Big-Little Chiplets for In-Memory Acceleration of DNNs: A Scalable Heterogeneous Architecture

Gokul Krishnan[1,†*], A. Alper Goksoy[2,†], Sumit K. Mandal[2,†], Zhenyu Wang[1],
Chaitali Chakrabarti[1], Jae-sun Seo[1], Umit Y. Ogras[2], Yu Cao[1*]
[1]Arizona State University, [2] University of Wisconsin-Madison, *Email: {gkrish19, yu.cao}@asu.edu

ABSTRACT

Monolithic in-memory computing (IMC) architectures face significant yield and fabrication cost challenges as the complexity of DNNs increases. Chiplet-based IMCs that integrate multiple dies with advanced 2.5D/3D packaging offers a low-cost and scalable solution. They enable heterogeneous architectures where the chiplets and their associated interconnection can be tailored to the non-uniform algorithmic structures to maximize IMC utilization and reduce energy consumption. This paper proposes a heterogeneous IMC architecture with big-little chiplets and a hybrid network-on-package (NoP) to optimize the utilization, interconnect bandwidth, and energy efficiency. For a given DNN, we develop a custom methodology to map the model onto the big-little architecture such that the early layers in the DNN are mapped to the little chiplets with higher NoP bandwidth and the subsequent layers are mapped to the big chiplets with lower NoP bandwidth. Furthermore, we achieve a scalable solution by incorporating a DRAM into each chiplet to support a wide range of DNNs beyond the area limit. Compared to a homogeneous chiplet-based IMC architecture, the proposed big-little architecture achieves up to 329× improvement in the energy-delay-area product (EDAP) and up to 2× higher IMC utilization. Experimental evaluation of the proposed big-little chiplet-based RRAM IMC architecture for ResNet-50 on ImageNet shows 259×, 139×, and 48× improvement in energy-efficiency at lower area compared to Nvidia V100 GPU, Nvidia T4 GPU, and SIMBA architecture, respectively.

1 INTRODUCTION

State-of-the-art deep neural networks (DNNs) have become more complex with deeper, wider, and more branched structures to cater to demanding applications [4, 34]. The growing complexity reduces hardware inference performance due to increased memory accesses and computations [34]. To boost the performance and energy efficiency, in-memory computing (IMC)-based architectures embed the matrix-vector-multiplications within the memory arrays [8, 18, 19, 25, 28]. However, IMC architectures with stationary weights stored on the chip result in significant area overhead and fabrication cost [9, 25]. Hence, 2.5D/3D architectures are adopted to design large-scale DNN accelerators using an array of small chips (i.e. chiplets) connected by a network-on-package (NoP) [10, 22].

Prior studies have demonstrated chiplet-based architectures based on both IMC and conventional multiply-and-accumulate (MAC) engines for DNN acceleration [5, 7, 9–13, 22, 26, 30, 32, 33, 35]. However, existing schemes do not consider the non-uniform distribution of weights and activations within DNNs while designing the chiplet-based architecture. Figure 1(a) and Figure 1(b) show the distribution of activations and weights (normalized) across all layers of ResNet-50 on ImageNet and VGG-19 on CIFAR-100. The initial layers have more activations between layers but have fewer weights. A larger number of activations lead to more on-chip data movement, while fewer weights imply reduced computations. In contrast, the latter layers have more weights and fewer activations, resulting in increased computations and reduced data movement. Hence, the chiplet-based IMC architectures should be optimized to match the non-uniform algorithm structure and maximize the efficiency of computation and data movement across the DNN layers.

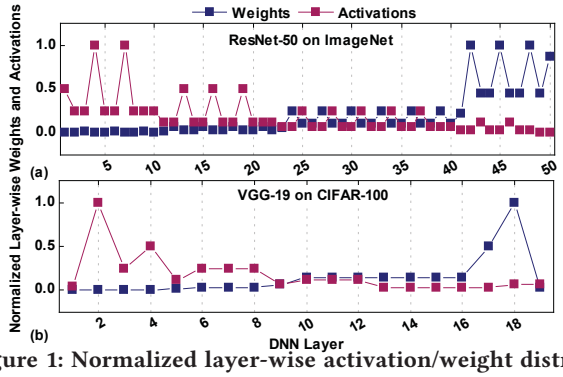

Figure 1: Normalized layer-wise activation/weight distribution for (a) ResNet-50 (ImageNet) and (b) VGG-19 (CIFAR-100). Initial/latter layers are activation/weight dominated.

Figure 2(a) shows the IMC utilization of four different DNNs using a homogeneous chiplet-based RRAM IMC architecture. The architecture utilizes chiplets with 16 tiles, where each tile consists of an array of 16 IMC crossbar arrays of size 256×256 [9]. The chiplets are interconnected by a 32-bit wide NoP operating at 250MHz, having the signaling scheme in [31]. Smaller DNNs like DenseNet-40 on CIFAR-10 have 29% IMC utilization, while larger DNNs like VGG-19 on CIFAR-100 achieve 40% IMC utilization. A lower IMC utilization leads to increased IMC array arrays and in turn, higher energy and latency. Furthermore, a single NoP structure results in significant area overhead due to the large NoP driver and interconnect cost. Figure 2(b) shows that for the homogeneous structure, the NoP accounts for 90% and 50% of the total area for VGG-19 on CIFAR-100 and DenseNet-40 on CIFAR-10, respectively. In addition, the increased NoP bus width leads to higher NoP energy with up to 53.75× higher cost relative to an 8-bit multiply-and-accumulate (MAC) operation in 16nm technology node [30].

This work addresses the inefficiency of homogeneous chiplet-based IMC architectures that fail to exploit the underlying distribution of weights and activations within DNNs. To this end, we propose a heterogeneous chiplet-based IMC architecture that integrates

† **Authors contributed equally.** This work was supported by C-BRIC, one of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA and Semiconductor Research Corporation under task ID 3012.001.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549447

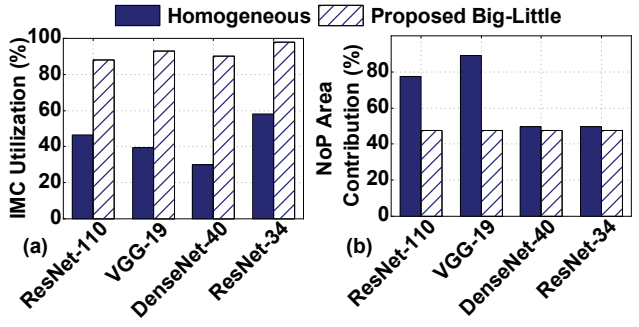

Figure 2: IMC utilization for different DNNs using a homogeneous chiplet RRAM IMC architecture [9] and the proposed heterogeneous big-little chiplet architecture. The heterogeneous big-little architecture improves the IMC utilization.

big and little-chiplet banks, as illustrated in Figure 3. Specifically, we develop an algorithm to determine the optimal configuration of the big-little IMC chiplet architecture. The little-chiplet bank consists of little chiplets interconnected by an interposer-based NoP (chiplets are placed closed to each other) [31]. Similarly, the big-chiplet bank consists of big chiplets interconnected by a bridge-based NoP [17]. Little chiplets consist of fewer/smaller IMC crossbars or processing element (PE) arrays, while the big chiplets have more/larger IMC crossbars or PE arrays. In addition, each chiplet (big/little) utilizes a local DRAM to store the weights of the DNN.

