ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Engineering Journals
Available on IEEE Xplore®

Research Article
https://doi.org/10.1631/ENG.ITEE.2025.0186

Efficient mapping and flexible interconnects: accelerating 3D CNN-based lung nodule segmentation and classification on multi-FPGA

Zhuang CAO[1,2], Tian ZHANG[1,2], Xiaowei HE[1,2], Sheng LIU[1,2], Junzhong SHEN[1,2]✉

[1]*College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China*
[2]*Key Laboratory of Advanced Microprocessor Chips and Systems, Changsha 410073, China*

Abstract: Three-dimensional convolutional neural networks (3D CNNs) show considerable promise for lung nodule detection. However, their high computational complexity and memory demands present substantial challenges for acceleration on a single field-programmable gate array (FPGA). To address this, we propose efficient mapping schemes for a multi-FPGA platform, leveraging its massive parallelism to maximize computational efficiency. Our system, integrating six customized FPGA boards, achieves state-of-the-art performance, delivering approximately 15.9 tera operations per second (TOPS) for nodule segmentation and approximately 3.8 TOPS for nodule classification. Compared to a central processing unit baseline, it achieves a 128.2× speedup while exhibiting 6.7× higher energy efficiency than a graphics processing unit implementation. Furthermore, the system attains a state-of-the-art recall rate of 87.1% on the real-world clinical benchmark.

Key words: Lung nodule detection; Three-dimensional convolutional neural networks (3D CNNs); Field-programmable gate array (FPGA); Multi-FPGA systems; Hardware acceleration; Parallel mapping

1 Introduction

Lung cancer continues to be one of the most prevalent and lethal malignancies globally, with recent clinical statistics underscoring its significant impact on public health (Siegel et al., 2024, 2025). Early and accurate detection of lung nodules via computed tomography (CT) scans is vital for improving patient survival rates.

Historically, interpreting volumetric CT scans requires radiologists to meticulously analyze hundreds of slices per patient, a process that is not only time-consuming but also prone to human error due to fatigue. To address these challenges, computer-aided detection (CAD) systems have been developed to automate localization and diagnosis of potential lesions (Messay et al., 2010; Tan et al., 2011; Isensee et al., 2021; Lin et al., 2024).

High-performance CAD systems typically decompose the workflow into two pivotal sub-procedures: segmentation and classification (Cui et al., 2025; Wang JX et al., 2025). The precision of both stages is fundamental to the overall diagnostic accuracy and clinical utility of the system.

In recent years, deep learning—particularly convolutional neural networks (CNNs)—has achieved remarkable success in medical image analysis (Litjens et al., 2017; Cui et al., 2025; Qin et al., 2025; Wu et al., 2025). However, early CNN-based methods often convert 3D medical volumes into 2D images to reduce computational overhead (Kuang et al., 2025), inevitably losing critical depth information and spatial continuity inherent in volumetric imaging.

To address these limitations, 3D CNN architectures have been proposed for dense volumetric segmentation, achieving superior accuracy (Shen et al., 2018b). Despite their performance advantages, 3D CNNs exhibit significantly higher computational complexity and memory bandwidth requirements than 2D CNNs—often by an order of magnitude. Consequently, the high inference latency of 3D CNNs remains a

✉ Junzhong SHEN, shenjunzhong@nudt.edu.cn
Zhuang CAO, https://orcid.org/0000-0001-6079-0313
Junzhong SHEN, https://orcid.org/0000-0001-6233-6800
CLC number: TP391.4

Received: Dec. 24, 2025; Revision accepted: May 15, 2026;
Crosschecked: May 21, 2026

Academic Paper of the 28th Annual Meeting of the China Association for Science and Technology

primary barrier to real-time clinical deployment.

Field-programmable gate arrays (FPGAs) provide an attractive solution for deep learning acceleration, offering a good balance of energy efficiency, low latency, and reconfigurability compared with central processing units (CPUs) and graphics processing units (GPUs) (Aydonat et al., 2017; Zhang C et al., 2019). Although single-FPGA implementations have successfully accelerated 3D kernels (Shen et al., 2018b), modern deep medical models often exceed the on-chip memory and logic capacity of a single device. Multi-FPGA platforms offer a scalable alternative, but existing multi-FPGA CNN accelerators often rely on rigid interconnect topologies, such as simple ring or linear array (Zhang C et al., 2016; Geng et al., 2018; Jiang WW et al., 2019; Shen et al., 2019; Zhang WT et al., 2019), leading to severe communication bottlenecks during inter-layer synchronization.

Inspired by the successful deployment of deep neural networks (DNNs) in cloud-scale FPGA data centers (Fowers et al., 2018; Microsoft, 2019), this work focuses on accelerating 3D CNN-based lung nodule detection using a multi-FPGA platform with a highly flexible topology. A preliminary version of this work was presented at the 56$^{\text{th}}$ Annual Design Automation Conference (Shen et al., 2019), and this paper substantially extends our preliminary studies (Shen et al., 2019, 2020) in three key aspects.

1. Topology innovation: a hybrid interconnect architecture with dedicated fast routing paths to alleviate multi-FPGA communication bottlenecks;

2. Algorithm advancement: a novel hybrid mapping strategy integrating model and data parallelism, beyond layer-wise mapping;

3. Pipeline integration: an end-to-end unified pipeline tightly coupling segmentation and classification.

These improvements deliver a throughput gain over the prior work (Shen et al., 2019) and a state-of-the-art recall rate of 87.1% on the LUNA16 benchmark. To the best of our knowledge, this work is the first end-to-end multi-FPGA acceleration framework for lung nodule detection that seamlessly integrates 3D CNN-based segmentation and classification on a multi-FPGA platform. Recent multi-FPGA studies in medical imaging (2023–2025) have focused on single-FPGA acceleration (Khan et al., 2023) and 2D CNN models (Lu et al., 2024), or on surveys only (Diaconu et al., 2025; Cheng, 2025), without realizing unified pipeline integration of the two core stages.

The key contributions of this paper are summarized as follows:

1. An end-to-end 3D CNN-based lung nodule detection pipeline optimized for multi-FPGA execution, integrating high-precision lung nodule segmentation network (LNS-net) and lung nodule classification network (LNC-net);

2. A flexible multi-FPGA architecture with hybrid interconnects and direct inter-FPGA fast paths, reducing communication latency by up to 17%;

3. Hybrid parallelism mapping schemes tailored to network characteristics, balancing workloads and maximizing resource utilization;

4. Multi-level communication optimizations that achieve over 80% accelerator utilization by overlapping computation and transmission.

2 Related works

2.1 3D CNN accelerators

FPGA-based acceleration of 3D CNNs has been extensively studied and has attracted growing attention, primarily for medical imaging (Shen et al., 2018b, 2020; Basalama et al., 2023) and video analysis (Hegde et al., 2018; Chen et al., 2019; Fan et al., 2019; Wang YC et al., 2019; Gao et al., 2025). Early works proposed uniform template-based accelerators supporting both 2D and 3D CNNs via algorithm–hardware co-design and optimized data reuse (Shen et al., 2018b). Later studies further improved the efficiency by fusing consecutive convolution layers and exploiting deep pipelining (Shen et al., 2020).

Several comprehensive surveys have reviewed FPGA-based CNN accelerators (Jiang JY et al., 2025; Varughese and Sridevi, 2025), offering insights into dynamic parallelism, pruning, quantization, and fast convolution algorithms that motivated our adoption of a sparsity-optimized Winograd algorithm. Advanced quantization schemes, such as static block floating-point representation (Fan et al., 2023), further complement our 8-bit fixed-point design by reducing logic resource usage with negligible accuracy loss. While prior works have also explored 3D CNN acceleration for video understanding (Hegde et al., 2018; Chen et al., 2019; Fan et al., 2019; Wang YC et al., 2019; Khan et al., 2023), most existing designs target single-FPGA platforms and face scalability bottlenecks when handling large models, motivating our scale-out approach based on multi-FPGA systems.

2.2 Scale-out acceleration for CNNs

As CNN models grow deeper and more complex, scale-out acceleration using multi-FPGA platforms has become increasingly important (Zhang C et al., 2016; Jiang WW et al., 2019; Shen et al., 2019; Liu et al., 2022; Lu et al., 2024). Early designs use deeply pipelined ring or star topologies (Zhang C et al., 2016; Shen et al., 2019), while Jiang WW et al. (2019) demonstrated super-linear speedup for DNN inference across multiple FPGAs. Recent works introduce automated end-to-end design flows for large-scale CNNs across multi-FPGA clusters (Lu et al., 2024), and bandwidth utilization and efficient computing engines have also been emphasized (Cheng, 2025). In industrial settings, FPGA clusters such as Microsoft's Project Brainwave have been deployed for large-scale DNN inference (Fowers et al., 2018; Microsoft, 2019), but mainly target 2D CNNs or recurrent networks.

While existing multi-FPGA solutions offer promising scalability, most focus on 2D CNNs or use rigid interconnect topologies, leaving 3D CNN-based medical image processing under-optimized. Our work extends these foundations by introducing flexible inter-FPGA connections and efficient mapping schemes to minimize communication overhead, presenting the first end-to-end multi-FPGA acceleration framework for 3D CNN-based lung nodule segmentation and classification.

3 Background

3.1 3D CNN-based lung nodule detection

The workflow for 3D CNN-based lung nodule detection, depicted in Fig. 1, comprises two principal stages: segmentation and classification, each facilitated by a dedicated 3D CNN. The pipeline commences with the application of image masking to raw 3D CT scans to delineate the region of interest (ROI) by excluding non-lung structures. Subsequent data preprocessing yields input images that are partitioned into fixed-size blocks for the segmentation module. This module outputs probability maps, which are then processed by connected component analysis to identify candidate nodules. The final classification stage evaluates these candidates to ascertain their status.

The architecture of LNS-net, the 3D CNN devised for lung nodule segmentation (LNS), is presented in Fig. 2. The network consists of 3D convolution (CONV), transposed convolution (TCONV), max-pooling (POOL), batch normalization (BN), and rectified linear unit (ReLU) activation layers. For the first dense block, the preceding convolution layer yields 64 input feature maps feeding into the block. Within each dense block, six $3 \times 3 \times 3$ CONV layers are implemented, each with subsequent BN and ReLU operations, and every internal convolution layer generates 12 new feature maps as the fixed growth rate. The feature maps are resized by transition layers, which are positioned between dense blocks and employ convolution and max-pooling. Each transposed convolution layer is configured with a $3 \times 3 \times 3$ kernel and is followed by BN and a ReLU layer.

