacing [17], we use 10 μm/10 μm for a general setup. Fig. 4 shows the layer stack of our modified PDK. For holistic extraction, we characterize this technology stack to generate an extraction-rule file. In an industrial design, this extraction-rule file would be provided by the packaging house through characterization of the chiplet–package technology combination that they support.

C. Overall CAD Flow

Our overall flow is illustrated in Fig. 2(b). When the RTL netlist is ready, the gate-level netlist is synthesized using a standard synthesis tool. The gate-level netlist is then fed to the partitioning tool along with the partition scheme settings. The partition tool takes into account the impacts of package wires on chiplet partitions. Next, we prepare the top-level plan of chiplets and the package together in the same design environment set up with the unified PDK. We determine package floorplan and chiplet pin arrangement with an algorithm that reduces the package routing issues, such as long wires or detours, and minimize package wire impacts on system performance. Next, we generate an initial package routing and estimate the package wireload on chiplet IO pins. We perform timing budget extraction of all chiplets and the package. Then, we split the overall design into individual chiplet and package subdesigns for parallel implementation. In Fig. 2(b), the "Chiplet Plan" boxes refer to plans of different chiplets, one plan for each chiplet. Though these chiplet plans are related through the top-level constraints, each plan is independent, and all plans can be implemented in parallel.

After coplanning and RDL routing, chiplets and the package can be implemented independently with contexts and constraints propagated from the top level. We perform package design utilizing the chiplet footprints, their connectivity, and the timing budget of package wires. The physical design of each chiplet is similar to the traditional 2-D chip design flow with some additional constraints imposed by the top-level plan. After placement and routing, design rule checking (DRC) is performed on all chiplets individually. If all chiplets pass the DRC, the entire system is assembled together for further optimizations and analyses.

The design assemble step combines chiplet and package designs into the same unified design environment as in the top-level planning stage. Because of this, optimization and analysis can capture chiplet–package interactions and perform adjustments to improve system performance and reliability. We perform holistic extraction, and the result is

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

TABLE I
TECHNOLOGY PARAMETERS OF OUR MODIFIED 65-NM LAYER STACK

Layer	Purpose	Width	Spacing	Thickness	Epsilon
M1-M7	Chip Internal Routing	TSMC	TSMC	TSMC	TSMC
ILD7	Inter-layer Dielectric	-	-	5 μm	2.0
M8	RDL1	10 μm	10 μm	5 μm	2.2
ILDR1	Inter-layer Dielectric	-	-	5 μm	2.0
M9	RDL2	10 μm	10 μm	5 μm	2.2
ILDR2	Inter-layer Dielectric	-	-	5 μm	2.0
M10	RDL3	10 μm	10 μm	5 μm	2.2
PP	Planar Passivation	-	-	1 μm	4.0
AP	Solder Pads	TSMC	TSMC	TSMC	TSMC

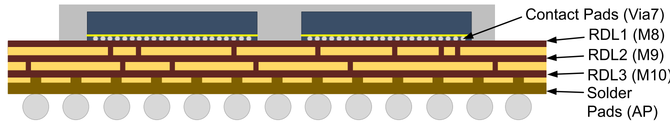


Fig. 4. Package RDL stack of our modified 65-nm PDK.

used for STA and timing context generation. These timing contexts are used to perform the next iterations of individual chiplets. This holistic optimization using standard tools improves system performance with buffer resizing, time borrowing, rerouting, and so on. The following iterations can be carried out if there is a scope of improvement, but, with a good estimation at the beginning, the second iteration is generally accurate enough. Finally, we assemble all the finished designs for full-system extraction, analysis, and sign-off verifications.

III. CHIPLET–PACKAGE COPLANNING AND MODELING

A. Top-Level Planning

The RDL routing problem of a 2.5-D package is different from the conventional chip routing problem. Existing works [24]–[26] try to solve the routability between chiplet pins in the system. However, compared to the chip routing problem, the number of nets on the package level is much fewer, and signal integrity issues are mainly caused by skewed long wires. As a result, minimizing total wire length is not always the primary concern. Several other factors, such as bus delay skew, signal-integrity, the inductive effect of long package wires, and EMI effects, can play a critical role. All these factors can be considered in the top-level planning stage of a 2.5-D system. In our strategy, we focus on developing a compact RDL routing plan with short and uniform wire lengths to minimize routing issues, such as congestion, detours, and unequal bus wire delays between chiplets.

Chiplet dimensions and pin pitch are determined based on the chiplet area and pin count. In our implementation, the Core chiplet has dimensions 520 μm $\times$ 475 μm and a total of 100 pins. The Mem chiplet has the dimension of 415 μm $\times$ 230 μm and a total of 60 pins. The pins of the Core chiplet are arranged in a 10×10 grid, and those of the Mem chiplet are arranged in a 6×10 grid. In both chiplets, the pin pitch is 40 μm in both directions of the grid. Without loss of generality, we consider two chiplets at a time in the coplanning step. In our strategy, we assign signals to chiplet pins after the package floorplan and routing are determined. The top-level

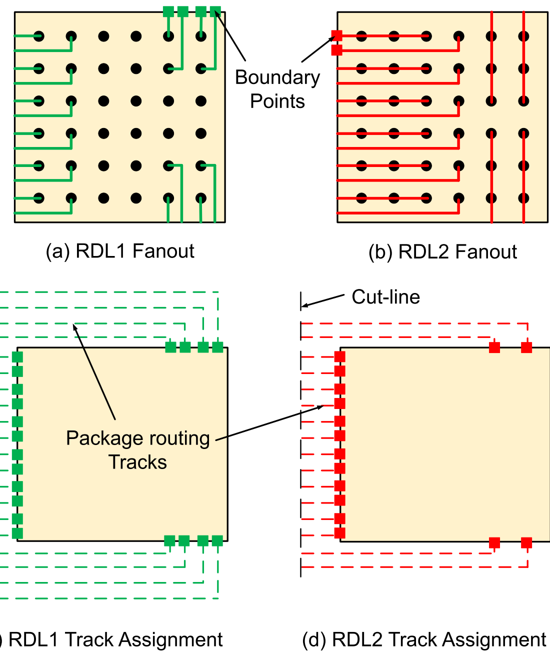


Fig. 5. Illustration of pin fan-out and track assignment of a chiplet with 6×6 pin grid and two RDLs.

package and chiplet plans are determined through pin fan-out of chiplets, RDL track assignment, package floorplanning and routing, slack-based greedy signal assignment of package wires, and package wireload estimation. Algorithm 1 describes our coplanning strategy.

