A 40.96-GOPS 196.8-mW Digital Logic Accelerator Used in DNN for Underwater Object Recognition

Chua-Chin Wang, *Senior Member, IEEE*, Ralph Gerard B. Sangalang, *Member, IEEE*, Chien-Ping Kuo, Hsin-Che Wu, Yi Hsu, Shen-Fu Hsiao, *Member, IEEE*, and Chia-Hung Yeh, *Senior Member, IEEE*

***Abstract*— This investigation presents a digital logic accelerator (DLA) design of a neural network hardware that utilizes output reuse. The DLA is used in the detection mechanism of underwater objects that was deployed in an underwater vehicle. A modified YoloV3-tiny network was also implemented to detect more than 20 underwater objects. The proposed DLA uses processing units that have parallel architectures of output windows, and output channels. Moreover, a new Inter-Controller is designed to control the direct memory access (DMA) together with a new Reshape module to improve the performance and power efficiency. A detailed description of the design as well as the measurements on silicon are presented. The chip is realized using a typical 180-nm CMOS process. It showed a performance result of 40.96 GOPS and the power consumption is 196.8 mW. The DLA was tested to demonstrate 19.88 frames per second and 40.96 GOPS.**

***Index Terms*— Deep learning, deep neural networks (DNN), hardware accelerators, energy efficient, reshape module.**

I. Introduction

ARTIFICIAL (AI) intelligence and nanotechnology are expected to be a 193.2 billion US dollar industries by 2025. These technologies were also identified by the United Nations as "frontier technologies" in its 2021 Technology and Innovation Report [1]. Convolutional neural networks (CNN) and deep neural networks (DNN) have been in the forefront of AI applications. Applications of CNN and DNN in computer vision includes tasks such as image detection and recognition. It has been used in areas as agricultural systems [2], biomedical robots [3], industrial robots [4], autonomous vehicles [5], aerial drones [6], and underwater vehicle [7]. With this, demands for computing resources also increased. State-of-the-art image processing architectures such as AlexNet [8], ZFNet [9], Inception [10], VGG [11], ResNet [12], ResNeXt [13], and SENet [14] normally wins the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). For this reason, academic and industrial communities pay close attention to AI-based developments primarily for its wide array of applications. The number of published papers related to AI and machine learning (ML) increased ten times annually during the period of 1998 to 2017 [15]. Notably, the biggest challenge in AI-related applications is to equip it into autonomous underwater vehicles (AUV) [16]. The overall power in these underwater applications is constrained by the battery of the vehicle systems. Training underwater images is more complex than that for those objects on land due to the different lighting conditions and shortage of underwater samples.

Several hardware accelerators have been reported in the previous years [17]–[28]. The systolic architecture proposed in [17] and [18] have reconfigurable designs for different convolution kernels. Tu *et al.* uses reconfigurable data path and convolution engine design which makes it more hardware efficient but it has a high area cost due to the complexity of the control structure [17]. The low power design in [18] allowed the systolic data flow to run only in the PE rows rather than the entire 2D plane, thus allowing data reuse of filters & input features along with convolution kernel reuse. The spatial architecture of [23] is an improved version of [19] that uses clustering of sparse CNNs to achieve higher throughput but requires a larger area. A filter and input reuse in the streaming architecture in [20] is reported to be energy efficient but it has a low hardware utilization. Lastly, filter-type structures are presented in [21] and [22]. More importantly, none of the above works were focused on underwater object recognition applications.

Major Contributions of the Proposed DLA

In this investigation, the authors proposed and achieved as follows:

1) A modified network toward power effectiveness based on the YoloV3-tiny was presented to detect different

Manuscript received 25 March 2022; revised 3 June 2022; accepted 26 June 2022. This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 110-2221-E-110-063-MY2, Grant MOST 110-2218-E-110-008-, and Grant MOST 110-2224-E-110-004-. This article was recommended by Associate Editor A. James. *(Corresponding author: Chua-Chin Wang.)*

Chua-Chin Wang is with the Department of Electrical Engineering and the Institute of Undersea Technology, National Sun Yat-sen University, Kaohsiung 80424, Taiwan (e-mail: ccwang@ee.nsysu.edu.tw).

Ralph Gerard B. Sangalang and Hsin-Che Wu are with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan.

Chien-Ping Kuo is with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, and also with MediaTek Inc., Hsinchu 30078, Taiwan.

Yi Hsu is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, and also with MediaTek Inc., Hsinchu 30078, Taiwan.

Shen-Fu Hsiao is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan.

Chia-Hung Yeh is with the Department of Electrical Engineering, National Taiwan Normal University, Taipei 10610, Taiwan, and also with the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3187450.

Digital Object Identifier 10.1109/TCSI.2022.3187450

TABLE I
MODEL DATASET USED FOR TRAINING

Object label	No. of Images	Object label	No. of Images
Fish	10,000	Sea cucumber	4,800
Lion-fish	6,000	Glass Bottle	3,500
Shark	7,000	Plastic Bottle	3,400
Turtle	5,000	Tire	3,700
Diver	10,000	Coral	6,478
Jellyfish	8,000	Octopus	5,000
Ray	2,500	Squid	2,500
Stone	4,000	Shrimp	5,000
Fish Net	4,500	Seahorse	5,000
Kelp	9,000	Starfish	5,000
		Total	110,378

underwater objects. The reason is underwater vehicles are battery-operated with limited battery storage.

2) A high throughput and low power hardware accelerator utilizing output reuse is implemented on silicon to assist in detection and recognition for underwater objects.
3) A new Inter-Controller is designed to minimize power dissipation of the DLA.
4) A hardware padding is proposed to reduce DRAM transmission between the hardware and CPU, achieving lower power dissipation performance.

The power constraint of the design is highly prioritized so that when deployed to an underwater vehicle, the object recognition would not consume a large chunk of power of the vehicle. Initial designs of the DLA was downloaded and implemented in the ZCU102 FPGA board. It was found out that the DLA alone consumes 4.322 W of power, hence it is impractical to deploy it for underwater missions.

The rest of the report is organized in the following manner. Section II presents the methodology used in designing the hardware accelerator. A discussion of the data set preparations is presented in this section as well as the proposed network for underwater object detection. A detailed discussion of all modules is presented. Section III presents the simulation and measurement results as well as system validation of the hardware accelerator. And to wrap the it up, the conclusion drawn in this study is presented in Section IV.

II. DLA DESIGN AND ANALYSIS

A. Data Set Preparation

An image database is needed to be able to train, verify, and test any neural network. In this investigation, we collected various images from repositories such as ImageNet [29] and The Fish Database of Taiwan [30]. A total of 110,378 images were used in the training, verification, and testing of the proposed network. A summary of the number of images used is presented in Table I. The network is trained to classify twenty objects, namely fish, lion-fish, sharks, turtles, divers, jellyfish, and so on, as shown in Table I.

B. Proposed DNN for the Underwater DLA Implementation

Because of the hardware architecture's limited resources, the size of the DNN model must be lowered as much as necessary without compromising too much accuracy. The authors proposed a lightweight network based on the YoloV3-tiny network [31]. Fig. 1 depicts the modified neural network to be used for the DLA. The symbol N represents the number of output channels of the proposed network given by $N = 3(4 + 1 + x)$, where x represents the number of object to be identified. The proposed network has 3 scales of prediction, 4 offsets of the bounding box, and an objectiveness of 1, similar to YoloV3 [32].

