Inference: take an already trained network and pass data through it to identify features based on the model etc.

Outline

- Context and Motivation
- Literature Review
- Hardware Accelerator Design
- Dataflow for 3D Systolic Architecture
- Experimental Results and Discussion
- Conclusions

Context and Motivation (1/2)

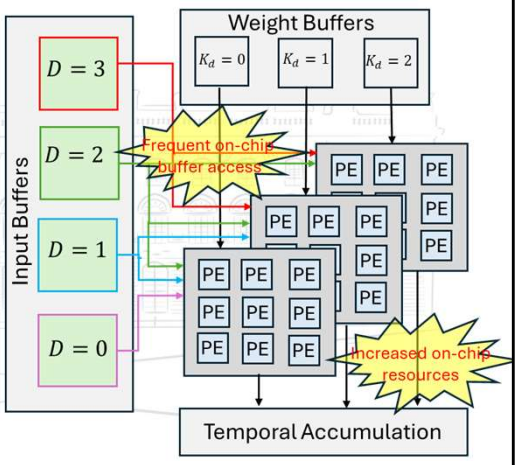
- 3D CNN are gaining popularity in applications involving video processing
  - Extract features across both spatial and temporal dimensions
- High computation throughput and significantly increased memory traffic
  - Difficult to deploy on resource-constrained and low-power platforms
- FPGAs are gaining prominence over other hardware platform
  - Reconfigurable and flexible architecture
- Limited availability of on-chip memory on FPGA
  - Makes it difficult to reuse filters and overlapped input data in all three dimensions
  - Frequent access to off-chip memory

3

- Recently, there has been a notable interest in computer vision research towards addressing more complex tasks, particularly those involving the processing of videos.
- To cater to videos, 3D Convolutional Neural Networks (CNNs) are gaining popularity due to their ability to extract features across both spatial and temporal dimensions.
- However, the high computation and memory complexity of 3D CNNs impedes their deployment on resource-constrained and low-power platforms like Field Programmable Gate Arrays (FPGAs).
- Currently, a wide range of devices use Graphics Processing Units (GPUs) to accelerate the CNNs at the edge. Here, the FPGA is preferred over other hardware platforms due to its reconfigurability,
- energy efficiency and easier path to system development.
- Due to limited availability of on-chip memory on FPGA, Tiling of incoming feature maps and weight blocks is indispensable
- In 3D CNNs, the 3D filter is slid in both spatial as well as temporal dimension to extract the time-domain features.
- In this scenario, the tiling of incoming data aggravates the difficulty to reuse filter and overlapped input data in all three dimensions.
- It also results in increased off-chip memory access

Context and Motivation (2/2)

- The conventional hardware accelerators for 2D CNNs emphasize the spatial reuse of input data only
- 2D systolic array-based architecture are inefficient for 3D CNNs
  - Clustering of multiple 2D PE arrays requires more on-chip memory and resources
  - Not suitable for low-power and resource-constrained scenarios
- Design of an energy-efficient 3D systolic array architecture
  - Supports data reuse in the temporal dimension
  - Eliminates conversion of 3D data into 2D matrices
  - Reduce both on-chip and off-chip memory traffic
- A corresponding dataflow control logic
  - Synchronize the propagation of data in multiple planes of a 3D systolic array



4

- The reported hardware accelerators for 2D CNNs primarily emphasize the spatial reuse of input data by arranging Processing Elements (PE) in a 2D plane.
- Consequently, this approach lacks efficiency when applied to 3D CNN inference tasks.
- Clustering: Multiple 2D arrays to process each frame
- The clustering of multiple 2D PE arrays to accelerate 3D CNNs requires more on-chip memory as well as more on-chip resources rendering these un-suitable for low-power and resource-constrained scenarios.
- In this paper, An energy efficient 3D systolic architecture for 3D CNNs is proposed that enhances the data reuse across their spatial and temporal dimensions.
- The 3D systolic array is designed to process the sliding of the 3D convolution filter in all three dimensions without converting it into a matrix form.
- Thus, reducing both on-chip and off-chip memory
- A generalized dataflow model has been developed for this 3D systolic architecture to synchronize the propagation of data in multiples planes of 3D array that maximizes hardware utilization and minimizes the latency in data transfer.