LNS-net adopts a high-level architecture comparable to U-net (Jiang WW et al., 2019), comprising contracting and expansive paths. A principal distinction lies in the contracting path, which is constructed from repeated dense blocks in lieu of unpadded convolutions. Moreover, the expansive path is implemented entirely with TCONV layers. Similar to U-net's skip connections, the inputs to most TCONV layers are sourced from two streams: the output of the immediately preceding layer and the feature maps from the corresponding dense block in the contracting path. A critical deviation is the fusion method, wherein the two data sources are combined by element-wise addition (denoted by the $\oplus$ symbol in Fig. 2), as opposed to the concatenation operation utilized in U-net. The

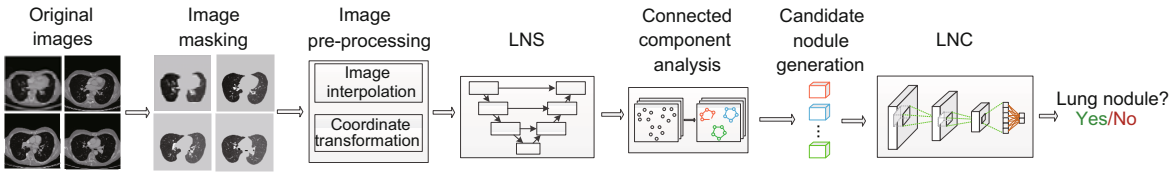

Fig. 1 Procedure of lung nodule detection

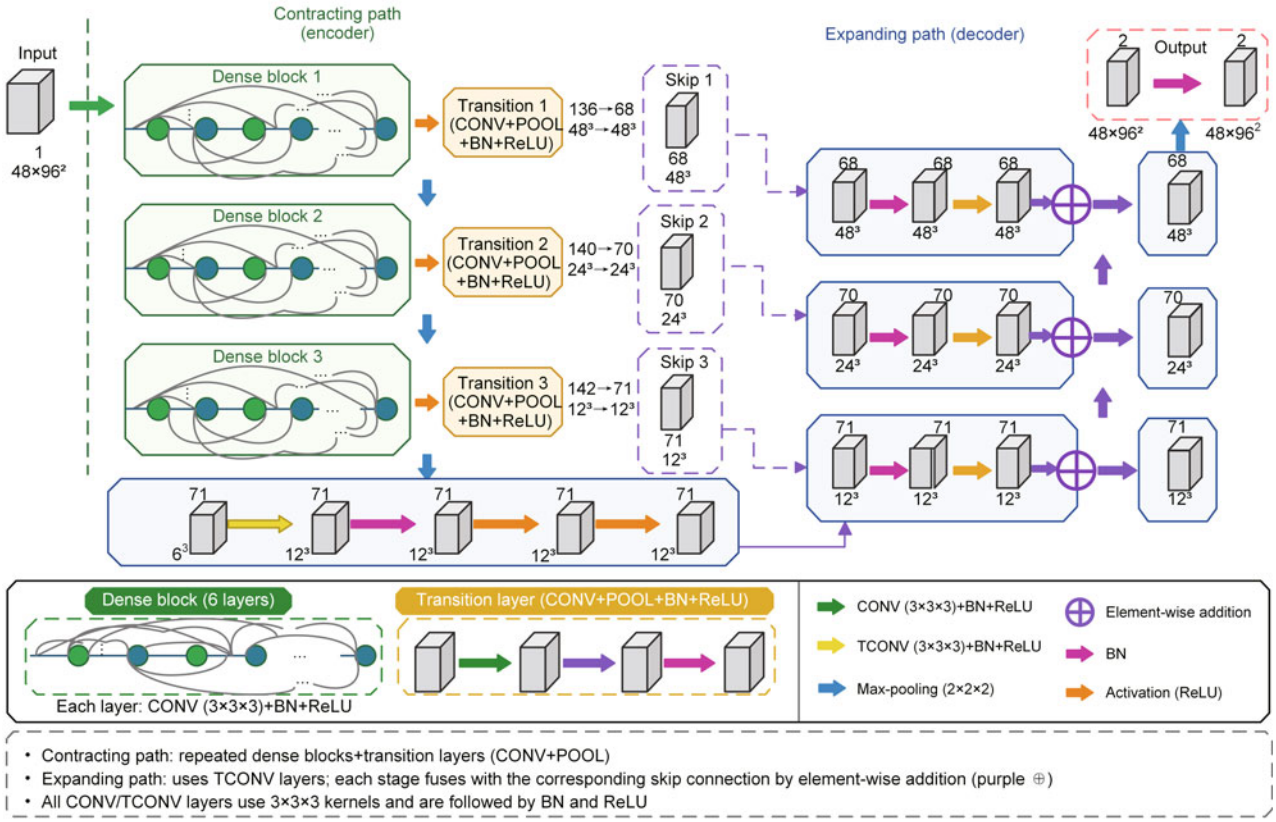

Fig. 2 Architecture of the proposed 3D CNN for LNS (LNS-net)

resultant complexity of LNS-net introduces significant challenges for its implementation on FPGAs.

In contrast to the computationally intensive LNS-net, the network designed for LNC employs a comparatively lightweight architecture. As illustrated in Fig. 3, the network is structured around three CONV blocks. Each block contains two CONV layers succeeded by a max-pooling operation. The architecture terminates in a fully connected (FC) layer, which receives the output of the final CONV block to yield the classification decision.

3.2 Algorithm analysis

3.2.1 Execution time distribution of sub-procedures

To assess the computational profile of the proposed algorithm, we measured the execution time of each sub-procedure. The results, obtained by processing a single image through multiple iterations on an NVIDIA GeForce RTX 2080 GPU, are presented in Table 1. The analysis reveals that LNS constitutes the most computationally intensive step, accounting for over 88% of the total time. Furthermore, LNS and LNC collectively represent more than 98% of the computational load, which justifies our dedicated effort to accelerate these two components.

It should be noted that the profiling results in Table 1 are obtained using 32-bit floating-point precision without hardware-specific optimizations, serving as a representative baseline for understanding the computational distribution across sub-procedures. This baseline is intentionally distinct from the optimized 8-bit fixed-point implementations reported in later sections, and the absolute time values are not directly compared.

Table 1 Time distributions of sub-procedures in lung nodule detection

Sub-procedure	Time (s)	Percentage
IM	0.45	0.4%
IPP	1.02	0.8%
LNS	111.55	88.5%
CCA	0.52	0.4%
LNC	12.51	9.9%

Time values are reported in seconds per full CT scan (512×512×300 voxels) using 32-bit floating-point precision. Percentages indicate the proportion of the total processing time attributable to each sub-procedure. IM: image masking; IPP: image pre-processing

3.2.2 Dense blocks

The architecture of the dense blocks in LNS-net is based on DenseNet (Huang G et al., 2017). A key characteristic of an L-layer dense block ($L = 6$ for LNS-net) is that the l^{th} layer receives l inputs. These inputs comprise two parts: the initial input feature maps to the dense block (shared across all layers) and the output feature maps produced by each of the preceding CONV layers. These inputs are concatenated to form the composite input for the l^{th} layer. All CONV layers within the dense blocks are configured to generate an identical number of output feature maps (i.e., 12 for LNS-net). Notably, since the input of a dense block is shared among all constituent CONV layers, each layer can initiate computation in advance without waiting for the output of preceding layers. Once the output of a CONV layer within the dense block is generated, all subsequent layers can proceed with computation concurrently—this architecture thus exhibits a high degree of inherent parallelism.

3.2.3 Transposed convolution layers

The TCONV operation is the reverse of standard convolution. Since the underlying procedures of CONV and TCONV layers are thoroughly documented in the literature, a detailed exposition is omitted here for brevity. This work instead focuses on the analysis of parallelism inherent in TCONV layers. The pronounced spatial separation between their dual data sources results in intricate dependencies. However, the distributivity of convolution over addition enables parallel computation for TCONV layers, as expressed in Eq. (1):

$$O = (\boldsymbol{i} + \boldsymbol{i}')\boldsymbol{w} = \boldsymbol{i}\boldsymbol{w} + \boldsymbol{i}'\boldsymbol{w}, \tag{1}$$

where $\boldsymbol{i}$ and $\boldsymbol{i}'$ represent two independent input feature maps to a TCONV layer (one from the preceding layer and one from the corresponding dense block), $\boldsymbol{w}$ denotes the 3D filter, and O is the output feature map. This equality holds due to the distributivity of convolution over addition, allowing the TCONV

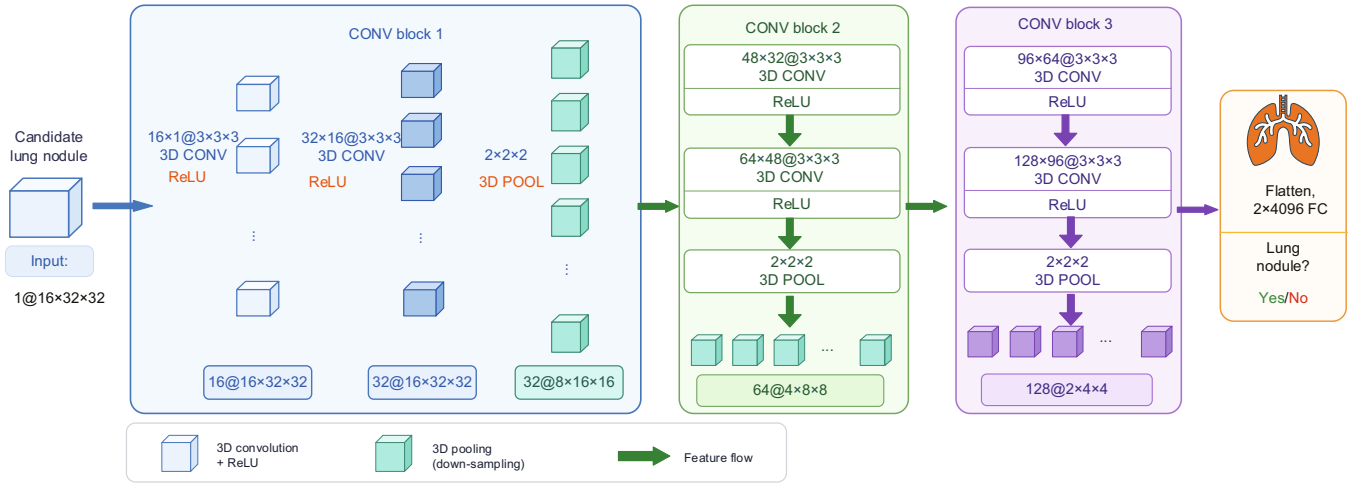

Fig. 3 Architecture of the proposed 3D CNN for LNC (LNC-net)

computation to start as soon as the dense-block output is ready without waiting for the preceding layer's result.