TABLE II
EFFECTS OF NUMBER FORMAT ON MAP OF THE NETWORK

Format	mAP
float-32	81.4%
float-16	81.2%
fixed-16	81.2%
fixed-8	17.9%

TABLE III
COMPARISON WITH OTHER NETWORKS FOR UNDERWATER DETECTION

	Zhang *et al.* [33]	Yeh *et al.* [16]	**Ours**
mAP (%)	79.54†/92.65‡	89.56‡/80.12⋆	81.2
Backbone	MobileNetV2	FPN	YoloV3-tiny
GFLOPs	N/A	5.06	2.06
No. of classes	4†/3‡	3⋆	20
FPS	44.22‡	0.5∼1	19.88
Hardware	CPU & GPU	Raspberry Pi 3	ASIC
Power	N/A	≈ 2 W	196.8 mW

† based on the URPC2020 datasets [34]
‡ based on the brackish datasets [35]
⋆ based on their own generated datasets [16]

The proposed network was firstly tested by software to see the effects of the data format. Results of the experiments were presented in Table II. It can be seen that lower bit resolution offered poor performance in terms of the mean average precision (mAP). A float-32 format obtained the best mAP for the network. However, floating point operations require more hardware resources. It was only 0.2% better than float-16 and fixed-16 format. Hence, the risk of more operations outweighs the performance benefits. Since, the float-16 and fixed-16 formats offer the same performance, it was best to choose the fixed-16 format to gain smaller area without sacrificing the performance. The computation amount of the proposed network is found to be 2.06 GFLOPs (giga-floating point operations) using the 16-bit fixed point precision data.

Table III shows a comparison of the proposed network to other networks that was implemented specifically to recognize underwater images. It can been seen that the proposed network can detect more objects at almost the same mAP values. It also has the smallest numbers of operations such that it will have lower power consumption. The network by Zhang *et al.* [33] was implemented in an Intel Xeon Gold 6130 CPU with RTX2080ti GPU. It offers a good object detection speed at the expense of using power hungry devices. The network by Yeh *et al.* [16] was implemented in a Raspberry Pi 3 B+ platform, a low-power mini-PC with an power consumption of around 2 W. Our implementation offered a very-low power performance at the expense of an acceptable real-time FPS (≈20 FPS).

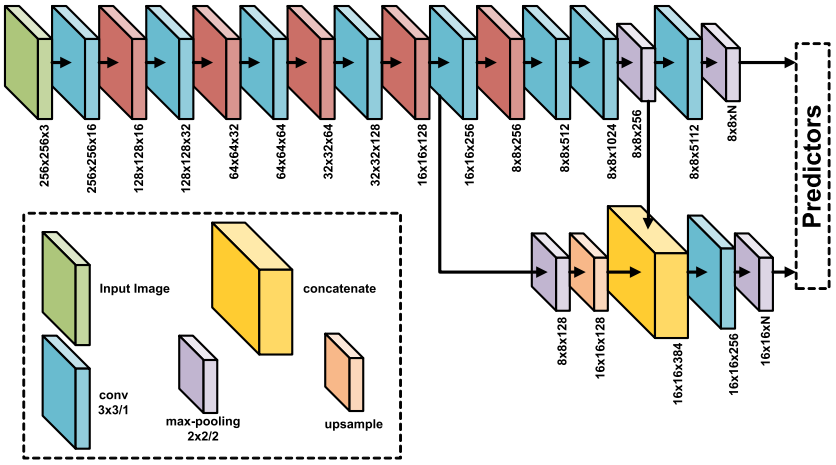

Fig. 1. Lightweight network based on YoloV3-tiny.

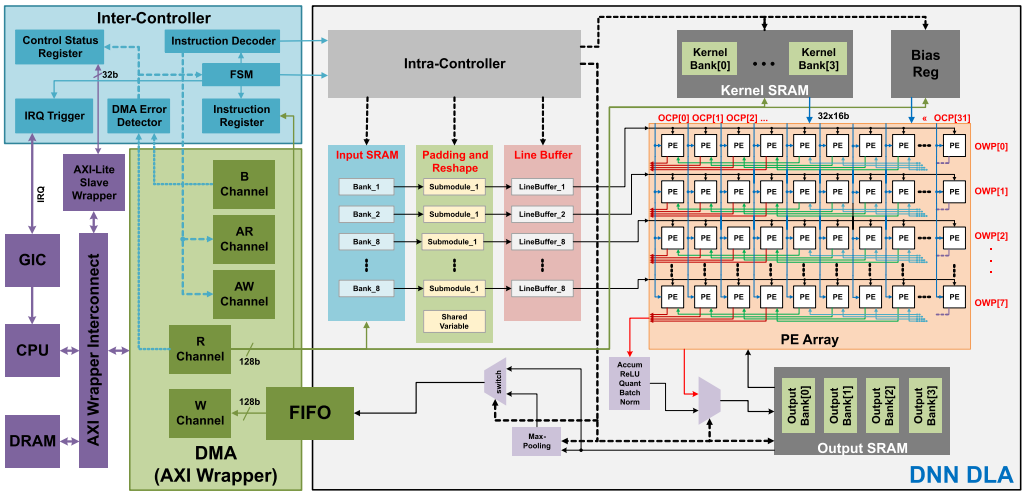

Fig. 2. Proposed hardware accelerator architecture (DLA) for DNN.

C. Neural Network Hardware Architecture

The proposed deep convolutional network architecture is presented in Fig. 2. A new Inter-Controller is used for the Direct Memory Access (DMA) as well as new Reshape modules in the DNN DLA to achieve low-power and high-performance for the hardware accelerator. These new modules will be used to control the convolution operations in the PE (processing element) Array in Fig. 2. Convolution operations are normally used to perform discrimination of difficult objects. This process is achieved by digital filters that are represented by 1D or 2D matrices. Depending on the filter weights, or kernel weights, the convolution can have different effects on an image. The image can be blurred, sharpened, recolored, or detected as boundaries or edges. Fig. 3 shows the structure of the convolution layer. It is composed of an input feature map, kernel, and output feature map. The convolution operation has M number of output channels (OM) and N number of input channels (IM) with kernel size, K. The size of the output feature map is H × W.

Even though large-scale computing models offers better recognition performance, the input feature map (IM) and the kernel generated after training cannot be stored in the on-chip buffer. The workaround to this is dividing the entire IM into multiple blocks to become smaller size. Tile-based

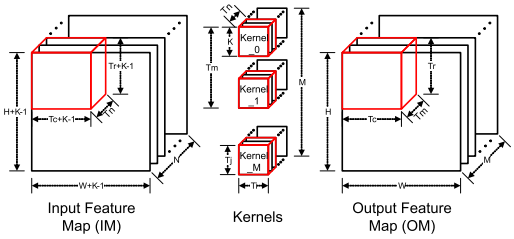

Fig. 3. Illustration of the convolution operation.

calculation is used to make the required memory size smaller. The internal memory required under batch calculation are Tm and Tn, the number of OCHs and ICH, respectively. The size of the OM is Tr × Tc, while for kernel it is Ti × Tj. These parameters will determine the overall size of the on-chip buffer. In actual hardware calculations, the data in this batch of tiles are calculated in numerous cycles. Hence a very large number of computation units are required. Usually, these operations require smaller number of calculations at higher computation cycles so that operation needs to be parallel. The corresponding parameter of the operation value is composed of Pm, Pn, Pr, Pc, Pi, Pj, which respectively represent the number of OCHs, the number of ICHs, the length & width of

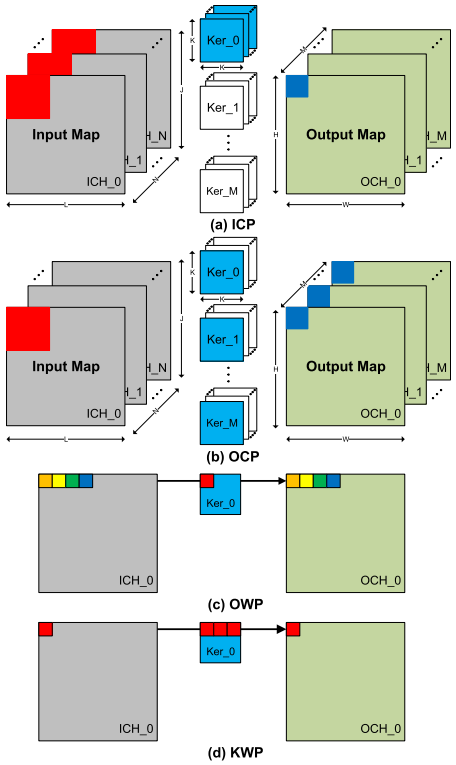

Fig. 4. Convolution operation and SRAM allocation in (a) ICP, (b) OCP, (c) OWP, and (d) KWP.

