Field Programmable Gate Array with virtualization and dynamic channel allocation: An efficient solution for Deep Learning algorithms

Shivaprasad Rai B **Dr. Chethana R Murthy**
Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering,
Bangalore, Karnataka, India -560069

Abstract— Field Programmable Gate Array (FPGA) is a computational unit that is used for custom logic implementation. These devices are special due to their flexibility in use. FPGA can be configured and reconfigured when there is a need for a change in the functionality of a connected device. To do any reconfiguration do not need to physically remove the connected FPGA hardware. FPGAs are also a better solution if looking for enhancing the existing hardware infrastructure for advanced usage. Increased execution lifetime is also an added advantage when we consider FPGAs for our implementation. Fetching data for computation is a very critical process when we execute any application on either our Personal Computer or in datacenter servers. If the CPU is involved in every data read/write operation then it becomes overhead for the processor because of this the total performance of the system will degrade. The direct Memory Access controller is an external device that takes care of data transfer in both CPU registers to memory and memory to memory. The MCDMA IP for PCI Express enables the efficient transfer of data using multiple DMA channels between the host and device. The flexible nature of an FPGA arises with the considerable overhead in size, time, and energy usage: an FPGA requires approximately much more area when compared with a normal Application Precise IC component, and also gives slower performance when considering the speed of computation with Application Specific IC and needs much more dynamic power. This drawback is due to FPGA's programmable routing fabric which needs more space, computation time, and energy to implement "immediate" fabrication. Reprogrammable hardware devices used for performance enhancement have the drawbacks that the connecting functionality provided by hardwired IP blocks covers only the lower protocol layers, and a huge effort is required to achieve high-throughput data transmission between host and accelerator devices. MCDMA for PCI Express operates using a FIFO data structure that stores descriptors, the descriptor queue is set up by the driver program to move data between FPGA and host. Its program fetches descriptors and processes them to get transmission data. To improve the performance, different descriptor queue structures can be used for host-to-device and device-to-host operations for each channel. On the user logic side, we can use memory-mapped and streaming interfaces which will allow for easy integration of the Multi-Channel Direct Memory Access Intellectual Property with other Platform Designer components. In summary, introducing Single Root IO Virtualization on the physical function which supports the creation of virtual functions, and using dynamic channel allocation, can achieve enhanced speed and performance efficiency. With this experiment, we got an 8.49% performance enhancement in terms of bandwidth, when SRIOV and DCA are enabled.

Keywords—Field Programmable Gate Array, Deep Learning, YOLO, Performance enhancement, Single Root IO Virtualization, Dynamic Channel Allocation

INTRODUCTION

Reconfigurable flatforms like Field Programmable Gate Arrays (FPGA) are getting more popular with the growth of reconfigurable hardware technologies. FPGAs have many benefits over Application-Specific Integrated Circuits (ASICs) like the ability to reconfigure the entire internal circuit as and when needed, and less development life cycle which results in faster time to market, using FPGA's efficient system models can be implemented and very good performance gain when compared with the same kind of programming applications running in ASIC's. Due to new developments in digital technologies, big data, the advancement of Artificial Intelligence, Deep Learning, and Neural Networking have developed and have demonstrated their ability and effectiveness in solving complicated learning problems ever seen before. Digital technology is evolving more than ever before, with new and advanced technology not only to process the data but also to create, transfer and store data there are very efficient devices one of the best examples is a smartphone. The amount of digital data created nowadays is increasing exponentially. These development open doors for new technologies like big data, Artificial Intelligence (AI), Deep Learning (DL), and Neural Networking. These technologies are capable to solve very complex learning problems. In particularly Convolutional Neural Networks (CNNs) are used very effectively in applications like not only object detection but also recognizing the detected objects from the input data and specifying the properties of the recognized objects. However, these complex algorithms need exhaustive CPU cycles and high memory bandwidth. So, by running these algorithms in general CPUs it will fail to achieve desired performance level. One of the best solutions is using the hardware accelerators like FPGAs or Graphical Processing Unit (GPU) to get the desired throughput for these DL algorithms. In particular, FPGAs are the best-known solution to get high performance from the implementation of DL algorithms in terms of computation speed. The reason for this high performance is each module can be executed parallel in a single FPGA chip. Part of FPGA can be configured to make it customized for a specific DL module at run time without disturbing the other modules running in the same Integrated Circuit (IC).