B. Pin Fan-Out and RDL Track Assignment

Before a chiplet pin can be routed externally, it needs to cross its chiplet boundary. We use a greedy strategy to fan-out and track the chiplet pin assignment, which tries to use short and straight wires within a minimum number of package layers. In Algorithm 1, lines 5–11 show our pin fan-out and track assignment strategy. For the sake of illustration and explanation, we assume a cut-line between two chiplets, as shown in Fig. 6. This cut-line acts as the routing target in this step. We bring as many internal pins as we can to the chiplet boundary using all the RDL routing tracks that cross the boundary. We name the boundary locations where the pins are routed to as “boundary points.” This is performed in a specific order on all sides of the chiplet. For each side, the layer touching the contact pads is routed first, followed by the subsequent RDLs. From line 3 of Algorithm 1, the side order is determined based on their rough distance from the cut-line. The number of rows/columns of pins that can be routed to boundary points depends on the pin pitch in terms of the package routing track. As shown in Fig. 5, if the pin pitch is two tracks, two rows/columns of pins adjacent to that side can be routed to the boundary points following those tracks.

Next, we assign tracks to these boundary points. The boundary point closest to the cut-line is assigned with its nearest track first. From line 9 of Algorithm 1, boundary points are sorted based on their distances from the cut-line. As a result, in the track assignment queue, the boundary points

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 RDL Planning Algorithm

```
1  Calculate area required for the chiplets
2  Generate pin array based on pin pitch and chiplet area
3  sideOrder = [near cut-line, top, bottom, opposite side]
4  layerOrder = [RDL layers from bottom to top]
5  foreach Chiplet do
6  |  foreach s in sideOrder do
7  |  |  foreach l in layerOrder do
8  |  |  |  Route pins to the Boundary Points of s on l
9  |  Sort BoundaryPoints in increasing order of their
   |  distance from cut-line
10 |  foreach bp in BoundaryPoints do
11 |  |  Assign the nearest available track to bp
12 while Floorplan not valid do
13 |  Floorplan = New relative position of the chiplets
14 |  Check if Floorplan is valid
15 Connect pin pairs routed to the same track
16 AssignSignals(Tracks, Nets, Slack)
17 WireLoadEst(Tracks, PDK.WireLoadModel)
18 Generate TCL script and SDC files
```

facing the cut-line come first, followed by boundary points on the perpendicular side and the opposite side, sorted in increasing order by their distances to the cut-line. The opposite side is least preferred because of the detours introduced to reach the cut-line. Fig. 5 shows the pin fan-out and track assignment of a chiplet with a 6×6 pin grid and two package layers.

C. Package Floorplan and Routing

Based on the track assignment of chiplets, their relative locations are determined. These relative locations will determine the package floorplans, chiplet connectivity, and RDL routing. In our current strategy, we accept a relative position between the chiplets that can produce sufficient overlap of the tracks to allow all their connections. Lines 12–14 of Algorithm 1 describe our strategy to determine the package floorplan.

Fig. 6 illustrates this strategy for connecting four pins between the chiplets using only one RDL, where the dashed white lines show available tracks crossing the cut-line. The thick lines connected to chiplets represent assigned tracks to chiplet pins. The track assignment strategy routes the pins of Chiplet-A and Chiplet-B to their nearest tracks crossing the cut-line separately. Next, while exploring different possible relative positions between chiplets, the floorplans similar to Fig. 6(b) are rejected as those have insufficient track overlap for four connections. Among two viable solutions, in this case, we arbitrarily pick the floorplan in Fig. 6(a), which supports the number of connections between the chiplets. After finding the relative position, we define the connectivity among the pins of the two chiplets that are routed to the same track crossing the cut-line. In this example, we connect pin A1 of Chiplet-A to pin B1 of Chiplet-B because they are routed to the same track. Similarly, pins A2, A3, and A4 of Chiplet-A will be connected to pins B2, B3, and B4 of Chiplet-B, respectively.

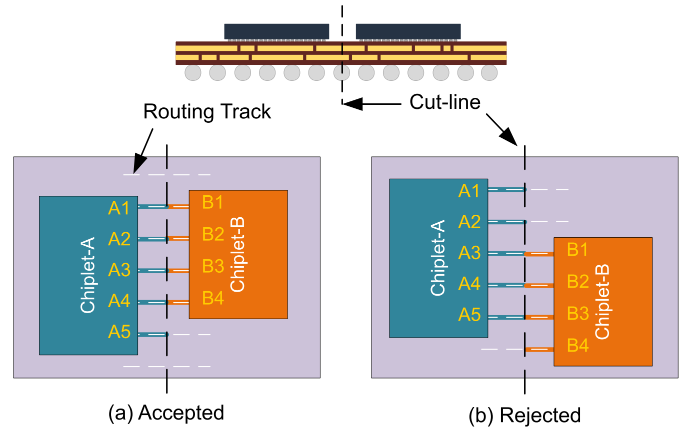


Fig. 6. Illustration of the floorplanning strategy. (a) Accepted solution that satisfies the pin connectivity requirement. (b) Rejected floorplan while finding the relative location.

Unconnected pins of the chiplets, such as pin A5 of Chiplet-A, can be used to connect with some other chiplets or to act as external I/O.

D. Signal Assignment

With the connectivity defined, both chiplet floorplan and pin assignment can be prepared in compliance with the rest of the package plan. One way to perform the signal assignment would be based on chiplet floorplans. In this strategy, a designer can prepare some initial floorplans and assign signals to the pins. Another way would be to determine the signal assignment of the chiplet pins based on timing requirements and then adjust chiplet floorplans to best suit the pin configurations. We follow the latter approach and apply a greedy algorithm for signal assignment. The AssignSignals() function in Algorithm 2 describes our signal assignment strategy. Performing the STA analysis on the synthesized gate-level netlist, timing slacks of all package wires are collected. Based on the floorplan and routing obtained in the previous steps, all track lengths connecting chiplet pins are calculated. As shown in lines 4 and 5, tracks and nets are sorted by their lengths and slacks, respectively. As a result, in lines 7–9, the net with the smallest slack is assigned to the track with the shortest length. This greedy strategy assigns timing-critical nets to shorter package wires, thus minimizes the package-wire delay overhead on them. This eventually improves overall system performance.

E. Package Wireload Estimation

When RDL routing and signal assignment are complete, parasitic loads at chiplet IOs due to package wires can be estimated. Being aware of the output load, during the optimization steps, chiplet design tools can make necessary adjustments, such as buffer insertion and cell resizing on IO nets. At this point, our goal is to perform a rough estimation of package wire loads to complete the first iteration of chiplet implementation. More accurate parasitics can be extracted from the assembled design for the second iteration of chiplet implementation. We calculate the package wireload as a linear function of the wirelength. The function WireLoadEst()

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

in Algorithm 2 describes our wireload estimation method. A wireload model is a list of values that represent the capacitance per unit length of package wires. These values are calculated from technology settings and package wire dimensions.