Literature Review of 3D CNN Hardware

- Tian et al. [4] presented an accelerator that consists of multiple 2D PE arrays
  - Implement convolution operations using different loop ordering strategies
- Hegde et al.[10] proposed a design based on multiple systolic arrays, coupled with hierarchical reconfigurable buffers
  - Leverage in-memory reusability and prevent the need for temporal data refetching
- Most of accelerators[3-9] decompose the 3D convolution into several 2D convolutions
  - Utilizes the spatial data locality
  - Additional data transfers between the 2D PE arrays and on-chip memory
- Wand et al. [11] presented the concept of a 3D systolic cube
  - Dataflow mode is output stationary
  - Limited data reuse and restricted parallelism
- Our work → Weight stationary mode for the 3D systolic array

5

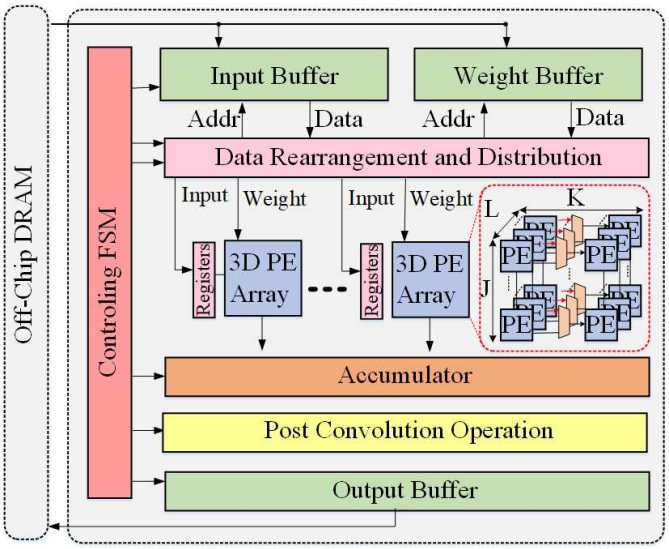
Tian et al. [4] recently presented an accelerator that consists of multiple 2D PE arrays to implement convolution operations using different loop ordering strategies.
In [10], Hegde et al. proposed a design based on multiple systolic arrays, coupled with hierarchical reconfigurable buffers, to leverage in-memory reusability and prevent the need for temporal data re-fetching.
Several specialized 3D CNN accelerators [3-9] have also been developed to address the issue of temporal reuse.
Most of these accelerators decompose the 3D convolution into several 2D convolutions and then process them on different PE arrays.
Each PE array utilizes the spatial data locality followed by the transmission of 3D input feature maps to additional on-chip buffers for exploiting temporal data locality.
These prior architectures lead to additional data transfers between the 2D PE arrays and on-chip memory, resulting in excessive energy costs.
The typical 2D systolic array is suited to mapping the complex and overlapped dataflow required for 2D convolution operation.
In our work, this idea has been extended for 3D CNNs by expanding the dimension of the systolic array to accommodate the temporal direction.
Previously, the authors in [11] presented the concept of a systolic cube. Their dataflow mode is output stationary which has limited data reuse and restricted parallelism.
Inferring 3D CNNs on their architecture for large-scale video datasets would necessitate either a significantly large systolic cube or a high number of iterations, resulting in substantial resource utilization or increased latency.

Our proposed accelerator enhances the data locality by proposing a dataflow mechanism for weight stationary mode for the 3D systolic array.
It can be easily scaled across different levels of parallelism to increase the throughput, as the stationary weights are reused by multiple processing elements simultaneously.

Proposed Hardware Accelerator Design (1/3)

- Input data buffer is on-chip memory
- The input data is rearranged and distributed in Rearrangement block
- 3D PE array modules of size $J \times K \times L$
- Supports varying kernel sizes across the network through muxes
  - Reconfigurable connections between the 3D PEs arrays
- Input and output level parallelism
  - Different levels of parallelism to increase the throughput by de/activitating multiple PE arrays