DL algorithms consume a lot of data, especially when it comes to the part of object detection and measurement deep learning algorithm uses thousands of input data sets, and getting these data from the memory is a time-consuming task for the processor. To make it simple and more time-efficient Direct Memory Access is introduced by using FPGA. If every data transfer request is monitored by the CPU, then the processor will be overloaded with data transfer requests only. Direct memory Access controller is an external hardware system that handles the data transfer monitoring once the CPU does initial handshaking with the Peripheral devices. In this methodology, any device that needs memory access first authenticates itself with the operating system, and then the operating System hands over the control to the DMA controller circuit so that the authenticated devices can directly access data from the memory using the system data bus. It drastically decreases the overhead for the CPU. CPU cycles can be used for other more productive operations so that overall throughput can be increased.

LITERATURE SURVEY

Reconfigurable circuits play a huge part in modern electronics and computation. In both wireless and wired communications, military and aeronautics, and industrial embedded applications, FPGAs give adequate resources for power consumption, reliability, and security. FPGAs demonstrate value in applications such as hardware acceleration, image processing, AI, and edge computing with strong digital signal processing and memory resources. When there is a requirement for a different configuration of hardware circuits to run our software program either need to choose multiple customs Integrated Circuits in which each IC is designed for a specific purpose or we can choose a single device that can be reconfigured for the custom logic whenever needed. Field Programmable Gate Arrays (FPGAs) are devices that can be fully or partially reconfigured without disturbing the remaining part of the execution. That is using SROM Object File or bitstreams we can flash or reconfigure the FPGA device without removing the device also we can configure part of the FPGA circuit without disturbing the modules running in the remaining part of the same FPGA. This reconfiguration can be done even remotely by terminals connected through the internet. This property of FPGA is utilized by many cloud service providers to provide FPGA devices as Infrastructure as a Service (IaaS). These silicon chips can be flashed using pre-defined bitstream files to convert them into any type of digital circuits or system required to run the applications over it. This means there is no need to change the underlying hardware if the software is upgraded with advanced Instruction Set Architecture. FPGA is used in many products like Television transceivers, Signal Processing Systems, the Internet of Things, in smartphones, and 5G technologies highly using FPGA for computation purposes and in datacentres to process high volume and complex structured/ unstructured data. Conventional CPUs follow the concept of “binary” changes using dynamic memory and the logic of execution. But providing the configuration bits for a chip will make it more general by providing a facility for programming the hardware circuit itself. An added advantage of reconfiguring the hardware device is programmer can create the standard binary bitstream files for a specific circuit and map that file to any of the configurations for corresponding FPGA devices without any extra cost.One more advantage of FPGA over predefined and fixed circuits- Application Specific Integrated Circuit (ASIC) is less development and marketing time. ASIs normally required several months to fabricate and also need very costly equipment for the fabrication of the first model. But when FPGAs have considered it takes only a few seconds to flash the bitstream files and make the device get up. Also, we have bitstreams that can be changed if any mistakes are made while programming them. This will drastically decrease the production cost of the FPGA circuits.The author in [1] Explains how accelerators that utilize application-specific integrated circuits, graphic processing units, and field-programmable gate arrays (FPGAs) have been used to enhance the throughput of Convolution neural Networks. in more detail this paper explains, how and why FPGAs have been getting significant usage for accelerating the execution of deep learning networks due to energy efficiency and ability to execute each neuron parallel. The author also emphasized how to improve acceleration performance using FPGA-based accelerators. Performance per watt of power consumption is maximum in FPGAs which makes FPGA is an excellent choice for devices that use power cells for their operations and also in cloud service providing high computational capable servers. The challenges for using FPGA for Deep learning are mainly about the storage and external memory bandwidth. To overcome this, we need carefully designed data access methods that will help to decrease the memory bandwidth requirements.While using the YOLO CNN model for object detection, the binary weight helps us to store the entire network model in Block RAMs of a field-programmable gate array (FPGA) to decrease off-chip entrees frequently and it gives much more performance improvement [2]. Object detection is an important mission in computer vision. Normally we use GPUs to execute Deep Learning models, but it is inefficient in tasks such as selecting the size of data bit and controlling data read/written by the external memory.FPGA is the best alternative to address these issues. FPGAs have been normally utilized for high-performance deep learning models due to their elastic design and minimum development cycles. If we optimize high-performance, Scalable CNN accelerators for depth separable convolution we get a smaller number of operations to need and also a remarkably less number of parameters we need to input. This modification creates a new possibility to run the CNN on portable devices. It makes the entire neural network model can be fit into large or small FPGA [3].FPGAs, hardware accelerators used with FPGAs are a hopeful key for Network Functions Virtualization (NFV) and 5G cloud environments due to their enhanced turnaround time and considerable acceleration possibility through parallel execution on the reconfigurable fabric [4]. Network Function Virtualization (NFV) is a major change for telecommunication operatives’ datacentres, also virtualization is used to low delay high-performance requirements such as broadcasting and broadband multimedia systems. FPGAs are ever more used in such environments, utilizing their computing power and energy efficiency ratio, improved transistor per unit area, and reconfigurability. The communication system of the FPGA Virtualization Manager enables acceleration provision for virtual machines, unikernel, and containers. If we can control the virtualization framework by the cloud stack then it supports virtualized hardware accelerator allowing the acceleration of a large number of guest applications. In this paper [4] author explained a new communication interface called the FPGA manager it allows dynamic remote methods using a set of instructions used to hardware accelerators and VNF status. Also, we can track the resources used using software inside the FPGA MCU.

