CHEF: A Framework for Deploying Heterogeneous Models on Clusters With Heterogeneous FPGAs

Yue Tang, Yukai Song, Naveena Elango, *Member, IEEE*, Sheena Ratnam Priya, *Member, IEEE*, Alex K. Jones, *Fellow, IEEE*, Jinjun Xiong, *Fellow, IEEE*, Peipei Zhou, *Senior Member, IEEE*, and Jingtong Hu, *Senior Member, IEEE*

***Abstract*—Deep neural networks (DNNs) are rapidly evolving from streamlined single-modality single-task (SMST) to multimodality multitask (MMMT) with large variations for different layers and complex data dependencies among layers. To support such models, hardware systems also evolved to be heterogeneous. The heterogeneous system comes from the prevailing trend to integrate diverse accelerators into the system for lower latency. FPGAs have high-computation density and communication bandwidth and are configurable to be deployed with different designs of accelerators, which are widely used for various machine-learning applications. However, scaling from SMST to MMMT on heterogeneous FPGAs is challenging since MMMT has much larger layer variations, a massive number of layers, and complex data dependency among different backbones. Previous mapping algorithms are either inefficient or over-simplified which makes them impractical in general scenarios. In this work, we propose CHEF to enable efficient implementation of MMMT models in realistic heterogeneous FPGA clusters, i.e., deploying heterogeneous accelerators on heterogeneous FPGAs (A2F) and mapping the heterogeneous DNNs on the deployed heterogeneous accelerators (M2A). We propose CHEF-A2F, a two-stage accelerators-to-FPGAs deployment approach to co-optimize hardware deployment and accelerator mapping. In addition, we propose CHEF-M2A, which can support general and practical cases compared to previous mapping algorithms. To the best of our knowledge, this is the first attempt to implement MMMT models in real heterogeneous FPGA clusters. Experimental results show that the latency obtained with CHEF is near-optimal while the search time is 10 000× less than exhaustively searching the optimal solution.**

***Index Terms*—Heterogeneous FPGA clusters, multimodality multitask (MMMT).**

Manuscript received 1 August 2024; accepted 1 August 2024. Date of current version 6 November 2024. This work was supported in part by NSF under Award 2213701, Award 2217003, Award 2133267, Award 2122320, Award 2324864, Award 2324937, Award 2328972, Award 2235364, and Award 2229873; and in part by NIH under Award R01EB033387. This article was presented at the International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS) 2024 and appeared as part of the ESWEEK-TCAD Special Issue. This article was recommended by Associate Editor S. Dailey. *(Corresponding author: Yue Tang.)*

Yue Tang, Yukai Song, and Jingtong Hu are with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: yut51@pitt.edu; yus190@pitt.edu; jthu@pitt.edu).

Naveena Elango, Sheena Ratnam Priya, and Jinjun Xiong are with the Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260 USA (e-mail: naveenae@buffalo.edu; sheenara@buffalo.edu; jinjun@buffalo.edu).

Alex K. Jones is with the Department of Electrical Engineering and Computer Science Department, Syracuse University, Syracuse, NY 13244 USA (e-mail: akj@syr.edu).

Peipei Zhou is with the School of Engineering, Brown University, Providence, RI 02912 USA (e-mail: peipei_zhou@brown.edu).

Digital Object Identifier 10.1109/TCAD.2024.3438994

I. Introduction

DEEP neural networks (DNNs) are increasingly used in complex machine learning applications, requiring diverse models and advanced hardware to meet new challenges [1]. On one hand, DNNs are rapidly evolving from simple, single-task systems to more complex, multitask systems, especially in fields like robotics [2], human–computer interactions [3], [4], virtual reality (VR)/augmented reality (AR) [5], [6], etc. Fig. 1(a) shows an example of an multimodality multitask (MMMT) model with three modality nets fusing at the end. The circle 1.1 represents the first layer of the first modality. As shown in Fig. 1(a), such MMMT models involve complex interblock connections between multiple backbones of different sizes [1], [7]. On the other hand, heterogeneous hardware acceleration components are increasingly integrated into state-of-the-art (SOTA) systems. FPGAs, known for their high-computing power and high flexibility, have been widely used for various machine-learning applications both at the edge level and at the cloud level [8], [9], [10], [11], [12], [13], [14]. For example, VMSS [12], an edge server composed of Xilinx U50+U30 FPGAs is proposed to build efficient video analytics in smart cities. Compared to other platforms, such as GPUs, TPUs, etc., VMSS can be reconfigured to satisfy codecs, streaming protocols, specialized DNNs, and other smart application needs efficiently. At the cloud level, UIUC XACC [13] has been designed to support high-performance computing, machine learning, and genomics applications equipped with modern FPGAs. However, while deploying single-modality single-task (SMST) DNNs on such multiaccelerator clusters has been well studied, scaling them into MMMT DNN applications has not been comprehensively investigated.

Compared with SMST, MMMT is more complex, with varied layers, a massive number of layers, and intricate data dependencies, presenting new challenges in accelerator design. First, MMMT models have much larger variations in terms of layer type and layer shape. For example, VFS [16], a typical MMMT model, involves convolutional (Conv) layers, and fully connected (FC) layers, and contains VGG and VD-CNN backbones. The input size of the VGG backbone is $3 \times 224 \times 224$, while the input size of the VD-CNN backbone is $64 \times 1014 \times 4$. When calculating the computation-to-communication (CTC) ratio of all Conv layers on a monolithic accelerator on the Xilinx U280 FPGA, the CTC ratio for VGG ranges from 48 to 448, while the CTC ratio of Conv layers in VD-CNN

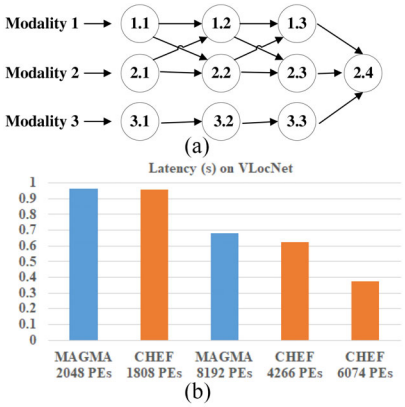

Fig. 1. (a) Abstracted MMMT model with three modalities. A circle represents a layer of a modality net and an arrow represents data dependency between two layers. The index *a.b* in the circle represents the *b*th layer of the *a*th modality. MMMT models include complex interblock connections between multiple backbones. (b) Latency comparison for MAGMA [15] and CHEF (ours) on VLocNet [2], an MMMT model. MAGMA focuses on mapping multiple DNNs on multiple accelerators but does not involve cross-backbone layer dependencies. Compared to MAGMA, CHEF achieves lower latency for MMMT models with fewer PEs.

ranges from 274 to 319. Existing multiaccelerator designs for SMST models [8], [9], [10], [11] partition available resources for each layer, and design customized subaccelerators for different types of layers. Such layer-wise pipelined dataflow accelerators (DFAs) can solve the large variation for shallower networks.

Second, since MMMT models contain multiple SMST backbones, the number of layers is also multiple times greater than that found in single DNNs. For example, VFS includes 48 Conv and FC layers, while VLocNet [2], another MMMT model, is composed of 141 layers. The traditional DFAs fail to address the large variation when the network becomes deeper because they would necessitate the design of numerous different small accelerators under a fixed FPGA resource constraint. As proved in DNNExplorer [17], more accelerators lead to fewer resources for each stage, which eventually leads to lower performance. DNNExplorer shows that when the number of Conv layers increases from 13 to 38, the performance of a 38-layer model decreases by 77.8% compared to a shallower network with 13 Conv layers.

Third, MMMT models include more complex interlayer dependency across different SMST backbones. While Herald [18] and MAGMA [15] were developed to alleviate the previous two challenges by running multiple networks on multiple accelerators in parallel instead of in pure pipeline fashion, the complex interlayer dependency across different SMST backbones makes them inefficient. Fig. 1(b) shows the comparison of the latency of VLocNet, a typical MMMT model, on MAGMA and our design. MAGMA targets a small accelerator with 32×64 processing elements (PEs) and a large accelerator with 128×64 PEs. CHEF targets Xilinx U280 (1808 PEs) and U250 (2458 PEs) FPGAs. In FPGA, five digital signal processors (DSPs) conduct a multiply-accumulate (MAC) operation and can be considered as one PE. As illustrated in Fig. 1(b), with fewer PEs, CHEF achieves lower latency for MMMT models than the SOTA SMST-based accelerator design.

H2H [1] is the first attempt to map MMMT models to different FPGA accelerators using an iterative heuristic algorithm. However, H2H cannot work for general scenarios due to the following limitations. First, H2H relies on the CPU host memory to store data when the DRAMs of FPGAs cannot hold all data, which cannot work for edge servers without a host. Second, in H2H, each FPGA is only deployed with one accelerator, while in a more general case, one FPGA is feasible to deploy with one or multiple subaccelerators. The limited design space prevents H2H from finding a more optimal mapping scheme with better-resource utilization. M5 [7] is the second MMMT mapping work but has the following limitations. First, M5 uses the number of DSPs to approximate the resource consumption and latency, while the actual relationship between the resource consumption and latency is not polynomial. Second, M5 only targets homogeneous clusters rather than heterogeneous clusters. Therefore, these two works are over-simplified and fail be applied in more complicated and practical design scenarios existing in heterogeneous systems.