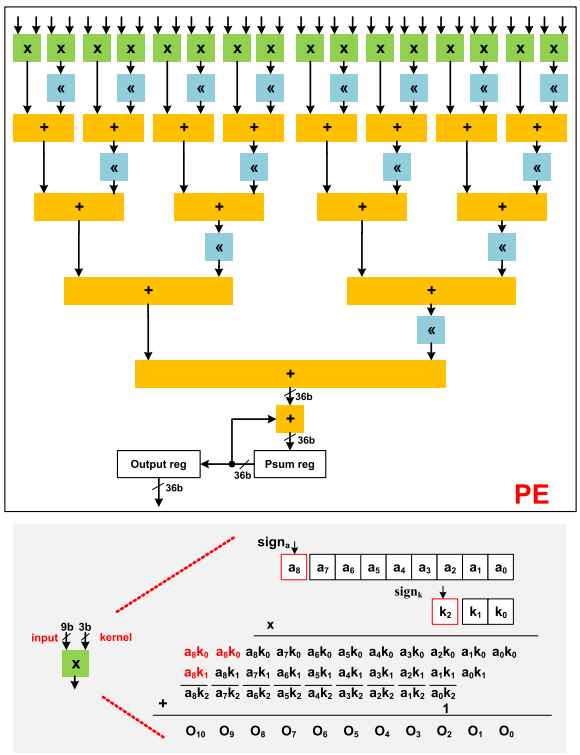

Fig. 5. Processing element (PE) internal architecture.

OM, and the length & width of the kernel to be operated in parallel in a cycle.

In this design, the number of arithmetic units (number of PEs) represents the number of multiply and accumulate (MAC) cycle operations. When PE parallel operations are in different positions, the size of the on-chip buffer will definitely change differently. Referring to Fig. 3, where Tm is in the grid, Tn, Tr, Tc, K parameters represent how many OCH, ICH, output windows, and kernels are calculated by the hardware in a unit time, respectively. The total number of arithmetic units in the hardware is the product of these parameters. The convolution operation usually has three parallel computing methods on the hardware classified as: 1. Input Channel Parallel (ICP), 2. Output Channel Parallel (OCP), and 3. Window Parallel (WP).

Referring to Input Channel parallel (ICP) in Fig. 4(a), the input features are placed in all channels and then convolved with the set of corresponding row kernels and the results are placed in a single portion of the output map. The Output Channel Parallel (OCP), on the other hand, saves multiple convolution operation results in different channels of the output map as shown in Fig. 4(b).

The third computing method is the (Window Parallel) WP, which is divided into Output Window Parallel (OWP) and Kernel Window Parallel (KWP). The OWP is done by calculating the same OCH value using PE operation and the same weight position. This is shown in Fig. 4(c). The PE operation using different weight position and the same output window is called KWP as shown in Fig. 4(d). However, this method requires more ports to support the calculation, the area of the input/output buffers of the two parallel calculation methods will not increase.

D. Sub-Circuit Design and Operation

1) Multi-Precision Processing Element (PE): The PE hardware supports MAC operations. The sub-PE is a 9 × 3 multiplier accepting signed 8-bit inputs and signed 2-bit kernels. The PE uses 16 sets of multipliers that are constructed to operate simultaneously. The adder tree presented in Fig. 5 supports multiple accumulation process according to the degree of computation needed. This design allows input precision to support 16 bits or 8 bits, while the kernel precision supports 16, 8, 4, 2 bits, so it can dynamically support Multi-Precision (MP). These excludes the sign bit for the precision. For the PE to support low precision operation, multiple input channel (ICH) data should be loaded at the same time in the multipliers for accumulation.

2) PE Array: The 8 × 32 PE array is composed of 8 rows of OWP and 32 columns of OCP as presented in Fig. 6. To reduce the area and power consumption of the array, the excitation functions (ReLU, Leaky ReLU, Max-pooling, Concatenation), quantization, and batch normalization of the pixels are performed outside the array prior to the output SRAM.

3) Input SRAM: Since our design is designed to be multi-precision, the input precision can be 16, 8, 4, and 2 bits precision. The PE array operates at 100% capacity when the input channel is using 16 bit precision. When 2-bit input is used, 8 times throughput is required, such that 8 banks are designed. Fig. 7 shows the input SRAM in this design, where 8 banks are included, each bank = 128 bits, which are all composed of two-port SRAM (1R1W). The bank is divided into two halves as double buffers to make the access time shorter. The double buffer design has an advantage of using

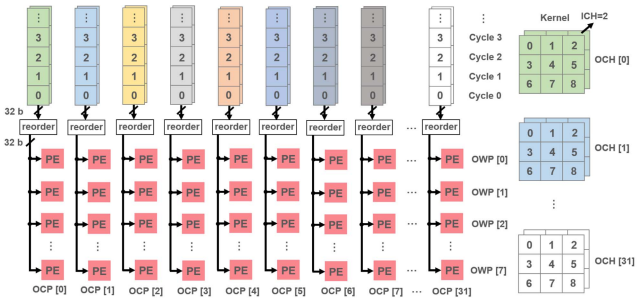

Fig. 6. PE array architecture.

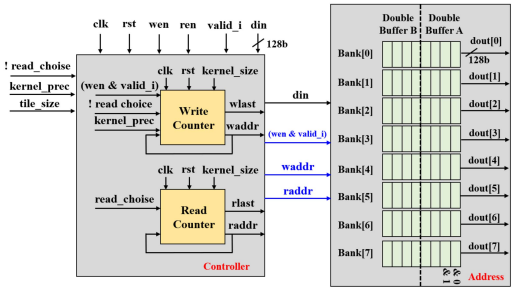

Fig. 7. Input SRAM architecture.

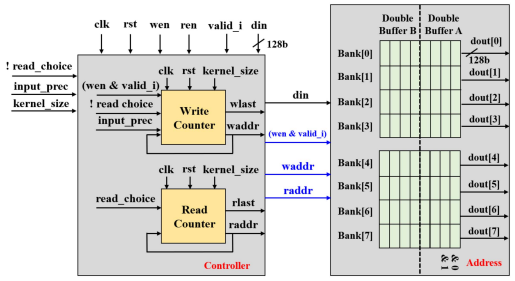

Fig. 8. Kernel SRAM architecture.

only half the bank capacity during the PE operation while the other half is being used to load the required data. Hence, the operation is faster and the throughput is doubled.

4) Kernel SRAM: The kernel SRAM is also designed as a double buffer composed of two-port registers. The kernel SRAM in this design uses 8 banks of 128 bits wide consisting of two-port register files, as shown in Fig. 8. Since the PE array has 32 columns of OCP, when the required throughput of the kernel is only 16×32 bits, 4 banks are enough to be utilized. To make the PE usage rate close to 100%, twice the weight can be realized.

5) Output SRAM: The output SRAM in this design, shown in Fig. 9, uses 4 banks with a bit width of 128 bits composed of dual-port SRAM (2R or 2W or 1R1W). Since there are 256 PEs in the PE array, each set of PEs output 16-bit data. When the throughput required is only 16×32 bits, 4 banks are enough to meet the requirements of the PE. When PE operation needs 8×32 bit values, it will take 8 cycles for the output SRAM to receive all incoming data.

6) Instruction SRAM: Instruction SRAM consists of single-port SRAM with a depth of 512 bits and a bit width of 64 bits corresponding to the bit width of the instruction set architecture. The instruction set architecture is introduced in

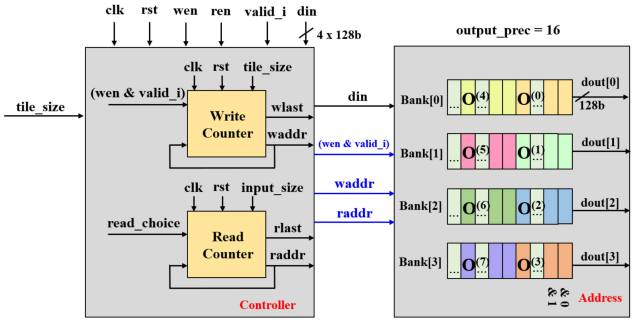

Fig. 9. Output SRAM architecture.