One more most known application of FPGAs is for graphics processing applications [5]. These reconfigurable devices support spatial and temporal parallelism. Parallelization depends on the processing architecture and hardware restrictions of the system. Those limits can restrict the designer to redevelop the algorithm. FPGAs are very popular as implementation platforms due to their frequent evolution in functionality and dimensions, particularly for image processing and video processing. The applications like edge detection can be executed more efficiently on FPGAs.Feed-Forward Neural Networks (FFNN) is one of the often-utilized machine learning algorithms, with many use cases naturally executed using PC-based software systems. If speed enhancement in execution is essential in real-time applications or better forecast, decision, or classification, PC-based models are unable to deliver the required result. In recent days, this becomes very normal due to FFNN sizes are growing because of the complication of the problems to be answered and big data. Programs, with huge sizes of inputs, neuron elements, and the count of layers. And also, energy usage and processing speed are vital questions; GPUs and CPUs are capable to execute data at a higher speed, but they need more energy and resources compared with FPGA and other custom embedded hardware platforms [6].Quantum Communication (QC) is the most evolving application of quantum technology and is now deploying in different applications. Both Quantum Random Number Generation (QRNG) and Quantum Key Distribution (QKD) are important Quantum Communication technologies. The combination of these two allows building the perfect security protocol system, resilient to any outside attack. The implementation of such kind of system needs the design and development of numerous mechanisms: from visual setup to the underlying electronic components, from the circuit that controls the digital signals to the administration software. In addition to the specific quantum implementations that will differ depending on various safety protocols, an important module of the entire arrangement is a regulation board capable to deliver deterministic behavior, high time-based resolution, and high-speed code execution. A Field Programmable Gate Array (FPGA) is the best choice for these types of applications [7]. FPGA also gives a benefit in terms of power saving, which can be an important feature for complex applications like the CubeSat project for Quantum Communication using Satellite. The use of DMA is to decrease the workload on the processor. As the term indicates, it accesses memory directly for peripheral devices. Data access from the memory is made by the DMA directly instead of the processor. DMA authorizes peripheral devices to read and write data in the memory directly, without depending on the processor. Hence, the processor can execute other tasks simultaneously, while the DMA is fetching the content of the memory. As such, the overall performance of the system is boosted [8]. DMA appears to be an easy concept, but system implementation with other hardware subsystems is a complex technique. DMA has different applications, such as network cards, graphics cards, and disk drive controllers. In addition to computer systems, DMA plays a vital role in Software on Chip (SoC) and embedded systems, facilitating good speed for moving data to externally coupled peripheral components.Graphics processing units (GPUs) can have better performance and better bandwidth in terms of memory access. But, to qualify parallel tasks to run on different types of clusters, the implementation of inter-accelerator communication between the nodes is essential. For this purpose, many memory replicas using the CPU are created. Normally when a small message is involved in communication it surges delay and negatively affects the performance of the application. As an accelerator GPU does not perform well in applications that use complex procedures using interrupts, non-single instruction, multiple data streams (SIMD), and moderately poor parallelism. FPGAs can give a solution for the above problem deficiently [9]. The hardware accelerator controlled by direct memory access (DMA) is greatly affected by the communication bandwidth from/to DRAM through on-chip buses. To facilitate the optimization of communication schemes (CS), a communication primitive (CP) is defined by the memory bank allocation and the number of activated DMACs [10].