At the end of the codesign steps, as depicted in line 18 of Algorithm 1, the RDL planner tool generates a TCL script to implement the package routing and SDC files for all chiplets specifying wireload on IO pins. In the SDC file, the capacitance estimated by the WireLoadEst() function is specified as the wireload of the corresponding pin. With this strategy, our RDL planner can directly handle one-to-one pin connections between two chiplets. We prioritize the point-to-point connection since this is the most commonly used connection type on the package level. A multipoint connection can be handled by breaking it down into multiple point-to-point connections and then applying our strategy. Multiple chiplets can be handled by grouping the chiplets that are already interconnected into a single chiplet-like entity and perform routing between the group and another chiplet.

Algorithm 2 Signal Assignment & Wireload Estimate

```
1  Function AssignSignals(Tracks, Nets, Slack):
2      foreach track in Tracks do
3          track.length = calc_path_len(track.path)
4      sorted_tracks = sort_by_length(Tracks)
5      sorted_nets = sort_by_slack(Nets, Slack)
6      set next_track = 0
7      foreach net in sorted_nets do
8          sorted_tracks[next_track].signal = net
9          next_track + = 1
10     return
11
12 Function WireLoadEst(Tracks, WireLoadModel):
13     foreach track in Tracks do
14         cap_per_len = WireLoadModel[track.layer]
15         track.load = track.length × cap_per_len
16     return
```

IV. Physical Design

The physical design of both chiplets and the package can be implemented using any commercial chip design environment that supports hierarchical design flow. We use Cadence Innovus to perform the hierarchical implementation of the package and chiplets in our 2.5-D system. We set up the modified TSMC 65-nm PDK and load the entire system into the design environment. Chiplets appear as modules in this design environment. Based on the plan generated by our RDL planner, we place the chiplets on the package and define their signal assignments. Using the scripts generated by our RDL planner, we route the chiplet pins on RDLs. Then, the timing budget of chiplets and the package are extracted. After this step, chiplets and the package are separated as hierarchical subdesigns and can be implemented in parallel in their own design environments.

A. Hierarchical Implementation

During implementation, each chiplet is treated as a single 2-D design with some extra constraints imposed by the top-level plan and designed using traditional chip design techniques. The initial SDC file, which defines the chiplet context (such as IO delay and output load), is modified to include the wireload on chiplet pins estimated by our RDL planner tool. In the top-level planning stage, initial chiplet floorplans are prepared. This floorplan can be adjusted if necessary without changing the pin configuration specified by the top-level plan. After fixing the floorplan, the power distribution network (PDN) is designed to ensure reliable power delivery to standard cells and memory macros. Standard tools are used for standard cell placement, clock network design, routing, and postrouting optimizations. Finally, filler cells and metal fills are used to fulfill the density requirement. Fig. 7(c) (top) shows the Core chiplet, which contains all logic blocks and 8-KB memory in the lower address range. Fig. 7(c) (bottom) shows the Mem chiplet, which contains the other 8-KB memory in the upper address range.

The package design can be implemented in parallel with chiplet designs. However, more accurate and reliable optimization of the package can be performed if interface timing models of chiplets extracted after their implementation are used. Our RDL planner generates routing scripts for interchiplet routing at the end of the coplanning step. We utilize these scripts to finish interchiplet routing. Based on the package floorplan and interchiplet routing, package external IOs are placed. Chiplet pins that are not used in interchiplet connectivity are assigned as external connections to package IO pads. Fig. 7(b) shows the package design of the 2.5-D system that integrates the chiplets shown in Fig. 7(c).

With routed chiplets and package layouts, they are imported into the integrated design environment again. To ensure manufacturability, DRC is performed on each of the chiplet, and the package before design assembly. Fig. 7(d) shows a zoomed-in view of the assembled design, which shows traces from both chiplets and the package in the unified environment. Holistic extraction is performed on this assembled design using the extraction-rule file characterized for the chiplet–package unified technology. As all chiplets and the package are combined together in the same environment, all interactions between chiplets and the package are captured accurately in the extraction.

B. Holistic Extraction

Holistic extraction can be performed using any commercial extraction tool that supports hierarchical extraction flow. We use Synopsys StarRC LEF/DEF flow to perform the holistic extraction. Table II shows the coupling capacitance extraction result of the final design. For readability, capacitance numbers from M1–M5 are merged. In the traditional die-by-die approach, it is not possible to accurately capture the interactions between chiplet and package routing layers. In our holistic method, these interactions are captured, which are presented in the last three columns of the table: RDL1, RDL2, and RDL3. If we notice, there exists sufficient coupling

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

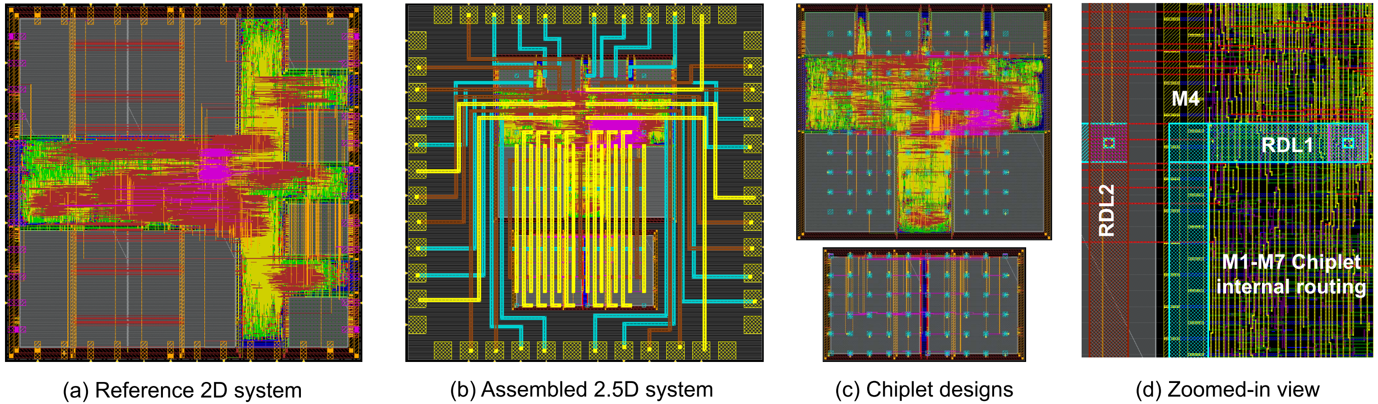


Fig. 7. Design layouts of (a) reference 2-D system, (b) assembled 2.5-D system with chiplet and package layers, (c) designs of Core chiplet (top) and Mem chiplet (bottom), and (d) zoomed-in view of the assembled design.

between RDL1 of the package and M6 and M7 of chiplets. Moreover, it is evident from the numbers that the coupling of RDL1 with M6 is greater than that with M7. As M7 is the topmost chiplet routing layer, it is expected that the coupling between RDL1 and M7 should be greater. However, as the routing on M6 is significantly greater than that on M7, and routing tracks of RDL1 and M6 run in the parallel direction, the effective overlap between RDL1 and M6 is much greater than that with M7. This detailed interaction can only be captured in a holistic extraction method. The extraction result can then be utilized to incrementally improve the system performance, signal integrity, and system reliability. In Section V, we present a set of design case studies that reveal the impact of chiplet–package interactions and how the holistic extraction result can be utilized to iteratively improve the design.