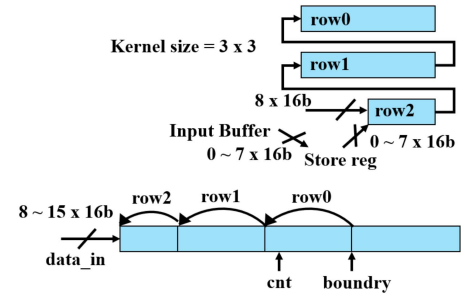

Fig. 10. Line buffer with 3×3 convolution.

detail in Section II-H. The total size is 4 kb and the maximum single transfer limit of AXI Burst Mode is aligned.

7) Line Buffer: Fig. 10 shows the schematic diagram of implementing the line buffer with 3×3 convolution. This was done because during the convolution operations, the same data will be repeated many times. If this happens to the SRAM, a large repeated number of repeated read will occur. The line buffer serves as a temporary storage for the input during the convolution operations. The line buffer uses a 16-bit register to match the width of a 128-bit two-port SRAM (1R1W), and pushes 8 groups of 16 bits at a time through the shift register. During convolution, the same data is repeated several times. To minimize the number of SRAM read/write operations, the input pixels are stored firstly in the line buffer.

8) Reshape Module: The size of the on-chip buffer in this design is limited so that a new Reshape module is proposed to resolve this problem. When the kernel size is >1, the input feature map is usually padded to maintain the input feature map size. Furthermore, since the hardware adopts tile-based design, the input feature map is divided into many small tiles, and these small tiles also need to have padding.

In addition, when tile <32, the Reshape module is used to merge N small tiles into a 32×32 large tile and then do the padding so that the transmission is not $N\times$ the number of transmission. This reduces the number of transfer and load burden of the CPU.

When a layer of a feature map is larger than the maximum tile, which is 32, the feature map will be cut into smaller tiles as shown in Fig. 11. The general detection model or the image recognition uses Stride2 pooling. This can make the on-chip next layer operation reduce the area of the tile by 4 times. The buffer usage is reduced while the number of transmissions is increased. Therefore, it is possible to use a Reshape module to merge multiple tiles with a tile size ≤ 32. For example,

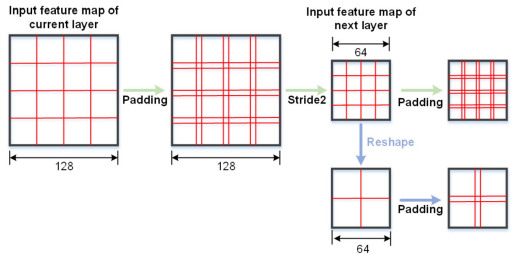

Fig. 11. Tile cutting and padding in different strides.

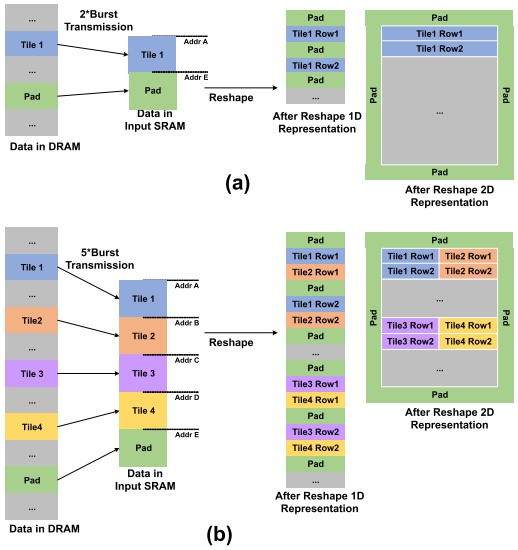

Fig. 12. Re-organization using the Reshape module after scheduling, (a) single-tile transmission, (b) multi-tile transmission.

assume a tile exceeds the maximum size by 8×8. A total of 4 combinations of 8×8 tiles can turn into 32×32 tile, and then perform the operation through the PE array.

If the input feature map is larger than the output map, additional padding is used to make them equal. The proposed design is implemented in tile-based way, where the tiles are stored in the DRAM. After the reshape mode, all the data returns to the original shape, as shown in Fig. 12.

E. Hardware Padding

Since the padding information exists in different sections of DRAM, Tile and Padding information is needed to be reorganized through the Reshape module. However, these information are extracted through the CPU, as shown in Fig. 13(a). Since the tile boundaries might be discontinuous, arranging these discontinuous memory addresses is time consuming. The authors proposed a hardware padding module which replaces adjacent tiles with a boundary of its own tile. This is called Reflection Padding. Fig. 13(b) shows the Reflection Padding scenario. Another benefit of Reflection Padding can be seen in the perspective of the DRAM transmission. If the CPU executes the padding for tile size greater than 32, it will require two burst transmissions while if padding is executed in the hardware, only one burst transmission is required.

F. Direct Memory Access

There will be a lot of data movement operations in the convolution operation. If these movement operations are executed

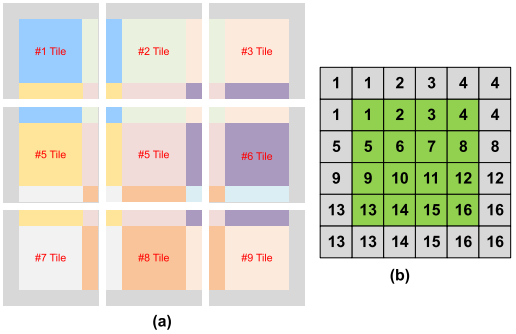

Fig. 13. (a) Tile-based Padding; (b) Reflection Padding architecture.

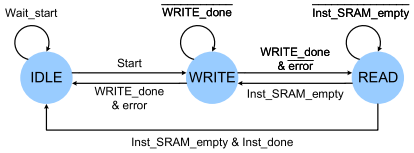

Fig. 14. Simplified FSM of the Inter-Controller.

by the CPU, the data access time will be much longer than the hardware calculation time. Therefore, we added a DMA between the DRAM and the hardware accelerator. The DMA is based on the AXI4 bus protocol. The burst mode of this protocol can reduce the requirements and set the number of cycles of the bus, and DMA belongs to the Master (CPU or GPU).

G. Controller

There are two controllers in the design; Inter- and Intra-Controller. Inter-controller is responsible for controlling the Direct Memory Access (DMA) and the Intra-controller is in the hardware accelerator. Its internal functions include instruction fetching and decoding, DMA error monitor, interrupt trigger, and control register. The control signals determine the operation of the connected modules, including the finite-sate machine (FSM) jump, circuit switching, and power consumption. The simplified FSM of the Inter-Controller is presented in Fig. 14. It shows three general states: 1. Idle State, 2. Write state, and 3. Read state. At Idle state, func $= 0$, the system is waiting for a command to proceed to take actions. During Write state, func $=$ (1, 2, or 3), the controller commands the DLA hardware to perform the calculations. Lastly, in the Read state, func $= 4$, the controller will then access the output SRAM when the write operation is done with no errors. The overall function of the hardware accelerator is shown in Fig. 15.