3.3 3D Winograd minimal algorithm with sparse inputs

3.3.1 Overview of the 3D Winograd algorithm

The Winograd algorithm has been utilized in many works (Dey et al., 2018; Shen et al., 2018b) for CNN acceleration on FPGAs, owing to its ability to lower computational complexity and conserve hardware resources such as digital signal processor (DSP) slices. Shen et al. (2018b) presented the first implementation of a 3D Winograd algorithm for FPGA-based 3D CNN acceleration, which is formulated as

$$\begin{cases} F(m^3, r^3) = (\boldsymbol{M}\boldsymbol{m}\boldsymbol{M}^{\mathrm{T}})^{\mathrm{R}}\boldsymbol{M}^{\mathrm{T}}, \\ \boldsymbol{m} = ((\boldsymbol{X}\boldsymbol{x}\boldsymbol{X}^{\mathrm{T}})^{\mathrm{R}}\boldsymbol{X}^{\mathrm{T}}) \odot ((\boldsymbol{W}\boldsymbol{w}\boldsymbol{W}^{\mathrm{T}})^{\mathrm{R}}\boldsymbol{W}^{\mathrm{T}}), \end{cases} \tag{2}$$

where $\boldsymbol{x}$ is an input tile of size $(m+r-1)^3$, $\boldsymbol{w}$ is a filter tile of size r^3, the scalar m (in $F(m^3, r^3)$) denotes the output tile size, and the bold symbol $\boldsymbol{m}$ represents the intermediate transformed tile. $\boldsymbol{X}$, $\boldsymbol{W}$, and $\boldsymbol{M}$ are constant transformation matrices determined by m and r (following the notation in Shen et al. (2018b)), and R represents a 90° rotation operation. $\odot$ represents the element-wise multiplication (Hadamard product).

3.3.2 Sparsity in transposed convolution

The adoption of the 3D Winograd algorithm is highly advantageous for reducing the computational complexity of LNS-net, given that CONV and TCONV layers account for over 90% of its operations. A significant challenge arises when applying this algorithm to TCONV layers, as transposed convolution requires explicit zero insertion between input elements before computation (Fig. 4a). These artificially inserted zeros introduce numerous invalid multiplications and redundant calculations in subsequent Winograd transforms. The intermediate data tiles within the Winograd transformation exhibit fixed sparsity patterns (Fig. 4b, zeros shown in blue), with the overall sparsity level reaching approximately 87.5% zeros. This structured sparsity originating from the zero insertion of transposed convolution (Fig. 4a) can be fully exploited to simplify transformation procedures and skip all redundant zero-related arithmetic operations. A detailed mathematical justification of the fixed sparsity pattern is provided in Section S1 in the electronic supplementary materials (ESM).

Our sparse-optimized transformation templates based on the Winograd algorithm achieve 54.8% adder reduction (complete resource data are provided in Section S2 in the ESM).

4 System architecture design

4.1 System overview

Fig. 5a presents an overview of the multi-FPGA system designed to accelerate 3D CNN-based LNS. The system comprises a host CPU, multiple FPGA nodes, and an interconnection network utilizing a central switch. The host CPU accesses the double data rate synchronous dynamic random access memory (DDR) of each FPGA node directly through the peripheral component interconnect express (PCIe) interfaces (PCIe IFs), enabling efficient loading of initial data from main memory into the FPGA nodes' DDR and retrieval of final results. Additionally, the CPU configures the network interface (NI) modules via the PCIe IFs to manage inter-FPGA communication links.

This work adopts the scalable star topology introduced in Shen et al. (2019), but extends it by incorporating direct inter-FPGA connections—referred to as the fast path—to support low-overhead data communication. Figs. 5b and 5c demonstrate the application of the fast path in two classical

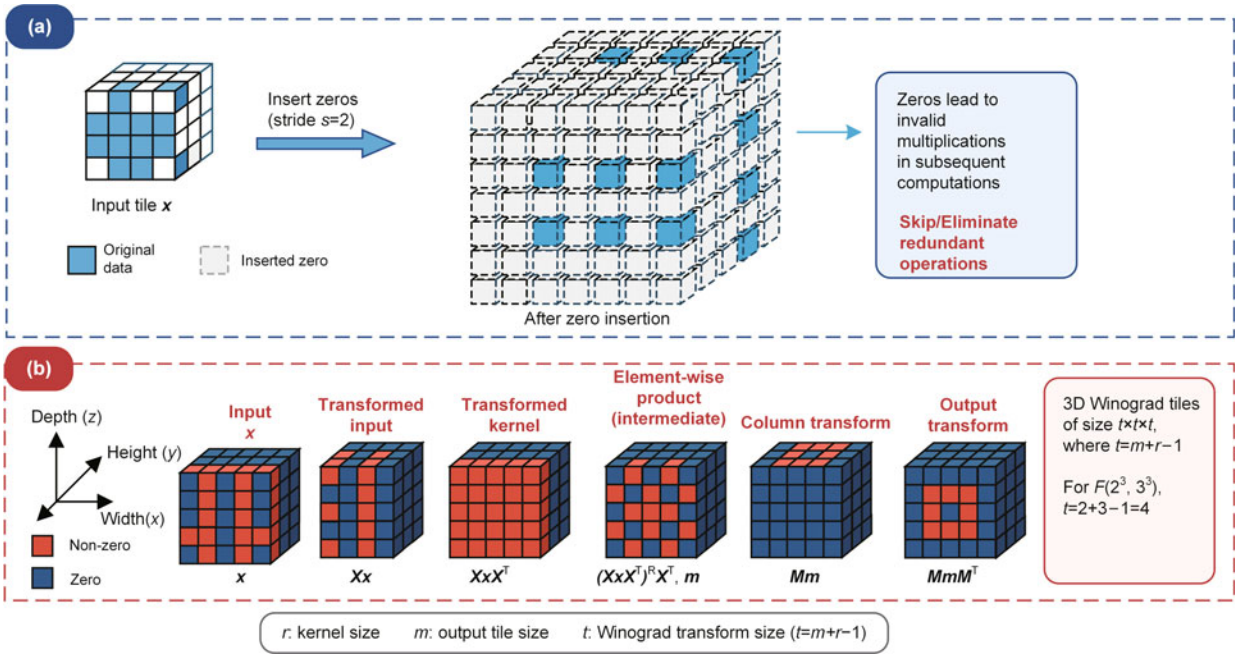

Fig. 4 Zero padding for transposed convolution (a) and sparsity patterns in the Winograd domain $F(2^3, 3^3)$ (b)

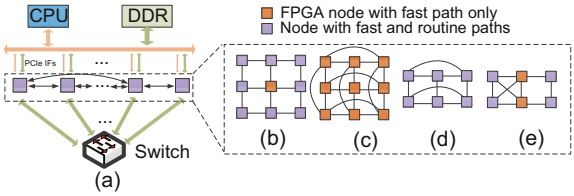

Fig. 5 Overview of the multi-FPGA system: (a) the system architecture; (b)–(e) different interconnect topologies among FPGA nodes

network topologies. Under the constraint of at most four interconnect interfaces per FPGA node, a node can connect to the switch only when it uses fewer than four direct FPGA-to-FPGA links. This study focuses on the hybrid topologies in Figs. 5d and 5e, where most nodes possess both routine path and fast path connections, ensuring robust local and global interconnectivity.

The system consists of $N = 6$ FPGA nodes, each equipped with four quad small form-factor pluggable plus (QSFP+) cages supporting up to 100 Gb/s per link. One port per node connects to a central switch via a 40-GbE routine path (GbE is short for gigabit Ethernet), and the remaining three ports are interconnected with other nodes via 100-GbE fast paths using optical cables. Detailed routing rules and latency breakdown are provided in Section S3 in the ESM. Data transmission over the routine path employs the standard transmission control protocol/Internet protocol (TCP/IP), while fast path uses a custom lightweight protocol optimized for low-latency communication between FPGA nodes. Accordingly, two types of network interface modules are implemented per FPGA node (refer to Section 4.3 for details).

4.2 Accelerator design

Fig. 6 presents an overview of the proposed 3D TCONV accelerator. Although the overall workflow is analogous to that in Shen et al. (2018b), the transformation of input data and multiplication products employs optimized templates. Specifically, the transform templates for $\boldsymbol{X}$ (input transform) and $\boldsymbol{M}$ (intermediate matrix transform) from the original Winograd implementation in Shen et al. (2018b) are replaced with our optimized templates for $\boldsymbol{X}'$, $\boldsymbol{X}''$, and $\boldsymbol{M}''$, which were first introduced in our prior work (Shen et al., 2019). Compared with the original templates in Shen et al. (2018b), our optimized approach reduces adder usage by 25%–75% and eliminates all redundant operations. Furthermore, two additional sparsity pattern identifier (SPI) modules are integrated to locate non-zero data in the register files, with all SPI modules sharing input signals from the tile index generator (TIG). Notably, the proposed architecture is also compatible with 3D CONV acceleration by bypassing components such as SPI and TIG and replacing the transformation templates with those reported in Shen et al. (2018b).

4.3 Network interface modules

As described in the preceding section, two distinct types of NI modules are implemented in each FPGA node to accommodate different data transmission requirements. One module adheres to the standard TCP/IP, whereas the other employs a customized lightweight protocol introduced in this work.

4.3.1 TCP/IP-based network interface module

The NI module manages communication links, packet encapsulation/parsing, and reliable transmission via TCP/IP (Shen et al., 2018a), using a standard three-way handshake for session initialization and packet retransmission to guarantee correctness. The principal components of the NI module are specified below.

Packet analysis (PA) validates Ethernet, IP, and TCP headers, detects packet errors and loss, and delivers valid

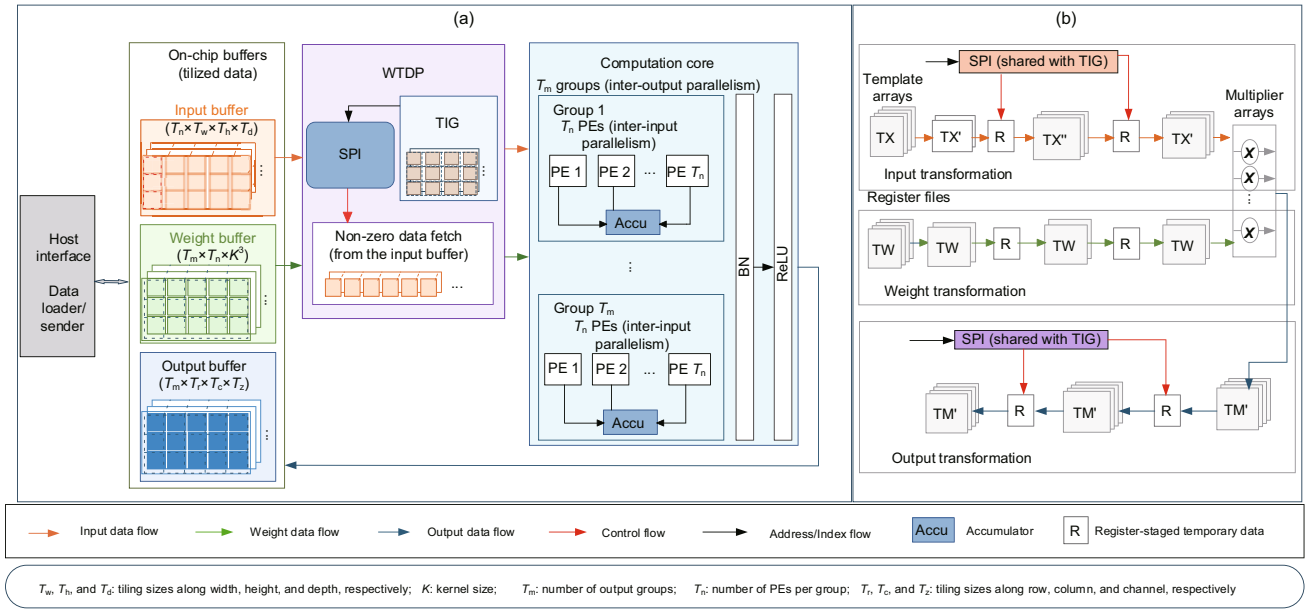

Fig. 6 Overview of the 3D TCONV accelerator: (a) block diagram of the accelerator architecture, where input and weight data are transformed using optimized Winograd templates (Shen et al., 2019) that achieve 25%–75% adder reduction over the original design (Shen et al., 2018b); (b) workflow of a single PE

payloads to the data first-in first-out (FIFO).