Table III shows a comparative study between die-by-die and holistic extraction results. The die-by-die extraction result is calculated by performing extractions on individual chiplets and the package separately and then adding capacitance values of corresponding layers. As seen from Table III, die-by-die extraction severely overestimates the ground capacitance, especially on package layers. This overestimation is due to the absence of chiplet routing layers between the package layers and the reference ground plane. More alarming errors are observed in coupling-capacitance. Die-by-die extraction severely underestimates the coupling capacitance on all layers as it cannot capture the interactions between chiplets and the package layers. This large error in parasitic extraction can cause severe signal integrity issues leading to system failure. Therefore, holistic consideration of chiplet and package interactions is a must in high-density 2.5-D packaging technologies.

One limitation of holistic extraction is that the existing commercial extraction tools cannot perform holistic extraction when heterogeneous technologies are involved. However, this limitation is not inherent to the holistic flow. This limitation can be overcome by extending the extraction tools to handle multiple heterogeneous technologies. An intermediate solution is to perform in-context parasitic extraction per technology and stitch them together carefully to create the holistic extraction result [27].

TABLE II
HOLISTIC CAPACITANCE (IN fF) EXTRACTION RESULTS

Metal Layer	M1-M5	M6	M7	RDL1	RDL2	RDL3
M1-M5	16348	222.5	446.7	185.3	18.61	10.18
M6	222.5	137.1	32.81	51.7	4.168	2.149
M7	446.7	32.81	371.1	32.43	1.459	1.891
RDL1	185.3	51.70	32.43	65.67	399.3	11.19
RDL2	18.61	4.168	1.459	399.3	103.3	390.5
RDL3	10.18	2.149	1.891	11.19	390.5	115.3
Ground Capacitance						
Metal Layer	M1-M5	M6	M7	RDL1	RDL2	RDL3
Capacitance	31842	1526	477	853	251	420

C. Iterative Optimizations

After design assembly and analyses, if the target performance is not achieved and discrepancies between estimated package parasitics with extraction results are observed, iterative implementation of chiplets can be conducted. If active packaging material is used, a similar optimization procedure can be performed on the package layer as well. In the first iteration of the chiplet design, the package wireload is a rough estimation based on package wirelength. Thus, we almost always expect some room for improvement. After design assembly and holistic extraction, the STA analysis is performed on the design with the holistic extraction result. Based on this analysis, new timing contexts are created for all the chiplets. In the STA analysis, timing paths through the package are modeled, including the cells within the driver and receiver chiplets. This makes the cross-boundary co-optimization between chiplets possible. One limitation of existing STA tools is that they only consider resistive and capacitive parasitic elements of the nets. However, package wires exhibit significant inductive behavior. Though, in this article, we only consider the capacitive impact of the package wires, this same methodology can be applied to consider other elements of the package wire, which affects the overall system performance.

As accurate parasitic information is available through holistic extraction, it is possible to generate a tighter timing budget for the next iteration. We use Synopsys PrimeTime to create the updated timing contexts utilizing its context characterization feature. We export the updated timing context

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

TABLE III
COMPARISON OF HOLISTIC VERSUS DBD GROUND (GCAP) AND COUPLING (CCAP) CAPACITANCE EXTRACTION RESULTS (IN fF)

Metal Layer	M1-M5	M6	M7	R1	R2	R3
DbD GCAP	31973	1568	485	1285	323	433
Holi GCAP	31842	1526	477	853	251	420
DbD GCAP Err	0.41%	2.77%	1.60%	50.77%	28.8%	3.19%
DbD CCAP	23366	394	864	547	830	517
Holi CCAP	23732	450	886	746	917	531
DbD CCAP Err	-1.54%	-12.5%	-2.57%	-26.5%	-9.55%	-2.75%

as an SDC file for each chiplet. This SDC file contains all the details of the timing contexts of each IO pin of a chiplet. For output pins, it specifies maximum transition time, wireload, pin-load, and output delay. For input pins, it specifies minimum/maximum allowed capacitance, maximum fan-out, driving cell, and input delay. For all the delay information, clock latency is also specified. Using these updated timing contexts, all chiplets are reimplemented and adjusted for the package overhead. These timing contexts can be used to perform iterative optimization of the package design as well. There can be several iterations of assembly, extraction, timing context creation, and reimplementation until it is no longer possible to improve the system performance or the target performance is met. However, with a good estimation in the first iteration, a second iteration is generally good enough to meet the best system performance.

V. Two-Way Partition Design Study

We prepared several design cases to study the impact of chiplet–package interactions on the system. We found that holistic extraction results can be utilized to significantly improve system performance. In this section, we present some of these designs and analysis results.

A. Design Case Variants

1) Case 1: Reference 2-D Design: We design a 2-D implementation of the microcontroller system as a reference design using TSMC 65-nm technology with lower seven metal layers. The gate-level netlist obtained after synthesis and before preparing chiplet partitions is used in this design. After trying out several floorplans, we settle with a square floorplan with a side length of 600 μm, as shown in Fig. 7(a). We performed PDN, cell placement, clock network design, routing, and postrouting optimizations using standard chip design tools. The finished design achieves 400-MHz maximum system frequency. In Table IV, Case 1 column shows the parameters of the finished design.

2) Case 2: Context-Free 2.5-D Design: This case is a context-free single-pass design that resembles the traditional die-by-die approach. Chiplets and the package are designed independently without using the context creation step as in our flow. Though our RDL planner generates the top-level plan, it does not perform package wireload estimation. However, we still perform design assembly and holistic extraction to capture chiplet–package interactions. This design case reveals the impact of the package on chiplets and the consequent degradation of the overall system performance. In Table IV, Case 2 column shows the parameters of this design case.