MOTIVATION

The main motivation for this research work is to identify the performance enhancement achieved by introducing FPGA In a deep learning algorithm and by introducing dynamic channel allocation and SRIOV. With the video file as input first, the model tries to detect the objects using the YOLO algorithm Customised data set is created to find the different real-world objects and the data set is created in such a way that during the in-detection phase itself the model will try to categorize the detected objects according to their real-world size. Once the model is trained with a data set then it will detect and categorize the objects with respect to their nature and roughly on their dimensional property. Once the object is detected YOLO algorithm will draw a Bounding Box For the detected object. This experiment will be done with and without FPGA and also with the support of virtualization and dynamic channel allocation through bit stream.

DESIGN

A. System Architecture

Here in this section architecture of the proposed system is presented. It is a conceptual model which classifies the behavior and structure, and also provides an abstract view of the system. The block diagram of the system is shown in figure 1.

Dynamic channel allocation helps for faster data transfer between CPU and main memory, Virtualization allows faster execution of the neural networks. Both of these features help FPGA to give enhanced performance results while running

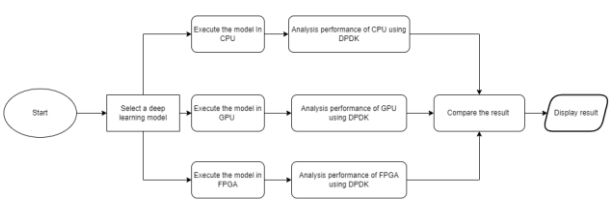

Figure 1. System Design

the DL algorithms. In this experiment different combinations of testing are done to compare the result, with only CPU, With GPU accelerator, and with FPGA accelerators experiment is done. Also, FPGA is loaded with bitstream which supports DCA and SRIOV and is tested for enhanced performance output. Intel dev cloud is used for all this experimental setup.

B. Data flow Diagram of the proposed FPGA system

The data flow diagram level 1 for the project implementation is shown in Fig.4.3. The level 1 DFD deals with sub-processes of the system represented in the context diagram discussed in the earlier section. In this project, the entire system is broken into 4 sub-processes namely generating bitstream, programming bitstream on an FPGA device, loading the driver, and finally running the application.