In addition to the hardware architecture, we also propose a new technique to map DNNs onto the big-little chiplet-based IMC architecture. Taking a cue from the non-uniform distribution of the weights and activations within the DNN, we propose to map the early layers within a DNN onto the little chiplet bank and the subsequent layers onto the big chiplet bank. The smaller structure of the weights in the early layers results in higher utilization within the little chiplet bank, while the larger layers towards the end of the DNN achieve high utilization on the big-chiplet bank. To achieve this, we develop a custom mapping algorithm that performs the mapping of the DNN on to the big-little architecture. We note that, the algorithm is universal and applies to the case when the resource in a given big-little chiplet is not enough to store all DNN weights.

We exploit the activation distribution by utilizing an interposer-based NoP with high bandwidth within the little chiplet bank, which houses the early layers with higher on-chip data movement. Simultaneously, the subsequent layers with lower on-chip data movement (fewer activations) utilize the bridge-based NoP with lower bandwidth within the big chiplet bank. Experimental evaluation of the proposed big-little chiplet-based RRAM IMC architecture on ResNet-50 on ImageNet shows up to 259×, 139×, and 48× improvement in energy-efficiency with lower area compared to Nvidia V100 GPU, Nvidia T4 GPU, and SIMBA [26] architecture, respectively.

The major contributions of this work are as follows:

- We propose a heterogeneous big-little chiplet-based IMC architecture that utilizes a big and little IMC-based chiplet compute structure coupled with an optimal NoP configuration (interposer and bridge).
- We present a custom mapping strategy of DNNs onto the big-little chiplet IMC architecture that exploits the non-uniform distribution of weights and activations,
- Our experiments of the proposed big-little chiplet-based RRAM IMC architecture on ResNet-50 on ImageNet achieve up to 259×, 139×, and 48× improvement in energy-efficiency and lower area compared to Nvidia V100 GPU, Nvidia T4 GPU, and SIMBA [26] architecture, respectively.

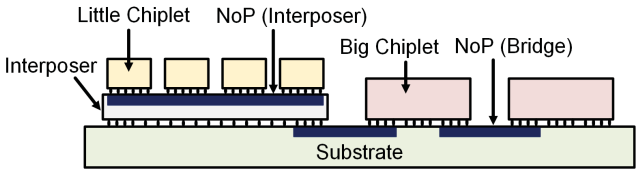

Figure 3: Cross-sectional view of the big-little chiplet-based IMC architecture. The architecture consists of a little chiplet bank with little chiplets (connected by an NoP within the interposer and a big chiplet bank with big chiplets connected by a bridge NoP. NoP properties: 1.5–8mm length, 2–4.5μm pitch, and 0.5–2μm width.

2 RELATED WORK

Chiplet-based architectures are well explored for high performance computing applications [7, 15, 22, 24, 32, 33, 35]. A co-design flow considering architecture, chip, and package for a chiplet-based system is proposed in [7]. A detailed design space exploration with the proposed co-design flow shows significant improvement in power consumption and area with respect to a monolithic design. Vivet et al. [32] proposed a chiplet-based system with 96 computing cores and a 3D memory are distributed over 6 chiplets. Another recent work proposed a 2,048 chiplet (14,336 cores) wafer-scale processor that utilizes a bridge-based integration [22]. The authors discuss the challenges of designing a wafer-scale processor and provide insights into power delivery, clock routing, and testing.

Chiplet-based architectures have proven to be both more energy-efficient and cost-effective than monolithic architectures for complex DNNs. Several prior studies proposed chiplet-based architectures for DNN acceleration [1, 26, 30]. The authors of [26] proposed a fine-grained 36-chiplet architecture for DNN inference acceleration. Each chiplet utilizes a homogeneous structure with 16 PEs that operate using a weight stationary dataflow. The chiplets are connected by a 6×6 NoP mesh that utilize the ground-referenced signaling technique [31]. The authors of [30] proposed a hierarchical and analytical framework, NN-Baton, to analyze DNN mapping and communication overheads in a chiplet-based DNN accelerator. NN-Baton supports different mapping schemes of DNNs onto the chiplets. Furthermore, an analytical model to quantify the communication overhead for NoP is also presented in NN-Baton. A family of chiplet topologies are proposed in [1]. The authors explored different chiplet topologies and compared their performance through an analytical metric which estimates latency. A chiplet-based IMC benchmarking tool for design space exploration, SIAM, is proposed in [9]. SIAM supports different chiplet architectures, IMC crossbar tile structures, NoP, NoC, and DRAM estimation. However, all prior studies assume a homogeneous chiplet structure across all chiplets interconnected by a single NoP. Furthermore, none of the prior works considered the non-uniform distribution of weights and activations in the DNN during the mapping process. Hence, many chiplets remain under-utilized while the large NoP leads to an increased area and energy overhead.

In contrast to prior works, we propose a heterogeneous big-little chiplet-based IMC architecture that combines big chiplet bank with a bridge-based NoP and a little IMC chiplet bank with a interposer-based NoP to enhance the IMC utilization and improve energy

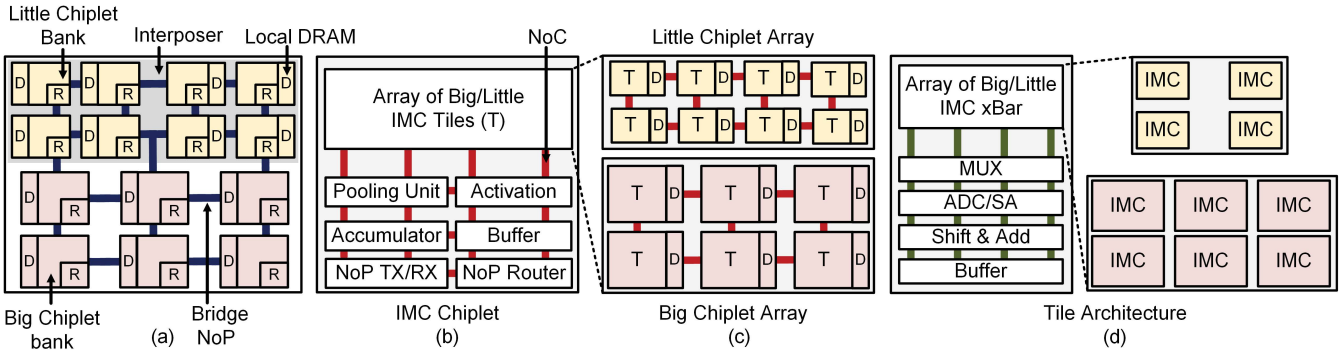

Figure 4: (a) Overview of the big-little chiplet IMC architecture. The little chiplet bank utilizes smaller chiplets connected by a interposer-based NoP while the big chiplet bank utilizes bigger chiplets connected by a bridge-based NoP. Each chiplet utilizes a local DRAM, (b) IMC chiplet architecture (big and little). Each chiplet consists of an array of IMC tiles and a dedicated NoP transceiver and router, (c) The little chiplet bank consists of fewer and smaller tiles while the big chiplet bank consists of more bigger tiles. Both chiplet structures utilize a mesh-based NoC for on-chip communication, and (d) Structure of each tile within the big and little chiplet. It consists of an array of IMC crossbar arrays and associated peripheral circuits with an interconnect similar to that in [25]. The little chiplet consists of fewer and smaller IMC crossbars while the big chiplet has larger and more IMC crossbar arrays.

efficiency. Furthermore, we propose a customized methodology that exploits the non-uniform distribution of DNN weights and activations in mapping the DNNs onto the big-little chiplet IMC architecture. To the best of our knowledge, this is the first heterogeneous chiplet-based IMC architecture that leverages different IMC structures collectively with a heterogeneous NoP coupled with a customized DNN mapping.