Session check (SC) consults the TCP table to route payloads destined for the current FPGA node to the ingress module, and requests retransmission upon detecting checksum failure or packet loss.

Session management agent (SMA) manages connection establishment/termination, records session information in the TCP table, and maintains a retransmission timer for timeout-based recovery.

Packet encapsulation (PE) constructs outgoing packets by appending headers to payloads from the egress module's data FIFO.

The NI module also integrates a 40-GbE media access control (MAC) unit as the physical and link layer interface. A 100-GbE MAC is under development for the platform's four 100-GbE links.

4.3.2 Network interface module with lightweight network transmission protocol

To minimize the data transmission overhead on the fast path between FPGA nodes, a customized lightweight network transmission protocol was designed. Similar to user datagram protocol (UDP), this protocol operates without handshaking between sender and receiver nodes. It forgoes packet retransmission in cases of loss or corruption, trading reliability for lower latency. This design is suitable for high-speed, low-congestion links. In our system, the inter-FPGA connections utilize dedicated 100 Gb/s links without intermediate congestion, making the lightweight protocol an ideal fit.

Fig. 7 illustrates the customized packet format, which is a simplified derivative of the Ethernet frame. Each packet comprises a 12-byte header and a payload of up to 1024 bytes. The header includes "Sid" and "Did" fields to identify the sender and receiver FPGA nodes, "on/off" and "DAddr" fields to specify the payload destination address, and a "total length" field defining the packet size in bytes. Additional fields facilitate packet loss and error detection.

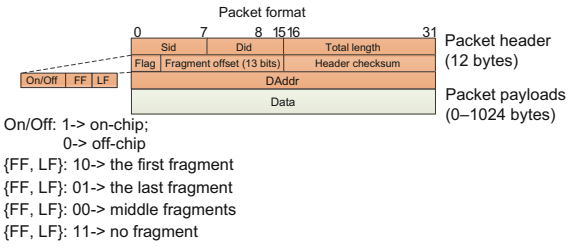

Fig. 7 Packet format. FF: first fragment; LF: last fragment

The architecture of the lightweight NI (LNI) module closely resembles its standard counterpart shown in Fig. 8. The key modifications include the removal of session management modules and the replacement of the TCP table with a node registration table (NRT). Furthermore, a 100-GbE MAC is integrated at the link layer instead of a 40-GbE MAC. The NRT records each node's adjacency relationships (connected nodes, ports, and connection path types "R"/"F") and is shared among all local LNI modules, as illustrated in the upper section of Fig. 9.

Fig. 9 illustrates the packet reception workflow within the LNI module. The lightweight packet analysis module extracts header information and detects packet loss or errors. It then performs an NRT lookup to determine whether the packet is destined for the current node. In the event of packet loss, the module duplicates the current payload as a compensation mechanism, storing the copy in a data duplication FIFO to avoid the overhead of retransmissions.

5 Implementations

5.1 Overall acceleration framework

We offload computationally intensive workloads—specifically, the computations associated with LNS-net and LNC-net—to FPGAs, while the host CPU undertakes fewer computationally demanding tasks such as image pre-processing and system management. Fig. 10 illustrates the complete end-to-end pipeline, and Table 2 summarizes the latency and data transfer breakdown.

The processing pipeline operates in four stages: (1) pre-processing on CPU; (2) LNS-net segmentation on FPGA1–4 using model parallelism; (3) connected component analysis (CCA) on CPU; (4) LNC-net classification on FPGA5–6 using data parallelism. Detailed stage-by-stage operations are provided in Section S4 in the ESM. The total end-to-end latency for a full CT scan is about 1.3 s. The CPU acts as a coordination hub between FPGA stages, performing tasks that are inefficient on FPGAs (pre-processing and CCA) while managing data movement. The data size reduction from 6 MB to 0.5 MB after CCA minimizes PCIe bandwidth contention.

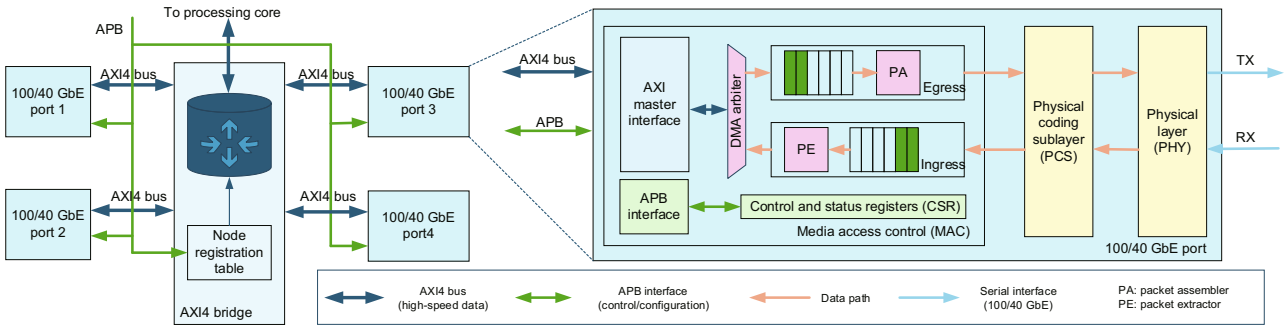

Fig. 8 Block diagram of the network interface module. RX: receive; TX: transmit; APB: advanced peripheral bus

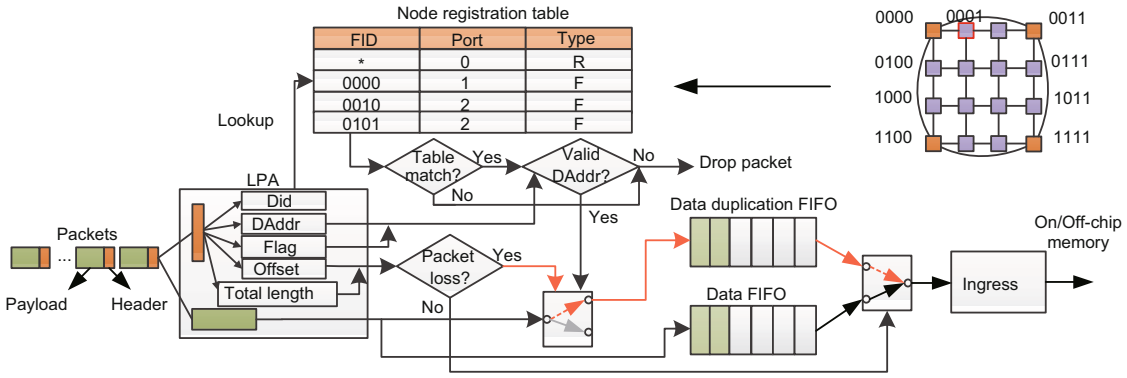

Fig. 9 Workflow of the proposed LNI module. FID: frame identifier; LPA: lightweight protocol address. * denotes a wildcard (i.e., any value matches)

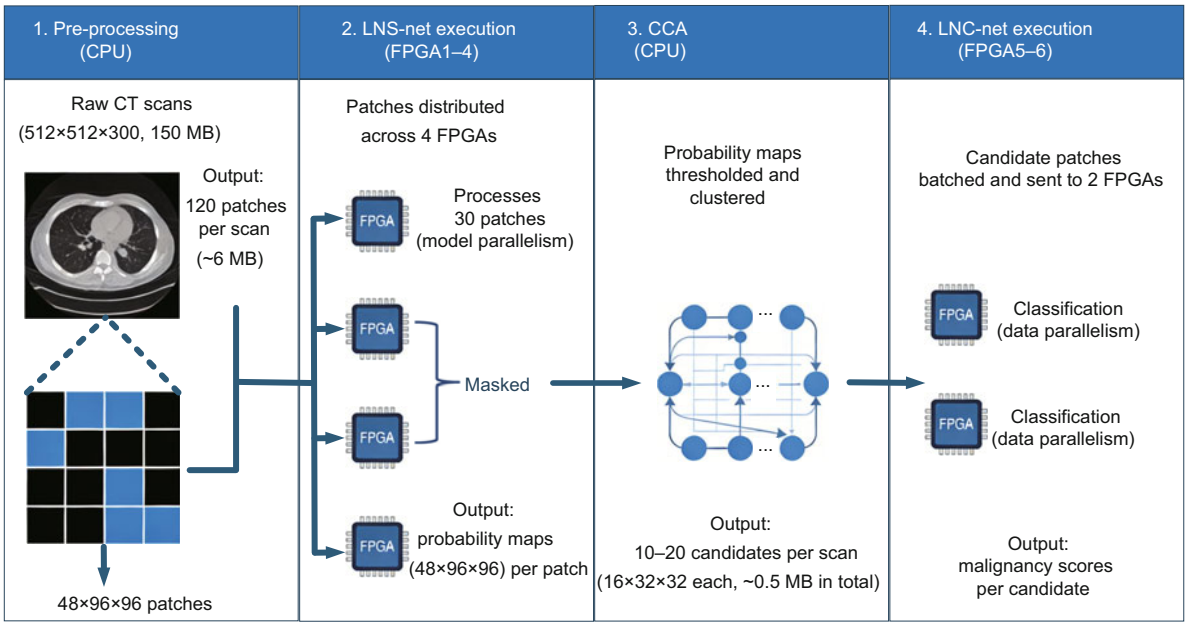

Fig. 10 End-to-end pipeline with data transfer details

Table 2 End-to-end latency and data transfer breakdown

Stage	Processing unit	Data size (per scan)	Transfer path	Latency
Pre-processing	CPU	150 MB → 6 MB	DRAM → CPU cache	45.0 ms
LNS-net	FPGA1–4	6 MB	PCIe → FPGA DDR	1.25 s
CCA	CPU	6 MB → 0.5 MB	FPGA DDR → PCIe → DRAM	1.5 ms
LNC-net	FPGA5–6	0.5 MB	PCIe → FPGA DDR	6.0 ms
Total				~1.30 s

DRAM: dynamic random access memory

5.2 CNN mapping scheme for LNS-net

We propose a workload-balancing mapping scheme for dense blocks across multiple accelerators, which forms the foundation of the overall LNS-net mapping strategy. The detailed load-balancing procedure is provided in Section S7 in the ESM; here, we summarize its key principles.