3) Case 3: Context-Aware Optimized 2.5-D Designs: This case is designed in our holistic flow and optimized using iterative context creation and reimplementation of chiplets, as discussed in Section IV-C. We try to include chiplet–package interactions as much as possible in the design and optimization steps. As discussed previously, our RDL planner prepares the top-level plan and calculates package wireload estimation, which is used in the first iteration of chiplets implementation. After design assembly and holistic extraction, extracted parasitics are used to perform STA and create chiplet timing contexts for the next iteration of chiplets implementation. The last two columns of Table IV show the parameters of two different iterations of this design.

B. Holistic Analysis and Optimization

We consider Case 1 2-D implementation as the reference design. Due to the interchiplet RDL wire overhead, it is expected that 2.5-D implementations will have worse performance. In the Case 2 design, which resembles the die-by-die design approach, after applying all possible traditional optimizations, all chiplets achieve 400-MHz operating frequency, the same as the 2-D design. However, the overall 2.5-D system can only run at a maximum frequency of 366 MHz. The slowest paths are between the chiplets through the package, resulting in a slower clock frequency. This result reveals that our holistic extraction method can capture the package impact on the overall system performance. This package overhead is overlooked in the die-by-die design approach. As a result, the die-by-die analysis will report an inaccurate system frequency. Our holistic extraction and analysis flow can accurately capture the package overhead on the system performance and report the frequency at which the system can run reliably.

In the first iteration of the Case 3 design, a predictive package wireload model is used in chiplet implementation. Though it is a very rough estimate based on a linear model, this design achieves an operating frequency of 384 MHz. Compared to the performance gap of 34 MHz between the 2-D implementation and the Case 1 2.5-D implementation, this is an approximately 50% reduction in the performance gap. This result reveals the importance of considering chiplet–package interactions, even in the early planning stage.

In the second iteration of the Case 2 design, timing contexts created using holistic extraction results are imported during chiplet implementation. These contexts have an accurate picture of the overall system. Using these contexts, the chiplet design tools can adjust chiplet designs to compensate for the delay introduced by package wires. As a result, in the second iteration, the 2.5-D system achieves a 395-MHz operating frequency, which is very close to the 2-D system performance. As the critical path is from the Core chiplet to the Mem chiplet through the address bus, the size and number of buffers in the Core chiplet increased, while the redundant buffers in the Mem chiplet are removed. All these optimizations are performed by the chip design tools without any special setting other than the timing contexts created using the holistic

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
ANALYSIS RESULT COMPARISON OF THE MICROCONTROLLER SYSTEM

Design Case	Case-1	Case-2		Case-3 first iteration		Case-3 2nd/final iteration	
Chip Design	2D Chip	Core Chiplet	Ext. Mem Chiplet	Core Chiplet	Ext. Mem Chiplet	Core Chiplet	Ext. Mem Chiplet
Logic Gates#	24141	23933	20	23918	15	23909	0
Buffer/Inverter#	4760	4684	20	4634	15	4653	0
Die Size (um × um)	600 × 600	520 × 475	415 × 230	520 × 475	415 × 230	520 × 475	415 × 230
Total Chip Wirelength (mm)	551.974	495.578	14.289	488.11	13.842	485.923	12.373
M6 Wirelength (mm)	15.128	12.978	2.847	13.607	4.052	11.866	4.579
M7 Wirelength (mm)	8.562	19.084	1.991	18.117	2.312	17.446	3.264
Max Frequency	400 MHz	366 MHz		384 MHz		395 MHz	
Performance Gap	0%	100%		47.05%		14.70%	
Chip Power	20.1 mW	18.4 mW	2.504 mW	18.2 mW	2.506 mW	18.2 mW	2.576 mW

extraction result. More iterations are performed afterward, but there is no significant improvement in system performance. That is why the second iteration is taken as the final design of Case 3. This result reveals that, with proper considerations of the chiplet–package interactions, it is possible to reduce the interchiplet overhead and optimize the overall 2.5-D system performance. In this design case, we reduce the performance gap between the 2-D implementation and the Case 2 2.5-D system by 85% through our holistic extraction and iterative optimization flow.

VI. AGILE MULTIWAY DESIGN TECHNIQUES

Though our design flow and planning strategy are illustrated based on a two-way partition design, it can be easily extended for multiway partitioned designs. To illustrate a multiway partitioned system, the application of novel design techniques enabled by 2.5-D integration, and the design flexibilities offered by our flow, we discuss a three-way partition implementation of the microcontroller system.

A. Three-Way Partition Design

In this implementation, the 8-KB Mem chiplet of the previous 2.5-D system is further divided into two 4-KB Mem chiplets. Fig. 8 shows the chiplets for this three-way partition design. This way, now, the 2.5-D system can have three different flavors with 16-, 12-, and 8-KB memory capacities. Fig. 9 shows all these flavors of the system. Fig. 9(b) shows the system with all three chiplets with 16-KB memory. The RDL plan of this design is prepared in two stages. In the first stage, only the two 4-KB Mem chiplets are considered. These two chiplets share 12 connections on the address bus. The RDL planning tool routes these nets using straight horizontal wires on RDL1. These wires can be seen in Fig. 9(b) as horizontal blue wires in the lower half of the package. In the second stage of RDL planning, these two chiplets are considered as a single chiplet-like group. For the RDL router, dummy pin locations are specified on the horizontal RDL1 wires between the two chiplets. The RDL planning tool routes the connections between the Core chiplet and this combined chiplet-like group to finish the interchiplet routing. The address bus is routed on RDL2 and RDL3 and form T-connections with the RDL1 wires between the Mem chiplets. Finally, the remaining pin locations of the Core chiplet are used as external IOs. With this top-level

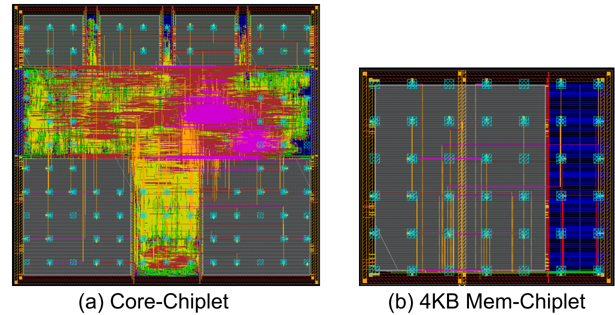


Fig. 8. Layouts of the chiplets for the three-way partition design study. (a) Core chiplet. (b) 4 KB Mem chiplet.

RDL plan, chiplets are implemented following our iterative optimization flow. As seen from the first row of Table V, the optimized 16-KB system achieves a maximum operating frequency of 380 MHz.