H. Software and Hardware Integration

The software and hardware integration of the proposed network consist of two stages, Initialization and Execution stages. The Initialization uses a single CPU thread and is divided into three main parts: 1. DNN Model Analysis, 2. DRAM Utilization & Optimization, and 3. Layer Switching. On the other hand, the Execution stage uses three threads that are operating in parallel. These are: 1. Image Pre-processing, 2. Hardware (DLA) control, and 3. Post-processing of results. Fig. 16 shows the diagram of the system operation of the

List		Detail
# of MACs		8x32
Reuse type		Output reuse
Parallel	Input channel	1
	Output channel	1~32
	Output windows	8
Tile Size (without padding)		8x32
Precision	Input	16 bits
	Weight	16 bits
	Output	16 bits
Support layer	Convolution (kernel size/stride)	1x1/1, 3x3/1, 3x3/2, 5x5/1
	Activation function	ReLU、Leaky ReLU、Max-pooling、Concatenation
	Batch Norm.	Pre-processing on software
On-chip Buffer	Input	Two-port, Bank=1, Width=128 bits, Depth=444
	Weight	Two-port, Bank=4, Width=128 bits, Depth=242
	Output	Dual-port, Bank=4, Width=128 bits, Depth=1024

Fig. 15. Overall functionality of the DLA.

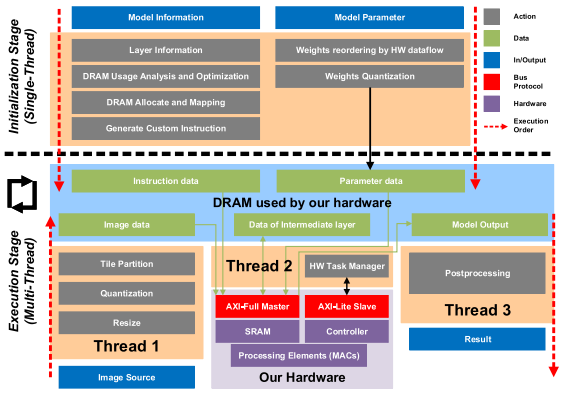

Fig. 16. Illustration of the Software/Hardware integration.

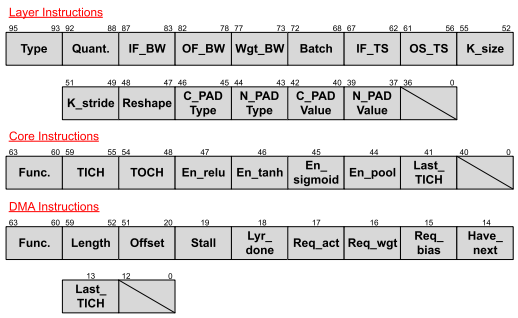

Fig. 17. Instruction set.

network with the software and hardware integrated with each other.

We decided to create an instruction set architecture based on direct control in order to make the hardware dynamically adaptable to fit the settings of different parameters of each layer of the DNN model. Fig. 17 depicts the instruction set architecture of the system.

1) Layer Instruction: The layer instruction set is used by the model corresponding to each layer that will be used by the DLA. It is composed of 96-bit instructions. Each layer that needs acceleration is equipped with an instruction, and once if an operation is completed, the layers can be replaced by another instruction. These instructions are executed only

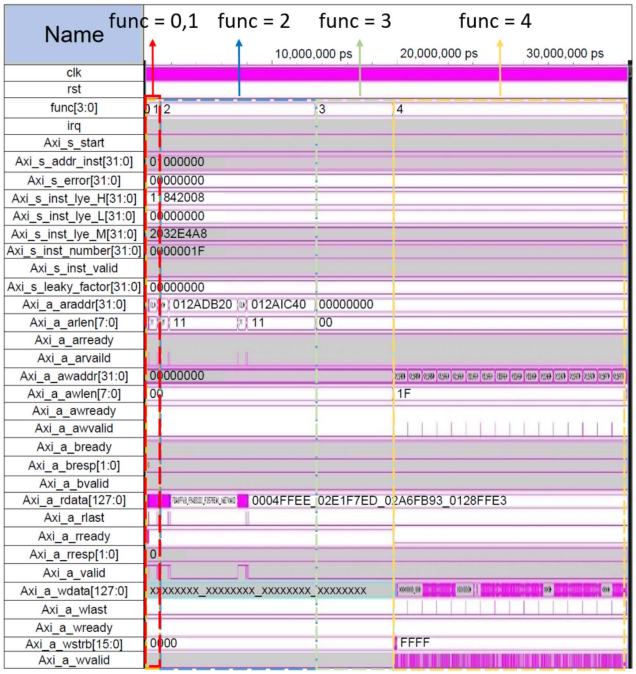

Fig. 18. Pre-layout simulation result after memory scheduling and reorganization by the Reshape module.

once per operation so that it will not mess up with the other 64-bit instructions

2) Core Instruction: The core instruction is based on the tile-based design. It is based on the operation of one tensor. It contains the instructions on how many ICH, OCH, and activation functions to be used in parallel hardware operation, or whether to use hardware accelerated pooling, etc. The main prupose of this set is to provide for the information required by the Intra-Controller of the DLA. With this, the control logic can be simplified and use the CPU to schedule instructions reducing the burden in hardware control.

3) DMA Instruction: The DMA proposed is designed using the AXI4 bus protocol, where commands must provide the starting address of the transfer (DRAM Address) and the transfer length (Length), while other fields provide hardware accelerator information, including classification, transmission and operation synchronization of incoming data, etc.

III. Simulation, Measurement, and Implementation

The proposed DLA architecture is implemented using a typical 180-nm CMOS process.

A. DNN Accelerator Simulation Results

The pre-layout simulation, shown in Fig. 18, operating at 100 MHz uses NC-Verilog to verify the functionality of the overall circuit. The states of the FSM are decribed as follows:

- Idle State: When func = 0, the Axi_s_start signal is logic HIGH. This means that the controller is awaiting command for the hardware to proceed. The state machine is said to be in the standby mode.
- Load State: When func = 1, the DRAM feeds the data on the on-chip buffer. No output is exported and Axi_a_wdata is unknown.

Fig. 19. Post-layout simulation results at the worst corner.

- Processing State: When func = 2, the on-chip buffer has been loaded a complete tile information and has started to send data to the PE array for calculation.
- Calculate State: When func = 3, there is no data being processed in the on-chip buffer from the DRAM. The DLA only performs calculation from the PE array.
- Store State: When func = 4, the DLA has finished all calculations and the signal Axi_a_wdata will now show a valid value. It will output all the data from the on-chip buffers. Once all data is out, the irq signal and the DLA will know that the tasks are done.

Fig. 19 shows the post-layout simulations in the worst case corner, namely, the slow-slow (SS) process corner, and 75° C at a frequency of 100 MHz. It shows a consistent result as those given by the pre-layout simulation in Fig. 18.

B. FPGA Emulation

Object detection was firstly implemented in FPGA to test the functionality of the proposed architecture. The FPGA board used in the implementation is Xilinx Zynq UltraScale+ MPSoC ZCU102 evaluation board. This will ensure that the proposed architecture operates based on the functionality it is designed. Fig. 20 shows the diagram of the object recognition system implemented in the FPGA. It can be seen from the figure that the proposed architecture is performing the correct object detection algorithm.

C. Error Analysis

To error analysis used in this investigation requires three (3) sets of data; 1.) results of YoloV3-tiny operation performed in a CPU; 2.) pre-layout simulation results, and; 3.) post-layout simulation results. The results obtained from the CPU and YoloV3-tiny is regarded as the benchmark.

$$e = \frac{x_{cpu} - x_{sim}}{x_{cpu}} \tag{1}$$

The simulation results x_{sim} (pre- or post-layout simulations) is compared with the CPU results x_{cpu} to get the error e using

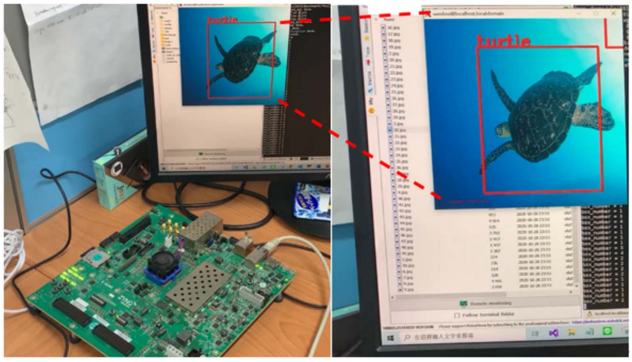

Fig. 20. FPGA emulation verification of the object recognition system.