3 BIG-LITTLE CHIPLET ARCHITECTURE

Figure 4(a) shows the top-level block diagram of the heterogeneous big-little chiplet IMC architecture. The architecture consists of two banks of IMC chiplets, a little bank (shown in yellow color) and a big bank (shown in light red color). The little IMC chiplet bank consists of chiplets with smaller and fewer IMC crossbar arrays compared to the big chiplets. It is placed on an interposer that houses the NoP. The NoP provides high bandwidth and a compact structure for on-package communication within the little chiplet bank. At the same time, the increased size and count within the big chiplet bank allow for higher computation capability. The big chiplets are directly connected to the substrate using microbumps. A bridge-based NoP is utilized within the big chiplet bank for on-package communication. Long wires of the bridge NoP allow easy integration of the big chiplets. We utilize the Y–X routing methodology for the NoP. Each chiplet (big and little) consists of a local DRAM (DDR4 in this work) that stores the weights required for the IMC crossbar arrays.

Figure 4(b) shows the structure of a IMC chiplet. Each chiplet utilizes a hierarchical structure that consists of an array of big (bottom of Figure 4(c)) or little IMC tiles (top of Figure 4(c)) and each tile consists of an array of IMC crossbars or PEs. In addition, the chiplet contains a pooling unit, non-linear activation unit, accumulator, and buffer. The accumulator is used for the partial sum accumulation across different tiles within the chiplet. Furthermore, the buffers allow for efficient data movement in and out of the chiplet. Each IMC chiplet consists of a dedicated NoP transceiver used for the transmission and reception of packets across the NoP. In this work, we adopt the NoP transceiver from [31]. Each transceiver consists of a local PLL circuit that provides the clock for the transceiver. A five-port router is utilized for routing of the data across the NoP.

Each IMC chiplet utilizes a local DRAM to store the weights. The local DRAM allows for external memory access, thus making our proposed big-little architecture a generic platform. If a DNN does not fit on the entire chip, the DRAM stores all the weights necessary for each chiplet. First, the DRAM loads the necessary weights into the IMC crossbar arrays. Next, while the computation is performed, the DRAM loads the next set of weights of the DNN. The buffer is designed to support a ping-pong operation [16]. The weights from the DRAM are loaded into the first buffer stage (ping) and then moved to the second buffer stage (pong). Therefore, the big-little IMC chiplet architecture masks the DRAM latency with the computation latency, achieving high throughput.

Finally, Figure 4(d) shows the structure of an IMC tile. Each array in the crossbar consists of PEs that perform the computations. In this work, we focus on a resistive random-access-memory (RRAM) based IMC crossbar array due to its superior energy-efficiency [25]. The computations are performed in the analog domain by turning on all wordlines (WL) together and performing accumulation along the bitline (BL). The inputs are given through the WL while the weights are stored within the RRAM cells. Each IMC array consists of specialized peripheral circuitry that assists the computation. The peripheral circuitry includes a column multiplexer (mux), an analog-to-digital converter (ADC), a shift and add circuit, and a buffer. The column mux is used to share the ADC across columns of the IMC array. The ADC converts the MAC output in the analog domain across each column into the digital domain. The big-little IMC architecture does not utilize a digital-to-analog converter (DAC) by employing bit-serial computing. The shift and add circuit handles the positional value of each bit within the multi-bit input activations that are computed using the IMC arrays. The buffers within the tile are utilized for storing the partial sums and the input activations.

4 PARAMETERS OF THE BIG-LITTLE ARCHITECTURE AND MAPPING

The underlying non-uniform distribution of weights and activations within a DNN results in an increased number of activations in the early layers and larger number of weights in the subsequent layers (Figure 1). This non-uniform weight distribution leads to under-utilization of chiplets in the early layers, thus a lower overall IMC

utilization. To improve the IMC utilization, crossbar arrays with smaller size (e.g. 32×32 instead of 128×128) can be used everywhere. However, using smaller crossbar arrays also leads to increasing number of chiplets in the system. In turn, larger number of chiplets in the system increases the area as well as energy consumption (due to higher relative area and energy of the peripheral circuits) masking the benefit of using chiplet-based system. Therefore, a balance between crossbar array size and number of chiplets in the system is necessary. To this end, we propose a technique to optimize the big-little chiplet configuration as discussed next.

4.1 Configuration of the big-little chiplets

We first determine the configuration of big-little chiplets by computing the tile utilization with different big-little chiplet configurations for a given DNN. Algorithm 1 shows our proposed technique to find the utilization. The inputs to the algorithm are

(1) the set of crossbar sizes for the little chiplets ($\mathcal{X}_{\mathcal{L}}$) and the big chiplets ($\mathcal{X}_{\mathcal{B}}$),
(2) set of number of tiles in the little chiplets ($\mathcal{T}_{\mathcal{L}}$) and the big chiplets ($\mathcal{T}_{\mathcal{B}}$),
(3) number of little chiplets ($\mathcal{N}_{\mathcal{L}}$) and big chiplets ($\mathcal{N}_{\mathcal{B}}$),
(4) the DNN structure,
(5) the total number of chiplets in the system.

We note that the initial layers of the DNN are mapped on to little chiplets since there are fewer weights in the initial layers. A DNN layer is mapped on to a chiplet when number of tiles required for that layer is less than the number of remaining tiles in the chiplet, i.e., the available resource on the chiplet is sufficient for the layer (as shown in line 13–17 of Algorithm 1). Once a layer (layer-j) is mapped on to a chiplet, the tile utilization is computed as:

$$IMC_j = \left\lceil \frac{K_j^x \times k_j^y \times N_j^{if}}{x} \right\rceil \times \left\lceil \frac{N_j^{of} \times Q}{x} \right\rceil$$

$$u_j = 100 \times \frac{K_j^x \times k_j^y \times N_j^{if} \times N_j^{of} \times Q}{IMC_j \times x \times x} \quad (1)$$

where K_j^x and K_j^y are the kernel sizes of layer-j, N_j^{if} and N_j^{of} are the number of i/p and o/p features for layer-j, Q is the quantization precision, IMC_j is the number of IMC crossbars required for layer-j and x is the IMC crossbar size ($x \times x$). Once the resources of a chiplet are exhausted, the next chiplet is considered for mapping. This process continues until no chiplet (little/big) is available.

In the proposed method, for each chiplet configuration, we obtain the average utilization for a particular DNN after each layer is mapped(line 36 of Algorithm 1). Then we sort (in descending order) the configurations based on the utilization and save the top K configurations. The above procedure is repeated for M different DNNs and the configuration with highest utilization which is common for all DNNs is considered as the final configuration for the big-little chiplet system. We note that K and M are user-defined parameters and our proposed technique is independent of these parameters.