Compared to H2H and M5 which are the only two existing works scheduling MMMT models on multiple FPGAs, our work targets more general and practical scenarios for the MMMT scheduling problem. It will be explained in Section III in detail. Our main contributions are as follows.

1) We propose CHEF-A2F (Section IV), a two-stage accelerators-to-FPGAs deployment approach to efficiently deploy heterogeneous accelerators to heterogeneous FPGAs supporting diverse accelerator types (DATs) (Feature ①) and search for an efficient solution in a nonlinear, multidimensional, multiple-knapsack (MDMK) design space (Feature ②).
2) We propose CHEF-M2A (Section V), an efficient mapping algorithm to map the MMMT models to the deployed accelerators considering both the variation among heterogeneous layers and the interlayer dependency. Compared to H2H and M5, CHEF-M2A supports more complicated scenarios as shown in Fig. 2(b) incorporating intra-FPGA bandwidth (BW) sharing (Feature ③), inter-FPGA-communication (Feature ④), DRAM budget during mapping (Feature ⑤), and addressing cross-backbone data dependencies (Feature ⑥).
3) Based on the CHEF algorithm, we develop a simulator to estimate the latency of MMMT models for different clusters. To the best of our knowledge, we are the first to attempt to validate the simulator with end-to-end implementation (Feature ⑦). Experimental results show that the deviation of the simulation result is only -7.81% compared to the end-to-end on-board measurement result, which validates that the estimated latency of CHEF is relatively accurate. Therefore, our work can be used as a benchmark for future mapping algorithms either in simulation or implementation.

TABLE I
COMPARISONS WITH SOTA HETEROGENEOUS ACCELERATORS DESIGNS

Features	① DAT	② Knapsack	③ Intra-FPGA	④ Inter-FPGA	⑤ DRAM budget	⑥ Cross-backbone	⑦ Implementation
CHARM [8]	×	MDSK	✓	×	×	×	✓
BLAST-R [9]	×	MDMK	×	✓	×	×	×
Elastic-DF [10]	✓	MDMK	✓	✓	×	×	✓
AIgean [11]	✓	MDMK	×	✓	×	×	✓
Herald [18]	✓	MDSK	✓	×	×	×	×
MAGMA [15]	✓	MDSK	✓	×	×	×	×
H2H [1]	✓	Fixed	×	✓	✓	✓	×
M5 [7]	✓	SDMK	×	✓	×	✓	×
CHEF (ours)	✓	MDMK	✓	✓	✓	✓	✓

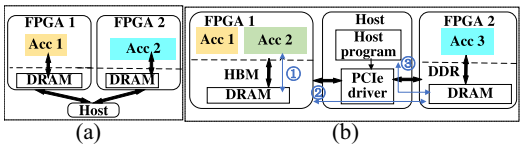

Fig. 2. (a) Architecture for the heterogeneous cluster in H2H [1]. It contains multiple FPGAs, and one FPGA is deployed with an accelerator. All FPGAs are connected to a main host with unlimited memory. (b) More general architecture is implemented in CHEF. Different from H2H, one FPGA can be deployed with one or multiple accelerators. We only store weights and immediate features on local DRAMs, and the memory constraint is considered. ① represents the intra-FPGA communication scheme. We support two inter-FPGA communication schemes: ② the direct P2P communication between two FPGAs without a host CPU and ③ the FPGAs are connected via a host.

II. RELATED WORKS

A. Evolving From SMST to MMMT

The development of DNNs enables easier fusing from different input signals, which makes it appealing to evolve from streamlined SMST models to MMMT models for better accuracy [19]. Currently, MMMT models are promising to be applied in various applications, such as robotics, human–computer interaction, and VR/AR for better performance [2], [3], [4], [5], [6], [20]. For example, VLocNet, a novel convolutional neural network (CNN) architecture has been proposed which takes two consecutive monocular images as input, regresses the 6-DoF global pose and 6-DoF odometry simultaneously, and outperforms task-specific localization models [2]. In the 3-D autonomous driving scenario, FULLER takes both the point cloud and image as inputs and achieves precision improvement in both map segmentation and 3-D detection [21]. However, apart from better-prediction accuracy, it is also necessary to reduce the inference implementation with the help of diverse accelerators. Compared with implementing SMST, implementing MMMT has larger layer variation, a massive number of layers, and more complex interlayer dependency, which increases the difficulties of efficiently deploying MMMT models on hardware platforms.

B. Effectiveness of Heterogeneous Accelerators in SMST Implementation and Limitations to Be Applied in MMMT Implementation

To solve the large variation in DNN layer shapes, heterogeneous accelerators are designed for better utilization and low latency [8], [9], [10], [11], [15], [18]. Table I compares SOTA heterogeneous accelerator designs considering the seven features mentioned in Section I. CHARM [8] provides a system-design methodology for composing heterogeneous matrix multiply (MM) accelerators on the Versal ACAP chip. Since the mapping targets a single FPGA with resource constraints, including PEs and on-chip block RAMs (BRAMs), the design space can be represented as a multidimensional, single-knapsack problem (MDSK). To efficiently map diverse sizes of MM layers on multiple accelerators, it partitions the MM layers of different workloads and generates resource partition candidates based on the workload assignment. BLAST-R [9] explores heterogeneous FPGA-based designs to effectively leverage both task and data parallelism to achieve the minimum cost while satisfying timing constraints. It models a CNN as a task graph and partitions Conv layers into pipeline stages by inserting buffers. Since it involves multiple FPGAs, the design space expands to MDMK which is more complex to find an optimal solution. However, the partition algorithms in CHARM and BLAST-R only focus on the monotone type of layers, while an MMMT model can be composed of Conv, FC, long short-term memory (LSTM) layers, etc. To implement MMMT models, a more general resource allocation approach supporting diverse layer types is needed.

Dlastic-DF [10] and AIgean [11] have achieved full end-to-end multi-FPGA implementations for traditional SMST models on the clusters with 100-GB/s network. They involve inter-FPGA data communication. However, AIgean only targets resource-abundant FPGA clouds whose on-chip memory can hold all data but have not considered the memory budget for resource-constrained edge clusters. Dlastic-DF implements SMST models in a pipelined manner. However, as mentioned in Section I, such a pipelined manner suffers from fewer resources for each stage, especially for MMMT DNNs involving multiple times of layers compared to SMST DNNs.

To support evolved networks with multiple inputs, Herald [18] and MAGMA [15] have been developed to deploy multiple SMST DNNs on multiple accelerators, achieving better utilization for heterogeneous layers. Unlike previous DFAs [8], [9], [10], [11], such approaches can address the former two challenges of MMMT models: 1) layer variation and 2) massive number of layers. However, unlike real MMMT models, the heterogeneous SMST models are independent of each other. As shown in Fig. 1, ignoring the last challenge, i.e., data dependency among different backbones will lead to suboptimal solutions.

C. Deploying MMMT Models on Multi-FPGA Systems

To the best of our knowledge, H2H [1] and M5 [7] are the only two works to map MMMT models to multi-FPGA systems. H2H provides an iterative heuristic algorithm to

map MMMT models on heterogeneous off-the-shelf FPGA-based accelerators with four steps, including computation prioritized mapping under zero local DRAM assumption; weight locality optimization buffering parts of weights to local DRAM; activation transfer optimization reducing immediate feature transmission latency for adjacent layer allocated on the same accelerator; and data locality aware re-mapping, to reduce inter-FPGA data communication overhead. Different from H2H which only assigns one accelerator on one FPGA board, M5 explores flexible accelerator configurations and possible resource sharing among layers. However, the algorithms of these works have not been validated on real hardware platforms. The limitations mentioned in Section I prevent both algorithms from being applied in practical scenarios. The proposed CHEF will address these limitations which will be discussed in Section III in detail. The main advantages of CHEF compared with all existing works are presented in Table I.

III. Motivation

As introduced in Section II-C, H2H and M5 are the only two works addressing the MMMT models to multi-FPGAs scheduling problem. However, the limitations in Section I prevent them from being used in a practical and general system. This section will first introduce the general system case and show how the H2H and M5 fail in the case. Then, the overview of CHEF is shown, including the challenges and solutions, to achieve MMMT models to heterogenous FPGAs scheduling in the general case.