….contd

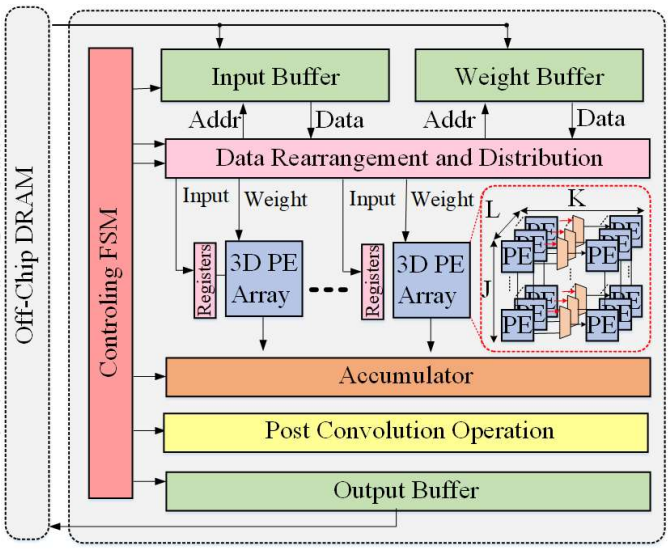


6

Figure shows the primary component included in the proposed design.
The weight buffer stores filter parameters, while inputs and partial outputs are held in the input and output buffers, respectively.
The input and weight data are rearranged and distributed to the attached 3D systolic arrays.
The main computation occurs within a 3D PE array module of size $J \times K \times L$ where PEs are interconnected in three dimensions.
This 3D systolic architecture directly loads 3D kernel blocks and performs convolution on the incoming input data.
The proposed architecture supports varying kernel sizes across layers of a 3D CNN by reconfiguring the connection between the 3D PEs arrays.
The multiplexers attached to the PEs control the reconfiguration of 3D systolic arrays.
It determines whether a PE receives input from an adjacent PE or directly from the input buffer.
The flexibility in PE connections allows the hardware to exploit input and output level parallelism.

Proposed Hardware Accelerator Design (2/3)

- Simultaneous Computation for multiple output and input feature maps
  - Small kernel blocks
  - Maximize resource utilization
- Partial results accumulator
- Controlling FSM
  - Synchronization of data traffic
- Activation and other post-convolutional operations
- Partial outputs are held in output buffers



7

Kernel blocks that are smaller than $J \times K \times L$ dimension can be computed for multiple output and input feature maps simultaneously to maximize resource utilization.

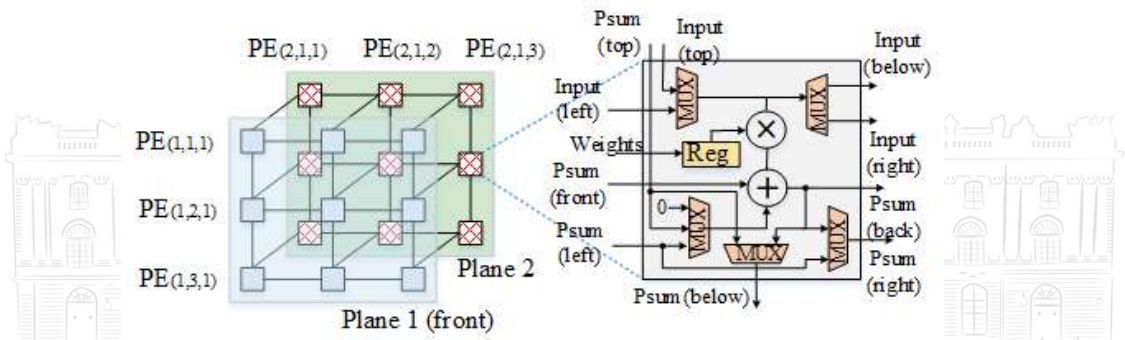
Partial results from different feature maps are accumulated in the accumulator.

A finite state machine (FSM) manages the synchronization of data traffic.

Activation functions and other post-convolution operations are executed at the final stage of the pipeline.

Partial outputs are stored in output buffers before being transferred to off-chip memory.

Proposed Hardware Accelerator Design (3/3)



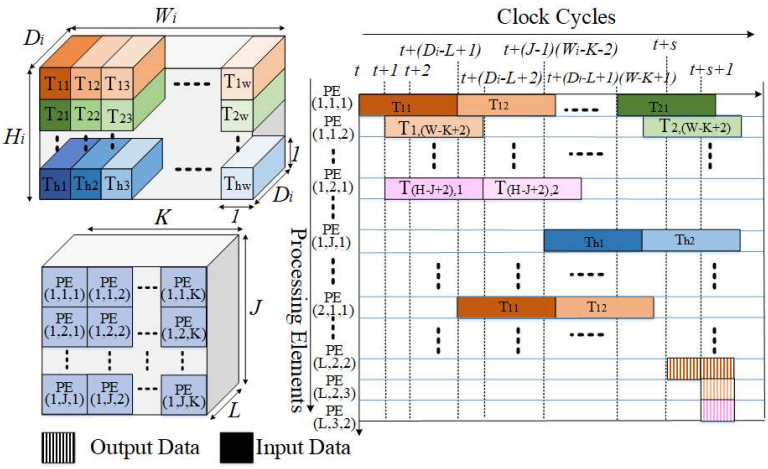
- Multiply-Accumulate (MAC) unit, control logic, data register, and multiplexers
  - Different inputs from the front side only
- A set of inputs is transferred to registers and shared by a column of PEs
- Input data is fully utilized across both spatial and temporal dimensions
  - Ensures maximum data reuse and efficient computation within the systolic array