4.2 Configuration of the big-little NoP

The heterogeneous chiplet configuration (discussed in Section 4.1) improves the overall chiplet utilization by using smaller chiplets that match well to the early layers with fewer weights. However, the initial DNN layers produce higher number of activations compared to later layers. Therefore, the volume of traffic between little chiplets (used for initial DNN layers) is higher than the traffic volume between big chiplets (used for later DNN layers). Hence, the network-on-package (NoP) configuration between little chiplets needs to be different than that of the big chiplets. To this end, we

Algorithm 1: Determining Big-Little Chiplet Configuration

1 **Input:** DNN structure, number of chiplets (N_C), set of crossbars sizes for the little chiplets ($\mathcal{X}_{\mathcal{L}}$) and the big chiplets ($\mathcal{X}_{\mathcal{B}}$); set of number of tiles in the little chiplets ($\mathcal{T}_{\mathcal{L}}$) and the big chiplets ($\mathcal{T}_{\mathcal{B}}$); number of little chiplets ($\mathcal{N}_{\mathcal{L}}$) and number of big chiplets ($\mathcal{N}_{\mathcal{B}}$)
2 **Output:** Tile utilization for each configuration i (U_i)
3 $N_{cfg} \leftarrow$ number of configurations in the set containing all possible combinations of the elements in $\mathcal{X}_{\mathcal{B}}, \mathcal{X}_{\mathcal{L}}, \mathcal{T}_{\mathcal{L}}, \mathcal{T}_{\mathcal{B}}, \mathcal{N}_{\mathcal{L}}, \mathcal{N}_{\mathcal{B}}$
4 $L \leftarrow$ number of DNN layers
5 **for** $i = 1 : N_{cfg}$ **do**
6 n_l = Number of little chiplets in Config-i
7 n_b = Number of little chiplets in Config-i
8 $j \leftarrow 0$ // Number of layers already mapped
9 $U \leftarrow 0$ // Sum of utilization
10 $n_l^u \leftarrow 0$ // Number of little chiplets used
11 **while** $n_l^u \leq n_l$ *and* $j < L$ **do**
12 $j_t \leftarrow$ Number of tiles required for layer-j
13 $r_t^l \leftarrow$ Number of remaining tiles in the little chiplet
14 **if** $j_t < r_t^l$ **then**
15 Map layer-j to the little chiplet
16 $u_j \leftarrow$ Tiles utilization for layer-j (Eq. 1)
17 $U = U + u_j;\ j = j + 1$
18 **end**
19 **else**
20 $n_l^u = n_l^u + 1$
21 **end**
22 **end**
23 $n_b^u \leftarrow 0$ // Number of big chiplets used
24 **while** $n_b^u \leq n_b$ *and* $j < L$ **do**
25 $j_t \leftarrow$ Number of tiles required for layer-j
26 $r_t^b \leftarrow$ Number of remaining tiles in the big chiplet
27 **if** $j_t < r_t^b$ **then**
28 Map layer-j to the big chiplet
29 $u_j \leftarrow$ Tiles utilization for layer-j (Eq. 1)
30 $U = U + u_j;\ j = j + 1$
31 **end**
32 **else**
33 $n_b^u = n_b^u + 1$
34 **end**
35 **end**
36 $U_i = \frac{U}{L}$
37 **end**

Algorithm 2: Determining Big-Little NoP Configuration

Input: DNN structure, number of chiplets (N_C), set of NoP bus widths for the little chiplets ($\mathcal{W}_\mathcal{L}$) and the big chiplets ($\mathcal{W}_\mathcal{B}$); set of NoP frequency for the little chiplets ($\mathcal{F}_\mathcal{L}$) and the big chiplets ($\mathcal{F}_\mathcal{B}$), mapping of layers to the big-little chiplet ($\mathcal{L} \rightarrow C$)
Output: NoP EDP for each configuration-i (E_i)
$N_{cfg} \leftarrow$ number of configurations in the set containing all possible combinations of the elements in $\mathcal{W}_\mathcal{L}$, $\mathcal{W}_\mathcal{B}$, $\mathcal{F}_\mathcal{L}$, $\mathcal{F}_\mathcal{B}$
$L \leftarrow$ number of DNN layers
n_l = Number of little chiplets
n_b = Number of big chiplets
for $i = 1 : N_{cfg}$ **do**
    w_l = Bus-width of little chiplets in Config-i
    w_b = Bus-width of big chiplets in Config-i
    f_l = NoP frequency of little chiplets in Config-i
    f_b = NoP frequency of big chiplets in Config-i
    $E_i \leftarrow 0$ // Initializing EDP of Config-i
    for $j = 1 : n_l$ **do**
        Compute edp_j by from Equation 2
        $E_i = E_i + edp_j$ // Communication EDP
    end
    for $k = 1 : (n_b - 1)$ **do**
        Compute edp_k from Equation 2
        $E_i = E_i + edp_k$ // Communication EDP
    end
end

propose a technique to determine optimal NoP configuration for a system with big-little chiplet targeted for a particular DNN. Algorithm 2 shows the technique to determine NoP configuration for a particular DNN. The inputs to the algorithm are:

(1) big-little chiplet configuration obtained from Algorithm 1,
(2) set of NoP bus width for the little chiplets ($\mathcal{W}_\mathcal{L}$) and the big chiplets ($\mathcal{W}_\mathcal{B}$),
(3) set of NoP frequency for the little chiplets ($\mathcal{F}_\mathcal{L}$) and the big chiplets ($\mathcal{F}_\mathcal{B}$),
(4) the DNN structure.

We evaluate the energy-delay product of communication for each NoP configuration in the set of configurations. An analytical expression based evaluation is incorporated to perform fast exploration in the NoP configuration space. First, we evaluate communication volume of each NoP configuration given a particular DNN. The communication volume is equivalent to the number of packets transferred between two chiplets, and the number of packets (P) is expressed as $P = \frac{b}{w}$, where b is the number of bits to be communicated and w is the NoP bus width. We divide the number of packets by NoP frequency (f) to obtain an approximation of NoP latency $d = \frac{P}{f} = \frac{b}{w \times f}$. Next, we compute NoP power consumption by assuming that it is proportional to cube of NoP frequency [21]; $p = f^3$. Then the approximate energy consumption (e) is computed by multiplying communication latency and communication power;

Algorithm 3: Mapping DNN Layers to Big-Little Chiplets

Input: DNN layers ($\mathcal{L}$), IMC crossbar size in Big chiplet (x_b), IMC crossbar size in little chiplet (x_l), number of tiles in big chiplets (t_b), number of tiles in little chiplets (t_l), number of available big chiplets (n_b), number of available little chiplets (n_l)
Output: Mapping of layers to of big-little chiplet ($\mathcal{L} \rightarrow C$)
Compute S_B by following Equation 3
Compute Pr by following Equation 4
for $j = 1 : Pr$ **do**
    $\mathcal{L}_j \rightarrow$ DNN layers for partition-j; $\mathcal{L}_j \in \mathcal{L}$
    for $i = 1 : |\mathcal{L}_j|$ **do**
        $a_L \rightarrow 1$ // Number of little chiplets used
        Compute utilization of i$^{\text{th}}$ (U_B^i) layer on a big chiplet using x_b, t_b
        Compute utilization of i$^{\text{th}}$ (U_L^i) layer on a little chiplet using x_l, t_l
        if $((U_B^i < U_L^i) \& (a_L \leq A_L))$ **then**
            Map i$^{\text{th}}$ layer to little chiplet.
            if *Resource in* a_L *is exhausted* **then**
                $a_L \rightarrow a_L + 1$
            end
        end
        else
            Compute number of big chiplets (a_B) required to map layer-i – layer-$|\mathcal{L}_|$|
            assert($(a_B \leq A_B)$, 'Error')
            for $k = i : |\mathcal{L}_j|$ **do**
                Map k$^{\text{th}}$ layer to big chiplet.
            end
            break;
        end
    end
end

$e = d \times p$. Finally, communication EDP between each pair of chiplet (edp) is computed as:

$$edp = e \times d = d \times p \times d = d^2 \times f^3 = \left(\frac{b}{w \times f}\right)^2 \times f^3 = \frac{b^2 \times f}{w^2} \quad (2)$$

The total communication EDP for each NoP configuration for a particular DNN is obtained by adding the communication EDP between each pair of chiplets. A total of K NoP configurations with lower EDP are saved and the above procedure is repeated for M different DNNs. The configuration with lowest cost which is common for all DNNs is considered as the final NoP configuration for the big-little chiplet system. Similar to the technique of selecting big-little chiplet configuration (described in Section 4.1), K and M are the user defined parameter and our proposed technique is independent of these parameters.