In H2H, it is limited to only one accelerator connection topology with a host shown in Fig. 2(a) and ignores how to deploy different heterogeneous accelerators to heterogeneous FPGAs, which prevents the algorithm from being applied in general FPGA systems. First, in Fig. 2(a), H2H only targets the situation in which all FPGAs are connected to the main host. The host stores weights and immediate data in the main memory and conducts data swapping between two FPGAs. However, numerous general cases are beyond H2H's capabilities. For example, in the cases of edge servers like VMSS, BLAST-R, etc., FPGAs can directly communicate with each other via diverse connection approaches, such as Ethernet, PCIe, high-speed serial (HSS), etc., [i.e., ② in Fig. 2(b)]. The lack of main host memory makes it necessary to store all data in the local DRAM of each FPGA. Some clusters like UIUC XACC [13] and UCLA VAST [14] can communicate with each other either via the main host (③ in the figure) or directly via the PCIe driver without requiring access to the host CPU (②). Second, H2H maps multimodal models to off-the-shelf accelerators. However, different acceleration designs adopt different scheduling methodologies, computation patterns, and communications patterns, so there is no guarantee that these accelerators can be compatible with each other. In addition, H2H only deploys one accelerator on one FPGA, which is not flexible and leads to suboptimal mapping schemes. Different from H2H, CHEF targets a more practical and general design situation, where users have some compatible accelerator design intellectual properties (IPs) with self-developed analytical models. An IP is an accelerator design that can be deployed on an FPGA with a given parallelism degree. This scenario is common in system design. For example, Xilinx has developed a group of parameterizable IP cores called deep-learning processor units (DPUs) which are preimplemented on FPGAs [22]. Since our work requires finding an optimized scheduling scheme during the design time before hardware implementation, an accurate analytical model, including the resource costs and latency for specific layers, is also indispensable. Given one or multiple FPGA platforms, users can select IPs and deploy them to the system based on application requirements. As shown in Fig. 2(b), an FPGA is flexible to either accommodate one big accelerator or multiple smaller accelerators that can execute independent layers in parallel.

M5 [7] is the second work to deploy the MMMT model on multiple FPGAs but is oversimplified and only targets homogeneous clusters. First, M5 is oversimplified which only uses the utilized DSPs for each accelerator to profile the resource consumption and latency. In practical system design, the relationship between latency and resource costs is not polynomial, which makes the mapping problem more complicated. Second, M5 only targets homogeneous clusters of FPGAs, while mapping heterogeneous models to heterogeneous clusters of FPGAs introduces a larger design space. To sum up, H2H and M5 fail to be applied in more complicated design scenarios existing in heterogeneous systems.

Compared to H2H and M5 which are the only two existing works scheduling MMMT models on multiple FPGAs, our work targets more general and practical scenarios. As shown in Fig. 2(b), we have a cluster with heterogeneous FPGAs, and each FPGA has a particular on-chip resource constraint, i.e., available DSPs and BRAMs. Each FPGA also has a fixed DRAM size and on-chip to off-chip communication scheme ①. All the data are stored in DRAMs and different FPGAs can achieve peer-to-peer (P2P) communication directly ②. The host is only used to call the functions for the on-chip accelerator kernels. It should be noted that this architecture can be extended to solve the architecture in Fig. 2(a) by using half of the BW parameter in ③ as the P2P communication BW, i.e., data between two FPGAs are relayed via the main CPU host. Therefore, this architecture can support general scenarios, including cloud, edge, and on-device clusters. Given the clusters, users have developed different compatible candidate template accelerator IPs with diverse computation resource costs and performance models. Unlike M5, which relies on a simple performance model only considering the computation parallelism of MAC based on the number of DSPs, our performance model involves accurate profiling of on-chip computation and on-chip to off-chip communication. The model can be calibrated during on-board experiments.

The main goal of this study is to optimize both hardware setup and accelerator mapping to ensure the efficient inference performance of multitask DNNs. Therefore, we introduce CHEF, a framework designed for the effective deployment of varied accelerators to FPGAs (CHEF-A2F) and for mapping complex DNNs to these accelerators (CHEF-M2A). The overview is shown in Fig. 3.

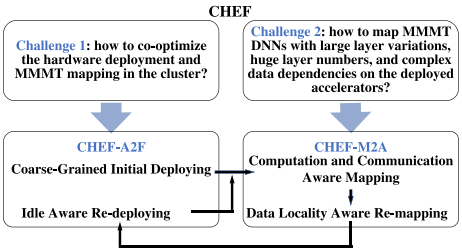

Fig. 3. Overview of CHEF. It includes CHEF-A2F to deploy heterogeneous accelerators on different FPGAs and CHEF-M2A to map MMMT DNNs on the deployed accelerators. CHEF-A2F includes two steps: coarse-grained initial deploying and idle aware re-deploying. CHEF-M2A includes two steps: computation and communication aware mapping and data locality aware re-mapping. CHEF-M2A mapping is conducted in every deploying to re-deploying iteration of CHEF-A2F. The four steps form a close-loop optimization workflow and work iteratively until no more beneficial scheduling scheme is acquired.

In the general and practical case shown in Fig. 2(b), we aim to select efficient accelerators to be deployed on heterogeneous FPGAs under hardware constraints and then map the MMMT model to the deployed accelerators for low latency. As shown in Fig. 3, the deploying and mapping problems need to be co-optimized. We use a running example of scheduling VFS on VMSS to illustrate CHEF.

There are two main challenges. The first is how to co-optimize the hardware deployment and accelerator mapping in the cluster. An FPGA can be deployed with one big accelerator or multiple small accelerators running in parallel. Different deployment leads to different mapping results, which is shown in Fig. 5. Balancing between accelerator architectures and available hardware resources is a tradeoff. To address this challenge, we propose CHEF-A2F, a two-stage accelerators-to-FPGAs deployment approach. It starts mapping with Coarse-Grained Initial Deploying and then conducts Idle Aware re-deploying based on the mapping results. It supports diverse layer types (Feature ①) and models the search space as an MDMK problem (Feature ②). This approach will be introduced in detail in Section IV.

The second challenge is that, unlike traditional streamlined DNNs, MMMT models have large layer variations, huge layer numbers, and complex data dependencies, so it is nontrivial to map MMMT DNNs on multiple FPGAs considering both computation and communication patterns. Given both computation and communication constraints, previous MMMT mapping algorithms [1], [7] are oversimplified. Therefore, we propose CHEF-M2A, a novel MMMT models-to-accelerators mapping algorithm. It generalizes H2H by considering the following additional configurations. First, one FPGA can be deployed with one or multiple accelerators, so accelerators can communicate with each other via intraboard communication (Feature ③) or interboard communication (Feature ④). Second, without relying on the host memory to buffer weights and intermediate data, the mapping algorithm will consider the impacts of local DRAM size (Feature ⑤). Compared to the four steps in H2H, CHEF-M2A achieves lower latency with only two steps: 1) the computation and communication aware mapping and 2) data locality aware re-mapping. This mapping algorithm will be introduced in Section V.

As illustrated in Fig. 3, the optimizations in CHEF-A2F and CHEF-M2A form a close-loop optimization workflow. During the initial deployment and each iteration of re-deploying in CHEF-A2F, CHEF-M2A mapping is conducted to update the mapping scheme based on the new accelerator-to-FPGA deployment. CHEF stops until no more beneficial mapping and deploying schemes can be obtained.

IV. CHEF-A2F

In this section, we propose CHEF-A2F, a two-stage accelerators-to-FPGAs deployment approach to address the first challenge discussed in Section III. The overall co-optimize problem can be formulated as follows. Given $i = 1, \ldots, m$ FPGAs with available DSPs and BRAMs constraints, i.e., DSP_i and BRAM_i for FPGA_i, we have already designed $t = 1, \ldots, n$ types of accelerator IPs, e.g., A_1 Conv IPs, A_2 FC IPs, A_3 LSTM IPs, etc. (Feature ①). Each IP has an analytical model which is composed of a resource and a performance model [23], [24]. The resource model is used to estimate its DSPs and BRAMs cost, e.g., $\text{DSP}_{a=1,\ldots,A_1}$ and $\text{BRAM}_{a=1,\ldots,A_1}$ for Conv IPs. The performance model estimates the latency for a DNN layer of the same type. The optimization problem can be illustrated in (1). The deployment scheme is shown as $(\{X_{\text{ita}}\}\ 1 \le a \le A_t,\ 1 \le t \le n,\ 1 \le i \le m)$, where X_{ita} is the number of the ath IP for the tth accelerator type deployed to FPGA_i, and $\{X_{\text{ita}}\}$ is a list of X_{ita} for all IPs. The goal of the optimization problem is to minimize the overall mapping latency of the deployed accelerators. The first two constraints indicate that for each FPGA i, the sum of DSPs and BRAMs costs of the deployed accelerators should not exceed the available DSPs and BRAMs for each FPGA. Constraint 3 indicates that the number of accelerators deployed on each FPGA for each IP should be a non-negative integer, and the same IPs can be selected multiple times. The last constraint ensures that for each type of accelerator, at least one IP should be selected and deployed in the multi-FPGA cluster

$$\begin{aligned} &\min \text{CHEF} - \text{M2A Mapping}(\{X_{\text{ita}}\},\ 1 \le a \le A_t \\ &\quad 1 \le t \le n,\ 1 \le i \le m) \\ &\text{s.t.} \begin{cases} \sum_{t=1}^{n} \sum_{a=1}^{A_t} \text{DSP}_{\text{ita}} \cdot X_{\text{ita}} < \text{DSP}_i\ \forall i \\ \sum_{t=1}^{n} \sum_{a=1}^{A_t} \text{BRAM}_{\text{ita}} \cdot X_{\text{ita}} < \text{BRAM}_i, \forall i \\ X_{\text{ita}} \ge 0 \text{ and integer} \\ \sum_{i=1}^{m} \sum_{a=1}^{A_t} X_{\text{ita}} \ge 1, \forall t. \end{cases} \end{aligned} \tag{1}$$

It is apparent that (1) can be represented as a nonlinear, MDMK problem, which is NP-hard and cannot be solved in polynomial time. Since the mapping function is also nonpolynomial, directly applying traditional knapsack-solving algorithms like dynamic programming (DP) to find an optimal solution is time-consuming. Therefore, we propose a two-stage accelerators-to-FPGAs deployment approach, CHEF-A2F, to search for an efficient deploying scheme in an acceptable time for this MDMK problem (Feature ②).