8

The 3D systolic architecture features a unique interconnection pattern for its PEs.
Each PE is connected to the PE directly to its right and the PE directly below it, facilitating efficient data flow in the spatial dimension.
To manage data flow in the temporal dimension, each 2D plane of PEs is connected to its corresponding PEs in the previous plane.
This inter-plane connectivity ensures that temporal data dependencies are efficiently handled.
Figure illustrates the detailed architecture of a PE, highlighting these interconnections and the dataflow paths within the 3D systolic array.
Each PE contains a Multiply-Accumulate (MAC) unit, associated control logic, data register, and multiplexers.
Each plane of PEs array receives different inputs from the front side, and a set of inputs is transferred to registers and shared by a column of PEs in each cycle.
Once the input data is fetched from the buffer, it is fully utilized across both spatial and temporal dimensions through the 3D interconnection of PEs.
This ensures maximum data reuse and efficient computation within the systolic array.
Input data is 8 bits, weights are 8 bits, output is 16 bits. Final truncation is at the end of accumulation for final results (not partial results).

Dataflow for 3D Systolic Architecture (1/3)

- Weights are first loaded due to Weight Stationary mode
- Arrival of input data block
- Rearrangement of input data
  - $H \times W$ temporal blocks
- Only front plane will receive the $T_{a,b}$ input block
- $T_{a,b}$ at each PE block is determined by the size of the filter and input block



Time-space diagram for 3D data Re-arrangement and Scheduling

9

It describes the dataflow within the 3D systolic architecture to fully exploit the data reusability and to reduce the frequency of data requests from on-chip buffers.
The overall flow of the input data and the generation of the output block is shown in Figure.
The weights are loaded into the systolic array in their existing arrangement as the weight stationary mode is adopted for the proposed dataflow.
The input data is rearranged along the temporal dimension. As a result, $H \times W$ temporal blocks are formed, as illustrated in Figure.
All these temporal blocks are fed to the 3D PE array through the front plane of the array.
Ta,b is the input temporal data chunk block. a, b is explained in next slide.

Each PE in the front plane will receive the $T_{a,b}$ block of the input data as explained in next slide.
The input data block will arrive at each PE in a systolic manner and the arrival of temporal blocks at each PE block is determined by the size of the filter and input block as explained by the equations in next slide.

Dataflow for 3D Systolic Architecture (2/3)

The terms a and b are defined for each PE as follows:

$$a = 1{:}H - (K_H - 1), b = 1{:}W - (K_W - 1) \rightarrow PE_{1,1,1} \quad (1)$$

$$a = H - K_H + j, \quad b = 1{:}W - (K_W - 1) \rightarrow PE_{1,j,1}$$
$$where\ j \in 2, 3, \dots, J \quad (2)$$

$$a = 1{:}H - (K_H - 1), b = W - K_W + k \rightarrow PE_{1,1,k}$$
$$where\ k \in 2, 3, \dots, K \quad (3)$$

$$a = H - (K_H - j), b = W - (K_W - k) \rightarrow PE_{1,j,k}$$
$$where\ j \in 2, 3, \dots, J\ and\ k \in 2, 3, \dots, K \quad (4)$$

(i in i,j,k is fixed at 1 for demonstrating the 1st plane)

10

These equations determines which input temporal block will arrive at which processing element.

H is height, W is width, K is size of kernel. i,j,k are 3D sizes of arrays

Here, it is assumed that J,$K$and L are exactly equal to K_D, K_H and K_W respectively.

The first equation defines the range of temporal blocks that the top-left PE of the first plane will receive.