Workloads are partitioned into $\frac{n}{p}$ groups, where n and p represent the number of layers per dense block and the number of accelerators, respectively. During each iteration, once an accelerator f_j completes its current task, it immediately broadcasts the results to all other accelerators. The unassigned workload with the highest computational cost from the subsequent group is then assigned to f_j. This dynamic load-balancing approach ensures equitable execution time across accelerators. Each FPGA node maintains a local result buffer for each layer it computes. When accelerator f_j completes a layer, it writes the output feature maps to its local on-chip buffer, multicasts the results to all other nodes via the switch (routine path), and waits for an acknowledgment bitmap from all receivers before marking the transmission as complete. Receiver nodes employ double-buffering for incoming inter-layer data and enforce barrier synchronization via layer dependencies.

The computational cost of each CONV layer in a dense

block is defined as $C(j) = \alpha \cdot \mathrm{OPs}(j) + \beta \cdot \mathrm{Dep}(j)$, where $\mathrm{OPs}(j)$ is the floating-point operation count and $\mathrm{Dep}(j)$ captures synchronization overhead ($\alpha = 1.0$ and $\beta = 0.2$ in our implementation). This hybrid metric ensures that both arithmetic intensity and synchronization delay are considered during workload assignment.

A key design decision involves mapping p accelerators onto f FPGA nodes. We adopt $p = f$ (one accelerator per FPGA node), achieving 18% higher system throughput compared to that of a hypothetical $p > f$ design. A quantitative comparison is provided in Section S5 in the ESM. This strategy distributes control logic and on-chip buffer management across all nodes, reducing individual node complexity.

For mapping multiple dense blocks, we assign a dedicated accelerator to each block on every FPGA node, preventing underutilization from reconfiguring a single accelerator for blocks with differing parameters. Similarly, distinct accelerators are implemented for each TCONV layer due to their parameter variations. Within an FPGA node, these accelerators are organized into a computational pipeline for independent operation; across different nodes, accelerators operate in parallel.

As a case study, we demonstrate the mapping scheme for LNS-net using four FPGA nodes ($p = f = 4$), as shown in Fig. 11. Execution is organized into seven pipeline stages covering the initial CONV layer, dense-block computation, inter-node communication, and subsequent TCONV layers, enabling multiple images to be processed concurrently. A detailed walk-through of each stage (initialization, dense-block workload assignment, and TCONV layer computation) is provided in Section S6 in the ESM. Dense-block computation, packet transmission, and TCONV processing proceed concurrently, improving pipeline utilization and reducing accelerator idle time. Double-buffering is employed to hide data loading/storing behind computation, with inter-stage double buffers implemented in external memory.

Fig. 12 presents the block diagram of the system on chip (SoC) in each FPGA node. A main controller module allocates computational tasks among accelerators and is directly configured by the host CPU via PCIe buses. Each accelerator is equipped with a task scheduler module to govern the execution of multiple computational tasks. All accelerators access external memory through the memory-mapped AXI4 bus and utilize on-chip buffers for inter-layer data. Multiple NI modules are introduced to eliminate resource contention, ensuring that each accelerator has exclusive access to a dedicated module. An optimization scheme for NI module type assignment is detailed in Section 5.4.

5.3 CNN mapping scheme for LNC-net

As discussed, LNC-net is mapped onto multiple FPGA nodes using a data parallelism strategy, where each participating node stores the complete set of model weights and processes different batches of candidate lung nodule images (size: $16 \times 32 \times 32$). Since the CONV layers within LNC-net exhibit significant size variations, employing a single accelerator for all layers would lead to low computational efficiency (Abadi et al., 2016). To address this, we assign a dedicated accelerator to each CONV block, as the two CONV layers within a block are of similar size (differing primarily in the number of input/output channels, as shown in Fig. 3). Accelerators for different blocks can be configured with distinct parameters (e.g., parallelism degree). Each accelerator executes CONV layers of its corresponding block sequentially.

To enhance throughput, the accelerators are organized into a computational pipeline. The entire set of LNC-net weights and intermediate data is stored in on-chip buffers to minimize access latency, and double-buffering is employed to support concurrent read/write operations. Fig. 13 illustrates the data flow of four consecutive images through the

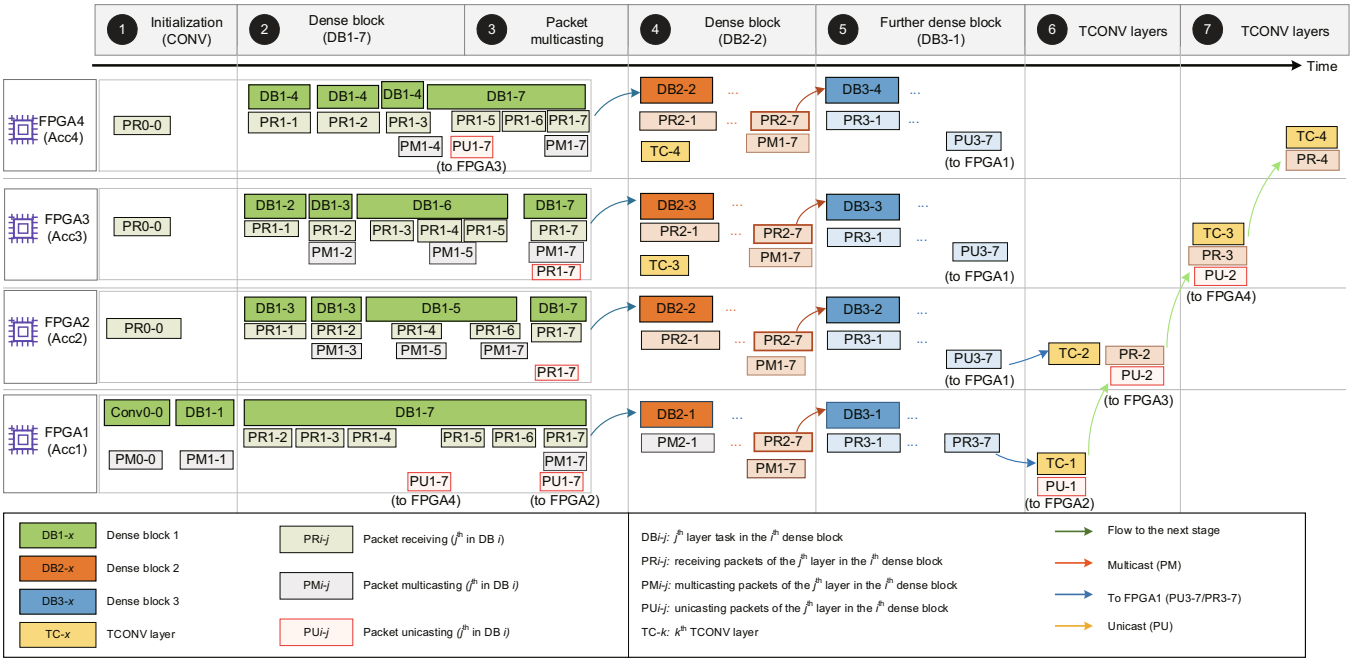

Fig. 11 Timing graph of the proposed mapping scheme. Conv0-0 denotes the initial CONV layer

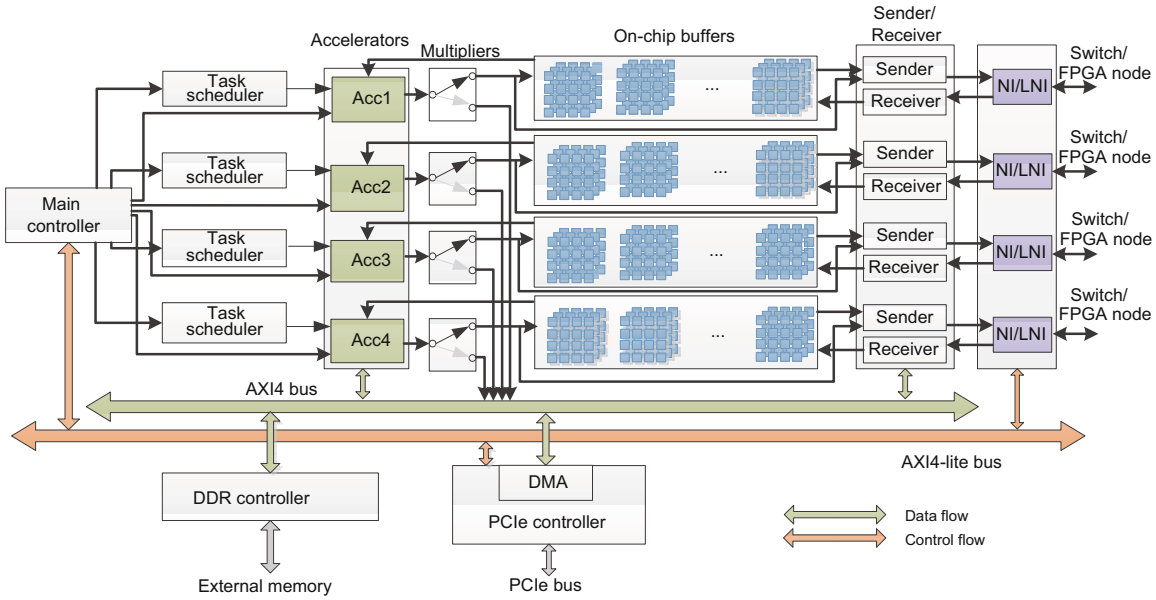

Fig. 12 SoC block diagram for LNS-net acceleration

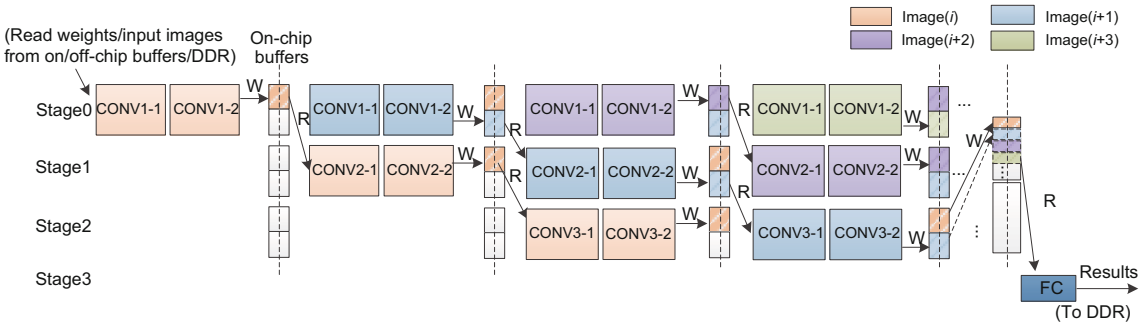

Fig. 13 Data flow among the accelerators for the LNC-net. Here, W and R denote write and read operations, respectively

accelerators within an FPGA node (weight data movement is omitted for clarity). As shown, only the input of CONV1-1 and the output of the FC layer involve off-chip memory access; all other inter-layer data reside in on-chip buffers, promoting high reuse and significantly reducing memory access overhead. Furthermore, the FC layer processes a batched set of images to increase weight reuse, transforming its computation from matrix–vector to more efficient matrix–matrix multiplication.