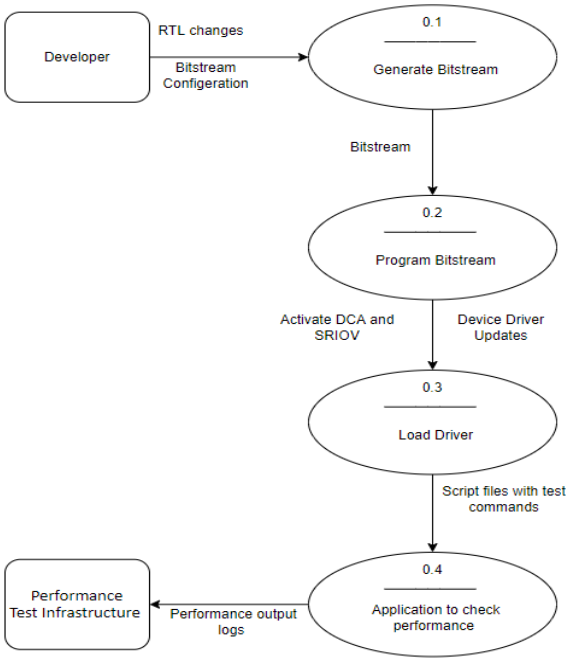

Figure 2. DFD for bitstream generation and testing

C. Algorithms Used

Algorithm to generate a bitstream

It defines the pseudocode for generating bitstream using the user input from the CSV file which defines all properties of the bitstream. The output expected is either the bitstream with specified capability or the detailed error message with a description

Input: Device type, number of pf, channels per pf, number of vf, channels per vf, number of active channels, bitstream type, SRIOV enable, disable (.csv format)

Output: Bitstream generated .sof file, software folder used for testing

Method:

```
begin
    for each entry in the csv file:
        featureOfBS: =Read the values of each entry;
    assign the feature for Quartus API;
    pointerToBS:= generateBSQuartus(featureList);
    wait until (BS is generated by Quartus);
    path_bitstream:=copy the software and BS to the local server
        where the FPGA device is connected using the flask server;
    return: path_bitstream;
end
```

Algorithm for loading device driver

This pseudocode defines the way how generated bitstream is programmed in the FPGA device. Then testing is done using the different available device drivers according to the requirement.

Input: Bitstream (BS) generated (.sof format)
Output: FPGA is programmed with BS, performance test result
Method:

```
begin:
    program bitstream using the Jtag cable;
    reboot machine;
    check lspci –d 1172: -v ;
      if ( device found ):
        check for features of the device:
          if (all required features are present):
            compile all program files (.c files) ;
                load required driver;
            check for driver loaded or not:
              if (driver loaded):
                get input value for performance test: Time,
                payload, number of channels, number of
                threads, execute performance test;
                return result;
              else:
                return error: "driver not loaded";
          else:
            return error: "bitstream has no required capabilities"
      else:
        return error: "device not loaded"
end
```

Algorithm for Dynamic Channel Allocation

This defines pseudocode for DCA, which allocates channels dynamically for different descriptor and virtual functions, each virtual functions have predefined virtual channels these virtual channels are mapped dynamically to the available physical channels.

Input: Maximum number of channels, number of vf, number of pf, payload size, number of queues
Output: Dynamically allocated channels for each queue
Method:

```
begin
    if request for payload transmission:
      for each descriptor queue:
        for each unallocated channel:
          currentAllocatedChannel:=selecet one channel from the
          pool
          Allocate the currentAllocatedChannel to the descriptor
          queue continue the data transmission
          if data transmission == done:
          currentAllocatedChannel:=
          currentAllocatedChannel_DiscriptorQueue
          deallocate all memory used by the queue descriptor
          return the currentAllocatedChannel into the pool
        done
      done
end
```

Algorithm for creating virtual functions on physical functions

This pseudocode defines steps to create virtual functions on the above available physical functions.