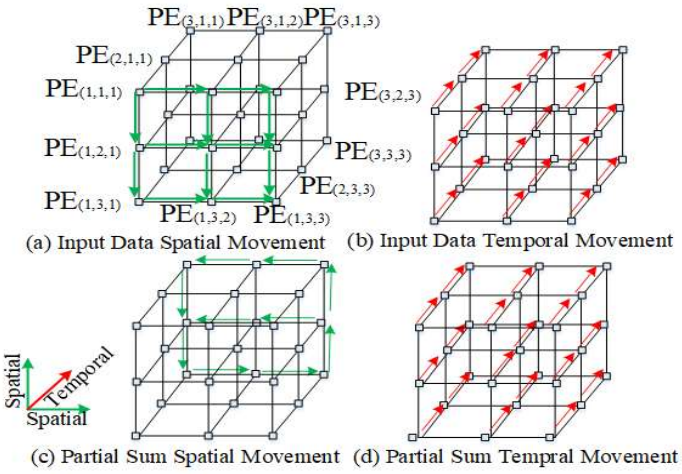
The second equation specifies the temporal block arriving at the leftmost column of the first plane.

The third equation determines the range of input temporal blocks that reach the first row of the first plane.

The fourth equation describes the temporal blocks that arrive at the remaining PE blocks.

Dataflow for 3D Systolic Architecture (3/3)

- Conditional data movement
  - $D - K_D + 1$ clock cycles
- Input data is loaded from the front plane only
- Back planes carry the temporal movement only
- Front plane share the calculated partial sum in the temporal dimension only
- Spatial accumulation of partial sum occurs in the Rear-most plane



11

In multi-dimensional systolic architecture, there is a conditional data movement in each direction. 1st plane is loaded, in green colour. Moves back plane by plane. Red arrows show temporal movement plane by plane.
The input block will be passed to its previous plane after every $D - K_D + 1$ clock cycles. D is temporal dimension. KD is kernel size in temporal D dimension.
The spatial moment of the input data depends on the arrival of the new data block.
When a new data block arrives at any PE, that PE does not receive the data from its top and left neighboring PEs. However, it transfers the incoming data to its neighboring PEs.
The input data is loaded from the front plane only, which is shared in both temporal and spatial dimensions.
The previous planes carry the temporal movement only, as shown in Fig. (b).
The movement of each partial sum is highlighted in Fig. (c) and (d).
The front plane share the calculated partial sum in the temporal dimension only whereas the spatial accumulation occurs in the last plane.

Case Study for 3D Systolic Architecture

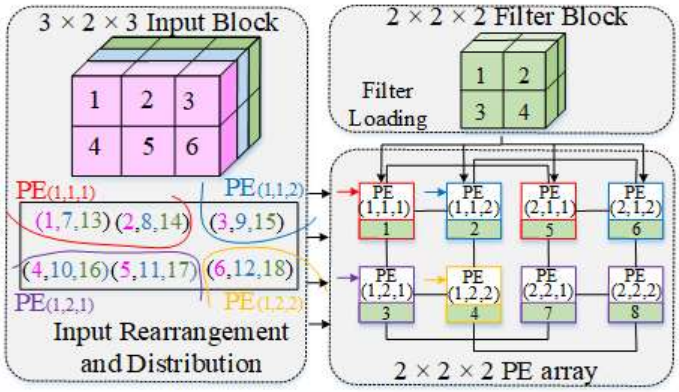


TABLE I. COMPUTATION IN EACH PE FOR EVERY CLOCK CYCLE

clk no.	PE (111)	PE (112)	PE (121)	PE (122)	PE (211)	PE (212)	PE (221)	PE (222)
1	1×1							
3	7×1	3×2	4×3		7×5			
5	2×1	9×2	10×3	6×4	13×5	9×6	10×7	
7	8×1	2×2	5×3	12×4	8×5	15×6	16×7	12×8
9		8×2	11×3	5×4	14×5	8×6	11×7	18×8
11				11×4		14×6	17×7	11×8
13								17×8

- 14 clock cycles to compute the output data.
- Whereas the work in [11] takes 20 clock cycles for the same example.

Input Block → Data Rearrange → Convolution → Output

12

As an example, the input block of 3 × 2 × 3 is mapped to 2 × 2 × 2 PE array to convolve with 2 × 2 × 2 filter block.
Figure shows the input data rearrangements in temporal blocks and the distribution of these blocks for each PE.
The filter block is loaded in their original arrangement onto the PE array.
The temporal blocks grouped by the red line, are fed to the PE(1,1,1).
The temporal blocks grouped by the purple line, are fed to the PE(1,2,1).
The temporal blocks grouped by the blue line, are fed to the PE(1,1,2).
The temporal blocks grouped by the yellow line, are fed to the PE(1,2,2).
Small arrows demonstrate the arrival of the input temporal block at the PE of the first plane.
Table I shows the computation in each PE block for the given example.
Every PE in the last plane generates the final output pixel and the order of the output is the same as that of the input temporal block.
Using this dataflow, any 3D CNN can be mapped on a 3D systolic array while exploiting maximum data reuse.
It takes only 14 clock cycles to compute the output data whereas the work in [11] takes 20 clock cycles for the same example.