4.3 Mapping a Previously Unseen DNN to a System on big-little Chiplets

So far, we described our proposed technique to determine the optimal configuration of big-little chiplet and the NoP. The optimal configuration is determined by performing design space exploration with several DNNs. However, an unknown DNN (not seen before) may be encountered at runtime. Moreover, there is no guarantee that all the weights of a given DNN will fit in the on-chiplet resources since the number of DNN parameters seem to be continuously growing. In these cases, we need to divide the entire DNN into multiple parts and load the weights of each part from DRAM before executing. Algorithm 3 shows the DNN partitioning as well as the mapping technique. The input to the algorithm is the DNN structure, big-little chiplet configuration and big-little NoP configuration. First, we compute the number of in-memory computing bits available on the system (S_B). Specifically, for each type (little/big) of chiplets, we multiply the number of available chiplets (n_l/n_b), the number of tiles in each chiplet (t_l/t_b), the number of crossbar array in each tile (16), and the size of IMC crossbar array for big and little chiplets (x_l/x_b). Then we add the product for big and little chiplets to obtain the total number of in-memory computing bits available on the system (S_B):

$$S_B = (n_l \times t_l \times 16 \times x_l \times x_l) + (n_b \times t_b \times 16 \times x_b \times x_b) \quad (3)$$

Next, we compute the number of bits required to store all the weights of the DNN (D_B). Assuming average utilization of $u(0 < u \leq 1)$, the total number of partitions (Pr) required for the DNN is computed by taking the ceiling of the quotient obtained by dividing the required number of bits to store all weights (D_B) by the available number of in-memory bits on the system (S_B):

$$Pr = \left\lceil \frac{D_B}{S_B \times u} \right\rceil \quad (4)$$

For each partition, first, we compute the utilization of i^{th} layer on a big chiplet (U_B^i) as well as on a little chiplet (U_L^i). We compute U_B^i and U_L^i using Equation 1. If the big chiplet utilization ((U_B^i)) is less than the little chiplet utilization (U_L^i) and the little chiplet bank is not exhausted, then the layer is mapped onto a little chiplet, as shown in lines 7–12 of Algorithm 3. Otherwise, we compute the number of big chiplets required (a_B) to map the rest of the layers. If a_B is less than or equal to the number of available big chiplets (A_B), then we map the rest of the layers to the big chiplet bank, else the algorithm throws an error since the resource requirement exceeds the available capacity (shown in line 18–23 of Algorithm 3). Thus, we ensure that the initial layers with fewer weights are mapped into little chiplets and the latter layers with higher number of weights are mapped onto big chiplets with more computation resources. Therefore, our proposed custom mapping of the DNN onto the big-little chiplet architecture ensures high IMC utilization.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

Evaluation platform: To evaluate the proposed heterogeneous big-little IMC chiplet architecture, we use a customized version of the open-sourced tool SIAM [9]. The customization includes the addition of the custom mapping scheme detailed in Section 4.

Table 1: Set of configurations considered to determine big-little chiplet and NoP structure.

Chiplet Configuration		NoP Configuration	
Parameter	Values in the Set	Parameter	Values in the Set
$\mathcal{X}_\mathcal{L}$	{32, 64}	$\mathcal{W}_\mathcal{L}$	{16, 32, 64}
$\mathcal{X}_\mathcal{B}$	{128, 256, 512}	$\mathcal{W}_\mathcal{B}$	{4, 8, 12, 16, 20, 24}
$\mathcal{T}_\mathcal{L}$	{9, 16, 25}	$\mathcal{F}_\mathcal{L}$	{600, 1000, 1400, 1800} MHz
$\mathcal{T}_\mathcal{B}$	{36, 49}	$\mathcal{F}_\mathcal{B}$	{600, 800, 1000} MHz

In addition, we handle the big-little chiplet IMC architecture by adding the number of each type (big/little) of chiplets, the number of tiles inside big and little chiplets, and the big-little IMC structure. Furthermore, we also assume that each type of chiplet can use different NoP width. The simulator performs the mapping of a given DNN onto the big-little IMC chiplet architecture. The outputs include area, energy, latency, throughput, energy efficiency, and IMC utilization (for all individual components in the architecture). Finally, we add support for intermediate DRAM access (DDR4 [20]) for each chiplet to handle the case where all weights do not fit on the system at once. We plan to open-source the tool and optimization methodology upon acceptance of the paper.

DNN algorithms and architectural parameters: We evaluate the proposed heterogeneous chiplet architecture with DenseNet-40 (0.26M) on CIFAR-10, ResNet-110 (1.7M) on CIFAR-10, VGG-19 (45.6M) on CIFAR-100, ResNet-34 (21.5M) and ResNet-50 (23M) on ImageNet. We utilize an RRAM-based IMC structure for DNN inference with the following parameters: one bit per RRAM cell, a R_{off}/R_{on} ratio of 100, ADC resolution of 4-bits with 8 columns multiplexed, operating frequency of 1GHz [6, 25], and a parallel read-out method. We use 8-bit quantization for the weights and activations, and a 32nm CMOS technology node. The chiplets are placed to achieve the least Manhattan distance. The NoP parameters include E_{bit} of 0.54pJ/bit [31], interconnect parameters width, length, and pitch for the interposer-based NoP from [31] and for bridge-based NoP from [3] (Figure 3), per lane NoP TX/RX area of 5,304 μm^2, and NoP clocking circuit area of 10,609 μm^2 [23]. In addition, we also model the μbump for both the interposer [29] and bridge-based [14] NoP by utilizing the PTM models [27].