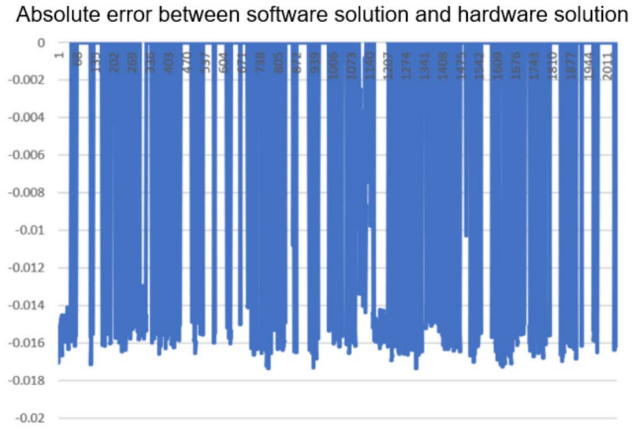

Fig. 21. Error analysis of the software and hardware solutions.

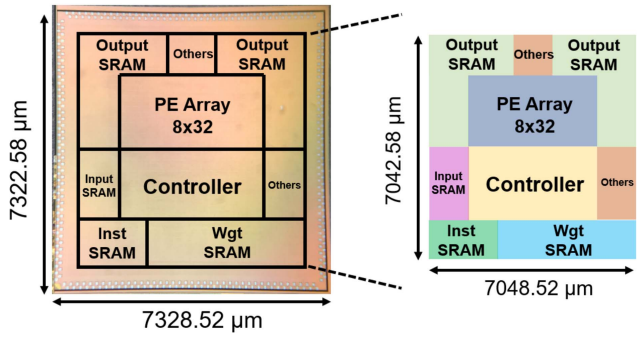

Fig. 22. Die photo of the DLA.

Eqn. (1). The simulation results are compared with each other to verify whether the results are consistent. The absolute error is found within 1.8% as shown in Fig. 21. Since the absolute error is relatively small, the fix16 format used by the proposed hardware was not an issue comparing to the float32 format of the CPU.

D. Prototype Implementation

Fig. 22 shows the die micrograph of the DLA chip. The size of the chip is $7332.58 \times 7328.52\ \mu m^2$ with a core $7042.58 \times 7048.52\ \mu m^2$. The implemented chip is composed of the PE Array, the Controller, the SRAMs (input, weight, instruction, & output SRAM), and other auxiliary blocks such as input & output buffers, controller registers, reshape module, padding, and bias circuits.

Fig. 23 shows the area utilization of the chip. It shows that the SRAMs occupied 55% of the overall chip area where 33%

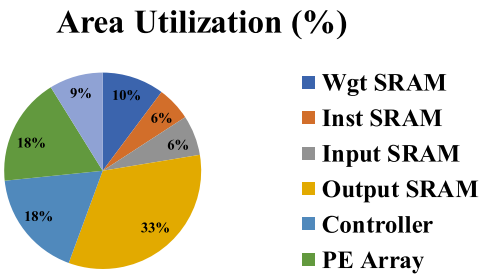

Fig. 23. Area utilization of the chip.

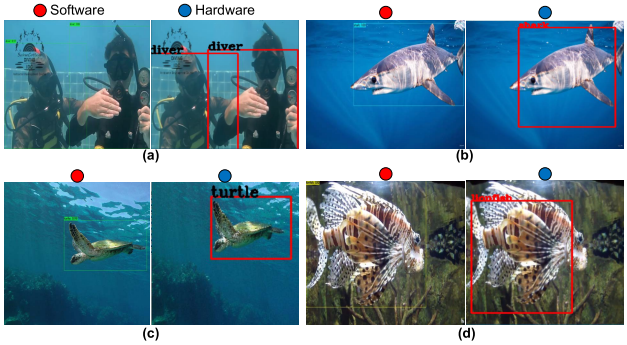

Fig. 24. Side-by-side comparison of the CPU vs. the proposed DLA hardware implementations in object recognition of (a) Divers, (b) Shark, (c) Turtle, and (d) Lionfish.

is for the output SRAM alone. The PE array and the controller occupies 18% of the chip area, respectively.

Fig. 24 shows the side-by-side comparison of the implementations in the CPU using the proposed YoloV3-tiny algorithm and the proposed DLA hardware. The DLA hardware reads the video image from a host controller (via Real-time Transport Protocol) or a streaming camera (via Real-time Streaming Protocol). The proposed DLA was designed using Verilog HDL. Both systems can successfully recognize all objects, including fish, turtle, divers, sharks, etc.

E. Measurement Results

The measurement was done using an SOC test platform (Advantest V93000 PS1600), as shown in Fig. 25. Measurements shows consistent results matching with the simulations. The measurement was carried out using 5 DLA chips measured 5 times each. Figs. 26 and 27 shows the hardware accelerator jump test during func = 1 & 2, and func = 3 & 4, respectively. The irq signal will be high for a short period during FSM jump and is shown in Fig. 28. Fig. 29 shows the shmoo plot of the DLA chip. The chips can reach the designed frequency of 100 MHz.

Fig. 30 shows the power utilization of the hardware accelerator at 100 MHz frequency. The biggest one comes from the PE array at 86.4 mW or 43.9% of the total power of the chip. The SRAMs utilized 45.3% of the total power consumed by the chip. Table IV shows the breakdown of the power consumption of each module. The overall power consumed by the chip is 196.8 mW, which is a 95% reduction from our initial FPGA implementation.

Table V shows the comparison with many recent NN accelerator works. The proposed design is implemented using a typical 180-nm CMOS process. The supply voltage used is

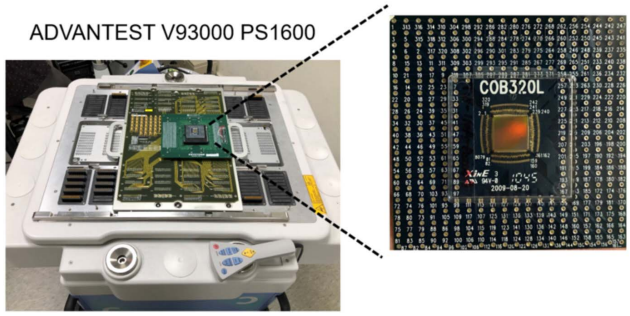

Fig. 25. DLA measurement environment.

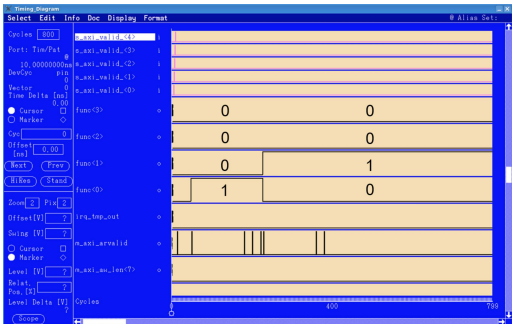

Fig. 26. DLA jump test, func = 1 and func = 2.

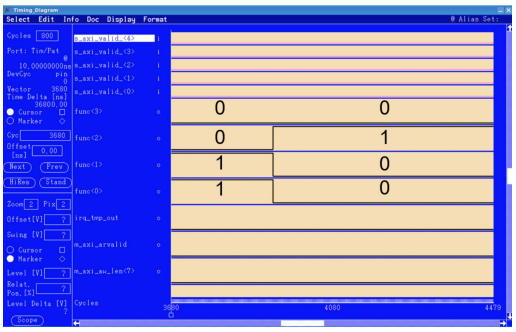

Fig. 27. DLA jump test, func = 3 and func = 4.