The overview of CHEF-A2F is shown in Fig. 4. It first allocates accelerators from the candidate IPs to the FPGA cluster in a Coarse-Grained manner. Then, an Idle Aware re-deploying algorithm is proposed to remove and replace some

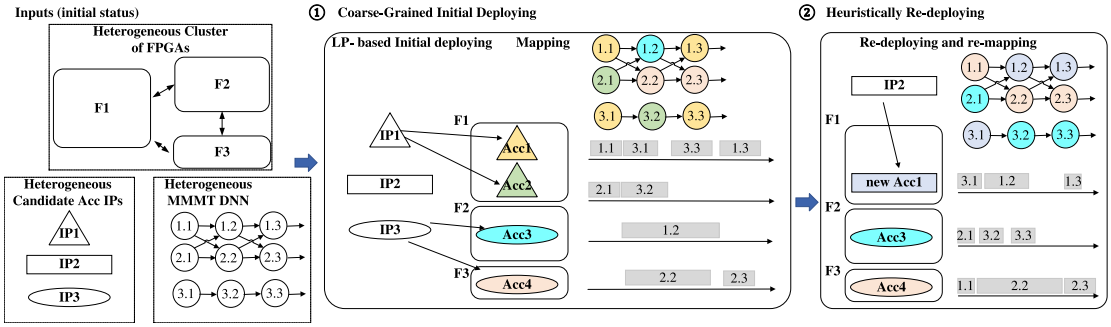

Fig. 4. Two-stage accelerators-to-FPGAs deployment approach (CHEF-A2F) visualization. It includes coarse-grained initial deploying and idle aware re-deploying. In every deploying or re-deploying iteration, MMMT is mapped to the deployed or re-deployed accelerators. The re-deploying stops until no beneficial mapping scheme can be acquired. We only show the former three Conv layers for each modality of a MMMT model.

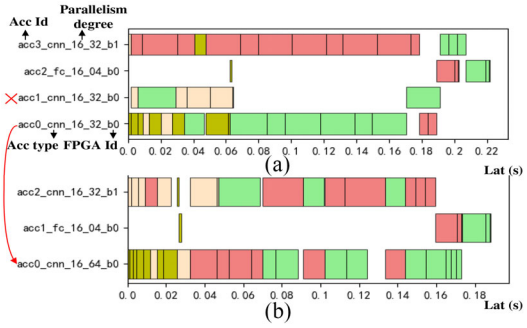

Fig. 5. Gantt charts of scheduling VFS on VMSS under 15 GB/s before re-deployment and after re-deployment. The "16_32" means the parallelism for the input channels is 16, while that for output channels is 32 for a Conv layer. Different bar colors represent layers from different modalities. (a) Mapping scheme after coarse-grained initial deploying. (b) Mapping scheme after idle aware re-deployment.

accelerators for better utilization. The Gantt charts of the mapping scheme are shown in Fig. 5.

A. Coarse-Grained Initial Deploying

Since the Mapping function in (1) is nonpolynomial, it is time-consuming to directly apply knapsack-solving algorithms. Therefore, we provide a coarse-grained approach to select the most powerful accelerators combination with the maximum overall throughput as an initial deploying strategy.

For each tth type of accelerator IP ta deployed in FPGA i, we estimate the maximum throughput the accelerator can achieve for each layer of the MMMT model thp_{ita}. Then, we approximate the mapping results in (1) using the sum of the estimated maximum throughput for all the deployed accelerators. The optimization goal after approximation is shown in (2), while the constraints remain unchanged

$$\max \sum_{t=1}^{n} \sum_{i=1}^{m} \sum_{a=1}^{A_t} thp_{\text{ita}} \cdot X_{\text{ita}}. \tag{2}$$

This problem is changed to a standard linear programming (LP) problem and can be solved by off-the-shelf LP tools. In this work, PuLP [25] is used as the LP solver. After the LP-based deployment, we apply the CHEF-M2A mapping algorithm in Section V to get an initial estimated latency. Current mapping scheme of the VFS running example is shown in Fig. 5(a).

B. Idle Aware re-deployment

In Section IV-A, we use the maximum throughput to approximate the mapping performance for each deployed accelerator. However, the accelerators cannot achieve the best performance since some of them will be idle for some layers after CHEF-M2A mapping. For example, in Fig. 5(a), "acc1" is idle after 0.06 s. Therefore, we re-deploy some accelerators based on the mapping results. We found that after removing idle accelerators [i.e., acc1 in Fig. 5(a)] we can leave space to replace smaller accelerators with lower-parallelism degrees [i.e., "acc0" in Fig. 5(a)] to bigger ones with higher-parallelism degrees [i.e., acc0 in Fig. 5(b)]. Based on this observation, we propose an idle-aware re-deployment after getting the initial deployment scheme, mapping, and estimated initial latency *Lat*. We search for an optimized deployment scheme by iteratively removing accelerators with longer idle time and then replacing small accelerators with larger and more powerful ones. We check the latency after mapping every time a replacement is conducted and only accept the replacement with shortened latency. The proposed re-deploying algorithm adopts the duty cycle to measure if an accelerator is under-utilized and idle.

The algorithm consists of the following steps. *Step 1:* For the accelerator of each type, starting from the accelerator with the least duty cycle, the algorithm will attempt to remove the accelerator from the located FPGA if the accelerator is not the only accelerator of the same type. For example, in Fig. 5(a), there are two accelerator types: Conv and FC. "acc2" is the only FC accelerator, while acc1 has the lowest-duty cycle for all Conv accelerators.

Step 2: For the remaining accelerators on the same FPGA, the algorithm attempts to pick one and replace it with another candidate IP as long as the DSPs and BRAMs constraints are met (e.g., replace acc0 on "b0" in Fig. 5(a) to acc0 in Fig. 5(b).

Step 3: For each replacement in step 2, we re-map the MMMT DNN using CHEF-M2A and choose the deployment

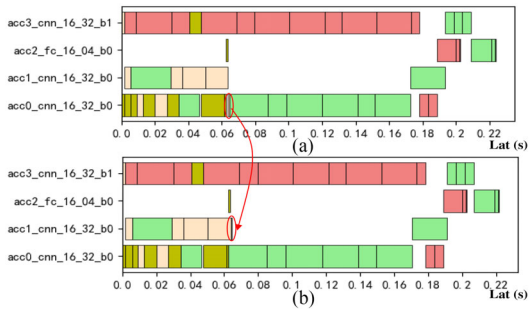

Fig. 6. Gantt charts of scheduling VFS on VMSS under 15 GB/s before re-mapping and after re-mapping under coarse-grained initial deploying. (a) Mapping scheme after computation and communication aware mapping. (b) Mapping scheme after data locality aware re-mapping.

parameters $\{X_{\text{ita}}\}$ with the lowest latency (e.g., the lowest *Lat* is shortened from 0.222 to 0.188 s in Fig. 5(b).

Step 4: If the overall latency is shortened after steps 1–3, we accept such replacement. The re-deploying scheme, re-mapping scheme, and *Lat* are updated. Then, we return to step 1 using the updated duty cycle for the next re-deploying iteration. If the latency is not shortened after the replacement, we will remove another accelerator with the second least-duty cycle and repeat steps 2 and 3. If no accelerator can be removed for lower-resultant latency, [e.g., *Lat* cannot be shorter than that in Fig. 5(b)], the algorithm will stop.

V. CHEF-M2A

This section will introduce the details of CHEF-M2A, which maps MMMT models to the accelerators obtained in Section IV. The input of the CHEF-M2A algorithm includes the model graph G_{model} and accelerator-to-FPGA deploying information $\{F_i\{\text{Acc}_j\}\}$. The nodes in G_{model} represent the layers of the MMMT model, while the edges represent layer dependencies. Unlike SMST, G_{model} involves multiple branches with complex data dependency. We use the same running example to illustrate the CHEF-M2A algorithm, and the Gantt charts are shown in Fig. 6.