5.2 Big-Little IMC Structure and NoP

This section demonstrates the parameters related to big-little IMC structure and big-little NoP. Specifically, we consider four DNNs (mentioned in Section 5.1) and execute Algorithm 1 to determine the top 10 (K=10) configurations with highest utilization for each DNN. We consider a system with 36 chiplets to limit the total area and power consumption of the system. Table 1 shows the input parameters ($\mathcal{X}_\mathcal{L}, \mathcal{X}_\mathcal{B}, \mathcal{T}_\mathcal{L}, \mathcal{T}_\mathcal{B}$) to the algorithm. We vary the number of little chiplets from 1 to 35 while maintaining the total number of chiplets to be 36. Then, we choose the best configuration which is common for all four DNNs. We observe that a system with *25 little chiplets* with a *64×64 IMC crossbar* and *25 tiles per chiplet*, and *11 big chiplets* with a *256×256* IMC crossbar and *36 tiles per chiplet* provides best utilization across all four DNNs. Figure 5 shows the utilization for a system with 25 little and 11 big chiplets with varying size of crossbars (both for big and little chiplet) for all four

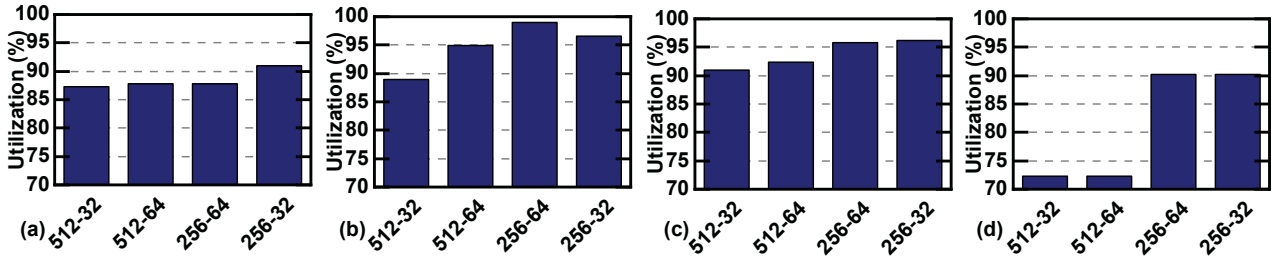

Figure 5: IMC utilizations for different DNNs across different big-little chiplet-based RRAM IMC configurations for (a) ResNet-110, (b) ResNet-34, (c) VGG-19, (d) DenseNet-40. Based on the utilization, we choose crossbar size of big chiplet as 256×256 and crossbar size of little chiplet as 64×64 (256–64).

Table 2: Performance comparison of each component of a homogeneous (Little only, Big only) chiplet architecture and the heterogeneous Big-Little IMC chiplet architecture for VGG-19 on CIFAR-100.

Configuration	Area					Energy					Latency				
	IMC (%)	NoP (%)	NoC (%)	Total (mm^2)	Normalized to big-little (×)	IMC (%)	NoP (%)	NoC (%)	Total (mJ)	Normalized to big-little (×)	IMC (%)	NoP (%)	NoC (%)	Total (ms)	Normalized to big-little (×)
Little only	11.9	88.0	0.1	952.1	10.9	99.7	0.2	0.1	1.3	4.1	99.7	0.1	0.2	1.6	1.3
Big only	44.0	55.5	0.5	597.2	6.8	78.6	11.0	10.4	0.43	1.3	99.6	0.1	0.3	3.2	2.7
Big-Little (this work)	**52.4**	**47.4**	**0.2**	**87.4**	**1.0**	**99.8**	**0.1**	**0.1**	**0.32**	**1.0**	**99.2**	**0.3**	**0.5**	**1.2**	**1.0**

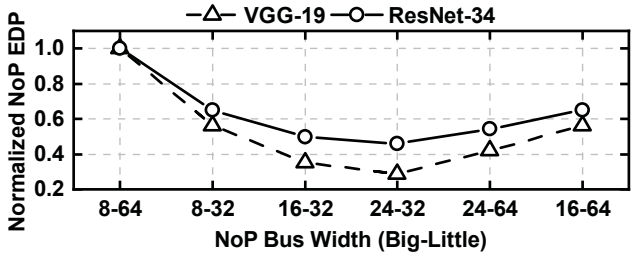

Figure 6: Normalized NoP EDP for different bus-widths for VGG-19 and ResNet-34. The NoP with bus width of 24 for big and 32 for little chiplets (24–32) shows lowest EDP.

DNNs. In this case, we also fixed the number of tiles per chiplet to 25 for the little chiplets and 36 for the big chiplets. Figure 5 reveals that the configuration where the crossbar size of the big chiplets is 256×256 and the crossbar size of the little chiplets is 64×64 (256–64, 256 denotes crossbar size of big chiplets and 64 denotes crossbar size of little chiplets) shows higher utilization than other configurations for three out of four DNNs. Only in the case of ResNet-110, the configuration 256–32 shows higher utilization than 256–64. However, we choose 256–64 over 256–32 since it provides more on-chip resources, lower area and energy efficiency for the IMC crossbar array (due to peripheral circuits).

Similarly, we execute Algorithm 2 for four DNNs to obtain the NoP configuration. Table 1 shows the set of different NoP parameters ($\mathcal{W}_{\mathcal{L}}, \mathcal{W}_{\mathcal{B}}, \mathcal{F}_{\mathcal{L}}, \mathcal{F}_{\mathcal{B}}$ used as inputs to Algorithm 2). The parameters are adopted from [2]. EDP for NoP is obtained for all NoP configurations for the four DNNs. Then, the NoP configuration having the lowest EDP for all four DNNs is chosen. Based on the EDP results, the big NoP frequency and the little NoP frequency is set to 600 MHz and 1 GHz, respectively; the big NoP bus width and the little NoP bus width is set to 24 and 32, respectively. Figure 6 shows the normalized NoP EDP for different combination of bus width for big and little chiplets. For illustration purpose, we show VGG-19 and ResNet-34 since these two DNNs utilize more than 34 out of 36 chiplets. From Figure 6, it is observed that the configuration with big NoP bus width of 24 and little NoP bus width of 32 shows the lowest EDP. Since little chiplets produce higher number of activations than the big chiplets, it is intuitive that little NoP are wider (larger bus width) than the big NoP.

5.3 Comparison with Baseline Architectures with Homogeneous Chiplets

We compare the performance of our proposed big-little chiplet architecture with respect to two baseline architectures with homogeneous chiplets [9]. **1) Little only:** In this configuration, we consider a system where the configuration of all chiplets as well as the NoP is same as that of the little chiplets. **2) Big only:** In this configuration, we consider a system where the configuration of all chiplets as well as the NoP is same as that of the big chiplets. We note that, the total number of chiplets with 'Little only' and 'Big only' configurations vary for different DNNs.

Table 2 shows the performance comparison for 'Little only', 'Big only' and the proposed big-little architectures for VGG-19 on CIFAR-100. In this table, the performance of each component of the architecture, i.e. IMC, NoP and NoC is shown. Our proposed big-little chiplet architecture results in a balanced distribution of the area among the circuit and NoP components, while the NoC accounts for a minimal portion (0.2%) of the total area. In 'Little-only' architecture, NoP becomes the bottleneck for area since the chiplets have smaller size, hence more number of chiplets are required which increases the NoP. In 'Big only' architecture, NoP consumes more energy due to higher volume of data movement between each pair of chiplets. In contrast, the proposed big-little architecture with its high IMC utilization and reduced on-chip communication as well as on-package data movement results in less total energy consumption and less inference latency. Overall, the proposed heterogeneous big-little architecture achieves up to 10.9× lower area, 4.1× lower energy, and 2.7× lower latency than 'Little only' and 'Big only' architectures.

Next, we compare the IMC utilization and the performance (area, energy and latency) for ResNet-110, VGG-19, DenseNet-40, and ResNet-34 against 'little only' and 'big only' architecture. For VGG-19, our proposed big-little architecture achieves the highest IMC

Table 3: Performance comparison of a homogeneous (Little only, Big only) chiplet architecture and the heterogeneous Big-Little IMC chiplet architecture for different DNNs.