Input: Bus: Device. Function (BDF), number of vfs need to create on the pf,
maximum number of channels allocated for each vf

Output: VF with a specified number of virtual channels

Method:

```
begin
    if SRIOV is enabled:
        if number of requested vf <= maximum num of allowed vf:
            for i=numberOfPF:
                vfi =createVirtualFunction on specified pf;
                vf i,number of channels = max number of channels;
                i++;
            repeat until: i <=num of vf required
            return true;
        else
            return " permission denaied";
    else:
      return "SRIOV not enabled"
```

FINAL OUTPUT

This section shows the final result of DCA and SRIOV and the performance comparison of different devices with FPGA when used with object detection modules.

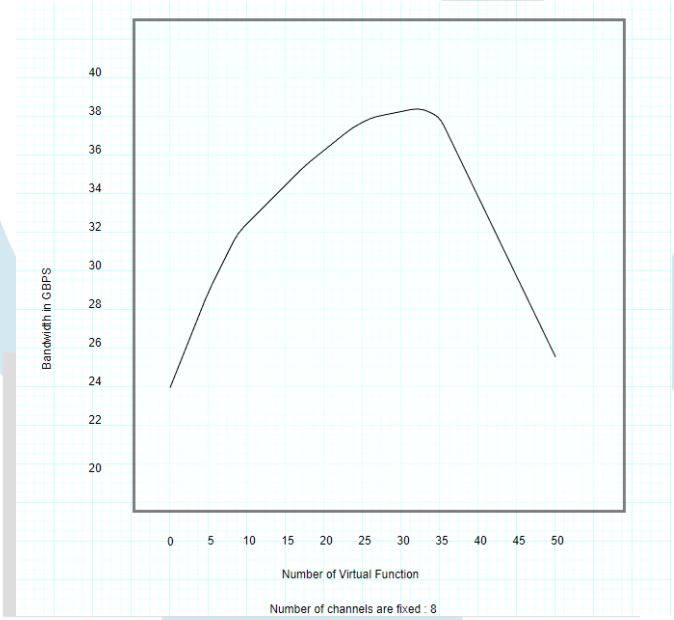

Fig 4 Bandwidth versus Number of vf

Fig 4 shows performance change with bandwidth with respect to a different number of virtual functions(vf). Here maximum bandwidth is reached at 32 vfs, after that the performance decrease is observed due to the overhead of executing vf management modules.

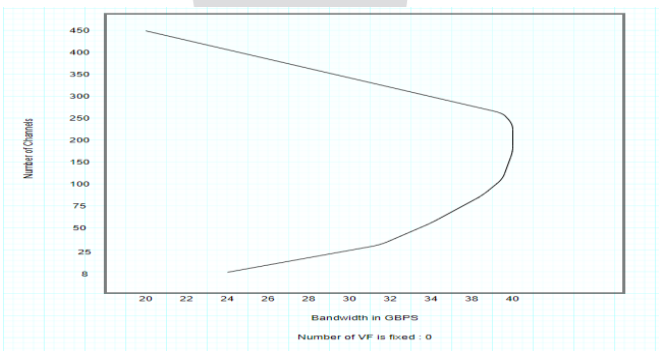

Fig 5: Bandwidth versus Number of channels

Fig 5 shows performance change with bandwidth with respect to a different number of channels, here maximum bandwidth is reached at 256 channels, after that the performance decrease is observed due to the overhead of modules used to control DCA.