Unlike H2H which only lists the performance models for each accelerator, CHEF-M2A models the accelerators-to-FPGAs deploying information $\{F_i\{\text{Acc}_j\}\}$ which is obtained from ($\{X_{\text{ita}}\}$ $1 \le a \le A_t$, $1 \le t \le n$, $1 \le i \le m$). For each FPGA F_i, it includes the list of deployed accelerators $\{\text{Acc}_j\}$, the number of DDR/HBM channels, and available DRAM size. Acc_j is the jth accelerator deployed in F_i. Each accelerator Acc_j records which IP it is applied to which FPGA and has a performance model $Perf_j$ using the layer information, BW between the FPGA chip and the DRAM BW_{DRAM} [i.e., ① in Fig. 2(c)], and the interaccelerators BW BW_{Inter} [i.e., ② or ③ in Fig. 2(c)] as inputs. The output of the mapping algorithm is a multiaccelerator graph $G^*_{\text{sys}} = \{G^*_{\text{Acc}_j}\}$, where each accelerator j holds a mapping graph $G^*_{\text{Acc}_j}$ representing the hardware dependency for each layer. Each node of $G^*_{\text{Acc}_j}$ has the information on which layer the accelerator is mapped to, the start time of the layer, and the end time, while the edges show the dependencies and orders of these mapped layers. An example of G^*_{sys} is shown in Fig. 6. The mapping latency *Lat* is the maximum value of the end time in G^*_{sys}. Our goal is to minimize *Lat*.

As introduced in Section III, H2H has four processes: 1) computation prioritized mapping; 2) weight locality optimization; 3) activation transfer optimization; and 4) data locality aware re-mapping. It assumes all data are stored in the host memory which is hypothetically unlimited in the first process and then buffers only a proportion of these data to the local DRAMs in the second and third processes to remove the data transmission latency. However, in a more general case without host memory, the zero local DRAM assumption cannot be applied, so data transmission between layers should be involved at the beginning. Therefore, compared to H2H, CHEF-M2A first conducts Computation and Communication Aware Mapping, which considers weights locality and activation transmission together. After that, Data Locality Aware re-mapping is applied to further shorten the overall latency.

A. Computation and Communication Aware Mapping

Since MMMT models involve cross-backbone dependency, each layer will have multiple predecessors and successors (Feature ⑥). To tackle this, in the initial mapping, we first consider all unmapped nodes without predecessors in the model graph and find the best-mapping combinations, and then the mapped nodes are removed from the graph. The detailed steps are as follows. In step 1, for all the unmapped nodes without predecessors in G_{model}, we enumerate all the mapping combinations to allocate these nodes on $\{F_i\{\text{Acc}_j\}\}$. Unlike H2H which assumes zero DRAM locality by storing all weights and immediate features in the host memory at the beginning and moving a proportion of these data to DRAM under the DRAM budget later, CHEF-M2A stores all data in DRAM. Therefore, CHEF needs to guarantee the DRAM budget can hold all data during mapping. Thus, in step 2, we conduct DRAM budget check (Feature ⑤), i.e., for each mapping candidate, we check if the current DRAM budget for FPGA is possible to accommodate weights and features for the rest of the layers. Only if for all layer types, the DRAM cost for the rest of the layers of the same type is smaller than the DRAM size for the FPGAs deployed with corresponding types of accelerators, and the FPGA with maximum DRAM budget can hold the data of the layer with maximum DRAM cost, CHEF-M2A will accept current mapping candidate and move to the next step.

Step 3 is to calculate the latency increment ΔLat for all the accepted mapping candidates. Assume layer l is mapped to accelerator Acc_j, and its predecessor layer l' in the MMMT model is mapped to accelerator $\text{Acc}_{j'}$. For layer l, its layer latency (e.g., the length of a box in Fig. 6) involves the intra-accelerator latency estimated by $Perf_j$ and the data transmission latency among accelerators. Unlike H2H, we store weights of l in the local DRAM of F_i, so there is no weight transmission. For feature transmission, the situation is also more complicated since we consider the fact that multiple accelerators are located on one FPGA. The detailed analysis is as follows (Features ③ and ④). First, if Acc_j and $\text{Acc}_{j'}$ share the same DDR/HBM bank of the same FPGA, there is

no feature transmission latency, but BW_{DRAM} will be divided by the number of accelerators sharing the same bank. Second, if they are on the same FPGA but connect to different banks, features are transmitted among banks via the FPGA chip. Third, if Acc_j and $Acc_{j'}$ are located on different FPGAs, the feature transmission latency is calculated via BW_{Inter}. ΔLat is the maximum layer latency of these unmapped nodes. In step 4, we select the mapping candidate that results in the minimum ΔLat and remove the mapped nodes from the MMMT model graph. G^*_{sys} is also updated with new mapped nodes.

B. Data Locality Aware re-mapping

Next, CHEF-M2A conducts a re-mapping operation that reallocates a layer from its source accelerator to a new destination accelerator, on which its neighbors (predecessors or successors) are mapped with the following steps.

Step 1: For all the nodes in G_{model}, if its neighbor is not on the same accelerator, we attempt to re-map the node to its neighbor's accelerator [e.g., the circled layer on acc0 in Fig. 6(a) is moved to acc1 in Fig. 6(b)].

Step 2: For each re-mapping attempt, we conduct the DRAM budget check. The latency after re-mapping is calculated only if the DRAM budget is satisfied.

Step 3: The re-mapping attempt is accepted if the MMMT model latency is shortened (e.g., *Lat* in Fig. 6 is shortened from 0.224 to 0.222 s). G^*_{sys} is also updated.

Steps 1–3 are repeated until no node can be re-mapped for better results.

VI. Simulator

Based on CHEF-A2F and CHEF-M2A, we develop a simulator to estimate the latency of MMMT models for different clusters. The simulator is composed of four parts: 1) the configuration of the FPGA cluster; 2) resources and performance models for accelerator IPs; 3) the definition of the MMMT model; and 4) our CHEF scheduling algorithm.

The Configuration of the Cluster: In this part, we first define the FPGAs that are used in the cluster. The information on each FPGA includes the frequency of the FPGA, the number of on-chip DSPs and BRAMs, the number of DDR/HBM banks in the off-chip DRAM, the size of each DDR/HBM bank, and the DDR/HBM BW BW_{DRAM}. Second, we include the P2P BW between arbitrary two FPGAs BW_{inter}. If FPGA 1 and FPGA 2 in Fig. 2(b) are connected via ②, BW_{inter} is the PCIe BW. If they are connected via ③, which is the same as Fig. 2(a), BW_{inter} is half of the FPGA-to-host BW.

Accelerator IPs: Our simulator enables users to design customized accelerator IPs with self-developed resource and performance models. Currently, we have established model templates for Conv IPs and FC IPs based on XFER [24] and LSTM IPs based on [26] with different parallelism degrees. For each template, the resource model estimates the number of DSPs, BRAMs costs under their parallelism degree, and the DRAM costs for each layer. The performance model calculates the latency for each layer, both including the features transfer with the predecessor layers and without interlayer data communication.

TABLE II
Heterogeneous Edge Clusters

Name	Used in	Configuration
Cluster 1	H2H [1]	VC707+ZCU102+ZC706+XCKU060+ XC7Z045+VCU118
Cluster 2	VMSS [12]	U50LV+U30
Cluster 3	XACC [13]	U280+U250

TABLE III
MMMT DNNs

Model	Backbone	Layer Types	Layers
VLocNet [2]	ResNet-50 variants	CNN, FC	141
CASUA-SURF [27]	ResNet-18 variants	CNN	44
VFS [16]	VGG and VD-CNN variants	CNN, FC	48
FaceBag [28]	ResNet variants	CNN, FC	51
CNN-LSTM [29]	ConvNet and LSTM variants	CNN, LSTM	20

Definition of the MMMT Model: In each MMMT model, we first define layer information, including the layer type and the layer parameters for each layer. For Conv layers, the layer parameters include the number of output channels, the number of input channels, output feature column size, output feature row size, weight kernel size, stride, and padding size. For FC layers, the parameters only involve the number of output channels and the number of input channels. For LSTM layers, embedded vector dimension, the number of hidden states, and the number of LSTM cells are involved. Second, the data dependency between different layers is defined.

CHEF Scheduling Algorithm: Based on the algorithms introduced in Sections IV and V, we generate the optimized scheduling strategy given the FPGA clusters, accelerator IPs, and the MMMT model. CHEF-A2F records the accelerator-to-FPGA deploying information $\{F_i\{Acc_j\}\}$ indicating which IP is deployed to which FPGA and how many accelerators of this IP are on the FPGA. Given the accelerator deployment, CHEF-M2A records the mapping scheme G^*_{sys}, including which layer, is mapped to which accelerator, and the start and end times for each layer. The resultant inference latency *Lat* is the end time of the last layer.

VII. Experiments

In this section, we first implement the MMMT model on the public XACC server and compare the end-to-end latency with that estimated by the simulator to validate the correctness of the algorithm. Second, we analyze the effectiveness of re-deploying and re-mapping optimization steps in CHEF on three practical FPGA clusters both at the edge level and at the cloud level. Then, we show the overall effectiveness of CHEF by comparing it with the SOTA MMMT models to heterogeneous clusters scheduling algorithm H2H. Finally, a series of ablation studies for search time (ST) and latency are presented to validate the effectiveness and efficiency of CHEF-A2F and CHEF-M2A.