B. Drop-In Approach

In fan-out wafer-level packaging (FOWLP), a reconstituted wafer is built using the KGD of the chiplets. In this step, a chiplet can be deliberately left out from the reconstituted wafer to design low-cost flavors of a system with limited capabilities while keeping all optical masks untouched. We call this the "drop-in" design approach because the whole system could be created if the missing chiplet is dropped into the package. In the three-way partitioned design, a 12-KB system can be designed if the second Mem chiplet is excluded from the package. This design approach requires zero design costs but offers the end-users to choose from several options as per their requirements. Our flow can handle this design approach. Fig. 9(c) shows the drop-in design with 12-KB memory. The second Mem chiplet is excluded from the package, and holistic extraction and analysis are performed after design assembly. The second row of Table V shows that, in the absence of the second Mem chiplet, the system performance improved from 380 to 390 MHz without any further optimizations on the chiplets.

C. Pay-as-You-Use Approach

This is another approach, similar to the drop-in approach, to develop several flavors of a 2.5-D system. In this approach,

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

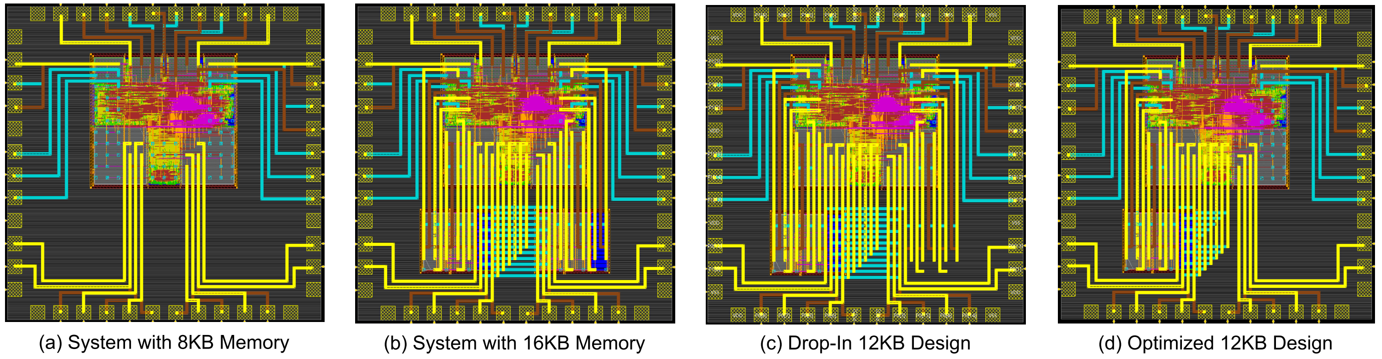


Fig. 9. Design layouts of (a) core-only system with 8-KB memory, (b) optimized full system with 16-KB memory, (c) 12-KB design using the drop-in approach, and (d) optimized 12-KB design using pay-as-you-use approach.

each design flavor is customized depending on the usage of systems components. The penalty paid in the system performance and power depends on the use of the system resources. Fig. 9(d) shows a 12-KB implementation of the three-way partitioned system designed in this approach.

Unlike the drop-in design, the package routing is modified to remove redundant package wires related to the second Mem chiplet. Then, we simply perform another incremental iteration of our holistic optimization flow, which automatically adjusts package wire drivers of the chiplets according to design needs. Since all steps are performed in a standard ASIC design environment, this process is fully automated and takes less than an hour, enabling agile design customization. As seen from the third row of Table V, the system performance improved from 390 MHz of the drop-in design to 396 MHz with the reimplementation. Though, in this design case, it is not a huge performance gain, this illustrates the flexibility and optimization capability of our flow, which can be utilized by system designers to quickly generate customized flavors of 2.5-D systems and reduce the turn-around time.

Of course, another flavor of the system can be designed by removing both the Mem chiplets and keeping the Core chiplet only in the package. This system is shown in Fig. 9(a). We perform holistic extraction and analysis on this system. This system achieves an operating frequency of 400 MHz, the same as that of the reference 2-D design, demonstrating no observable delay overhead introduced by package wires.

VII. Silicon Validation With Tape-Out

To validate our flow in silicon, we taped out a shared-block design containing a 2-D system and a 2.5-D implementation of the microcontroller. We used TSMC 65-nm as the implementation technology. We modified the top two routing layers of the chip design PDK to be used as RDLs. As this chip is designed to be manufactured by mimicking the attributes of 2.5-D RDLs, several design considerations are made. The system architecture of the ARM Cortex-M0 microcontroller is originally designed to be implemented as a System-on-Chip (SoC) at a target frequency of 100 MHz in TSMC 65-nm technology. For this mimicked technology

TABLE V

Comparison of Three-Way Partition Design Cases

Design Flavor	LPD	Frequency	Power	RDL Wirelength
16KB Optimized	2.62 ns	380 MHz	19.7 mW	46826 µm
12KB Drop-in	2.56 ns	390 MHz	18.8 mW	46826 µm
12KB Optimized	2.52 ns	396 MHz	18.8 mW	35541 µm
8KB Core-only	2.50 ns	400 MHz	18.1 mW	20905 µm

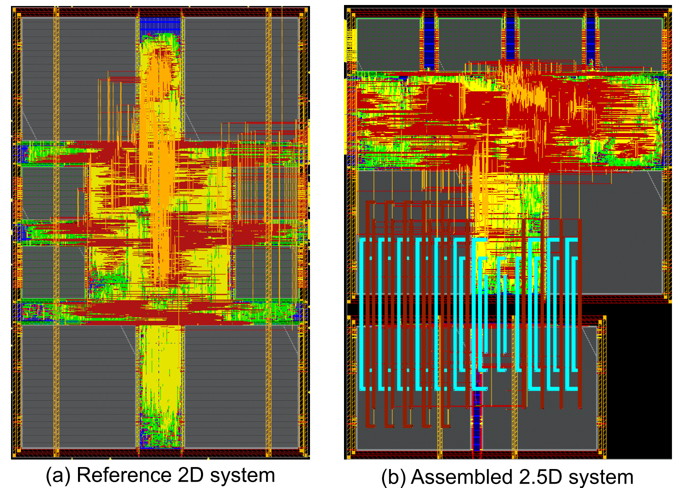


Fig. 10. System designs for tape-out. (a) Reference 2-D system. (b) Assembled 2.5-D system.

stack, we do not have a foundry-provided extraction-rule file. Moreover, for the shared-IO design and chip testing using simple logic analyzers, several testing logic are embedded within the chip. For all these reasons, though the target frequency is 400 MHz, the taped-out system runs at around 100 MHz.