Configuration	Utilization (%)				Area (mm^2)				Energy (mJ)				Latency (ms)			
	Res-110	VGG-19	Dense-40	Res-34	Res-110	VGG-19	Dense-40	Res-34	Res-110	VGG-19	Dense-40	Res-34	Res-110	VGG-19	Dense-40	Res-34
Little only	69	92	58	93	171.7	952.1	71.5	657.8	1.4	1.3	0.22	41.1	23.0	1.6	1.6	13.1
Big only	44	59	32	82	220.0	597.2	220.2	595.9	0.28	0.43	0.11	3.7	1.1	3.2	0.02	20.2
Big-Little (this work)	**88**	**93**	**90**	**98**	**87.4**	**87.4**	**87.4**	**87.4**	**0.18**	**0.32**	**0.06**	**8.2**	**1.1**	**1.2**	**0.03**	**48.6**

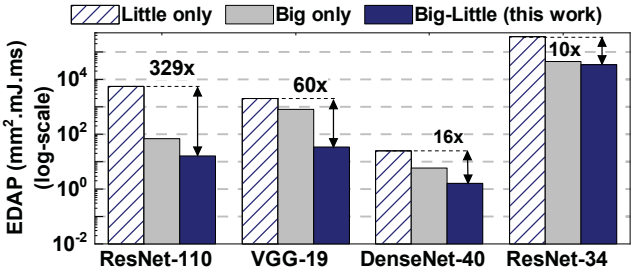

Figure 7: EDAP comparison (log-scale) of the big-little chiplet-based RRAM IMC architecture to 'Little only' and 'Big only' chiplet-based RRAM IMC architectures. The big-little architecture achieves up to 329× improvement compared to 'Little only' architecture.

utilization of 93% compared to 92% and 59% for 'Little only' and 'Big only', respectively. Similarly, the big-little architecture achieves 88%, 90%, and 98% IMC utilization for ResNet-110, DenseNet-40, and ResNet-34, respectively, up to 2.8× greater than 'Little only' and 'Big only' architectures. We observe that the big-little architecture provides *up to 7.8× improvement in energy and up to 21× improvement* in inference latency with respect to baseline homogeneous architectures. 'Big only' architecture consumes less energy and less latency than big-little architecture for ResNet-34, but in this case, the area of 'Big only' is 6.8× higher than big-little architecture. To better analyze the performance comparison, we plot the energy-delay-area product (EDAP) for all DNNs, as shown in Figure 7. The big-little chiplet architecture provides up to 329× lower EDAP than the 'Little only' and 'Big only' architectures across all four DNNs. Although 'Big only' architecture shows improvement in energy consumption and inference latency with respect to big-little for ResNet-34, the EDAP with 'Big only' is 1.3× higher than big-little architecture in this case. Hence, the proposed big-little IMC architecture achieves optimal performance through reduced EDAP at higher IMC utilization across different DNNs.

5.4 Results with DRAM (DDR4)

In this section, we show the performance results when the resource on a big-little chiplet-based system is not sufficient to store all the weights of a given DNN. In that case, the DNN is divided into multiple partitions. One partition is mapped on to the big-little chiplets at a time. While the computations of a partition of the DNN are performed, the weights corresponding to the next partition are loaded from the DRAM into the ping-pong buffer. The additional DRAM accesses result in increased energy. At the same time, the impact on latency is reduced through the ping-pong buffers [16]. Table 4 shows the ratio between DRAM energy and compute energy for VGG-16 and VGG-19 with systems having different number of chiplets. We observe that, the ratio of DRAM energy to computation energy increases with reduction in the system sizes for both the DNNs. With decreasing system size, more weights need to be stored and loaded from DRAM, thereby increasing DRAM energy.

Table 4: Ratio between DRAM energy and compute energy for VGG-16 and VGG-19 with systems having different number of chiplets (All weights of VGG-19 fit on chip with this configuration, significantly reducing the DRAM energy).**

# Chiplets	VGG-16		VGG-19	
	#partitions	Ratio	#partitions	Ratio
36	2	1.1	1	0.08**
25	2	2.1	2	131
16	3	3.6	2	161

Table 5: Comparison with other platforms for ResNet-50 on ImageNet (*reported in [26]).

Platform	Area (mm^2)	Energy Efficiency (Images/s/W)
Nvidia V100 GPU*	815	8.3
Nvidia T4 GPU*	525	15.5
SIMBA [26]	215	45
Big-Little (this work)	**85**	**827**

5.5 Comparison with State-of-the-art Work

Table 5 shows the comparison of the proposed heterogeneous big-little RRAM IMC chiplet architecture with an Nvidia T4 and V100 GPU, and SIMBA [26]. The big-little chiplet architecture achieves a lower area for the architecture due to the custom RRAM-based IMC and the optimized NoP structure. Compared to the Nvidia V100, Nvidia T4, and SIMBA architecture, the big-little IMC architecture achieves 9.6×, 6.2×, and 2.5× area improvement and 99.6×, 53.4×, and 18.4× energy-efficiency improvement, respectively. The improved energy efficiency is attributed to the higher IMC utilization, analog computation within the RRAM-based IMC, reduced NoP data movement and bus width, and the absence of intermediate DRAM transactions for weights and partial sums.

6 CONCLUSION

This paper proposed a heterogeneous big-little chiplet-based IMC architecture driven by the non-uniform nature of DNN layers. To the best of our knowledge, this is the first heterogeneous chiplet-based IMC architecture that leverages different IMC compute structures coupled with a heterogeneous NoP. We show that mapping the early layers to the little chiplet bank and the subsequent layers to the big chiplet bank, achieve up to 2.8× higher IMC utilization and up to 329× improvement in energy-delay-area product compared to homogeneous chiplet IMC architectures. Experimental evaluation of the proposed big-little chiplet-based RRAM IMC architecture for ResNet-50 on ImageNet shows 18.4× energy-efficiency improvement compared to SOTA chiplet-based architecture SIMBA.

REFERENCES

[1] Jieming Bharadwaj, Srikant Yin, Bradford Beckmann, and Tushar Krishna. 2020. Kite: A Family of Heterogeneous Interposer Topologies Enabled via Accurate Interconnect Modeling. In *ACM/IEEE DAC*.

[2] CHIPS Alliance (INTEL). 2021. EMIB PHY RTL. https://github.com/chipsalliance/aib-phy-hardware. [Online; Accessed 20-April-2022].

[3] David Greenhill et al. 2017. 3.3 A 14nm 1GHz FPGA with 2.5 D Transceiver Integration. In *2017 IEEE ISSCC*. IEEE.

[4] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for Mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1314–1324.

[5] Ranggi Hwang, Taehun Kim, Youngeun Kwon, and Minsoo Rhu. 2020. Centaur: A Chiplet-based, Hybrid Sparse-Dense Accelerator for Personalized Recommendations. In *2020 ACM/IEEE 47th Annual ISCA*. IEEE, 968–981.

[6] Saransh Imani, Mohsen Gupta, Yeseong Kim, and Tajana Rosing. 2019. FloatPIM: In-memory Acceleration of Deep Neural Network Training with High Precision. In *ACM/IEEE ISCA*.

[7] Gauthaman Kim, Jinwoo Murali, Heechun Park, Eric Qin, Hyoukjun Kwon, Venkata Chaitanya Krishna Chekuri, Nael Mizanur Rahman, Nihar Dasari, Arvind Singh, Minah Lee, et al. 2020. Architecture, Chip, and Package Codesign Flow for Interposer-Based 2.5D Chiplet Integration Enabling Heterogeneous IP Reuse. *IEEE TVLSI* (2020).