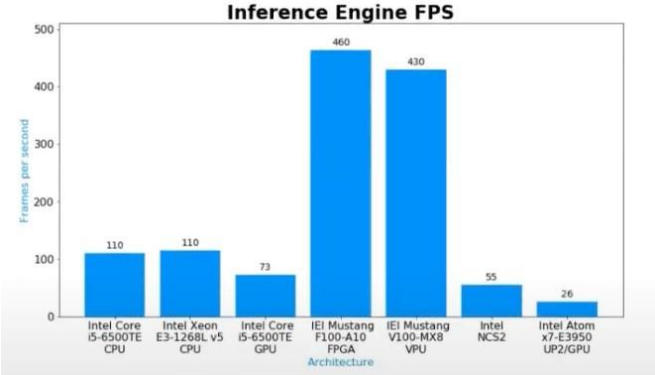

Fig 6: Comparison of different accelerators

Fig 6 shows a comparison of different accelerators which are tested using CNN object detection modules. Here we can see FPGA is 6 times faster than GPU when tested with the same input parameters.

CONCLUSION

To improve the data transfer performance of the accelerator in terms of speed and efficient use of resources, and enhanced design of device driver software is required. After a literature survey and research on the existing system, it is derived that to get better performance virtualization and introducing the dynamic method of channel allocation an adequate methodology. This method introduces the virtualization of descriptor functions and dynamic channel allocation and reallocation by mapping between the physical channel and virtual channels and also by utilizing the reprogramming capability of the FPGA device

With SRIOV and DCA enabled there will be a performance enhancement of 8.49% In terms of bandwidth. This gave six times faster frame processing with object detection algorithm when compared with GPU performance with the same input parameter.

REFERENCES

[1] Shawahna A, Sait SM, El-Maleh A. FPGA-based accelerators of deep learning networks for learning and classification: A review. IEEE Access. 2018 Dec 28;7:7823-59.

[2] Nguyen DT, Nguyen TN, Kim H, Lee HJ. A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2019 Apr 1;27(8):1861-73.

[3] Bai L, Zhao Y, Huang X. A CNN accelerator on FPGA using depthwise separable convolution. IEEE Transactions on Circuits and Systems II: Express Briefs. 2018 Aug 17;65(10):1415-9.

[4] Pinneterre S, Chiotakis S, Paolino M, Raho D. vFPGAmanager: A virtualization framework for orchestrated FPGA accelerator sharing in 5G cloud environments. In2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) 2018 Jun 6 (pp. 1-5). IEEE.

[5] Nguyen DT, Nguyen TN, Kim H, Lee HJ. A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2019 Apr 1;27(8):1861-73.

[6]. S. S. Ankad, B. Shivaprasad Rai, A. Nasreen, S. Mathur, P. Ramakanth Kumar and K. Sreelakshmi, "Object Size Measurement from CCTV footage using deep learning," 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2021, pp. 1-5, doi: 10.1109/CSITSS54238.2021.9683671.

[7] Medus LD, Iakymchuk T, Frances-Villora JV, Bataller-Mompeán M, Rosado-Muñoz A. A novel systolic parallel hardware architecture for the FPGA acceleration of feedforward neural networks. IEEE Access. 2019 Jun 5;7:76084-103.

[8] Stanco A, Santagiustina FB, Calderaro L, Avesani M, Bertapelle T, Dequal D, Vallone G, Villoresi P. Versatile and concurrent FPGA-based architecture for practical quantum communication systems. arXiv preprint arXiv:2107.01857. 2021 Jul 5.

[9] Ahmed MA, Aljumah A, Ahmad MG. Design and Implementation of a Direct Memory Access Controller for Embedded Applications. International Journal of Technology. 2019;10(2):309-19.

[10] Kobayashi R, Fujita N, Yamaguchi Y, Nakamichi A, Boku T. GPU-FPGA heterogeneous computing with opencl-enabled direct memory access. In2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) 2019 May 20 (pp. 489-498). IEEE.

[11] Wang J, Park S, Park CS. Optimization of Communication Schemes for DMA-Controlled Accelerators. IEEE Access. 2021 Oct 5;9:139228-47.

[12] Dave M, Jagtap S. Design and Verification of Configurable Multi-channel DMA controller. IJARIIEISSN(O)-2395-4396, Vol-3Issue-22017