A. Experimental Setup

Heterogeneous Cluster of FPGAs: Table II summarizes 3 heterogeneous FPGA clusters that have been applied in previous research and industrial applications. Cluster 1 adopts

$m = 6$ different Xilinx FPGAs that are applied in H2H with 6 developed IPs. Cluster 2 uses the VMSS edge server developed by Xilinx for smart cities ($m = 2$) with 12 IPs. The XACC in Cluster 3 is a public multi-FPGA cloud server established by UIUC ($m = 2$) with 8 IPs. H2H uses a CPU host to connect all FPGAs via Ethernet, while VMSS is equipped with PCIe interfaces for direct P2P communication. XACC supports both P2P communication and host-to-FPGA communication. To show the training performance on various P2P communication BWs, we test CHEF on these clusters with different $BW_{inter} = 0.125$, 3, and 15 GB/s. 0.125 GB/s represents a low-BW communication approach, such as Gigabit Ethernet [30] (1Gbps=0.125GB/s), while 3 GB/s represents a medium BW connection, such as PCIe (tested in Section VII-B). 15 GB/s represents a high BW. For example, with TCP/IP stack, the 100 GbE Smart NICs achieve around 100–140 Gbps BW with an average of 15 GB/s [31].

Heterogeneous MMMT Models: Table III summarizes 5 heterogeneous DNNs used in the evaluation, spanning different domains. The models use CNNs, FCs, or LSTMs as backbones (Feature ①) and typically involve 3 or 4 backbones with cross-backbone data dependencies Feature ⑥). For VLocNet, VFS, FaceBag, and CNN-LSTM, the number of layer types $n = 2$, while $n = 1$ for CASUA-SURF.

Baselines: The baselines are as follows.

1) To show the effectiveness of the re-deploying and re-mapping optimizations in Section VII-C, we compare the resultant latency with the following steps: Step 1 is the latency with initial deploying in CHEF-A2F and initial mapping in CHEF-M2A, step 2 is the latency with re-mapping during the initial deploying; and step 3 is the latency with both re-deploying in CHEF-A2F and re-mapping in CHEF-M2A.
2) To show the overall effectiveness of CHEF in Section VII-D, we compare it with the SOTA work H2H.
3) In the ablation studies in Section VII-E, to validate the effectiveness of CHEF-A2F, we compare our deployment search approach with DP, which is a commonly used search algorithm to find optimal solutions for Knapsack problems. To validate the effectiveness of CHEF-M2A, the optimal solution is provided by enumerating all possible layer-to-accelerator mapping combinations.

B. End-to-End Implementation (Feature ⑦)

The proposed work is evaluated on the public UIUC HACC Cluster [13] with one U280 and one U250 FPGA shown in Fig. 2(b). The working frequency for U280 is 200 MHz, while U250 works on 150 MHz. The U280 is equipped with 32 HBM banks, while the U250 is equipped with 4 DDR banks. We first measure the on-board BW_{DRAM}. The HBM BW is around 12 GB/s under 200 MHz, and the DDR BW is around 8 GB/s under 150 MHz. Then, we test the PCIe-based P2P communication BW BW_{inter}, and BW_{inter} is measured to be 3 GB/s. To validate the accuracy and correctness of CHEF, we conduct end-to-end implementation on the CASUA-SURF [27], VFS [16], and FaceBag [28] models, and each model contains more than 40 layers. We developed eight candidate IPs and established analytical models. Each IP is designed and coded with TAPA [32]. The obtained IP cores have bitstream generated in Xilinx Vitis (v2022.2). For each IP, we compare the latency estimated by the performance model and that measured on-board. The deviations for estimated latency compared to on-board tested latency for these IPs range from -0.94% to -6.66%, which proves that the performance models are accurate for latency estimation.

TABLE IV
END-TO-END RUNTIME OF MMMT MODELS ON HACC: MODELING VERSUS ON-BOARD MEASUREMENT

Model	Modeling (s)	On-board Measurement(s)	Error Rate
CASUA-SURF	0.0307	0.0336	-8.63%
VFS	0.1475	0.1600	-7.81%
FaceBag	0.0159	0.0168	-5.36%

With the configuration of the FPGA cluster, the acceleration IPs, and the definition of the MMMT model, we apply CHEF to acquire the scheduling information, including the accelerator deploying information $\{F_i\{\text{Acc}_j\}\}$ and the mapping scheme G^*_{sys} automatically generated by the simulator. The simulator also generates an estimated latency *Lat*. Then, we implement end-to-end inference of the three models on the cluster using the scheduling information and measure the on-board execution latency. As shown in Table IV, the deviation for the end-to-end testing of each complete model is less than 10% which validates that the estimated latency of CHEF is relatively accurate. After validating the correctness of the CHEF and the accuracy of its simulator, we use the simulator for a series of comparisons and ablation studies in the following experiments.

C. Effectiveness of re-deploying and re-mapping

Fig. 7 shows the system latency of the MMMT models listed in Table III. The *x*-axis represents different optimization steps in CHEF, i.e., without re-deploying or re-mapping, with only re-mapping, and with both re-deploying and re-mapping. We test the latency of the three clusters listed in Table II under various P2P BWs. It can be seen that the re-deploying in CHEF-A2F and re-mapping in CHEF-M2A can significantly reduce the overall latency in these cases. For example, in Fig. 7(a) for Cluster 1, the maximum latency reductions caused by the combination of re-deploying and re-mapping are 84%, 87%, and 88% under low, medium, and high P2P BW, respectively. re-deploying and re-mapping effectively improve the inference performance for different FPGA clusters under various P2P BWs.

D. Comparison With H2H [1]

Fig. 8 shows the inference latency speedup of the MMMT models compared to H2H [1]. We compare the latency on the three heterogeneous cluster platforms under low, medium, and high BW. H2H adopts fixed accelerators, while our work searches optimized accelerator combinations and deploys them on the clusters. The inference latency achieved by CHEF is significantly shortened compared to the H2H baseline.

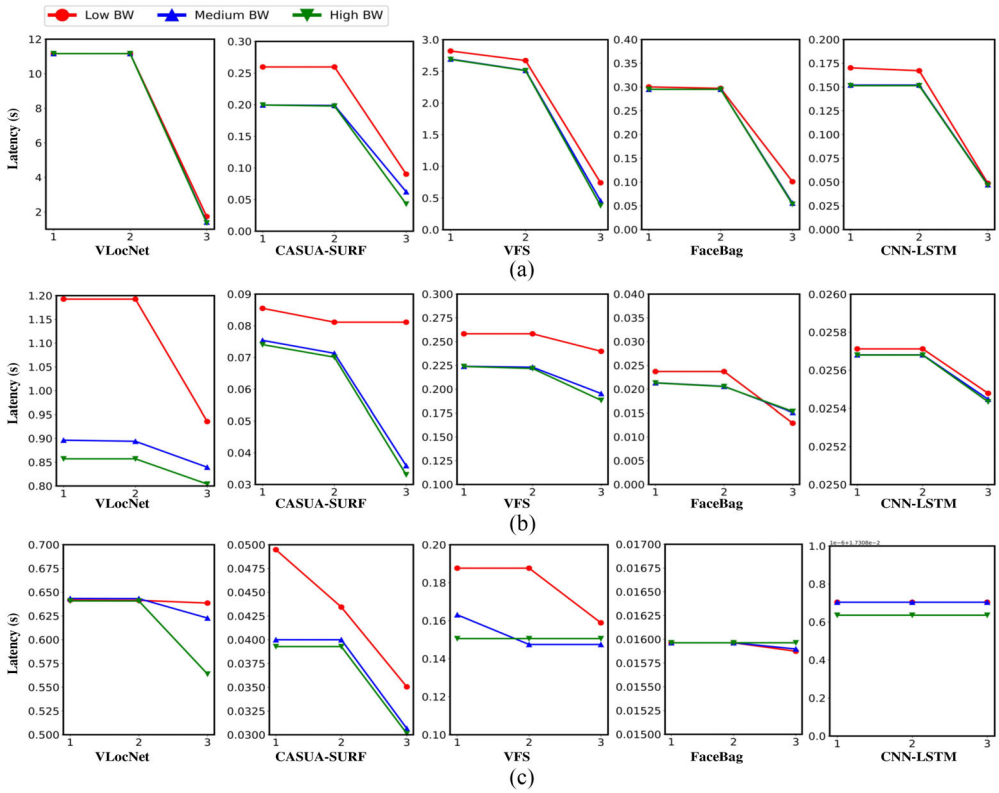

Fig. 7. Latency comparison of re-deploying and re-mapping. The x-axis represents three optimization steps: Step 1 is after initial deploying and initial mapping, step 2 is after the re-mapping under the initial accelerator deployment, and step 3 is the final result after both re-deploying and re-mapping. (a) Comparisons for Cluster 1. (b) Comparisons for Cluster 2. (c) Comparisons for Cluster 3. The re-deploying in CHEF-A2F and re-mapping in CHEF-M2A can significantly decrease the resultant inference latency for these clusters.