Experiments and Results (1/4)

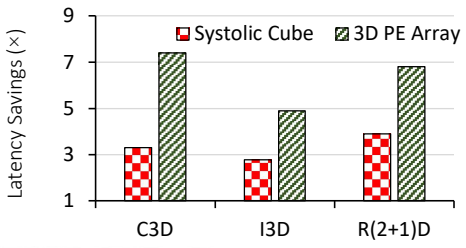
- Xilinx Virtex-7 FPGA VC709
- Cadence Genus synthesis tool in 180 nm technology
  - Comparison with systolic cube [11]
- Popular 3D CNNs, including I3D, C3D, and R(2+1)D
  - UCF101 dataset for video data
  - 8-bit precision
- Analysis is based on two main factors: (i) latency and (ii) energy consumption
- Tiling configuration for each model is obtained through experimentation

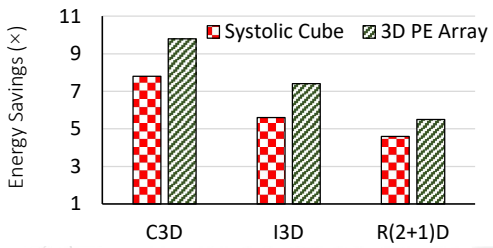
13

The Xilinx Virtex-7 FPGA VC709 was chosen as the implementation platform.
It features a Virtex-7 690T FPGA and two 4GB DDR3 DRAMs operating at 200 MHz.).
For comparison with systolic cube [11], the ASIC area for the proposed hardware architecture is determined using the Cadence Genus synthesis tool in 180 nm technology.
The experiments were conducted on popular 3D CNNs, including I3D, C3D, and R(2+1)D, using the UCF101 dataset.
A detailed analysis of the proposed methodology is based on two main factors: (i) latency and (ii) energy consumption.
The tiling configuration for each model is obtained through experimentation. Tiling size determined through experimentation and an in-house tool based on minimizing DRAM access.

Experiments and Results (2/4)

Latency





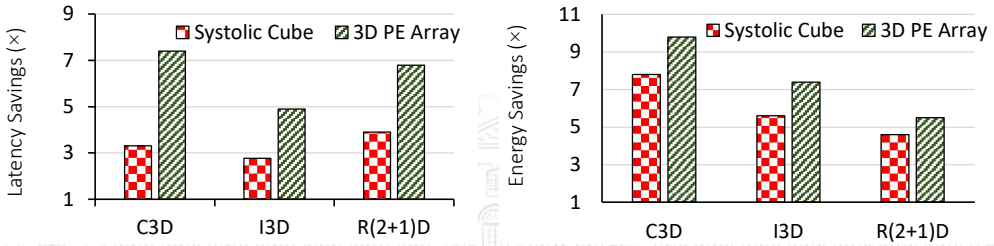
Energy

- 3D PE array comprised of 9 rows, 9 columns, and 9 planes
  - Accommodate all kernel sizes of the 3D CNNs under consideration
- Same configuration for baseline 2D array
  - Systolic cube configuration [11] is also 9×9×9 but with an output stationary dataflow
- The networks, C3D and I3D have network structures with sufficient temporal depth
  - Contributing to higher utilization of 3D systolic array

14

The proposed 3D systolic hardware is compared below with both the classical 2D systolic array and a newly introduced systolic cube architecture.
Our 3D PE array comprised of 9 rows, 9 columns, and 9 planes to accommodate all kernel sizes of the 3D CNNs under consideration.
baseline 2D array consists of the same configuration with 9 rows and 9 columns, and 9 of such 2D arrays are used for a fair comparison in throughput and resource consumption.
The systolic cube configuration is also 9 × 9 × 9, but with an output stationary dataflow as mentioned earlier.
Figure (a) shows the latency savings of systolic cube and our proposed architecture against the traditional 2D PE arrays.
Since 3D convolutions are carried out using multiple 2D convolutions on standard 2D accelerators, only a portion of the data locality is leveraged by 2D-plane accelerators.
However, the 3D systolic architecture shares the data in temporal dimension using neighboring PE planes.
The networks, C3D and I3D have network structures with sufficient temporal depth, contributing to higher hardware utilization.