TABLE IV
POWER DISTRIBUTION AT 100 MHZ

Module	Power (mW)	%
Input SRAM	3.22	1.64%
Weight SRAM	8.856	4.5%
Output SRAM	77.22	39.24%
Bias Reg.	0.9876	0.5%
PE Array	86.4	43.9%
Line Buffer	7.56	3.84%
Reshape / Padding	6.892	3.5%
FIFO	2.986	1.5%
Pooling	0.8244	0.43%
Ctrl	1.854	0.95%
TOTAL POWER	**196.8**	**100%**

1.8 V operating at 100 MHz. The proposed design consists of 256 MACs and uses 16-bit precision format. The measured performance of the proposed design is 40.96 GOPS at 196.8 mW power. Since the proposed network is found to be 2.06 GFLOPs, the frame rate of the system is 19.88 FPS. It has an area efficiency of 0.7638 GOPS/mm^2 and power efficiency of 0.2081 TOPS/W. The figure of merit (FOM) used to compare the designs is the power efficiency and normalized

TABLE V
COMPARISON TABLE WITH PREVIOUS WORKS

	JSSC [19]	ISSCC [24]	TCAS-I [25]	TVLSI [26]	ISSCC [27]	JETCAS [28]	Ours
Year	2017	2017	2018	2020	2020	2020	2022
Process (nm)	65	28	65	65	65	40	**180**
Verification	Meas.	Meas.	Simu.	Meas.	Meas.	Simu.	Meas.
Voltage Supply (V)	1.0	1.1	1.2	1.2	0.6	0.9	1.8
Area (mm^2)	16	1.87	10.6	16	2.56	200	53.63
Frequency (MHz)	200	200	200	60	0.25	200	100
On-chip Buffer (kb)	181.5	144	139.6	848	16	118	150
No. of MACs	168	1024	64	512	512	128	256
Activation bit-width	16	16	16	16	8	16	16
Kernel bit-width	16	16	16	16	8	16	16
Performance (GOPS)	42	76	23.4	61.44	0.471	51.2	40.96
Power (mW)	278	300	93.4	173	0.0106	153.94	196.8
Area eff. (GOPS/mm^2)	2.6250	40.6417	2.2075	3.8400	0.1840	0.2560	0.7638
Power eff. (TOPS/W)	0.1511	0.2533	0.2505	0.3551	44.4340	0.3326	0.2081
*FOM	3.367	2.943	8.042	3.420	0.446	3.696	**20.813**

$$*\text{FOM} = \frac{\text{Frequency(MHz)} \times \text{GOPS}}{\text{Power(mW)}} \times \left(\frac{\text{process}}{180\text{nm}}\right) \times \left(\frac{\text{voltage}}{1.8\text{V}}\right)^2$$

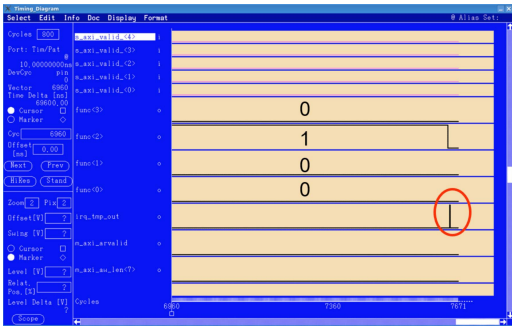

Fig. 28. irq signal jump of the hardware accelerator during FSM jump.

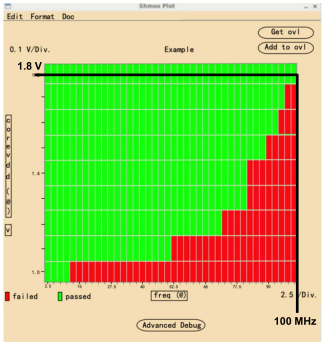

Fig. 29. Shmoo plot of the hardware accelerator.

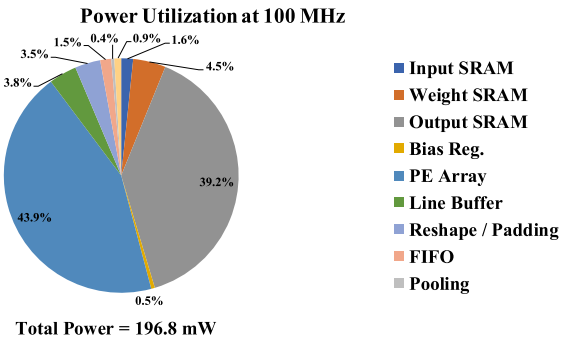

Fig. 30. Power utilization of the chip at 100 MHz.

with the operating frequency, CMOS process feature size, and supply voltage. It can be seen from Table V that the proposed design has the best FOM, 20.813 TOPS-GHz/W.

TABLE VI
SYMBOLS USED IN TEXT

Symbol	Meaning
M	no. of output channels (OCH)
N	no. of input channels (ICH)
K	kernel size
J	input feature map height
L	input feature map width
H	output feature map height
W	output feature map width
Tm	no. of OCH required for computation
Tn	no. of ICH required for computation
Tr	height of output feature map for computation
Tc	width of output feature map for computation
Pm	no. of OCH for parallel operations
Pn	no. of ICH for parallel operations
Pr	length of OM for parallel operations
Pc	width of OM for parallel operations
Pi	length of kernel for parallel operations
Pj	width of kernel for parallel operations

IV. CONCLUSION

A high performance and low-power deep learning hardware accelerator is presented in this investigation. The proposed DLA has parallel implementation of the convolution operations. A new Inter-Controller and a Reshape module is presented in this investigation. This reduces the area of the tiles by four folds. A comparison of the implemented hardware with a benchmark CPU implementation to show obscure error with the benchmark. The performance of the hardware was found to be 40.96 GOPS at 196.8 mW power consumption operating at 100 MHz clock rate. It also shows a power efficiency of 0.2081 TOPS/W and an area efficiency of 0.7638 GOPS/mm^2. An FOM based on the power efficiency, operating frequency, normalized process, and normalized voltage showed that our design is the best so far. The same methodology can be utilized in a more advanced CMOS process for better performance.

ACKNOWLEDGMENT

The authors would like to thank Taiwan Semiconductor Research Institute (TSRI) and Taiwan Ocean Research Institute (TORI) in National Applied Research Laboratories (NARLabs), Hsinchu, Taiwan, for the support of the chip

fabrication and measurement, and R/V Legend ship in open water testing.

APPENDIX

Table VI shows all the symbols that are used in the text.

REFERENCES

[1] *UN, Technology and Innovation Report 2021: Catching Technological Waves Innovation with Equity*, United Nations Publications, New York, NY, USA, May 2021. [Online]. Available: https://unctad.org/system/files/official-document/tir2020_en.pdf

[2] A. L. P. D. Ocampo and E. P. Dadios, "Mobile platform implementation of lightweight neural network model for plant disease detection and recognition," in *Proc. IEEE 10th Int. Conf. Humanoid, Nanotechnol., Inf. Technol., Commun. Control, Environ. Manage. (HNICEM)*, Nov. 2018, pp. 1–4.

[3] M. Chang, T.-W. Kim, J. Beom, S. Won, and D. Jeon, "AI therapist realizing expert verbal cues for effective robot-assisted gait training," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 12, pp. 2805–2815, Dec. 2020.

[4] S. Ji *et al.*, "Learning-based automation of robotic assembly for smart manufacturing," *Proc. IEEE*, vol. 109, no. 4, pp. 423–440, Apr. 2021.

[5] C. Chatzikomis, A. Sorniotti, P. Gruber, M. Zanchetta, D. Willans, and B. Balcombe, "Comparison of path tracking and torque-vectoring controllers for autonomous electric vehicles," *IEEE Trans. Intell. Vehicles*, vol. 3, no. 4, pp. 559–570, Dec. 2018.

[6] H. Lim *et al.*, "Deep learning-aided synthetic airspeed estimation of UAVs for analytical redundancy with a temporal convolutional network," *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 17–24, Jan. 2022.

[7] Y.-C. Chou, H.-H. Chen, C.-C. Wang, H.-M. Chou, and C.-C. Wang, "An AI AUV enabling vision-based diver-following and obstacle avoidance with 3D-modeling dataset," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2021, pp. 1–4.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2014, pp. 818–833.