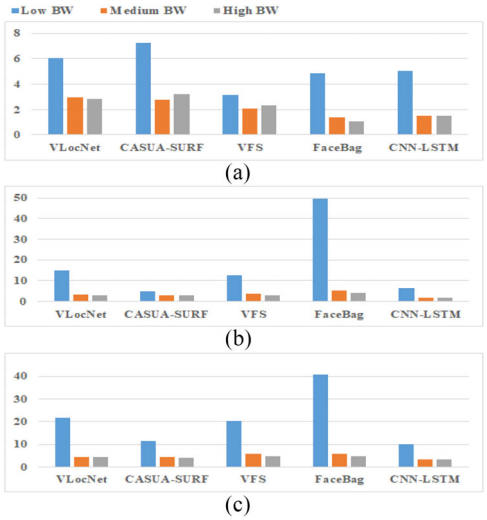

Fig. 8. Latency speedup compared to H2H. (a) Speedup in Cluster 1. (b) Speedup in Cluster 2. (c) Speedup in Cluster 3. Compared to H2H, the proposed CHEF achieves 1.09–49.43 inference speedup in these clusters.

The speedup comes from two aspects. On one hand, H2H deploys one fixed accelerator on one FPGA, while our proposed CHEF is flexible to deploy one or multiple accelerators on each FPGA. Take Cluster 2 under the low-P2P BW as an example, CHEF deploys 3 accelerators for VLocNet and 5 accelerators for CASUA-SURF. Compared to H2H, CHEF considers a larger design space, and it is feasible to search for a beneficial deployment scheme with superior mapping results.

On the other hand, H2H suffers from an intenser inter-FPGA communication bottleneck compared to CHEF. In CHEF, all weights are stored in local DRAM, while H2H stores parts of weights in the host memory which leads to extra weights transfer workload (Feature ⑤). Besides, immediate features are transmitted back and forth via the CPU host if adjacent layers are not located on the same accelerators. However, as mentioned in Section V, in CHEF, features do not need to be moved as long as two adjacent layers share the same HBM/DDR bank (Feature ③). If data-dependent layers are not allocated to the same memory bank but are mapped to accelerators on the same FPGA, features are transmitted via the FPGA chip. The on-chip transmission is more efficient than FPGA-to-FPGA transmission. Only when adjacent layers mapped to different FPGAs, the interboard feature transmission is required (Feature ④). Therefore, as presented in Fig. 8, CHEF achieves significant speedup, especially under lower-P2P BW. To further validate that Features ③–⑤ of CHEF successfully reduces communication overhead, we present

TABLE V
INTERACCELERATOR COMMUNICATION RATIO BETWEEN H2H AND CHEF (OURS)

Model	P2P BW	Cluster 1		Cluster 2		Cluster 3	
		H2H	CHEF	H2H	CHEF	H2H	CHEF
VLocNet	low	54.91%	14.07%	82.76%	2.22%	83.22%	0.05%
	medium	11.76%	2.88%	12.83%	5.62%	13.20%	4.04%
	high	2.87%	0.61%	2.86%	1.85%	2.95%	0.41%
CASUA -SURF	low	49.16%	5.40%	75.44%	11.43%	71.33%	5.82%
	medium	11.22%	0.94%	11.34%	4.30%	9.39%	1.89%
	high	2.88%	0.80%	2.50%	0.93%	2.03%	0.61%
VFS	low	49.97%	3.72%	82.60%	9.74%	79.86%	0.30%
	medium	6.11%	0.74%	17.75%	3.12%	15.85%	0.58%
	high	1.60%	0.24%	3.98%	0.84%	3.30%	0.33%
FaceBag	low	80.92%	3.19%	90.84%	0.00%	89.21%	0.00%
	medium	22.74%	0.89%	28.55%	1.82%	24.99%	1.70%
	high	6.61%	0.23%	7.10%	0.52%	5.99%	0.04%
CNN- LSTM	low	22.47%	0.10%	79.36%	0.10%	76.51%	0.06%
	medium	4.98%	0.83%	13.85%	0.00%	11.95%	0.00%
	high	1.07%	0.17%	3.11%	0.04%	2.65%	0.00%

TABLE VI
DEPLOYMENT PERFORMANCE AND ST COMPARISON

		DP (Optimal)		CHEF-A2F (ours)	
Model	FPGAs	Lat. (s)	ST (s)	Lat. (s)	ST (s)
VLocNet*	2	0.0446	3.62 (16X)	0.0505 (1.13X)	0.220
	3	0.0386	223 (326X)	0.0386 (1X)	0.683
	4	0.0304	1.97E4 (9610X)	0.0304 (1X)	2.05
CASUA -SURF*	2	0.0119	3.24 (21X)	0.0131 (1.10X)	0.151
	3	0.00838	196 (261X)	0.00838 (1X)	0.750
	4	0.00691	1.72E4 (9556X)	0.00691 (1X)	1.80
VFS*	2	0.0257	9.27 (22X)	0.0260 (1.01X)	0.424
	3	0.0212	810 (323X)	0.0212 (1X)	2.48
	4	0.0212	9.62E4 (10378X)	0.0212 (1X)	9.27
FaceBag*	2	0.000955	3.87 (18X)	0.00117 (1.23X)	0.220
	3	0.000888	226 (309X)	0.000931 (1.05X)	0.732
	4	0.000808	1.93E4 (9650X)	0.000808 (1X)	2.00
CNN- LSTM*	2	0.0103	2.61 (17X)	0.0110 (1.07X)	0.154
	3	0.0103	184 (322X)	0.0103 (1X)	0.572
	4	0.0103	1.65E4 (10060X)	0.0103 (1X)	1.64

the interaccelerator communication ratio of CHEF and H2H in Table V. The ratio is calculated using the sum of the accelerator-to-accelerator communication latency divided by the accumulation of layer time costs for each accelerator.

As illustrated in Table V, the interaccelerator communication ratio is less than 15% under various P2P BWs for different clusters, while the ratio for H2H is severely impacted by the P2P BW. For example, under low BW, the interaccelerator communication takes up 22%-91% of the accumulated inference latency. Compared to H2H, CHEF suffers less from the cross-accelerator communication overhead.

E. Analysis of CHEF-A2F and CHEF-M2A Algorithms

As mentioned in Section IV, finding an optimal solution is time-consuming, so we propose a two-stage accelerators-to-FPGAs deployment approach to co-optimize hardware deployment as well as accelerator mapping and thus search for a near-optimal solution. In this section, we validate the effectiveness and efficiency of CHEF-A2F and CHEF-M2A solving the MDMK problem in (1) by comparing the resultant inference latency (Lat.) and ST with the optimal solution (Feature ②). Results show that CHEF can achieve near-optimal solutions with significantly less searching time.

TABLE VII
MAPPING PERFORMANCE AND ST COMPARISON

		Optimal		CHEF-M2A (ours)	
Model	Accs.	Lat. (s)	ST (s)	Lat. (s)	ST (s)
VLocNet*	2	0.0547	2.85 (130X)	0.0555 (1.01X)	0.0220
	3	0.0555	159 (6115X)	0.0599 (1.08X)	0.0260
	4	0.0522	3107 (21727X)	0.0522 (1.00X)	0.1430
CASUA -SURF*	2	0.0133	1.23 (87.9X)	0.0134 (1.01X)	0.0140
	3	0.0125	46.4 (1719X)	0.0128 (1.02X)	0.0270
	4	0.0125	694 (8165X)	0.0128 (1.02X)	0.0850
VFS*	2	0.0235	2.75 (131X)	0.0261 (1.11X)	0.0210
	3	0.0212	157 (1826X)	0.0246 (1.16X)	0.0860
	4	0.0212	3453 (11472X)	0.0212 (1X)	0.3010
FaceBag*	2	0.00120	2.68 (223X)	0.00121 (1.01X)	0.0120
	3	0.00109	158 (4647X)	0.00120 (1.10X)	0.0340
	4	0.000997	3390 (25299X)	0.00117 (1.17X)	0.1340
CNN- LSTM*	2	0.0104	2.21 (205X)	0.0110 (1.06X)	0.0108
	3	0.0104	127 (5799X)	0.0104 (1X)	0.0219
	4	0.0103	2295 (18811X)	0.0103 (1X)	0.122

We first validate the effectiveness of the proposed CHEF-A2F deployment approach. Table VI shows the deployment comparison of CHEF-A2F and DP. We compare the estimated system latency and the ST on Cluster 3 with three candidate IPs and increase the number of FPGAs m from 2 to 4. For both DP and the proposed CHEF-A2F deployment approach, we apply the same CHEF-M2A mapping algorithm to generate the mapping scheme for each iteration of hardware deployment. Since DP searches all possible deployment schemes and finds an optimal solution, it will be time-consuming to map all layers of the MMMT models in each iteration. Therefore, we perform the comparison mapping subnetworks that contain only 9 or 10 layers for different models, and we use * to indicate only part of the layers is involved in the models in later results. As shown in Table VI, CHEF-A2F can achieve near-optimal performance for all the scenarios, while searching for an optimal deploying scheme demands a great mass of time. For example, when the number of FPGAs increases to 4, DP takes nearly 27 h to find an optimal solution for a subnetwork of VFS with ten layers, which is 10378 times compared to CHEF-A2F. Thus, searching for an optimal solution via DP for the whole VFS with 48 layers is estimated to take around 5 days, which is inefficient in practical applications. Compared to DP, CHEF-A2F can find a near-optimal solution for the whole VFS in 15.6 s. It should be noted that we only consider three IP candidates. When we increase the IPs to 4, DP fails to complete the search within 15 days even for subnetworks, while CHEF is efficient in searching for complete MMMT models with more IPs in an acceptable time. The ST will be discussed in detail in Fig. 9.