The architecture of the 3D TCONV accelerator is repurposed for LNC-net, with the SPI and Winograd tile data prefetcher (WTDP) modules bypassed due to the absence of input sparsity. Similarly, the PE designs adopt the templates from (Shen et al., 2018b). For the FC layer, we implement a multi-array matrix-multiplication architecture based on the one in Microsoft (2018), which offers high configurability, allowing its parallelism to be tuned to match the throughput of the preceding CONV accelerators.

5.4 Inter-FPGA communication optimizations

The use of reliable TCP/IP on the routine paths inherently introduces communication overhead between FPGA nodes. Optimizing these inter-FPGA communications is therefore critical to minimizing accelerator idle time and improving the overall computational efficiency of the system. This work implements optimizations at the hardware, scheduling, and network levels. It is important to note that, as established in Section 4.2, the FPGA nodes accelerating LNC require no inter-node communication; thus, the following optimizations apply solely to the nodes handling LNS.

1. Hardware-level optimization. We reduce inter-FPGA communication latency by modifying the NI module's workflow: the receiver sends a single acknowledgment (ACK) for a batch of packets instead of per-packet ACKs. Consequently, the sender transmits multiple packets before awaiting a collective ACK, drastically reducing waiting time. This approach assumes stable communication links, as it may increase the cost of retransmissions in case of errors.

2. Schedule-level optimization. The core principle is to overlap data transmission with accelerator computation. Specifically, the transmission of tiled result data begins immediately upon their generation by an accelerator, rather than waiting for all results to be produced. As Fig. 11 shows, packet multicasting/unicasting (PM/PU) and computation are largely concurrent, significantly reducing the time receivers spend waiting for data.

3. Network-level optimization. As described in Section 4.2, each FPGA node is equipped with four NI modules

(three LNIs and one TCP/IP-based NI) to eliminate the contention present in our prior design. This raises a new challenge: optimally assigning NI module types to accelerators. Based on the analysis in Section 2.2, dense blocks involve more complex communication patterns than TCONV layers and account for the majority (∼76%) of LNS-net's computation. Therefore, we assign a dedicated LNI module to each accelerator handling dense blocks to prioritize low-latency communication. The accelerator for the TCONV layers is assigned the TCP/IP-based NI module, as its communication demands are less critical.

6 Results and evaluation

6.1 Experimental setting

1. Hardware system. We built a system based on a tower server integrating an Intel® Xeon® E5-2680 CPU (12 cores, 2.5 GHz) and 128 GB random access memory (RAM). Additionally, the server provides eight PCIe slots, allowing the CPU to communicate with six customized FPGA evaluation boards, as well as four NVIDIA GeForce RTX 2080 GPUs (each with 2944 compute unified device architecture (CUDA) cores, 8-GB graphics double data rate 6 (GDDR6), 1.80 GHz). The FPGA boards are connected to the server via a PCIe hub. Each FPGA board integrates an XCVU9P FPGA, four dual in-line memory modules (DIMMs) of 64-GB double data rate 4 (DDR4) RAM, and four QSFP+ cages, each of which can provide up to 100 Gb/s network bandwidth. One of the QSFP+ ports on each FPGA board is connected to a switch equipped with 16 QSFP+ (40 GbE) ports. The other QSFP+ ports on each FPGA board are fully interconnected via 100-GbE optical cables according to the prototype illustrated in Fig. 5.

2. Software counterparts. All the software results were obtained using TensorFlow (Abadi et al., 2016) running on the CPU and GPU accelerators. To reduce the computational complexity and memory footprints of LNS-net and LNC-net, we obtained quantized versions of LNS-net and LNC-net with the help of the TensorFlow quantization tool. Experimental results showed that 8-bit fixed-point precision in LNS-net and LNC-net achieves similar accuracy to the original models. Therefore, we used 8-bit fixed-point precision for both activations and weights of LNS-net and LNC-net to evaluate the performance of our FPGA accelerators.

3. Benchmarks. We used the dataset from the LUNA16 (LUng Nodule Analysis 2016) challenge. The dataset contains 888 3D CT images, with a total of 1186 annotated positive lung nodules. The experiments were conducted on four randomly selected subsets (0–3) from the official 10-fold splits of the LUNA16 dataset. These subsets are chosen to be representative of the overall data distribution, and performance across them is highly consistent, demonstrating the robustness of our method.

4. Implementation details for reproducibility. To ensure reproducibility, our designs were implemented using Xilinx Vivado high-level synthesis (HLS) and Vivado™ Design Suite (2019.1), targeting a 200-MHz operating frequency on all FPGA nodes. Key HLS optimization directives include loop pipelining (II=1), loop unrolling, and array partitioning (cyclic for input buffers and block for weight buffers) to exploit parallelism and meet timing. The original floating-point models were quantized to 8-bit fixed-point precision using the TensorFlow quantization toolkit with symmetric per-layer calibration, introducing less than 0.5% accuracy loss. LNS-net is a custom hardware-oriented architecture combining elements from DenseNet (Huang XJ et al., 2017) and U-Net (Ronneberger et al., 2015), while LNC-net is adapted from Dey et al. (2018) with kernel size and filter count modifications for resource efficiency. Complete network specifications, HLS configuration scripts, and source code are provided in the ESM.

6.2 Analysis of experimental results

6.2.1 Evaluation of different topologies among FPGAs

Experiments are conducted to evaluate the effectiveness of adopting fast paths in the multi-FPGA system by testing different topologies among FPGA nodes. As illustrated in Fig. 14, three topologies are evaluated: (1) topology1, where no fast path exists (i.e., it uses the star topology); (2) topology2, where each FPGA node connects to three other FPGA nodes via fast paths and to the switch via a routine path; (3) topology3, where FPGA2 and FPGA4 only have fast paths, while the other FPGA nodes feature inter-FPGA connections similar to those in topology2.

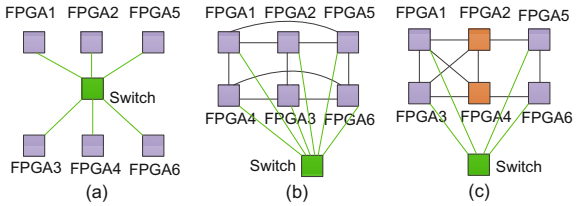

Fig. 14 The tested topologies among FPGA nodes: (a) topology1; (b) topology2; (c) topology3

We accelerated LNS-net on the tested prototypes and measured the latency for processing a single image ($48 \times 96 \times 96$) to evaluate system performance. To ensure experimental consistency with the prior benchmark, only FPGA1 to FPGA4 were utilized in this experiment.

Fig. 15 presents the experimental results. Both topology2 and topology3 achieved lower processing latency than topology1, with latency reductions of approximately 7% and 17%, respectively (results averaged over 10 independent runs). This demonstrates that integrating fast paths between FPGA nodes effectively reduces communication latency and improves the overall system performance.

6.2.2 Accelerator configuration and tiling parameters

Four accelerators (Acc1–Acc4) are implemented on FPGA1–FPGA4 for LNS, while three accelerators (Acc5–Acc7) are deployed on FPGA5 and FPGA6 for classification. The cross-layer parameters of these accelerators are listed in Table 3 (the configuration for the FC layer accelerator in the LNC-net is omitted from the table; it is configured with 2 PE links, each containing 32 PEs.) The tiling factors are derived using the performance model from Dey et al. (2018). To balance the execution time across the pipeline stages shown in

Fig. 11, we allocated different resource budgets to Acc1–Acc7, resulting in distinct tiling configurations.

As shown in Table 3, Acc4-1–Acc4-4 exhibit a higher degree of parallelism, reflecting greater computational complexity of the TCONV layers. For all accelerators, we set $T_r = R$ and $T_c = C$ to maximize parallelism, as the input image sizes are relatively small. Block RAMs (BRAMs) and LUTs emerge as the limiting resources on FPGA1–FPGA3; all available ultraRAMs (URAMs) are used to store inter-layer data of the dense blocks. On FPGA5 and FPGA6, URAMs are unused, as the available BRAM capacity is sufficient for storing LNC-net weights and inter-layer data.

6.2.3 Hardware utilization

Table 4 summarizes the full post-implementation resource utilization and operating frequency of all six FPGA nodes after place and route. All nodes adopt Xilinx XCVU9P FPGA and meet the 200-MHz design target. FPGA1–4, implementing the LNS-net segmentation accelerators, exhibit higher BRAM, URAM, and LUT usage due to larger intermediate feature maps. In contrast, FPGA5–6, implementing the lightweight LNC-net classification accelerators, require fewer resources and no URAMs. Benefiting from the proposed mapping schemes and inter-FPGA communication optimizations, most accelerators achieve a runtime hardware utilization above 80%. The exception is Acc7, whose utilization is limited by data dependency on Acc6—the throughput bottleneck of the classification pipeline.

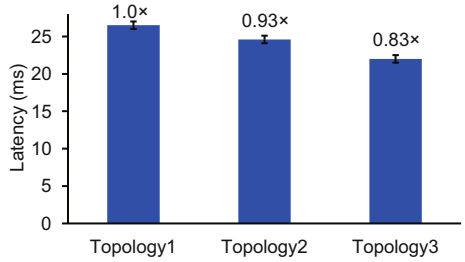

Fig. 15 Processing latency of a single image on different system prototypes

6.2.4 Latency breakdown

Fig. 16 presents the latency breakdown across different platforms. Our multi-FPGA system achieves significant latency reduction compared to both CPU and GPU implementations: segmentation latency is reduced to 9.2 ms (vs. 1187.8 ms on CPU and 16.5 ms on GPU) and classification latency to 0.3 ms (vs. 10.4 ms on CPU and 0.5 ms on GPU).

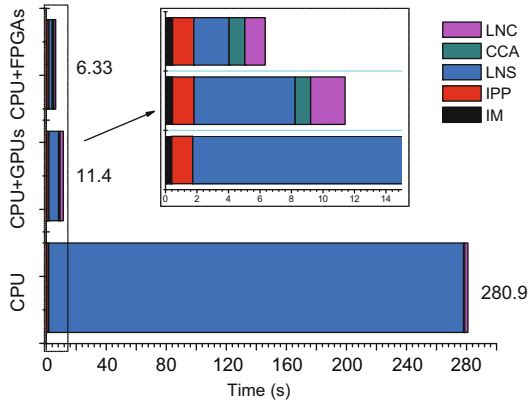

Fig. 16 Latency breakdown of lung nodule detection on different platforms

6.2.5 Comparisons with CPU and GPU

Table 5 compares the performance of the CPU, GPU, and our multi-FPGA system in accelerating LNS-net and LNC-net, using throughput, latency, and energy efficiency as key metrics. The latency values represent the average processing time per image across a batch of eight images; larger batch sizes were infeasible on the GPU due to memory constraints. To ensure fair comparison, all results in Table 5 use 8-bit fixed-point precision with appropriate optimizations: TensorRT for GPU and custom hardware acceleration for FPGA.