A. 2-D and 2.5-D System Designs

A reference 2-D system is designed along with a 2.5-D system for tape-out. Lower six metal layers are used for designing the 2-D system and for performing the internal routings of the 2.5-D chiplets. The 2-D system is designed using traditional chip design flow. The chiplets of the 2.5-D system are designed in our holistic design and optimization

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

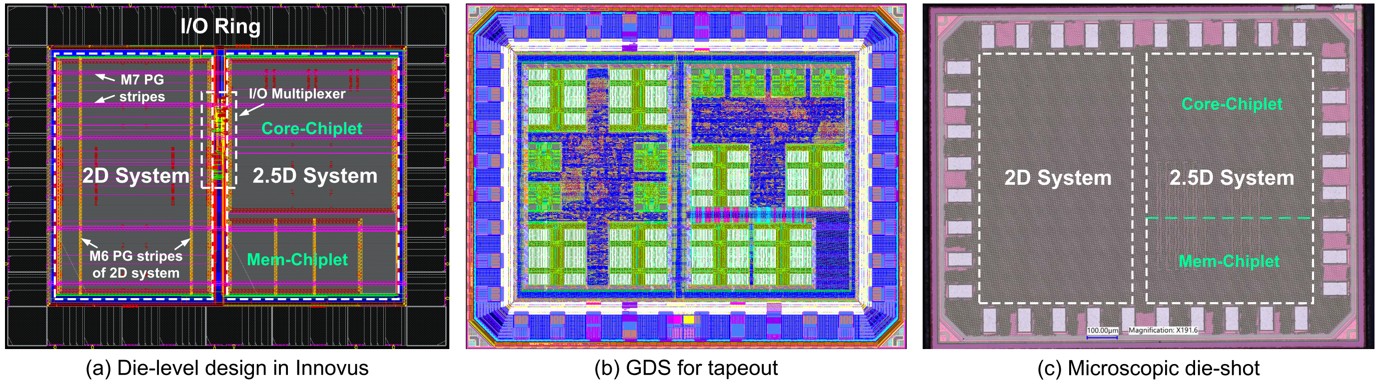


Fig. 11. Final design for tape-out and the fabricated die. (a) Die-level design in Innovus. (b) Combined GDS for tape-out. (c) Microscopic image of the taped-out die.

flow. In both systems, M5 and M6 are used for designing the PG ring around the core. The PG rails on M1 and PG stripes on M6 supply power to the standard cells and macros, respectively. Because of the shared-block design approach, only the interchiplet connections in the 2.5-D system are routed using RDLs. The external IOs of both systems are placed on M7, next to an IO multiplexing module. Fig. 10(a) shows the finished 2-D system. Fig. 10(b) shows the assembled 2.5-D system.

B. Shared IO Design

To save the pin area, we use a shared-block approach in our tape-out design. This also helped us satisfy the minimum area requirements of the foundry while saving the IO cell area. The 2.5-D implementation is combined with the 2-D implementation, as shown in Fig. 11(a). The systems share the same IO driver cells to communicate with the outside world. An IO multiplexing module is placed in between the two systems, which allows either one of the two systems to use the driver cells to communicate. The die-level power delivery network is designed on M7. As both systems have their PG rings on M5 and M6, horizontal M7 stripes are used to connect these rings with the IO ring around the die. Several stripes on M7 are used to ensure reliable power delivery to both systems.

C. Sign-Off Verifications

DRC verifications are performed on chiplet designs and the assembled system designs. After fixing system-level design errors, both systems are merged together into the die-level GDS. Fig. 11(b) shows the combined GDS. To pass the sign-off verifications of the foundry, we had to make some adjustments to the final design. For example, to pass the antenna rule check, some of the wide wires on M8 (RDL1) and M9 (RDL2) were adjusted to reduce the antenna area. To fulfill the density requirement, special filler cells and metal fills were used. After all tests are successfully passed, the design was sent out for fabrication. Fig. 11(c) shows the microscopic die-shot of the fabricated die.

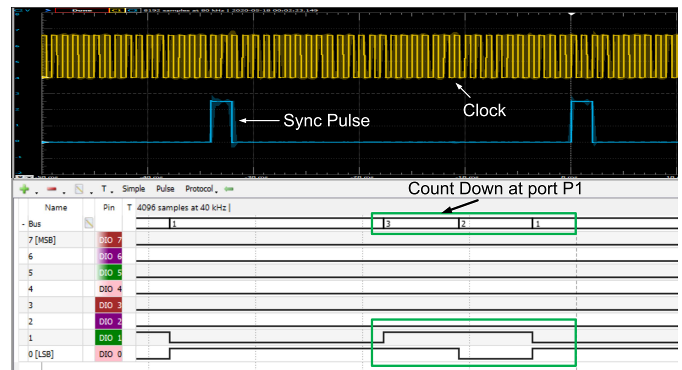


Fig. 12. Chip testing waveforms from the logic analyzer.

D. Chip Testing and Flow Validation

Both systems in the fabricated chip are tested using test vectors generated using a logic analyzer. Among several test cases, Fig. 12 shows the waveforms of the GPIO countdown test. In this test, the system reads the GPIO for a top value and then counts down to 1. At the end of the countdown, it generates a pulse (Sync Pulse) at a specific pin. The clock signal, Sync Pulse, and the countdown values on a bus are shown in Fig. 12. Both systems in the die are tested individually, and both of them passed all the tests successfully. This silicon design proves that our holistic flow is fully compatible with the current ASIC CAD flow and foundry model. Our agile design approach can make designing custom 2.5-D multichiplet systems as easy as designing 2-D modular ASICs.

VIII. Conclusion

In this article, we present our holistic design, analysis, and optimization flow for 2.5-D systems. Through several design case studies, we show that chiplet–package interactions in high-density integration technologies significantly affect the overall system performance. The traditional die-by-die approach is not sufficiently accurate for advanced packaging solutions. Our holistic extraction flow can accurately capture these cross-boundary interactions and optimize the system to reduce the overhead of package wires on system performance.

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.

We reduced the performance gap between a reference 2-D implementation and a traditional die-by-die 2.5-D implementation by 85% using our iterative optimization flow. Moreover, as illustrated using a three-way partitioned system, our flow supports several agile design approaches that can be exploited to generate different flavors of a 2.5-D system with almost zero design cost. These holistic design methodologies offer designers and application engineer's flexibility, low-cost customization, and performance. The other detailed analysis steps, such as power integrity, signal integrity, and thermal analyses, are not performed in this study. Those steps will be included in the future versions of this flow. Our flow is tested in silicon and can be utilized to implement 2.5-D systems in commercial technologies using standard VLSI CAD tools.

REFERENCES

[1] (Mar. 2021). *2.5D Holistic Design Flow*. [Online]. Available: https://e3da.csce.uark.edu/release/2.5D_Design_Flow

[2] C.-C. Hsieh, C.-H. Wu, and D. Yu, "Analysis and comparison of thermal performance of advanced packaging technologies for state-of-the-art mobile applications," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2016, pp. 1430–1438.