[8] Gokul Krishnan, Sumit K Mandal, Chaitali Chakrabarti, Jae sun Seo, Umit Y Ogras, and Yu Cao. 2020. Interconnect-aware Area and Energy Optimization for In-Memory Acceleration of DNNs. *IEEE Design & Test* 37, 6 (2020), 79–87.

[9] Gokul Krishnan, Sumit K Mandal, Manvitha Pannala, Chaitali Chakrabarti, Jae-Sun Seo, Umit Y Ogras, and Yu Cao. 2021. SIAM: Chiplet-based Scalable In-Memory Acceleration with Mesh for Deep Neural Networks. *ACM TECS* (2021).

[10] Liangzhen Kwon, Hyoukjun Lai, Michael Pellauer, Tushar Krishna, Yu-Hsin Chen, and Vikas Chandra. 2021. Heterogeneous Dataflow Accelerators for Multi-DNN Workloads. In *IEEE HPCA*.

[11] Yuan Li, Ahmed Louri, and Avinash Karanth. 2021. Scaling Deep-Learning Inference with Chiplet-based Architecture and Photonic Interconnects. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 931–936.

[12] Yuan Li, Ahmed Louri, and Avinash Karanth. 2022. SPACX: Silicon photonics-based scalable chiplet accelerator for DNN inference. In *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.* 1–13.

[13] Yuan Li, Ke Wang, Hao Zheng, Ahmed Louri, and Avinash Karanth. 2022. Ascend: A Scalable and Energy-Efficient Deep Neural Network Accelerator With Photonic Interconnects. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2022).

[14] Chester Liu, Jacob Botimer, and Zhengya Zhang. 2021. A 256Gb/s/mm-shoreline AIB-Compatible 16nm FinFET CMOS Chiplet for 2.5 D Integration with Stratix 10 FPGA on EMIB and Tiling on Silicon Interposer. In *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 1–2.

[15] Leila Ma, Yenai Delshadtehrani, Cansu Demirkiran, José L Abellán, and Aiav Joshi. 2021. TAP-2.5 D: A Thermally-Aware Chiplet Placement Methodology for 2.5 D Systems. In *IEEE DATE*.

[16] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. 2017. Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Monterey, California, USA) *(FPGA '17)*. ACM, New York, NY, USA, 45–54. https://doi.org/10.1145/3020078.3021736

[17] Robert Mahajan, Ravi Sankman, Neha Patel, Dae-Woo Kim, Kemal Aygun, Zhiguo Qian, Yidnekachew Mekonnen, Islam Salama, Sujit Sharan, Deepti Iyengar, et al. 2016. Embedded Multi-Die Interconnect Bridge (EMIB)–A High Density, High Bandwidth Packaging Interconnect. In *IEEE ECTC*.

[18] Sumit K Mandal, Gokul Krishnan, Chaitali Chakrabarti, Jae-Sun Seo, Yu Cao, and Umit Y Ogras. 2020. A Latency-Optimized Reconfigurable NoC for In-Memory Acceleration of DNNs. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, 3 (2020), 362–375.

[19] Sumit K Mandal, Gokul Krishnan, A Alper Goksoy, Gopikrishnan Ravindran Nair, Yu Cao, and Umit Y Ogras. 2022. COIN: Communication-Aware In-Memory Acceleration for Graph Convolutional Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* (2022).

[20] MICRON. 2014. Datasheet for DDR4 Model. https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/4gb_ddr4_dram_2e0d.pdf Accessed 29 Mar. 2021.

[21] Trevor Mudge. 2001. Power: A First-class Architectural Design Constraint. *Computer* 34, 4 (2001), 52–58.

[22] Jingyang Pal, Saptadeep Liu, Irina Alam, Nicholas Cebry, Haris Suhail, Shi Bu, Subramanian S Iyer, Sudhakar Pamarti, Rakesh Kumar, and Puneet Gupta. 2021. Designing a 2048-Chiplet, 14336-Core Waferscale Processor. In *ACM/IEEE DAC*.

[23] William J Poulton, John W Dally, Xi Chen, John G Eyles, Thomas H Greer, Stephen G Tell, and C Thomas Gray. 2013. A 0.54 pJ/b 20Gb/s Ground-Referenced Single-Ended Short-Haul Serial Link in 28nm CMOS for Advanced Packaging Applications. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*.

[24] Uneeb Rathore, Sumeet Singh Nagi, Subramanian Iyer, and Dejan Marković. 2022. A 16nm 785GMACs/J 784-Core Digital Signal Processor Array With a Multilayer Switch Box Interconnect, Assembled as a 2× 2 Dielet with 10μm-Pitch Inter-Dielet I/O for Runtime Multi-Program Reconfiguration. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 52–54.

[25] Anirban Shafiee, Ali Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with in-situ Analog Arithmetic in Crossbars. *ACM/IEEE ISCA* (2016).

[26] Jason Shao, Yakun Sophia Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, et al. 2019. SIMBA: Scaling Deep-Learning Inference with Multi-Chip-Module-based Architecture. In *IEEE/ACM MICRO*.

[27] Saurabh Sinha, Greg Yeric, Vikas Chandra, Brian Cline, and Yu Cao. 2012. Exploring Sub-20nm FinFET Design with Predictive Technology Models. In *DAC 2012*. IEEE, 283–288.

[28] Xuehai Song, Linghao Qian, Hai Li, and Yiran Chen. 2017. Pipelayer: A Pipelined Reram-based Accelerator for Deep Learning. In *IEEE HPCA*. 541–552.

[29] Michael Su, Bryan Black, Yu-Hsiang Hsiao, Chien-Lin Changchien, Chang-Chi Lee, and Hung-Jen Chang. 2016. 2.5 D IC micro-bump materials characterization and IMCs evolution under reliability stress conditions. In *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*. IEEE, 322–328.

[30] Hongyu Tan, Zhanhong Cai, Runpei Dong, and Kaisheng Ma. 2021. NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators. In *ACM/IEEE ISCA*.

[31] John W Turner, Walker J Poulton, John M Wilson, Xi Chen, Stephen G Tell, Matthew Fojtik, Thomas H Greer, Brian Zimmer, Sanquan Song, Nikola Nedovic, et al. 2018. Ground-Referenced Signaling for Intra-Chip and Short-Reach Chip-to-Chip Interconnects. In *IEEE CICC*.

[32] Eric Vivet, Pascal Guthmuller, Yvain Thonnart, Gael Pillonnet, César Fuguet, Ivan Miro-Panades, Guillaume Moritz, Jean Durupt, Christian Bernard, Didier Varreau, et al. 2020. IntAct: A 96-core Processor with Six Chiplets 3D-stacked on an Active Interposer with Distributed Interconnects and Integrated Power Management. *IEEE JSSC* (2020).

[33] Mengdi Wang, Ying Wang, Cheng Liu, and Lei Zhang. 2021. Network-on-Interposer Design for Agile Neural-Network Processor Chip Customization. In *ACM/IEEE DAC*.

[34] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. 2019. Exploring Randomly Wired Neural Networks for Image Recognition. In *IEEE/CVF ICCV*.

[35] Hao Zheng, Ke Wang, and Ahmed Louri. 2020. A Versatile and Flexible Chiplet-based System Design for Heterogeneous Manycore Architectures. In *ACM/IEEE DAC*.