Our multi-FPGA system demonstrates superior performance in both acceleration tasks. For LNS, it achieves 128.2× and 1.8× higher throughput than the CPU and GPU implementations, respectively. In LNC, the system

Table 3 Tiling factors and Winograd algorithm configurations of accelerators

Accelerator	Winograd configuration	T_m	T_n	T_z	T_c (= C)	T_r (= R)
Acc1	$F(4^3, 3^3)$	12	6	8	48	48
Acc2	$F(2^3, 3^3)$	12	4	4	24	24
Acc3	$F(2^3, 3^3)$	6	2	4	12	12
Acc4-1–Acc4-4	$F(4^3, 3^3)$	16/16/16/2	8/8/8/8	4/8/8/8	12/24/48/96	12/24/48/48
Acc5–Acc7	$F(2^3, 3^3)$	16/16/16	4/4/4	8/8/4	32/16/8	32/16/8

C and R denote the total number of channels and the spatial resolution, respectively. Acc4-1–Acc4-4 are four distinct accelerators implemented on the same FPGA4 node, each with its own tiling configuration

Table 4 Post-implementation resource utilization and operating frequency of all FPGA nodes

FPGA	Target accelerator	DSP	BRAM	URAM	LUT	Frequency (MHz)
FPGA1	Acc1 (LNS-net)	3138 (46%)	1289 (60%)	340 (35%)	830k (70%)	200
FPGA2	Acc2 (LNS-net)	3138 (46%)	1545 (72%)	340 (35%)	837k (71%)	200
FPGA3	Acc3 (LNS-net)	3138 (46%)	1613 (75%)	340 (35%)	866k (73%)	200
FPGA4	Acc4-1–Acc4-4 (LNS-net)	2418 (35%)	1191 (55%)	340 (35%)	612k (52%)	200
FPGA5	Acc5–Acc6 (LNC-net)	1842 (27%)	872 (41%)	0	438k (37%)	200
FPGA6	Acc7 (LNC-net)	1842 (27%)	872 (41%)	0	438k (37%)	200

here, k denotes 1000

Table 5 Comparisons with CPU/GPU solutions

Sub-procedure	Metric	CPU (E5-2680)	GPU (GeForce RTX 2080)	Multi-FPGA (MASA-BDv1 (VU9P))
LNS	Precision	8-bit fixed	8-bit fixed	8-bit fixed
	Power (W per device)	115	225	60 (LNS), 48 (LNC)
	Device count	1	4	4
	Latency* (ms)	1187.8	16.5	9.2
	Total throughput (GOPS)	124 (1.0×)	8902.4 (71.8×)	15 894 (128.2×)
	Per-device throughput (GOPS)	124	2225.6	3973.5
	Per-Watt throughput (GOPS/W)	1.1 (1.0×)	9.9 (9.0×)	66.2 (60.2×)
LNC	Device count	1	2	2
	Latency* (ms)	10.4	0.5	0.3
	Total throughput (GOPS)	118 (1.0×)	2145.1 (18.2×)	3773.2 (32.0×)
	Per-device throughput (GOPS)	118	1972.6	1886.6
	Per-Watt throughput (GOPS/W)	1.0 (1.0×)	8.6 (8.6×)	39.3 (39.3×)

* latency values are reported in milliseconds per image patch (48×96×96 for LNS-net and 16×32×32 for LNC-net), averaged across a batch of eight patches. GOPS: gita operations per second

outperforms the CPU and GPU by factors of 32.0× and 1.8× in throughput, respectively. Despite leveraging the CuDNN library for GPU optimization, the computational efficiency remains suboptimal for both networks (NVIDIA, 2019). This can be attributed to two primary factors: (1) the substantial volume of inter-layer data in LNS-net increases memory access overhead on the GPU; (2) the relatively low computation-to-communication ratio in LNC-net limits GPU acceleration efficacy. Furthermore, compared to our prior work, the current system achieves higher throughput for LNS, attributable to the introduction of fast paths between FPGA nodes and the use of 8-bit fixed-point precision.

In terms of energy efficiency, our multi-FPGA system also ranks the highest. For LNS, it is 60.2× and 6.7× higher energy-efficient than the CPU and GPU, respectively. Similarly, for LNC, it achieves 39.3× and 4.6× higher energy efficiency compared to the CPU and GPU, respectively.

6.2.6 Detection recall

We evaluated detection accuracy using the recall rate, defined as the proportion of retrieved true positives to all annotated positive nodules (negative nodules are excluded from this evaluation). Fig. 17 compares the recall rate of our FPGA-based solution and the GPU baseline.

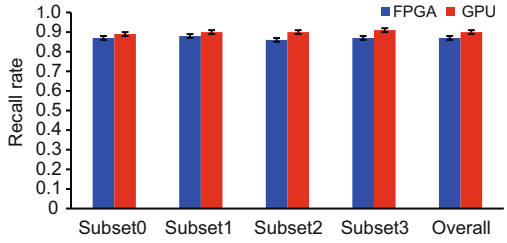

Fig. 17 Recall rate of multi-FPGA and GPU solutions

The FPGA implementation achieved recall rate closely matching the GPU solution across the evaluated subsets, with an overall recall of 0.87—approximately 3% lower than the GPU's 0.90. Results are reported for the four evaluated subsets (subset0–subset3) from the official LUNA16 split. The consistent trend across these subsets indicates that the observed performance gap is stable. This minor discrepancy stems from the use of the 3D Winograd algorithm in our FPGA accelerators for computing CONV and TCONV layers. As the Winograd method is an approximation of standard convolution, a marginal accuracy loss is inherent. Nevertheless, both implementations attain state-of-the-art recall rates comparable to top-performing entries in the LUNA16 challenge (Wikipedia, 2019), underscoring the competitive detection accuracy of our multi-FPGA system.

6.2.7 Reliability analysis of lightweight protocol

To validate that the lightweight protocol's reliability trade-off does not compromise segmentation accuracy, we conduct an ablation study comparing three configurations.

1. Baseline: TCP/IP only (no fast paths);

2. Lightweight without duplication: fast paths enabled, but without the data duplication mechanism;

3. Lightweight with duplication: fast paths + data duplication FIFO (as described in Section 4.3.2).

We first measured packet loss rate under sustained 100-Gb/s traffic over a 24-h period. The results showed that the TCP/IP baseline achieved 0% loss (guaranteed by retransmission). For lightweight network without duplication, the average loss rate was 4.7×10^{-7}. For lightweight network with duplication, the effective loss rate after compensation was less than 1×10^{-9}.

We then evaluated the impact on segmentation recall (LUNA16 subset0). As shown in Table 6, the lightweight protocol with duplication achieved recall identical to the TCP/IP baseline (87.1% vs. 87.0%), while the version without duplication suffers a marginal drop (from 87.1% to 86.8% over 10 000 test samples) due to rare packet loss affecting boundary voxels in TCONV layers. This confirmed that our duplication mechanism effectively masks the reliability loss without incurring the latency overhead of full TCP retransmission.

6.2.8 Ablation study: impact of Winograd algorithm on segmentation accuracy

To explicitly identify the source of the approximately 3.4% recall gap between our FPGA implementation (87.1%) and the GPU baseline (90.2%), we conducted a controlled ablation study on LUNA16 subset0 with three configurations: configuration A (GPU, standard CONV), configuration B (FPGA, standard CONV, Winograd bypassed),

Table 6 Recall comparison across different convolution implementations

Configuration	Recall	Relative drop	Note
Configuration A: GPU (standard CONV)	90.2%		Baseline
Configuration B: FPGA (standard CONV)	89.8%	0.4%	Hardware-induced quantization loss
Configuration C: FPGA (Winograd)	87.1%	3.4%	Winograd approximation + quantization

and configuration C (FPGA, Winograd). All configurations use identical 8-bit fixed-point quantization and the same LNS-net model architecture. Results are summarized in Table 6.

The recall of configuration B reaches 89.8%, only 0.4% lower than the GPU baseline (configuration A, 90.2%), attributed to fixed-point quantization and hardware-specific rounding. Configuration C with Winograd optimization yields 87.1% recall, representing an additional 3.0% drop relative to configuration B. These results confirm that the dominant source of accuracy loss is the approximate computation inherent to the Winograd algorithm.

6.3 Scalability analysis

Our analysis shows that the system scales effectively from 2 to 6 nodes. Extrapolation indicates three potential bottlenecks: interconnect saturation ($N \geq 8$), memory bandwidth contention ($N \geq 12$), and load imbalance ($N \geq 16$). A detailed analytical model and mitigation strategies are provided in Section S8 in the ESM.

6.4 Clinical deployment analysis

As summarized in Table 2, the end-to-end latency for a full CT scan is approximately 1.30 s, meeting the requirement for real-time clinical feedback. Detailed physical deployment characteristics (rack size, power envelope, and cooling) are reported in Section S9 in the ESM.

7 Conclusions

This paper demonstrates that multi-FPGA acceleration is a viable path toward clinical deployment of 3D CNNs for lung nodule detection. Our hybrid interconnect topology with fast paths and dual-parallelism mapping strategy achieves 128.2× throughput improvement over CPU and 1.8× over GPU, with 6.7× higher energy efficiency than GPU solutions. The system processes a full CT scan in approximately 1.30 s while achieving 87.1% recall on LUNA16, which indicates competitive detection accuracy relative to the GPU baseline.

1. Real-time viability. In clinical workflows, CAD systems must return results within seconds. Our approximately 1.30-s end-to-end latency meets this requirement, and the pipelined architecture further improves steady-state throughput by overlapping scan processing across stages.

2. Key insights. This work offers three takeaways: (1) FPGAs provide a practical accuracy–efficiency trade-off, delivering competitive detection accuracy together with substantially better energy efficiency than the GPU baseline; (2) scalability requires holistic optimization of interconnect, workload balancing, and communication; (3) clinical deployment is feasible with standard rack mounting, < 500 W power, and silent operation.

3. Limitations and future work. The 3.4% recall gap from Winograd approximation calls for hybrid convolution strategies. Interconnect saturation beyond eight FPGAs motivates exploration of fat-tree topologies. Migrating CPU stages to FPGA could further reduce latency. Future work includes multi-center validation and regulatory pathways.

Supplementary information

The online version contains supplementary materials available at https://doi.org/10.1631/ENG.ITEE.2025.0186.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. U23A20301) and the Fourth Cohort Funding for High-Level Talent Cultivation Program (No. ZC514C0882503).

Author contributions

Zhuang CAO and Junzhong SHEN designed the research and processed the data. Tian ZHANG drafted the paper. Xiaowei HE and Sheng LIU revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data for the LUNA16 challenge (https://luna16.grand-challenge.org/) are publicly available under the Creative Commons Attribution 4.0 International License. The other data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration on the use of generative AI tools

During the preparation of this work, the authors used DeepSeek to improve language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Abadi M, Barham P, Chen JM, et al., 2016. TensorFlow: a system for large-scale machine learning. Proc 12th USENIX Conf on Operating Systems Design and Implementation, p.265-283.