Then, we validate the effectiveness of CHEF-M2A. Table VII shows the mapping comparison of CHEF-M2A and the optimal solution in terms of estimated system latency and ST when the number of deployed accelerators increases from 2 to 4. The optimal solution is obtained by enumerating all possible mapping combinations and finding the one with the shortest system latency. Given N available accelerators and an MMMT model with M layers, the complexity of finding the optimal solution is N^M. Mapping all layers of MMMT models with four accelerator candidates ranging from 40 layers to 150 layers is time-consuming. Therefore, we also compare the performance mapping only 9 and 10 layers for the MMMT

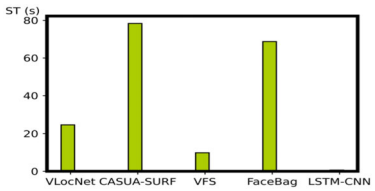

Fig. 9. ST of CHEF with eight candidate IPs when the number of FPGAs increases to 4. The ST ranges from 0.66 to 78 s.

models. As shown in Table VII, CHEF-M2A achieves near-optimal performance for all the scenarios, while searching for an optimal mapping scheme requires tremendous time. For example, deployed with 4 accelerators, finding an optimal solution mapping a mere 10-layer VLocNet takes nearly 1 h, while CHEF-M2A costs only 0.1430 s. CHEF-M2A achieves 21727 times speedup in ST for mapping. If we search for the optimal solution for the whole 141 layers, the optimal solution is estimated to take $6.4 \times 1e78$ h, which is impossible to apply in practical applications, while CHEF-M2A can find a near-optimal solution in 13.5 s.

To further present the overall efficiency of CHEF, we display the ST when CHEF is applied to schedule complete MMMT models on $m = 4$ FPGAs with eight candidate IPs. As can be seen in Fig. 9, CHEF searches for near-optimal deploying and mapping schemes for MMMT models ranging from 20 layers to 141 layers in minutes or seconds. If new models are given in the application, our proposed approach is feasible and efficient to generate the scheduling scheme in an acceptable design time.

VIII. Conclusion

This work proposes CHEF to enable efficient heterogeneous MMMT models deploying on heterogeneous clusters with FPGAs. We propose CHEF-A2F, a two-stage accelerators-to-FPGAs deployment approach to select efficient accelerators IPs and deploy them on given heterogeneous clusters of FPGAs considering the mapping performance. Then, we propose CHEF-M2A to map MMMT models to the deployed accelerators. We are the first attempt to implement end-to-end MMMT model inference in heterogeneous clusters of FPGAs which provides benchmarks and baselines for future works.

References

[1] X. Zhang, C. Hao, P. Zhou, A. Jones, and J. Hu, "H2H: Heterogeneous model to heterogeneous system mapping with computation and communication awareness," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 601–606.

[2] A. Valada, N. Radwan, and W. Burgard, "Deep auxiliary learning for visual localization and odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 6939–6946.

[3] S. Tripathi, S. Tripathi, and H. Beigi, "Multi-modal emotion recognition on IEMOCAP dataset using deep learning," 2018, *arXiv:1804.05788*.

[4] D. Fan, H. Lu, S. Xu, and S. Cao, "Multi-task and multi-modal learning for RGB dynamic gesture recognition," *IEEE Sensors J.*, vol. 21, no. 23, pp. 27026–27036, Dec. 2021.

[5] L. Hoang, S.-H. Lee, E.-J. Lee, and K.-R. Kwon, "GSV-NET: A multi-modal deep learning network for 3-D point cloud classification," *Appl. Sci.*, vol. 12, no. 1, p. 483, 2022.

[6] A. Mehler, G. Abrami, C. Spiekermann, and M. Jostock, "VAnnotatoR: A framework for generating multimodal hypertexts," in *Proc. 29th Hypertext Social Media*, 2018, pp. 150–154.

[7] A. K. Kamath, S. Abi-Karam, A. Bhat, and C. Hao, "M5: Multi-modal multi-task model mapping on multi-FPGA with accelerator configuration search," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2023, pp. 1–6.

[8] J. Zhuang et al., "CHARM: Composing heterogeneous accelerators for matrix multiply on versal ACAP architecture," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, 2023, pp. 153–164.

[9] W. Jiang, E. H.-M. Sha, Q. Zhuge, L. Yang, X. Chen, and J. Hu, "Heterogeneous FPGA-based cost-optimal design for timing-constrained CNNs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2542–2554, Nov. 2018.

[10] T. Alonso et al., "Elastic-DF: Scaling performance of DNN inference in FPGA clouds through automatic partitioning," *ACM Trans. Reconfig. Technol. Syst.*, vol. 15, no. 2, pp. 1–34, 2021.

[11] N. Tarafdar et al., "AIgean: An open framework for deploying machine learning on heterogeneous clusters," *ACM Trans. Reconfig. Technol. Syst.*, vol. 15, no. 3, pp. 1–32, 2021.

[12] "Bring video Analytics to a new level with Xilinx's VMSS and VCK5000 platform." Xilinx. 2022. [Online]. Available: https://www.xilinx.com/developer/articles/video-analytics-with-vmss-and-vck5000.html

[13] "The heterogeneous accelerated compute cluster at the University of Illinois, Urbana-Champaign," 2024. [Online]. Available: https://xilinx-center.csl.illinois.edu/xacc-cluster/

[14] "The heterogeneous accelerated compute cluster at the University of California, Los Angeles," 2024. [Online]. Available: https://wiki.ucla-vast.com/books/cluster-user-guide/page/getting-started

[15] S.-C. Kao and T. Krishna, "MAGMA: An optimization framework for mapping multiple dnns on multiple accelerator cores," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2022, pp. 814–830.

[16] S. Thuseethan, S. Janarthan, S. Rajasegarar, P. Kumari, and J. Yearwood, "Multimodal deep learning framework for sentiment analysis from text-image Web data," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol. (WI-IAT)*, 2020, pp. 267–274.

[17] X. Zhang et al., "DNNExplorer: A framework for modeling and exploring a novel paradigm of FPGA-based DNN accelerator," in *Proc. 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.

[18] H. Kwon, L. Lai, M. Pellauer, T. Krishna, Y.-H. Chen, and V. Chandra, "Heterogeneous dataflow accelerators for multi-DNN workloads," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2021, pp. 71–83.

[19] Y. Huang, C. Du, Z. Xue, X. Chen, H. Zhao, and L. Huang, "What makes multi-modal learning better than single (provably)," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 10944–10956.

[20] D. Antotsiou, G. Garcia-Hernando, and T.-K. Kim, "Task-oriented hand motion retargeting for dexterous manipulation imitation," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2018, pp. 1–15.

[21] Z. Huang et al., "FULLER: Unified multi-modality multi-task 3-D perception via multi-level gradient calibration," 2023, *arXiv:2307.16617*.

[22] "Deep-learning processor unit." 2024. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Deep-Learning-Processor-Unit

[23] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field–Programmable Gate Arrays*, 2015, pp. 161–170.

[24] W. Jiang et al., "Achieving super-linear speedup across multi-FPGA for real-time DNN inference," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5S, pp. 1–23, 2019.

[25] S. Mitchell, M. OŠSullivan, and I. Dunning, *PuLP: A Linear Programming Toolkit for Python*, Univ. Auckland, Auckland, New Zealand, vol. 65, 2011.

[26] X. Zhang, W. Jiang, and J. Hu, "Achieving full parallelism in LSTM via a unified accelerator design," in *Proc. IEEE 38th Int. Conf. Comput. Design (ICCD)*, 2020, pp. 469–477.

[27] S. Zhang et al., "A dataset and benchmark for large-scale multi-modal face anti-spoofing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 919–928.

[28] T. Shen, Y. Huang, and Z. Tong, "FaceBagNet: Bag-of-local-features model for multi-modal face anti-spoofing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 0–0.

[29] X. Li et al., "Concurrent activity recognition with multimodal CNN-LSTM structure," 2017, *arXiv:1702.01638*.

[30] "Gigabit Ethernet controller." 2024. [Online]. Available: https://docs.amd.com/r/en-US/ug1137-zynq-ultrascale-mpsoc-swdev/Gigabit-Ethernet-Controller

[31] M. Ruiz, D. Sidler, G. Sutter, G. Alonso, and S. López-Buedo, "Limago: An FPGA-based open-source 100 GbE TCP/IP stack," in *Proc. 29th Int. Conf. Field Programmable Logic Appl. (FPL)*, 2019, pp. 286–292.

[32] Y. Chi, L. Guo, J. Lau, Y.-K. Choi, J. Wang, and J. Cong, "Extending high-level synthesis for task-parallel programs," in *Proc. IEEE 29th Annu. Int. Symp. Field–Programmable Custom Comput. Machines (FCCM)*, 2021, pp. 204–213.