Experiments and Results (3/4)



- 6.4× and 1.92× performance speedup
- Energy reduction in the systolic cube and 3D systolic array design
- Better data locality in both the temporal and spatial dimensions
  - Significantly reducing buffer accesses
- Area efficiency of 0.74 and 0.70 FPS/mm2 for I3D and C3D models
  - Improvement of 9.1× and 2.8× over the work reported in [11]

15

Compared to 2D systolic array and cube architectures, our proposed hardware achieves 6.4× and 1.92× performance speedup on average, respectively.
We have estimated the energy consumption of the three 3D CNN networks on these systolic hardware implementations.
The energy reduction in the systolic cube and 3D systolic array design is primarily due to better data locality in both the temporal and spatial dimensions, significantly reducing buffer accesses.
The energy efficiency analysis, demonstrated in Figure (b), shows that the 3D systolic array architecture offers more energy savings compared the state-of-the-art approaches, perfectly aligning with the design goals for an embedded application.
The ASIC implementation of the proposed hardware design is also carried out for a fair comparison with [11] as their work targets an ASIC implementation.
We have achieved an area efficiency of 0.74 and 0.70 FPS/mm^2 for I3D and C3D models, respectively that maps to an improvement of 9.1× and 2.8× over the work reported in [11].

Experiments and Results (4/4)

Architecture	[5]	[6]	[4]	[8]		[3]	[9]		Proposed Work		
Model	C3D	C3D	C3D	C3D	R(2+1)D	I3D	C3D	R(2+1)D	C3D	I3D	R(2+1)D
Throughput (GOP/s)	172	334	357	79	35	1145	424	185	1024	1898	684
Latency(ms)	35.3	115	107	487	243	96	91	46	73	69	35
Energy/clip(J)	1.9	1.8	-	3.2	1.6	1.4	1.72	1.72	1.02	0.95	0.93
DSP(%)	93	99	96	48	48	98	98	98	0	0	0
LUT(%)	45	62	-	54	54	85	62	62	59	59	59
BRAM(%)	-	26	52	100	100	79	63	65	62	62	62

- This design minimizes the latency by 44%, 29%, and 55% for C3D, I3D and R(2+1)D, respectively
- Saving on average 49% 33%, and 45% of the energy for C3D, I3D and R(2+1)D, respectively

16

Table provides a detailed comparison with the previous implementation.
The proposed 3D systolic cube architecture significantly enhances the efficiency of 3D CNN computations by leveraging superior data locality in both temporal and spatial dimensions.
This design minimizes the latency by 44%, 29%, and 55% while saving on average 49% 33%, and 45% of the energy for C3D, I3D and R(2+1)D, respectively.
Through comprehensive experiments on these popular 3D CNN networks, we have demonstrated that our proposed approach achieves significant speedup and energy savings compared to traditional 2D systolic architectures.
We don't use DSP blocks. This gives better speed, lower power. Our design is not dependent on DSP blocks hence it is scalable.

Conclusion

- Performance enhancement of 3D CNN acceleration
  - 3D systolic architecture
- Optimize data reuse across both spatial and temporal dimensions
- Address the limitations of traditional 2D systolic array architectures
- Generalized dataflow model for 3D systolic array architecture
  - Efficient 3D CNN acceleration on resource-constrained and low-power hardware platforms
- Experimental results on popular networks such as C3D, I3D, and R(2+1)D
- FPGA and ASIC implementation

17

The proposed 3D systolic architecture significantly enhances the performance of 3D CNNs by optimizing data reuse across all dimensions.

Our design addresses the limitations of traditional 2D systolic array architectures by reducing their data transfer requirements.

The integration of a generalized dataflow model further maximizes hardware utilization

Enabling effective acceleration of computationally intensive 3D CNNs on resource-constrained and low-power hardware platforms.

Experimental results on popular networks such as C3D, I3D, and R(2+1)D demonstrate the superiority of our approach in terms of throughput and energy efficiency.