[3] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony, "2.2 AMD chiplet architecture for high-performance server and desktop products," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2020, pp. 44–45.

[4] J. H. Lau *et al.*, "Fan-out wafer-level packaging for heterogeneous integration," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 8, no. 9, pp. 1544–1560, Sep. 2018.

[5] S. Dwarakanath *et al.*, "Evaluation of fine-pitch routing capabilities of advanced dielectric materials for high speed panel-RDL in 2.5 D interposer and fan-out packages," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2019, pp. 718–725.

[6] Y. Xie, C. Bao, and A. Srivastava, "Security-aware 2.5D integrated circuit design flow against hardware IP piracy," *Computer*, vol. 50, no. 5, pp. 62–71, May 2017.

[7] S. Patnaik, M. Ashraf, O. Sinanoglu, and J. Knechtel, "Best of both worlds: Integration of split manufacturing and camouflaging into a security-driven CAD flow for 3D ICs," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2018, pp. 1–8.

[8] J. H. Lau, "Recent advances and trends in fan-out wafer/panel-level packaging," *J. Electron. Packag.*, vol. 141, no. 4, pp. 040801-1–040801-27, Dec. 2019.

[9] W. Ki *et al.*, "Chip stackable, ultra-thin, high-flexibility 3D FOWLP (3D SWIFT technology) for hetero-integrated advanced 3D WL-SiP," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2018, pp. 580–586.

[10] C.-F. Tseng, C.-S. Liu, C.-H. Wu, and D. Yu, "InFO (wafer level integrated fan-out) technology," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2016, pp. 1–6.

[11] J. Lin *et al.*, "Scalable chiplet package using fan-out embedded bridge," in *Proc. IEEE Electron. Compon. Technol. Conf.*, Jun. 2020, pp. 14–18.

[12] J. Kim *et al.*, "Architecture, chip, and package codesign flow for interposer-based 2.5-D chiplet integration enabling heterogeneous IP reuse," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 28, no. 11, pp. 2424–2437, Nov. 2020.

[13] W.-H. Liu, M.-S. Chang, and T.-C. Wang, "Floorplanning and signal assignment for silicon interposer-based 3D ICs," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.

[14] D. Yu, "A new integration technology platform: Integrated fan-out wafer-level-packaging for mobile applications," in *Proc. Symp. VLSI Technol.*, Jun. 2015, pp. T46–T47.

[15] N.-C. Chen *et al.*, "A novel system in package with fan-out WLP for high speed SERDES application," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2016, pp. 1495–1501.

[16] H.-P. Pu, H. J. Kuo, C. S. Liu, and D. C. H. Yu, "A novel submicron polymer re-distribution layer technology for advanced InFO packaging," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2018, pp. 45–51.

[17] C.-T. Wang *et al.*, "Signal integrity of submicron InFO heterogeneous integration for high performance computing applications," in *Proc. IEEE Electron. Compon. Technol. Conf.*, May 2019, pp. 688–694.

[18] P. Vivet *et al.*, "2.3 A 220GOPS 96-core processor with 6 chiplets 3D-stacked on an active interposer offering 0.6ns/mm latency, 3Tb/s/mm^2 inter-chiplet interconnects and 156mW/mm^2@ 82%-peak-efficiency DC-DC converters," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2020, pp. 46–48.

[19] J. Wang, S. Ma, P. D. S. Manoj, M. Yu, R. Weerasekera, and H. Yu, "High-speed and low-power 2.5 D I/O circuits for memory-logic-integration by through-silicon interposer," in *Proc. IEEE Int. 3D Syst. Integr. Conf.*, Oct. 2013, pp. 1–4.

[20] D. Xu, H. Huang, N. Yu, and H. Yu, "An energy-efficient 2.5 D through-silicon interposer I/O with self-adaptive adjustment of output-voltage swing," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2014, pp. 93–98.

[21] M. A. Kabir and Y. Peng, "Chiplet-package co-design for 2.5 D systems using standard ASIC CAD tools," in *Proc. Asia South Pacific Design Automat. Conf.*, Jan. 2020, pp. 351–356.

[22] G. Karypis and V. Kumar, "Multilevel *k*-way hypergraph partitioning," *VLSI Des.*, vol. 11, no. 3, pp. 285–300, Jan. 2000.

[23] J. Cong, S. K. Lim, and C. Wu, "Performance driven multi-level and multiway partitioning with retiming," in *Proc. Conf. Design Autom.*, 2000, pp. 274–279.

[24] T.-C. Lin, C.-C. Chi, and Y.-W. Chang, "Redistribution layer routing for wafer-level integrated fan-out package-on-packages," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 2017, pp. 561–568.

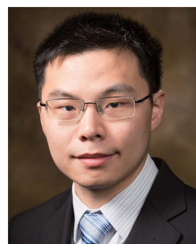
[25] H.-T. Wen, Y.-J. Cai, Y. Hsu, and Y.-W. Chang, "Via-based redistribution layer routing for InFO packages with irregular pad structures," in *Proc. ACM/IEEE Design Autom. Conf.*, Jul. 2020, pp. 1–6.

[26] C.-H. Chiang, F.-Y. Chuang, and Y.-W. Chang, "Unified redistribution layer routing for 2.5 D IC packages," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 331–337.

[27] M. A. Kabir, D. Petranovic, and Y. Peng, "Coupling extraction and optimization for heterogeneous 2.5 D chiplet-package co-design," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–8.

MD Arafat Kabir (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering (EEE) from the Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2017. He is currently pursuing the Ph.D. degree with the Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, AR, USA.

He studies design methodologies and algorithms for accurate extraction, analysis, and optimization of high-performance and high-density 2.5-D systems. His research interest is in computer-aided design (CAD) tool development for VLSI.



Yarui Peng (Member, IEEE) received the B.S. degree from Tsinghua University, Beijing, China, in 2012, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2014 and 2016, respectively.

He is currently an Assistant Professor with the Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, AR, USA. He studies design methodologies and optimization algorithms for parasitic extraction, signal integrity, power integrity, and thermal reliability. He also develops design automation tools for power electronics to improve performance, reliability, and productivity. His research interests are computer-aided design, analysis, and optimization for emerging technologies and multichip packages, such as 2.5-D/3-D ICs and wide bandgap power electronics.

Dr. Peng received the best paper awards at SRC TECHCON 14, ICPT 16, and EDAPS 17. He is also an NSF CAREER Award Winner in 2021.

Authorized licensed use limited to: University of Arkansas. Downloaded on May 30,2021 at 19:20:08 UTC from IEEE Xplore. Restrictions apply.