[10] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–4.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[13] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[14] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[15] S. D. Erokhin, "A review of scientific research on artificial intelligence," in *Proc. Syst. Signals Generating Process. Field Board Commun.*, Mar. 2019, pp. 1–4.

[16] C.-H. Yeh *et al.*, "Lightweight deep neural network for joint learning of underwater object detection and color conversion," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 26, 2021, doi: 10.1109/TNNLS.2021.3072414.

[17] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2220–2233, Aug. 2017.

[18] Y. Huan, J. Xu, L. Zheng, H. Tenhunen, and Z. Zou, "A 3D tiled low power accelerator for convolutional neural network," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[19] Y. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[20] L. Du *et al.*, "A reconfigurable streaming deep convolutional neural network accelerator for Internet of Things," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 198–208, Jan. 2018.

[21] Y.-J. Lin and T. S. Chang, "Data and hardware efficient design for convolutional neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 5, pp. 1642–1651, May 2018.

[22] J. Wang, J. Lin, and Z. Wang, "Efficient hardware architectures for deep convolutional neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 6, pp. 1941–1953, Nov. 2018.

[23] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[24] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[25] J. Jo, S. Kim, and I.-C. Park, "Energy-efficient convolution architecture based on rescheduled dataflow," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4196–4207, Dec. 2018.

[26] J. Sim, S. Lee, and L. S. Kim, "An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 87–100, Jan. 2020.

[27] J. S. P. Giraldo, S. Lauwereins, K. Badami, and M. Verhelst, "Vocell: A 65-nm speech-triggered wake-up SoC for 10-μW keyword spotting and speaker verification," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 868–878, Apr. 2020.

[28] S.-F. Hsiao, K.-C. Chen, C.-C. Lin, H.-J. Chang, and B.-C. Tsai, "Design of a sparsity-aware reconfigurable deep learning accelerator supporting various types of operations," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 3, pp. 376–387, Sep. 2020.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[30] K. T. Shao, *The Fish Database Taiwan*. Accessed: Feb. 8, 2019. [Online]. Available: www.fish4knowledge.eu

[31] J. Redmon. (2013). *DarkNet: Open Source Neural Networks in C*. Accessed: 2016. [Online]. Available: http://pjreddie.com/darknet/

[32] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[33] M. Zhang, S. Xu, W. Song, Q. He, and Q. Wei, "Lightweight underwater object detection based on Yolo v4 and multi-scale attentional feature fusion," *Remote Sens.*, vol. 13, no. 22, p. 4706, Nov. 2021.

[34] C. Liu *et al.*, "A dataset and benchmark of underwater object detection for robot picking," 2021, *arXiv:2106.05681*.

[35] M. Pedersen, J. B. Haurum, R. Gade, and T. B. Moeslund, "Detection of marine animals in a new underwater dataset with varying visibility," in *Proc. CVPR Workshops*, Jun. 2019, pp. 18–26.

Chua-Chin Wang (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the State University of New York (SUNY) at Stony Brook, USA, in 1992.

He then joined the Department of Electrical Engineering, National Sun Yat-sen University (NSYSU), Taiwan, where he was elevated to be a Distinguished Professor in 2010. He was nominated as the ASE Chair Professor in 2013 and elected to be the Dean of Engineering Collage in 2014. In 2018, he was assigned as the Director General of Underwater Vehicle Research and Development Center. He is now the Vice President at the Office of Research and Development, NSYSU. His research interests include memory and logic circuit design, communication circuit design, and interfacing I/O circuits.

Dr. Wang became an IET Fellow in 2012. He was named as a Distinguished Lecturer of IEEE Circuits and Systems Society (CASS) (2019–2021). He chaired the IEEE CASS Nanoelectronics and Giga-Scale Systems (NG) Technical Committee (2008–2009). He was the General Chair of the 2015 Symposium on Engineering Medicine and Biology Application (2015 SEMBA), the 2012 IEEE Asia–Pacific Conference on Circuits and Systems (2012 APCCAS), and the 2011 IEEE International Conference on IC Design and Technology (2011 ICICDT). He was the General Co-Chair of the 2010 IEEE International Symposium on Next-Generation Electronics (2010 ISNE). He was the General Chair of the 2007 VLSI/CAD Symposium. He was an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS from 2010 to 2011.

Ralph Gerard B. Sangalang (Member, IEEE) received the B.S. degree in electronics and communications engineering and the M.S. degree in electronics engineering from Batangas State University, Philippines. He is currently pursuing the Ph.D. degree in electronics engineering and the Ph.D. degree in electrical engineering under the double degree program with Batangas State University and the National Sun Yat-sen University, Taiwan. He has been with Batangas State University since 2009 and was the Student Outcome Committee Chair of the College of Engineering, Architecture and Fine Arts from 2014 to 2021. He was the Program Chair of B.S. Electronics Engineering from 2017 to 2021 and the Interim Program Chair of the B.S. Biomedical Engineering. He is a member of Batangas State University's CenTraL or the Center for Transformative Learning. His research interests include memory design, digital systems, control systems, computational modeling, fractional circuits, and engineering education.

Chien-Ping Kuo received the B.S. degree in electrical engineering from the National Chiayi University (NCYU), Chiayi, Taiwan, in 2019, and the M.S. degree from the National Sun Yat-sen University (NSYSU), Kaohsiung, Taiwan, in 2021. He is now connected to MediaTek Inc., Taiwan. His research interests include low-power SRAM circuit design, mix-signal IC, and design and high performance AI accelerator design.

Hsin-Che Wu received the B.S. degree in electrical engineering from the National Sun Yat-sen University (NSYSU), Kaohsiung, Taiwan, in 2019, where he is currently pursuing the M.S. degree. His recent research interests include negative charge pump circuit design, mix-signal IC design, and high performance AI accelerator design.

Yi Hsu received the B.S. degree from the Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Taiwan, in 2019, and the M.S. degree from the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, in 2021. He is now connected to MediaTek Inc., Taiwan. His research interest is AI hardware accelerator design.

Shen-Fu Hsiao (Member, IEEE) received the B.S. degree in electrical engineering from the National Taiwan University, Taiwan, in 1985, the M.S. degree in electrical engineering from the National Chiao Tung University, Taiwan, in 1987, and the Ph.D. degree from Yale University, USA, in 1993. Since 1993, he has been with the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan, where he is currently a Professor. From 2002 to 2003, he was a Research Fellow with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison. He was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle, from August 2007 to July 2008. His current research interests include computer arithmetic, deep learning, and VLSI design.

Chia-Hung Yeh (Senior Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Electrical Engineering, National Chung Cheng University, Chiayi, Taiwan, in 1997 and 2002, respectively. He was an Assistant Professor (from 2007 to 2010), an Associate Professor (from 2010 to 2013), and a Professor (from 2013 to 2017) at the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan. He is now a Distinguished Professor at the National Taiwan Normal University (NTNU), Taipei, Taiwan. He has coauthored more than 250 technical international conference papers and journal articles. He holds 47 patents in the U.S., Taiwan, and China. His research interests include image/video processing, deep learning, 3-D reconstruction, and video coding. He was elected as a fellow of the Institution of Engineering and Technology (IET) in 2017. He received the IEEE Multimedia Signal Processing (MMSP) Top 10% Paper Award in 2013, the IEEE Global Conference on Consumer Electronics (GCCE) Outstanding Poster Award in 2014, the Asia–Pacific Signal and Information Processing Association (APSIPA) Distinguished Lecturer in 2015, the NTNU Distinguished Professor Award in 2017, and the Outstanding Technical Achievement Award from IEEE Tainan Section. He was the Tainan Section Chair of the 2017 IEEE Signal Processing Society (SPS). He has served as an Associate Editor for the *Journal of Visual Communication and Image Representation*, the *EURASIP Journal of Advances in Signal Processing*, and the *International Journal of Pattern Recognition and Artificial Intelligence*.