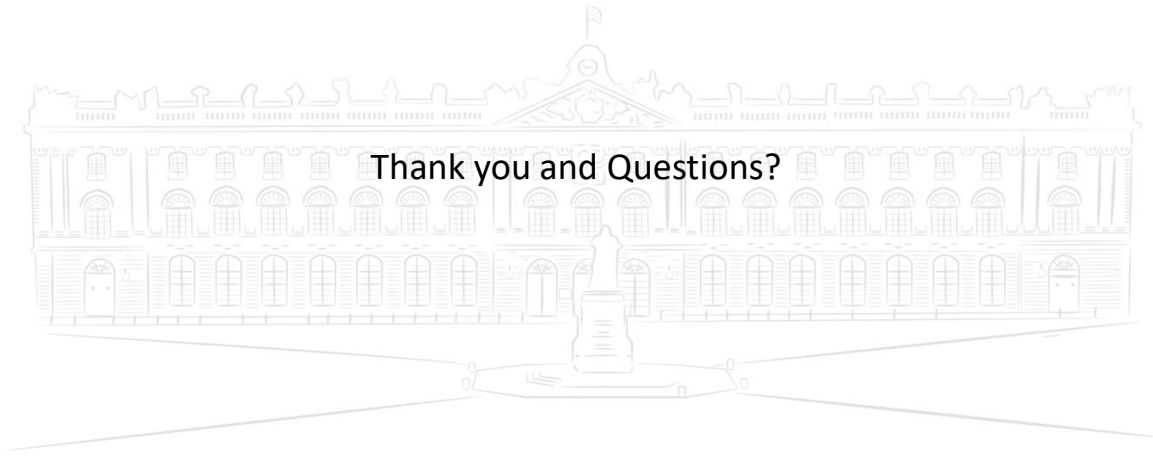
The results reported from FPGA and ASIC implementation

corroborate the advantages of our proposed approach

Acknowledgment

This work was funded by **LUMS University** through **Internal Research Grant** and the Conference Travel was made possible through **Faculty Travel Grant (FTG)**

The authors are grateful for this support.



Thank you and Questions?

References

[1] S. Mittal and Vibhu, “A Survey of Accelerator Architectures for 3D Convolution Neural Networks,” Journal of Systems Architecture, vol. 115, p. 102041, 2022.

[2] Y. Hu, Y. Liu and Z. Liu, "A Survey on Convolutional Neural Network Accelerators: GPU, FPGA and ASIC," 14th International Conference on Computer Research and Development (ICCRD), Shenzhen, China, 2022.

[3] F. H. Khan, M. A. Pasha and S. Masud, “Towards Designing a Hardware Accelerator for 3D Convolutional Neural Networks”, Computers and Electrical Engineering,, vol. 105, pp. 108489,2023.

[4] T. Tian, X. Jin, L. Zhao, X. Wang, J. Wang and W. Wu, "Exploration of Memory Access Optimization for FPGA-based 3D CNN Accelerator," Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020, pp. 1650-1655, 2020.

[5] H. Fan et al., "F-E3D: FPGA-based Acceleration of an Efficient 3D Convolutional Neural Network for Human Action Recognition," IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP), USA, pp. 1-8, 2019.

[6] Z. Liu, P. Chow, J. Xu, J. Jiang, Y. Dou and J. Zhou, "A Uniform Architecture Design for Accelerating 2D and 3D CNNs on FPGAs", Electronics MDPI, vol. 8, no. 1, 1 2019.

[7] J. Shen, Y. Huang, M. Wen and C. Zhang, "Toward an Efficient Deep Pipelined Template-Based Architecture for Accelerating the Entire 2-D and 3-D CNNs on FPGA," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 7, pp. 1442-1455, July 2020.

[8] M. Sun, P. Zhao, M. Gungor, M. Pedram, M. Leeser and X. Lin, "3D CNN Acceleration on FPGA using Hardware-Aware Pruning," 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2020, pp. 1-6,2020.

[9] P. Toupas, A. Montgomerie-corcoran, C.-s. Bouganis and D. Tzovaras, "Harflow3d: A Latency-Oriented 3D-CNN Accelerator Toolflow for HAR on FPGA Devices", Proceedings of International Symposium on Field-Programmable Custom Computing Machines FCCM, 2023.

[10] K. Hegde, R. Agrawal, Y. Yao and C. W. Fletcher, "Morph: Flexible Acceleration for 3D CNN-Based Video Understanding," 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, Japan, pp. 933-946, 2018.

[11] Y. Wang, Y. Wang, C. Shi, L. Cheng, H. Li and X. Li, "An Edge 3D CNN Accelerator for Low-Power Activity Recognition," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 5, pp. 918-930, May 2021.

Only [11] is related to 3D CNN dedicated architecture based on Output stationary and ASIC based. Ours is Weight Stationary and FPGA based.