Aydonat U, O'Connell S, Capalija D, et al., 2017. An OpenCL™ deep learning accelerator on Arria 10. Proc ACM/SIGDA Int Symp on Field-Programmable Gate Arrays, p.55-64. https://doi.org/10.1145/3020078.3021738

Basalama S, Sohrabizadeh A, Wang J, et al., 2023. FlexCNN: an end-to-end framework for composing CNN accelerators on FPGA. *ACM Trans Reconfig Technol Syst*, 16(2):23. https://doi.org/10.1145/3570928

Chen HX, Song MC, Zhao JC, et al., 2019. 3D-based video recognition acceleration by leveraging temporal locality. Proc 46th Int Symp on Computer Architecture, p.79-90. https://doi.org/10.1145/3307650.3322260

Cheng X, 2025. FPGA-based accelerator for convolutional neural networks. Proc Int Conf on Digital Analysis and Processing, Intelligent Computation, p.10-14. https://doi.org/10.1109/DAPIC66097.2025.00008

Cui N, Wu YJ, Xin GJ, et al., 2025. Application of quantitative interpretability to evaluate CNN-based models for medical image classification. *IEEE Access*, 13:89386-89398. https://doi.org/10.1109/ACCESS.2025.3567775

Dey R, Lu ZJ, Hong Y, 2018. Diagnostic classification of lung nodules using 3D neural networks. Proc IEEE 15th Int Symp on Biomedical Imaging, p.774-778. https://doi.org/10.1109/ISBI.2018.8363687

Diaconu D, Lin X, Blott M, et al., 2026. A survey of FPGA-based 3D CNN accelerators and hardware-aware algorithmic optimizations. *NACM Comput Surv*, 58(6):155. https://doi.org/10.1145/3777366

Fan HX, Luo C, Zeng CL, et al., 2019. F-E3D: FPGA-based acceleration of an efficient 3D convolutional neural network for human action recognition. Proc IEEE 30th Int Conf on Application-Specific Systems, Architectures and Processors, p.1-8. https://doi.org/10.1109/ASAP.2019.00-44

Fan HX, Liu SL, Que ZQ, et al., 2023. High-performance acceleration of 2-D and 3-D CNNs on FPGAs using static block floating point. *IEEE Trans Neur Netw Learn Syst*, 34(8):4473-4487. https://doi.org/10.1109/TNNLS.2021.3116302

Fowers J, Ovtcharov K, Papamichael M, et al., 2018. A configurable cloud-scale DNN processor for real-time AI. Proc ACM/IEEE 45th Annual Int Symp on Computer Architecture, p.1-14. https://doi.org/10.1109/ISCA.2018.00012

Gao L, Luo ZQ, Wang L, 2025. Convolutional neural network acceleration techniques based on FPGA platforms: principles, methods, and challenges. *Information*, 16(10):914. https://doi.org/10.3390/info16100914

Geng T, Wang TQ, Sanaullah A, et al., 2018. FPDeep: acceleration and load balancing of CNN training on FPGA clusters. Proc IEEE 26th Annual Int Symp on Field-Programmable Custom Computing Machines, p.81-84. https://doi.org/10.1109/FCCM.2018.00021

Hegde K, Agrawal R, Yao YL, et al., 2018. Morph: flexible acceleration for 3D CNN-based video understanding. Proc 51st Annual IEEE/ACM Int Symp on Microarchitecture, p.933-946. https://doi.org/10.1109/MICRO.2018.00080

Huang G, Liu Z, Van Der Maaten L, et al., 2017. Densely connected convolutional networks. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.2261-2269. https://doi.org/10.1109/CVPR.2017.243

Huang XJ, Shan JJ, Vaidya V, 2017. Lung nodule detection in CT using 3D convolutional neural networks. Proc IEEE 14th Int Symp on Biomedical Imaging, p.379-383. https://doi.org/10.1109/ISBI.2017.7950542

Isensee F, Jaeger PF, Kohl SAA, et al., 2021. nnU-Net: a self-configurationuring method for deep learning-based biomedical image segmentation. *Nat Methods*, 18(2):203-211. https://doi.org/10.1038/s41592-020-01008-z

Jiang JY, Zhou YA, Gong YH, et al., 2025. FPGA-based acceleration for convolutional neural networks: a comprehensive review. https://doi.org/10.48550/arXiv.2505.13461

Jiang WW, Sha EHM, Zhang XY, et al., 2019. Achieving super-linear speedup across multi-FPGA for real-time DNN inference. *ACM Trans Embed Comput Syst*, 18(5s):67. https://doi.org/10.1145/3358192

Khan FH, Pasha MA, Masud S, 2023. Towards designing a hardware accelerator for 3D convolutional neural networks. *Comput Electr Eng*, 105:108489. https://doi.org/10.1016/j.compeleceng.2022.108489

Kuang HL, Wang YH, Tan XZ, et al., 2025. LW-CTrans: a lightweight hybrid network of CNN and Transformer for 3D medical image segmentation. *Med Image Anal*, 102:103545. https://doi.org/10.1016/j.media.2025.103545

Lin CY, Guo SM, Lien JJJ, 2024. Development of a modified 3D region proposal network for lung nodule detection in computed tomography scans: a secondary analysis of lung nodule datasets. *Cancer Imaging*, 24(1):40. https://doi.org/10.1186/s40644-024-00683-X

Litjens G, Kooi T, Bejnordi BE, et al., 2017. A survey on deep learning in medical image analysis. *Med Image Anal*, 42:60-88. https://doi.org/10.1016/j.media.2017.07.005

Liu SL, Fan HX, Ferianc M, et al., 2022. Toward full-stack acceleration of deep convolutional neural networks on FPGAs. *IEEE Trans Neur Netw Learn Syst*, 33(8):3974-3987. https://doi.org/10.1109/TNNLS.2021.3055240

Lu YH, Qi XY, Li Y, et al., 2024. Automatic implementation of large-scale CNNs on FPGA cluster based on HLS4ML. Proc IEEE Int Symp on Parallel and Distributed Processing with Applications, p.1080-1087. https://doi.org/10.1109/ISPA63168.2024.00143

Messay T, Hardie RC, Rogers SK, 2010. A new computationally efficient CAD system for pulmonary nodule detection in CT imagery. *Med Image Anal*, 14(3):390-406. https://doi.org/10.1016/j.media.2010.02.004

Microsoft, 2018. Project Catapult. https://www.microsoft.com/en-us/research/project/project-catapult [Accessed on May 10, 2026].

Microsoft, 2019. Project Brainwave. https://www.microsoft.com/en-us/research/project/project-brainwave/ [Accessed on May 10, 2026].

Motamedi M, Gysel P, Akella V, et al., 2016. Design space exploration of FPGA-based deep convolutional neural networks. Proc 21st Asia and South Pacific Design Automation Conf, p.575-580. https://doi.org/10.1109/ASPDAC.2016.7428073

NVIDIA, 2019. NVIDIA cuDNN. https://developer.nvidia.com/cudnn [Accessed on May 10, 2026].

Qin JJ, Xiong J, Liang ZT, 2025. CNN-Transformer gated fusion network for medical image super-resolution. *Sci Rep*, 15(1):15338. https://doi.org/10.1038/s41598-025-00119-x

Ronneberger O, Fischer P, Brox T, 2015. U-Net: convolutional networks for biomedical image segmentation. Proc 18th Int Conf on Medical Image Computing and Computer-Assisted Intervention, p.234-241. https://doi.org/10.1007/978-3-319-24574-4_28

Shen JZ, Qiao Y, Huang Y, et al., 2018a. Towards a multi-array architecture for accelerating large-scale matrix multiplication on FPGAs. Proc IEEE Int Symp on Circuits and Systems, p.1-5. https://doi.org/10.1109/ISCAS.2018.8351474

Shen JZ, Huang Y, Wang ZL, et al., 2018b. Towards a uniform template-based architecture for accelerating 2D and 3D CNNs on FPGA. Proc ACM/SIGDA Int Symp on Field-Programmable Gate Arrays, p.97-106. https://doi.org/10.1145/3174243.3174257

Shen JZ, Wang DG, Huang Y, et al., 2019. Scale-out acceleration for 3D CNN-based lung nodule segmentation on a multi-FPGA system. Proc 56th Annual Design Automation Conf, Article 207. https://doi.org/10.1145/3316781.3317906

Shen JZ, Huang Y, Wen M, et al., 2020. Toward an efficient deep pipelined template-based architecture for accelerating the entire 2-D and 3-D CNNs on FPGA. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 39(7):1442-1455. https://doi.org/10.1109/TCAD.2019.2912894

Siegel RL, Giaquinto AN, Jemal A, 2024. Cancer statistics, 2024. *CA Cancer J Clin*, 74(1):12-49. https://doi.org/10.3322/caac.21820

Siegel RL, Kratzer TB, Giaquinto AN, et al., 2025. Cancer statistics, 2025. *CA Cancer J Clin*, 75(1):10-45. https://doi.org/10.3322/caac.21871

Tan MXN, Deklerck R, Jansen B, et al., 2011. A novel computer-aided lung nodule detection system for CT images. *Med Phys*, 38(10):5630-5645. https://doi.org/10.1118/1.3633941

Varughese DA, Sridevi S, 2026. Optimization strategies and algorithms for accelerating CNN on FPGA: a comprehensive review. *Arch Comput Methods Eng*, 33(1):1205-1226. https://doi.org/10.1007/s11831-025-10348-y

Wang JX, Zhang XW, Tang W, et al., 2025. A multi-view CNN model to predict resolving of new lung nodules on follow-up low-dose chest CT. *Insights Imaging*, 16(1):138. https://doi.org/10.1186/s13244-025-02000-x

Wang YC, Wang Y, Li HW, et al., 2019. Systolic cube: a spatial 3D CNN accelerator architecture for low power video analysis. Proc 56th Annual Design Automation Conf, Article 210. https://doi.org/10.1145/3316781.3317919

Wikipedia, 2019. Ipv4. https://en.wikipedia.org/wiki/IPv4 [Accessed on May 10, 2026].

Wu LH, Zhang M, Piao Y, et al., 2025. CNN-Transformer rectified collaborative learning for medical image segmentation. *IEEE Trans Circ Syst Video Technol*, 35(5):4072-4086. https://doi.org/10.1109/TCSVT.2024.3523316

Zhang C, Wu D, Sun JY, et al., 2016. Energy-efficient CNN implementation on a deeply pipelined FPGA cluster. Proc Int Symp on Low Power Electronics and Design, p.326-331. https://doi.org/10.1145/2934583.2934644

Zhang C, Sun GY, Fang ZM, et al., 2019. Caffeine: toward uniformed representation and acceleration for deep convolutional neural networks. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 38(11):2072-2085. https://doi.org/10.1109/TCAD.2017.2785257

Zhang WT, Zhang JX, Shen MH, et al., 2019. An efficient mapping approach to large-scale DNNs on multi-FPGA architectures. Proc Design, Automation & Test in Europe Conf & Exhibition, p.1241-1244. https://doi.org/10.23919/DATE.2019.8715174